Минобрнауки России
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ И ПРАКТИЧЕСКИХ РАБОТ

по дисциплине

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

специальности СПО
09.02.07 Информационные системы и программирование

2021

УТВЕРЖДЕНЫ

Цикловой комиссией информационных технологий

Протокол от «14» октября 2021 г. № 3

Председатель цикловой комиссии _____ И.В. Миляева

Авторы: Афанасьева С.М., канд. техн. наук

Содержание

Практическое занятие № 1
Составление ветвящихся алгоритмов

Цель работы: научиться составлять одну из базовых структор алгоритмов – ветвящиеся алгоритмы.

Краткие теоретические сведения.

Разветвляющийся называется алгоритм, в котором действие выполняется по одной из возможных ветвей решения задачи, в зависимости от выполнения условий. В отличие от линейных алгоритмов*, в которых команды выполняются последовательно одна за другой*, в разветвляющиеся алгоритмы входит условие, в зависимости от выполнения или невыполнения которого выполняется та или иная последовательность команд (действий) – ветка (рисунок 1).

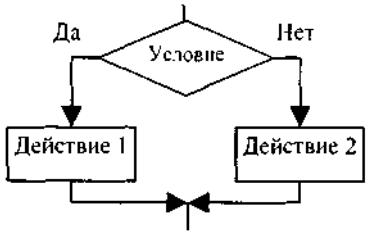


Рисунок 1

В качестве условия в разветвляющемся алгоритме может быть использовано любое понятное исполнителю утверждение, которое может соблюдаться (быть истинно) или не соблюдаться (быть ложно). Такое утверждение может быть выражено как словами, так и формулой. Таким образом, алгоритм ветвления состоит из условия и двух последовательностей команд.

Ветвящийся процесс, включающий в себя две ветви, называется простым, более двух ветвей — сложным. Сложный ветвящийся процесс можно представить с помощью простых ветвящихся процессов.

Примером может являться разветвляющийся алгоритм, изображенный на рисунке 2. Аргументами этого алгоритма являются числа *A* и *B*, а результатом — число *X*. Если условие *A >= B* истинно, то выполняется операция умножения чисел *(X = A* * B), в противном случае выполняется операция сложения *(X = A + B).* В результате печатается то значение *X,* которое получается в результате выполнения одного из действий.

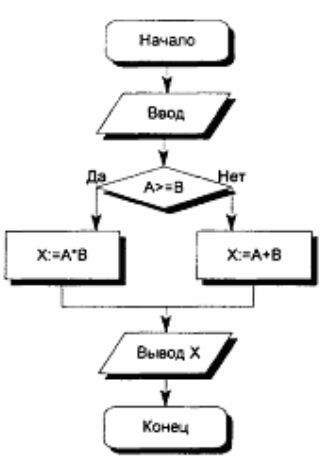


Рисунок 2

Пример выполнения задания

Задание: по заданным координатам *x* и *y* определить, где находится точка (рисунок 3): внутри заштрихованной области; вне заштрихованной области; на границе этой области.

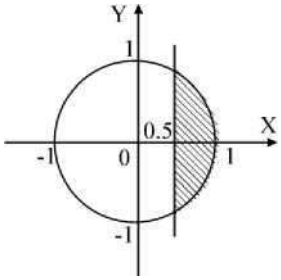


Рисунок 3 — Постановка задачи

Метод решения задачи:

1. Для решения задачи будем использовать уравнение окружности $x^2 + y^2 = R^2$. Так как R=1, то уравнение принимает вид $x^2+y^2 = 1$.
2. Определяем условие, при котором точка будет находиться внутри заштрихованной области: (x>0.5) и (x^2+y^2<1).
3. Определяем условие, при котором точка будет находиться вне заштрихованной области: (x<0.5) или (x^2+y^2>1).

Составим блок-схему решения задачи (рисунок 4).

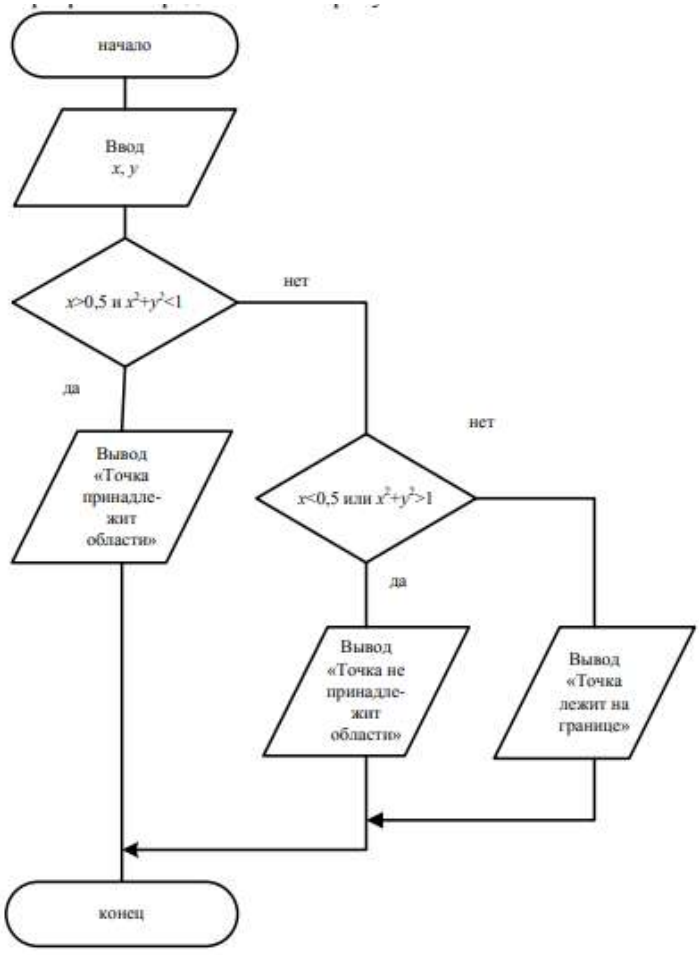


Рисунок 4

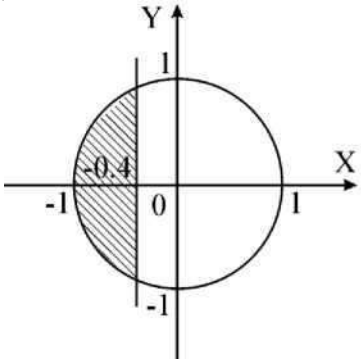
Варианты заданий

По заданным координатам точки определить, где находится точка:

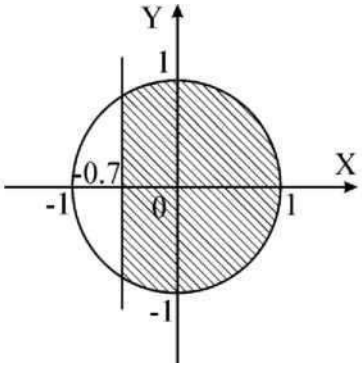
1) внутри заштрихованной области;

2) вне заштрихованной области;

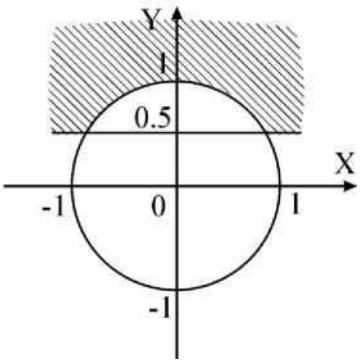
3) на границе этой области.



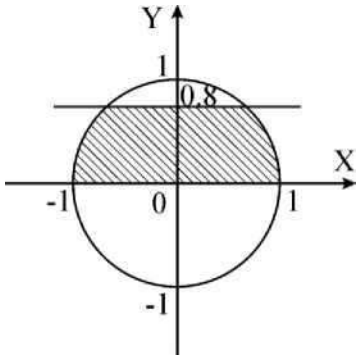
1.



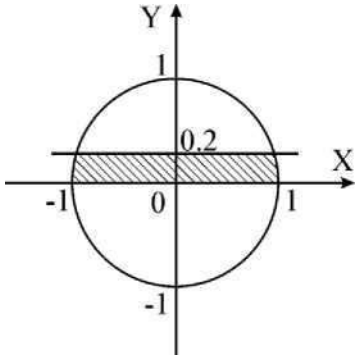
2.



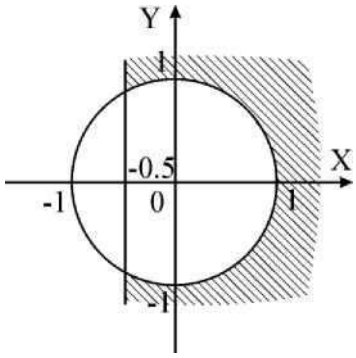
3.



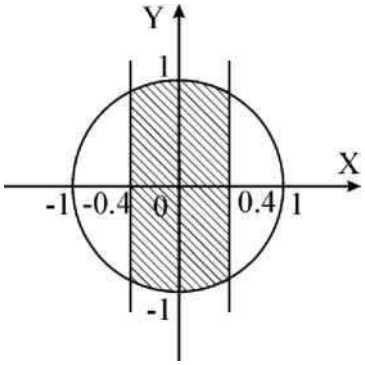
4.



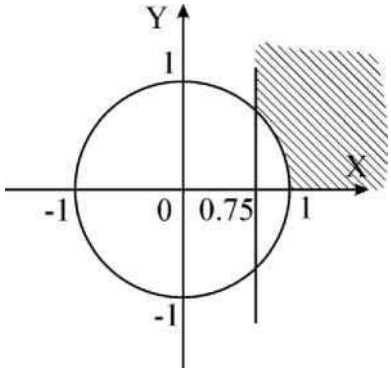
5.



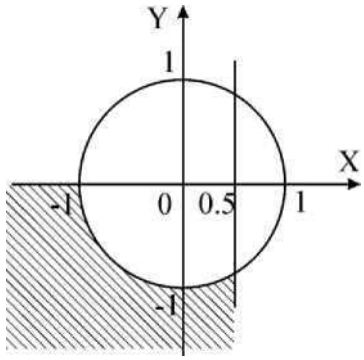
6.



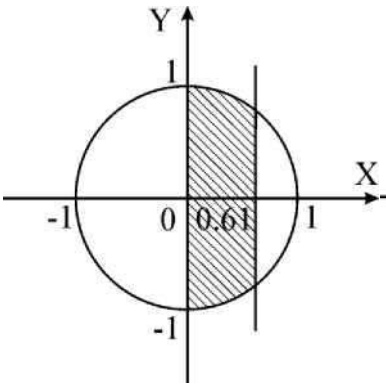
7.



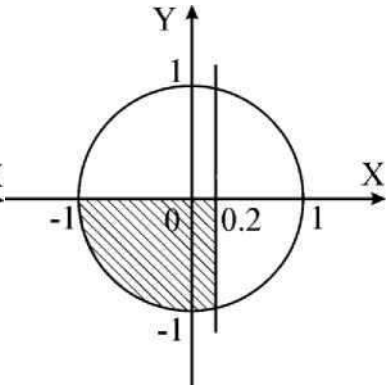
8.



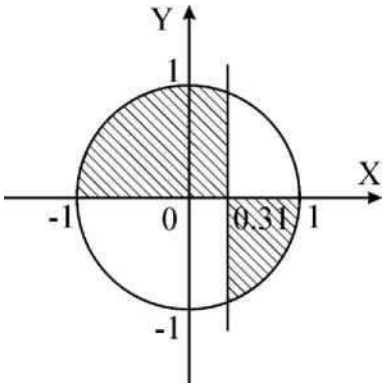
9.



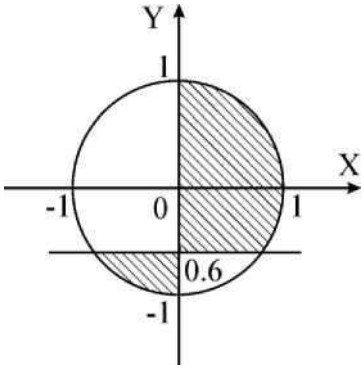
10.



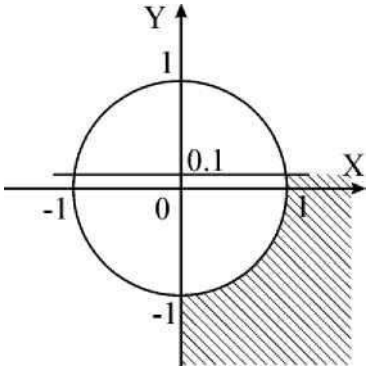
11.



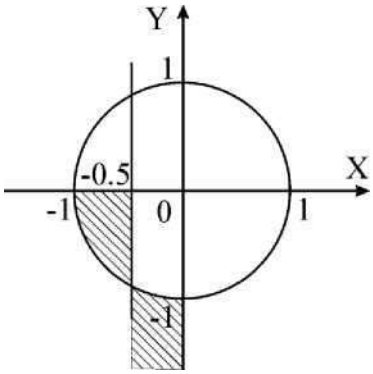
12.



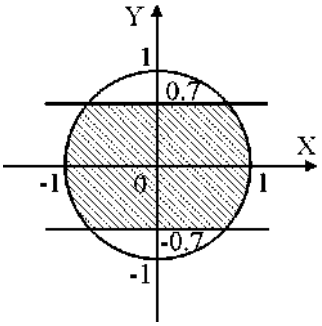
13.



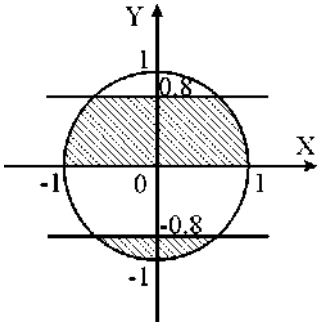
14.



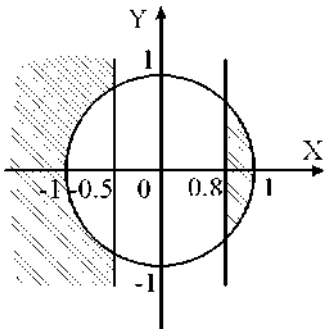
15.



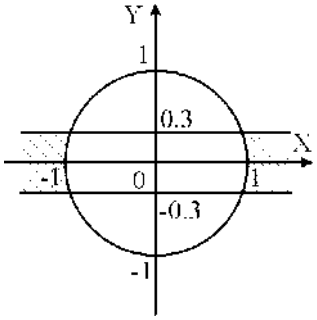
16.



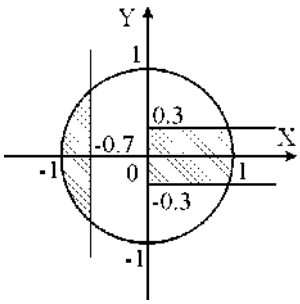
17.



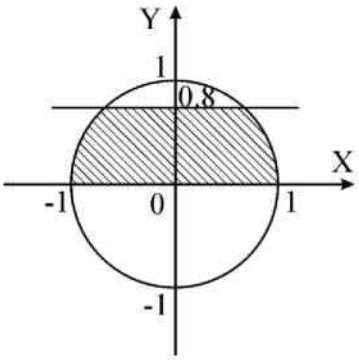
18.



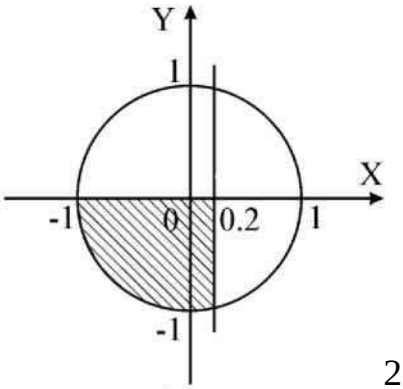
19.



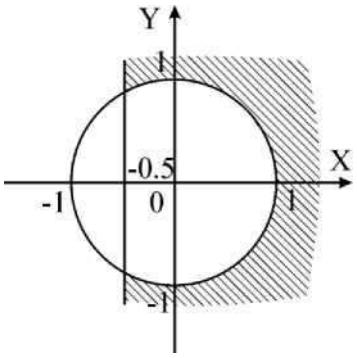
20.



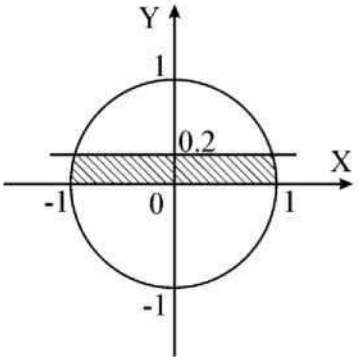
21.



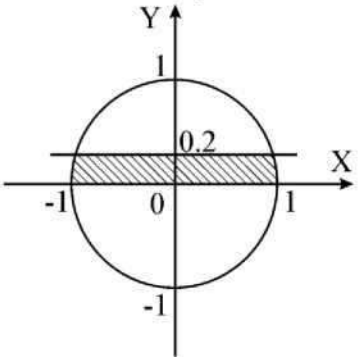
22.



23.



24.



25.

Практическое занятие № 2

Составление циклических алгоритмов

Цель работы: научиться составлять одну из базовых структор алгоритмов – циклические алгоритмы

Краткие теоретические сведения.

Цикл – процесс повторения одних и тех же операций (группы команд).

Тело цикла–это последовательность операций (команд), которая повторяется многократно заданное количество раз или до тех пор, пока не будет выполнено условие. Существуют три схемы представления циклических алгоритмов.

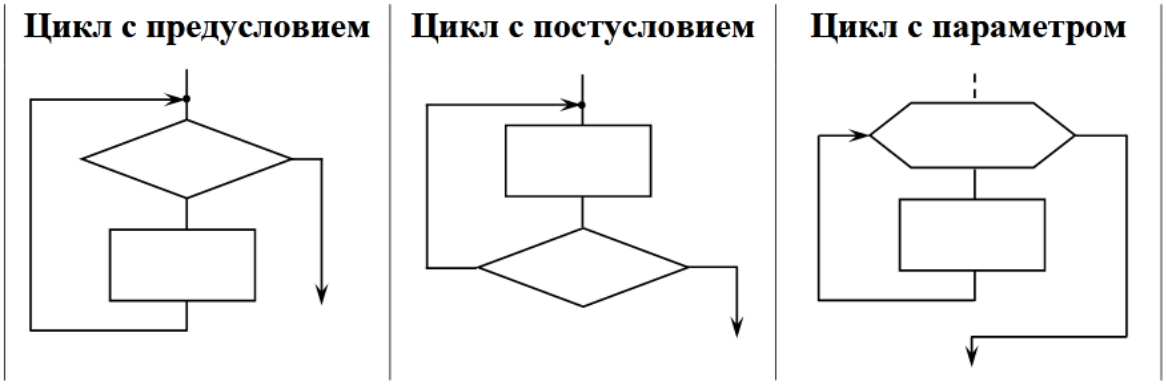


Рисунок 1

Цикл с предусловием

Особенностью первой схемы (цикл с предусловием) является то, что проверка условия проводится до тела цикла. В том случае, если условие выхода из цикла выполняется, то тело цикла не выполняется ни разу. Например, по условию необходимо вывести все точки графика $y=x^2$ на отрезке для x от -5 до 5 с шагом 1 сантиметр. Алгоритм решения задачи может быть представлен блок-схемой (рисунок 2).

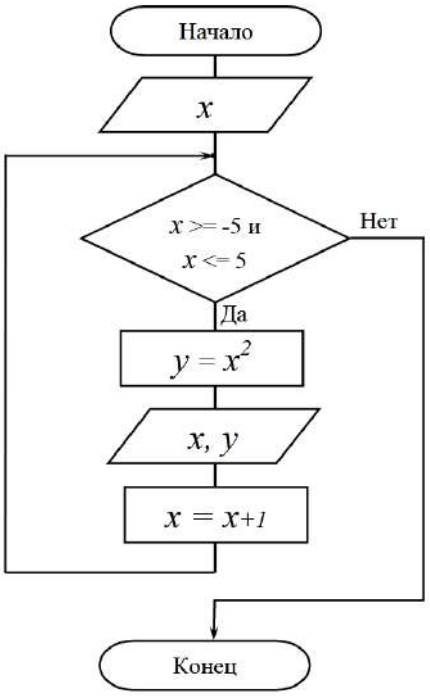


Рисунок 2

По представленному алгоритму с предусловием, если с клавиатуры будет введено стартовое значение хне удовлетворяющее условию $-5 \leq x \geq 5$, то решение задачи закончится после первого же неверного шага.

Цикл с постусловием

При решении задачи по циклическому алгоритму с постусловием в любом случае цикл будет выполнен хотябы один раз, так как первая проверка условия выхода из цикла осуществляется после того, как тело цикла выполнено. Например, блок-схема циклического алгоритма с постусловием для решения той же выше поставленной задачи представлена на рисунке 3.

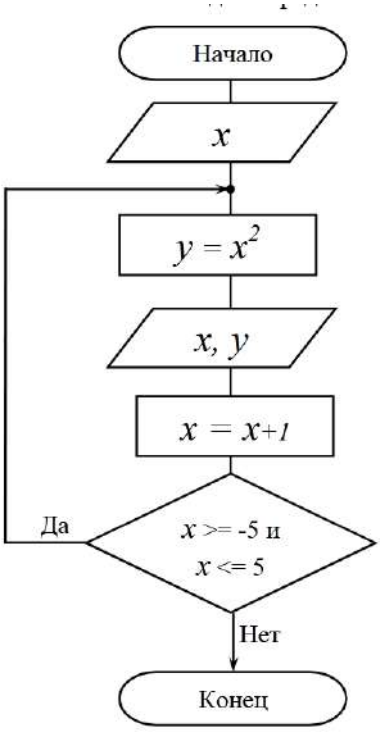


Рисунок 3

На практике обе схемы равноценны по применению частота и логичность их применения зависит от условия решаемой задачи.

Цикл с параметром

В условии многих задач четко оговаривается количество повторений операций тела цикла или, например, задано выполнить действия для всех значений переменной от первого до последнего с заданным шагом изменении.

Параметр цикла – это переменная, через которую организуется (описывается) цикл. Параметр цикла определяет количество повторений (итераций), так как у него обязательно известно - первое значение, при котором тело цикла будет выполнено первый раз, последнее значение, при котором тело цикла будет выполнено последний раз и величина изменения параметра (шаг), на которую надо увеличить параметр после каждой итерации. Шаг – числовой показатель изменения параметра после каждой итерации. Циклы с параметром более просты в понимании и организации. В блок–схеме выделена специальная фигура для обозначения цикла с параметром. В блоке через «;» прописываются все значения параметра - «первое; последнее; шаг». Применительно к решению выше поставленной задачи, параметром цикла, по условию, является *x*.

Для всех значений x от -5 до 5 через 1 сантиметр рассчитывается $y=x^2$ и выводится точка графика (координаты x, y). Блок-схема решения задачи с применением цикла с параметром, представлена на рисунке 4.

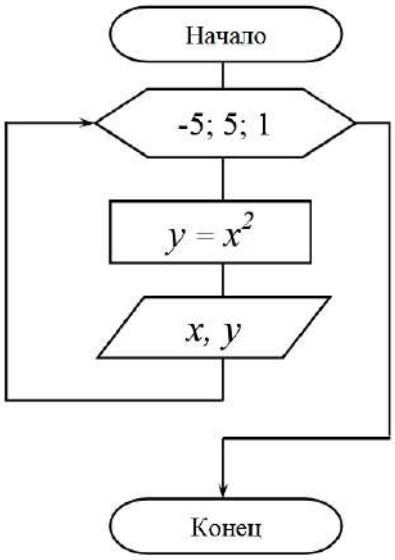


Рисунок 4

Варианты заданий

1 Составьте программу для вычисления степени числа *a* с целым показателем n.
2 Вычислите сумму всех двухзначных чисел.
3 Выведите на экран все чётные числа от 2 до n включительно и подсчитайте их количество.
4 Вычислите значение функции y=cos(x)*tg(pi*x) на интервале [0; pi), где шаг изменения равен 0.1.
5 Имеется товар в ящиках по 16, 17 и 21 кг. Как получить 185 кг этого товара, не вскрывая ящиков?
6 Найдите все делители некоторого целого числа.
7 Определите 5 первых совершенных чисел. Натуральное число называется совершенным, если оно равно сумме всех своих делителей за исключением самого себя. Например, 6 – совершенное число, т.к. 6=1+2+3, число 8 – несовершенное, т.к. 8<>1+2+4).
8 Найти сумму чётных натуральных чисел, меньших 100.
9 Найти сумму нечётных натуральных чисел, меньших 100.
10 Найти сумму целых положительных чисел, больших 20, меньших 100 и кратных 3.
11 Вычисление F = 10! выполнить каждым из трёх операторов цикла.
12 Начав тренировки, спортсмен в первый день пробежал 10 км, каждый последующий день он увеличивал дневную норму на 10% от нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?
13 Самолёт летит из пункта А в пункт Б со средней скоростью υ. Найти время полёта t, если есть встречный ветер, скорость которого $\upsilon_в$ изменяется от 0 до 15 м/с с шагом $\Delta\upsilon_в$= 0.5 м/с. Расстояние между пунктами А и Б S= 437 км, υ= 720 км/ч.(Для тех, кто плохо знает физику: t=S/ (υ-$\upsilon_в$))
14 Составить программу, печатающую таблицу значений градусов температуры по Цельсию и Фаренгейту. Значения градусов температуры по Цельсию изменяются от 0˚до 20˚с шагом 1˚. (℉ = ℃ * 1.8 + 32)

15 Пользователь сберегательной кассы внёс в неё вклад S_0= 3000 руб. До какой суммы он возрастёт через N= 5 лет, если процент годовых начислений P= 30 % ?

16 Даны действительные числа x, y. Вывести в порядке возрастания все целые числа, расположенные между x и y, а также количество этих чисел.

17 Даны действительные числа x, y. Вывести в порядке убывания все целые числа, расположенные между x и y, а также количество этих чисел

18 . Даны действительные числа x, y. Найти произведение всех целых чисел, расположенных между x и y, а также количество этих чисел.

19 Даны действительные числа x, y. Найти сумму квадратов всех целых чисел, расположенных между x и y, а также количество этих чисел.

20 Даны действительные числа x, y. Найти сумму кубов всех целых чисел, расположенных между x и y, а также количество этих чисел.

Лабораторная работа № 1
Составление программ линейной структуры

Цель работы: изучить классификацию типов и их внутреннее представление в языке С++, научиться работать со стандартными функциями. Изучить структуру программы на языке С++. Ознакомиться с программированием математических формул.

Краткие теоретические сведения.

Базовые типы – это типы данных, которые предопределены *стандартом языка*, указываются зарезервированными ключевыми словами, характеризуются одним значением и внутренним представлением.

Вещественный тип – это *базовый тип данных*, который применяется для хранения дробных чисел в формате с плавающей точкой.

Логический (булевый) тип – это *базовый тип данных*, который применяется для хранения значений двузначной логики.

Неявное приведение типа – это преобразование значения переменной к новому типу, которое происходит автоматически, по правилам, заложенным в языке программирования.

Перечисляемый тип – это *производный тип данных*, он определяется как набор идентификаторов, являющихся именованными целыми константами, которым приписаны уникальные обозначения

Преобразование типов – это приведение значения переменной одного типа в *значение* другого типа.

Производные типы – это типы данных, которые задаются пользователем.

Символьный тип – это *базовый тип данных*, который применяется для хранения символов или *управляющих последовательностей* в виде кода.

Тип данных – это множество допустимых значений, которые может принимать тот или иной *объект*, а также множество допустимых операций, которые применимы к нему.

Типы класса – это типы данных, экземплярами которых являются объекты.

Целочисленный тип – это *базовый тип данных*, который применяется для хранения целых чисел.

Явное приведение типа – это преобразование значения переменной к новому типу, при котором указывается *тип переменной*, к которому необходимо привести исходную переменную.

1. Для организации хранения данных и корректного выполнения операций над ними в языках программирования определены типы данных.
2. Типы характеризуются схожим внутренним представлением данных в памяти компьютера; объемом памяти, выделяемым под данные; множеством (диапазоном) принимаемых значений; допустимыми операциями и функциями.
3. В языке С++ типы классифицируются на базовые, производные и классы.
4. Для базовых типов определены их подмножества и расширения, что обеспечивает повышение точности расчетов или экономный расход памяти.
5. Над типами данных определена операция преобразования типов. Ее следует применять с осторожностью при переходе к типу, у которого меньше по модулю границы диапазонов.

Варианты заданий

При выполнении лабораторной работы для каждого задания требуется написать программу на языке С++, которая получает на входе числовые данные, выполняет их обработку в соответствии с требованиями задания и выводит результат на экран. Ввод данных осуществляется с клавиатуры с учетом требований к входным данным, содержащихся в постановке задачи. Ограничениями на *входные данные* является допустимый*диапазон* значений используемых *числовых типов* в языке С++.

1. Найдите сумму первых трех цифр дробной части *вещественного числа*. Например, для числа 23,16809 она равна 15.

2. Составьте программу вычисления стоимости поездки на автомобиле на дачу (туда и обратно). Исходными данными являются: расстояние до дачи (в километрах); количество бензина, которое потребляет автомобиль на 100 км пробега; цена одного литра бензина. Ниже представлен рекомендуемый вид диалога во время работы программы.
    1. Вычисление стоимости поездки на дачу.
    2. Расстояние до дачи (км) – 67
    3. Расход бензина (л на 100 км) – 8.5
    4. Цена литра бензина (руб.) – 23.7
    5. Поездка на дачу обойдется в 269 руб. 94 коп.
3. Составьте *линейную программу*, печатающую значение 1, если указанное *высказывание* является истинным, и 0 – в противном случае. Величина d является корнем только одного из уравнений $ax^2 + bx + c = 0$ и $mx + n = 0$ относительно x.
4. Составьте программу, которая преобразует введенное с клавиатуры дробное число в денежный формат. Например, число 12,348 должно быть преобразовано к виду 12 руб. 35 коп. Ниже представлен рекомендуемый вид диалога во время работы программы. Данные, вводимые пользователем:
    1. Преобразование числа в денежный формат.
    2. Введите дробное число – 23,6
    3. 23.6 руб. – это 23 руб. 60 коп.

Указания к выполнению работы

Каждое задание необходимо решить в соответствии с изученными методами обработки данных и преобразования типов данных в языке C++.

Следует реализовать каждое задание в соответствии с приведенными этапами:

1. изучить словесную постановку задачи, выделив при этом все виды данных;
2. сформулировать математическую постановку задачи;
3. выбрать метод решения задачи, если это необходимо;
4. разработать графическую схему алгоритма;
5. записать разработанный алгоритм на языке C++;
6. разработать контрольный тест к программе;
7. отладить программу;
8. представить отчет по работе.

Требования к отчету

Отчет по лабораторной работе должен соответствовать следующей структуре.

1. Титульный лист.
2. Словесная постановка задачи. В этом подразделе проводится полное описание задачи. Описывается суть задачи, анализ входящих в нее физических величин, область их допустимых значений, единицы их измерения, возможные ограничения, анализ условий при которых задача имеет решение (не имеет решения), анализ ожидаемых результатов.
3. Математическая модель. В этом подразделе вводятся математические описания физических величин и математическое описание их взаимодействий. Цель подраздела – представить решаемую задачу в математической формулировке.
4. Алгоритм решения задачи. В подразделе описывается разработка структуры алгоритма, обосновывается абстракция данных, задача разбивается на подзадачи. Схема алгоритма выполняется по ЕСПД (ГОСТ 19.003-80 и ГОСТ 19.002-80).

5. Листинг программы. Подраздел должен содержать текст программы на языке программирования C++, реализованный в среде MS Visual Studio 2010.
6. Контрольный тест. Подраздел содержит наборы исходных данных и полученные в ходе выполнения программы результаты.
7. Выводы по лабораторной работе.
8. Ответы на контрольные вопросы.

Контрольные вопросы

1. Почему в языке C++ определена строгая типизация данных, используемых в программе?
2. Как определяются границы диапазона базового типа в зависимости от выделяемой под этот тип памяти?
3. С какой целью в C++ определен тип void?
4. Какой объем памяти выделяется под переменную типа void? Какие значения может принимать переменная типа void?
5. Почему наблюдается *асимметрия* значений границ диапазонов целочисленных типов?
6. Чему будет равно значение операции *инкремента* для максимального числа в целочисленном типе? А каков результат*декремента* для минимального значения в таком же типе?
7. Почему запись целых чисел нельзя начинать с незначащих нулей?
8. Каким образом представлено число ноль в вещественных типах?
9. Почему в C++ *символьный тип* считается подмножеством целочисленного типа?
10. Каким образом можно инициализировать переменную перечисляемого типа?
11. При преобразовании целого со знаком к целому без знака всегда ли будет получено исходное числовое значение? Ответ обоснуйте.

Лабораторная работа № 2
Составление программ разветвляющейся структуры

Цель работы: научиться писать приложения на языке с++, используя операторы ветвления

Оператор if

```
int main()
{
setlocale(LC_ALL, "Russian");
double num;
cout << "Введите произвольное число: ";
cin >> num;
if (num < 10) { // Если введенное число меньше 10.
cout << "Это число меньше 10." << endl;
} else { // иначе
cout << "Это число больше либо равно 10." << endl;
}
return 0;
}
```

Если вы запустите эту программу, то при вводе числа, меньшего десяти, будет выводиться соответствующее сообщение. Если введенное число окажется большим, либо равным десяти — отобразится другое сообщение.

Оператор if служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным. Условная конструкция в С++ всегда записывается в круглых скобках после оператора if. Внутри фигурных скобок указывается тело условия. Если условие выполнится, то начнется выполнение всех команд, которые находятся между фигурными скобками.

Оператор переключатель switch

Оператор выбора имеет следующий синтаксис

```
switch выражение_целого_типа
{
case значение_1: оператор1;break;
...
case значение_n: оператор_n;break;
default: оператор_d;
}
```

Выполняется оператор следующим образом.

Вычисляется выражение, записанное как условие.

Если значение условия равно значение_i , то выполняется оператор_i и далее - выход из оператора switch.

Если значение условия не равно ни одному значение_i , то выполняется оператор_d .

Пример 1.

```
int x=2;
switch x
{
case 1: cout<<1<<endl;break;
case 2: cout<<2<<endl;break;
default: cout<<”x not equal 1 or 2”<<endl;
}
```

Ввод-вывод в С++

Обмен данными между программой и внешними устройствами осуществляется с помощью операций ввода-вывода. Типичным внешним устройством является консоль. Другим типичным устройством является жесткий диск, на котором расположены *файлы*.

Программа может создавать *файлы*, в которых хранится информация. Другая (или эта же) программа может читать информацию из *файла*.

В языке C++ нет особых операторов для ввода или вывода данных. Вместо этого имеется набор классов, стандартно поставляемых вместе с компилятором, которые и реализуют основные операции ввода-вывода.

Причиной является как слишком большое разнообразие операций ввода и вывода в разных операционных системах, особенно графических, так и возможность определения новых типов данных в языке C++. Библиотека классов для ввода-вывода решает две задачи. Во-первых, она обеспечивает эффективный ввод-вывод всех встроенных типов и простое, но тем не менее гибкое, определение операций ввода-вывода для новых типов, разрабатываемых программистом. Во-вторых, сама библиотека позволяет при необходимости развивать её и модифицировать.

Потоки. Механизм для ввода-вывода в C++ называется ***потоком***. Название произошло от того,что информация вводится и выводится в виде ***потока*** байтов – символ за символом.

Класс istream реализует *поток* ввода, класс ostream – *поток* вывода. Эти классы определены в файле заголовков iostream.h. Библиотека *потоков* ввода-вывода определяет три глобальных объекта: cout, cin и cerr. cout называется стандартным выводом, cin – стандартным вводом, cerr – стандартным *потоком* сообщений об ошибках. cout и cerr выводят на консоль и принадлежат к классу ostream, cin имеет тип istream и вводит с консоли.

Вывод осуществляется с помощью операции <<, ввод с помощью операции >>. Выражение

```
cout << "Пример вывода: " << 34;
```

напечатает на строку "Пример вывода", за которым будет выведено число 34. Выражение

```
int x;
cin >> x;
```

введет целое число в переменную x.

Манипуляторы и форматирование ввода-вывода. Часто бывает необходимо вывести строку или число в определенном формате. Для этого используются так называемые ***манипуляторы***.

Манипуляторы – это объекты особых типов, которые управляют тем, как ostream или istream обрабатывают последующие аргументы. Некоторые *манипуляторы* могут также выводить или вводить специальные символы.

endl	при выводе перейти на новую строку;
dec	выводить числа в десятичной системе (действует по умолчанию);
oct	выводить числа в восьмеричной системе;
hex	выводить числа в шестнадцатиричной системе счисления;
setw (int n)	установить ширину поля вывода в n символов (n – целое число);

Использовать *манипуляторы* просто – их надо вывести в выходной *поток*. Предположим, мы хотим вывести одно и то же число в разных системах счисления:

```
int x = 53;
cout << "Десятичный вид: " << dec
 << x << endl
 << "Восьмиричный вид: " << oct
 << x << endl
 << "Шестнадцатиричный вид: " << hex
 << x << endl
```

Аналогично используются *манипуляторы* с параметрами. Вывод числа с разным количеством цифр после запятой:

```
double x;
// вывести число в поле общей шириной
// 6 символов (3 цифры до запятой,
// десятичная точка и 2 цифры после запятой)
cout << setw(6) << setprecision(2) << fixed << x << endl;
```

Те же *манипуляторы* (за исключением endl) могут использоваться и при вводе. В этом случае они описывают представление вводимых чисел.

```
int x;
// ввести шестнадцатиричное число
cin >> hex >> x;
```

Поля вывода. Функция **width()** устанавливает минимальное число символов, использующееся в последующей операции вывода числа или строки. Так в результате следующих операций

```
cout.width(4);
cout << '(' << 12 << ')'; //получим число 12 в поле размером 4 символа.
```

Функция width() задает минимальное число символов. Если появится больше символов, они будут напечатаны все, поэтому

```
cout.width(4);
cout << '(' << "121212" << ")\n";
```

напечатает (121212)

Причина, по которой разрешено переполнение поля, а не усечение вывода,в том, чтобы избежать зависания при выводе.

3. Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант задания у преподавателя.
3. Создание проекта. В строке меню выберите Файл> Создать> Проект. В левой области диалогового окна Новый проект последовательно разверните узлы Установленные шаблоны и Visual C++, а затем щелкните Win32. В списке установленных шаблонов в центральной области выберите шаблон Консольное приложение Win32. В поле Имя введите имя проекта. Вы можете принять расположение по умолчанию в раскрывающемся списке Расположение, ввести другое расположение или нажать кнопку Обзор, чтобы перейти к каталогу, в котором требуется сохранить проект. Во время создания проекта среда Visual Studio помещает проект в решение. По умолчанию имя решения совпадает с именем проекта. Можно изменить имя в поле *Имя решения*, но для этого примера оставьте имя по умолчанию. На странице *Обзор* окна *Мастер приложений Win32* нажмите кнопку *Далее*. На странице *Параметры приложения* выберите в поле *Тип приложения* пункт *Консольное приложение*. Нажмите кнопку *Готово*, чтобы создать проект. Добавьте код:

```
#include "stdafx.h"
#include <iostream>
#include <cmath> // Работа с математическими функциями
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
  setlocale(LC_ALL, "Russian");
  .
  .
  .
  cin.get();
```

return 0;}

4. Выполнить задание по варианту.
5. Продемонстрировать выполнение работы преподавателю.
6. Оформить отчет.

Контрольные вопросы

1. Что такое решение (solution) в Visual Studio .NET? Зачем они нужны?
2. Как создать решение? Как добавить туда проект?
3. Что такое список задач? Как можно добавить задачу?
4. Зачем нужно окно свойств?
5. Зачем нужно окно вывода и окно ошибок? Чем они отличаются?
6. Зачем нужно окно решения?
7. Как добавить в проект новый или существующий файл.
8. Какой синтаксис имеет цикл while?.
9. Какой синтаксис имеет цикл do-while?.
10. Какой синтаксис имеет цикл for?
11. Для чего используется оператор break?
12. Каково назначение оператора continue?

Варианты заданий.

1. Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень — отрицательные.

2. Даны две точки A(x_1, y_1) и B(x_2,y_2). Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.

3. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник, и если да, то будет ли он прямоугольным.

4. Даны действительные числа x и y, не равные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее — их удвоенным произведением.

5. На плоскости XOY задана своими координатами точка A. Указать, где она расположена (на какой оси или в каком координатном угле).

6. Даны целые числа m, n. Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.

7. Подсчитать количество отрицательных среди чисел a, b, c.

8. Подсчитать количество положительных среди чисел a, b, c.

9. Услуги телефонной сети оплачиваются по следующему правилу: за разговоры до A минут в месяц — B руб., а разговоры сверх установленной нормы оплачиваются из расчета C руб. за минуту. Написать программу, вычисляющую плату за пользование телефоном для введенного времени разговоров за месяц.

13. Грузовой автомобиль выехал из одного города в другой со скоростью v_1 км/ч. Через t ч в этом же направлении выехал легковой автомобиль со скоростью v_2 км/ч. Составить программу, определяющую, догонит ли легковой автомобиль грузовой через t_1 ч после своего выезда.

14. Перераспределить значения переменных x и y так, чтобы в x оказалось большее из этих значений, а в y - меньшее.

15. Определить правильность даты, введенной с клавиатуры (число — от 1 до 31, месяц — от 1 до 12). Если введены некорректные данные, то сообщить об этом.

16. Составить программу, определяющую результат гадания на ромашке — «любит—не любит», взяв за исходное данное количество лепестков n.

17. Написать программу — модель анализа пожарного датчика в помещении, которая выводит сообщение «Пожароопасная ситуация», если температура в комнате превысила 60° C.

18. Рис расфасован в два пакета. Масса первого — n кг, второго — m кг. Составить программу, определяющую:

а) какой пакет тяжелее — первый или второй;

б) массу более тяжелого пакета.

19. Написать программу, которая анализирует данные о возрасте и относит человека к одной из четырех групп: дошкольник, ученик, работник, пенсионер. Возраст вводится с клавиатуры.

20. Составить программу, определяющую, пройдет ли график функции $y = ax^2 + bx +$ с через заданную точку с координатами (m, n).

21. К финалу конкурса лучшего по профессии «Специалист электронного офиса» были допущены трое: Иванов, Петров, Сидоров. Соревнования проходили в три тура. Иванов в первом туре набрал m_1 баллов, во втором — n_1, в третьем — p_1. Петров — m_2, n_2, p_2 соответственно; Сидоров — m_3, n_3 , p_3. Составить программу, определяющую, сколько баллов набрал победитель.

Лабораторная работа № 3
Составление программ циклической структуры

Оператор цикла while

Для организации повторения, управляемого счетчиком, требуется задать:

– Имя управляющей переменной (или счетчика циклов).

– Приращение (или уменьшение), на которое изменяется управляющая переменная в каждом цикле.

– Условие проверки, не достигнуто ли конечное значение управляющей переменной (т.е. надо ли продолжать циклы).

Рассмотрим простую программу, которая выводит на консоль числа от 1 до 10:

```
// повторение, управляемое счетчиком
#include <iostream>
main() {
        int counter = 1; // задание начального значения
        while(counter <= 10) { // условие повторения
                count << counter << endl;
                ++counter; // увеличение
        }
return 0;
}
```

Вывод.

1 2 3 4 5 6 7 8 9 10

Объявление int counter = 1; определяет имя управляющей переменной (counter), объявляет переменную как целую, резервируя для нее соответствующее место в памяти, и задает ее начальное значение 1. Объявления, в которых задается начальное значение, являются фактически выполняемыми операторами. Это объявление и задание начального значения переменной counter можно было бы сделать операторами int counter; counter = 1;

Здесь само объявление не является выполняемым оператором, а присваивание осуществляется отдельным оператором. Можно использовать оба эти метода для задания начальных значений переменных. Оператор ++counter увеличивает счетчик на 1 при каждом выполнении цикла. Условие продолжения цикла в структуре while проверяет, меньше или равно значение управляющей переменной числа 10 (последнего значения, при котором условие истинно). Заметьте, что тело этой структуры while выполняется и при значении управляющей переменной, равном 10. Выполнение цикла заканчивается, когда значение управляющей переменной превысит 10 (т.е. переменная counter станет равной 11).

Программа может быть сделана более компактной, если переменной counter задать начальное значение 0 и заменить структуру while следующей:

```
while(++counter <= 10)
        cout << counter << endl;
```

Этот код экономит один оператор, поскольку инкремент выполняется непосредственно в условии структуры while до того, как это условие проверяется. Этот код также устраняет скобки, заключающие в себе тело while, поскольку теперь while содержит всего один оператор.

Оператор цикла for

Структура повторения for содержит все элементы, необходимые для повторения, управляемого счетчиком. Чтобы проиллюстрировать мощь структуры for, перепишем программу.

```
// Повторение, управляемое счетчиком, со структурой for
int main() {
        // задание начального значения,условие повторения и
        // приращение – включено в заголовок структуры for
        for(int counter = 1; counter <= 10; counter++)
```

```
        cout << counter << endl;
    return 0;
}
```

Когда структура for начинает выполняться, управляющей переменной counter задается начальное значение 1. Затем проверяется условие продолжения цикла counter <= 10. Поскольку начальное значение counter равно 1, это условие удовлетворяется, так что оператор тела структуры печатает значение counter, равное 1. Затем управляющая переменная counter увеличивается на единицу в выражении counter++ и цикл опять начинается с проверки условия его продолжения. Поскольку значение counter теперь 2, предельная величина не превышена, так что программа снова выполняет тело цикла. Этот процесс продолжается, пока управляющая переменная counter не увеличится до 11 — это приведет к тому, что условие продолжения цикла нарушится и повторение прекратится. Выполнение программы продолжится с первого оператора, расположенного после структуры for (в данном случае с оператора return в конце программы).

Общая форма структуры for: for (выражение1; выражение2; выражение3) оператор, где выражение1 задает начальное значение переменной, управляющей циклом, выражение2 является условием продолжения цикла, а выражение3 изменяет управляющую переменную. В большинстве случаев структуру for можно представить эквивалентной ей структурой while следующим образом:

```
Выражение1;
while(Выражение2) {
        оператор
        Выражение3;
        }
```

«Приращение» структуры for может быть отрицательным (в этом случае в действительности происходит не приращение, а уменьшение переменной, управляющей циклом). Если условие продолжения цикла с самого начала не удовлетворяется, то операторы тела структуры for не выполняются и управление передается оператору, следующему за for. Управляющая переменная иногда печатается или используется в вычислениях в теле структуры for, но обычно это делается без изменений ее величины. Чаще управляющая переменная используется только для контроля числа повторений и никогда не упоминается в теле структуры. Хотя управляющая переменная может изменяться в теле цикла for, избегайте делать это, так как такая практика приводит к неявным, неочевидным ошибкам.

Оператор цикла do/while

Структура повторения do/while похожа на структуру while. В структуре while условие продолжения циклов проверяется в начале цикла, до того, как выполняется тело цикла. В структуре do/while проверка условия продолжения циклов производится после того, как тело цикла выполнено, следовательно, тело цикла будет выполнено по крайней мере один раз. Когда do/while завершается, выполнение программы продолжается с оператора, следующего за предложением while. Отметим, что в структуре do/while нет необходимости использовать фигурные скобки, если тело состоит только из одного оператора. Но фигурные скобки обычно все же ставят, чтобы избежать путаницы между структурами while и do/while. Например,

```
while (условие)
```

обычно рассматривается как заголовок структуры while. Структура do/while без фигурных скобок и с единственным оператором в теле имеет вид

```
do
оператор while(условие);
```

что может привести к путанице. Последняя строка — while (условие); может ошибочно интерпретироваться как заголовок структуры while, содержащий пустой

оператор. Таким образом, чтобы избежать путаницы, структура do/while даже с одним оператором часто записывается в виде

```
do {
оператор } while (условие);
```

Некоторые программисты всегда включают фигурные скобки в структуру do/while, даже если в них нет необходимости. Это помогает устранить двусмысленность, проистекающую из совпадения предложений структуры while и структуры do/while, содержащей один оператор.

Следующая программа использует структуру do/while, чтобы напечатать числа от 1 до 10. Обратите внимание, что к управляющей переменной counter в проверке окончания цикла применяется инкремент в префиксной форме. Отметьте также использование фигурных скобок, заключающих единственный оператор в теле do/while.

```
// Применение структуры повторения do/while
int main() {
int counter = 1;
do {
cout << counter << " ";
} while(++counter <= 10);
return 0;
}
```

Операторы break и continue

Операторы break и continue изменяют поток управления. Когда оператор break выполняется в структурах while, for, do/while или switch, происходит немедленный выход из структуры. Программа продолжает выполнение с первого оператора после структуры. Обычное назначение оператора break — досрочно прерывать цикл или пропустить оставшуюся часть структуры switch.

Оператор continue в структурах while, for или do/while вызывает пропуск оставшейся части тела структуры и начинается выполнение следующей итерации цикла. В структурах while и do/while немедленно после выполнения оператора continue производится проверка условия продолжения цикла. В структуре for выполняется выражение приращения, а затем осуществляется проверка условия продолжения.

Оператор переключатель switch

Оператор выбора имеет следующий синтаксис

switch *выражение_целого_типа*

{

case значение_1: оператор1;break;

...

case значение_n: оператор_n;break;

default: оператор_d;

}

Выполняется оператор следующим образом.

Вычисляется выражение, записанное как условие.

Если значение условия равно значение_i , то выполняется оператор_i и далее - выход из оператора switch.

Если значение условия не равно ни одному значение_i , то выполняется оператор_d .

Пример 1.

```
int x=2;
switch x
{
case 1: cout<<1<<endl;break;
case 2: cout<<2<<endl;break;
default: cout<<”x not equal 1 or 2”<<endl;
```

}

Варианты заданий

1. Имеется серия измерений элементов треугольника. Группы элементов пронумерованы. В серии в произвольном порядке могут встречаться такие группы элементов треугольника:

1) основание и высота;

2) две стороны и угол между ними (угол задан в радианах);

3) три стороны.

Разработать программу, которая запрашивает номер группы элементов, вводит соответствующие элементы и вычисляет площадь треугольника. Вычисления прекратить, если в качестве номера группы введен 0.

2. Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый день он увеличивал дневную норму на 10% нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?

3. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько амеб будет через 3, 6, 9, 12,..., 24 часа.

4. Около стены наклонно стоит палка длиной *x* м. Один ее конец находится на расстоянии *y* м от стены. Определить значение угла *a* между палкой и полом для значений *x*=к м и *y*, изменяющегося от 2 до 3 м с шагом h м.

5. У гусей и кроликов вместе 64 лапы. Сколько может быть кроликов и гусей (указать все сочетания)?

6. Сколько можно купить быков, коров и телят, платя за быка 10 т. руб., за корову — 5 т. руб., а за теленка — 0,5 т.руб., если на 100 т.руб. надо купить 100 голов скота?

8. Составить программу для проверки утверждения: «Результатами вычислений по формуле x^2 +x+ 17 при 0<=x< =15 являются простые числа». Все результаты вывести на экран.

9. Составить программу для проверки утверждения: «Результатами вычислений по формуле x^2 + x + 41 при 0<x<=40 являются простые числа». Все результаты вывести на экран.

10. Составить программу-генератор простых чисел, в основу положить формулу $2x^2$ + 29 при 0 <= x <= 28.

12. Составить программу-генератор чисел Пифагора a, b, c ($c^2=a^2+b^2$). В основу положить формулы: $a = m^2 - n^2$, b = 2mn, $c = m^2+ n^2$ (m, n - натуральные, 1 < m< к, 1<n< к, к — данное число). Результат вывести на экран в виде таблицы из пяти столбцов: m, n, a, b, c.

13. Покупатель должен заплатить в кассу 5 руб. У него имеются купюры по 1, 5, 10, 50, 100, 500, 1000 и 10000 руб. Сколько купюр разного достоинства отдаст покупатель, если он начинает платить с самых крупных купюр?

14. Ежемесячная стипендия студента составляет А руб., а расходы на проживание превышают стипендию и составляют В руб. в месяц. Рост цен ежемесячно увеличивает расходы на 3%. Составьте программу расчета суммы денег, которую необходимо единовременно попросить у родителей, чтобы можно было прожить учебный год (10 месяцев), используя только эти деньги и стипендию.

15. Составить программу, которая печатает таблицу умножения и сложения натуральных чисел в десятичной системе счисления.

16. Составить программу, которая печатает таблицу умножения и сложения натуральных чисел в шестнадцатеричной системе счисления.

17. Найти сумму всех n-значных чисел (1 <= n < =4).

18. Найти сумму всех n-значных чисел, кратных к (1 <= n <= 4).

19. Заменить буквы цифрами так, чтобы соотношение оказалось верным (одинаковым буквам соответствуют одинаковые цифры, разным — разные):

ХРУСТ*ГРОХОТ = РРРРРРРРРРР.

201. Составить программу, которая запрашивает пароль (например, четырехзначное число) до тех пор, пока он не будет правильно введен.

21. Составить программу, которая находит наибольшее значение отношения трехзначного числа к сумме его цифр.

22. Составить алгоритм решения ребуса РАДАР = $(Р + А + Д)^4$ (различные буквы обозначают различные цифры, старшая — не 0).

23. Составить алгоритм решения ребуса МУХА + МУХА + МУХА = СЛОН (различные буквы обозначают различные цифры, старшая — не 0).

24. Составить алгоритм решения ребуса ДРУГ — ГУРД = 2727 (различные буквы обозначают различные цифры, старшая — не 0).

25. Составить алгоритм решения ребуса КОТ + КОТ = ТОК (различные буквы обозначают различные цифры, старшая — не 0).

Лабораторная работа № 4
Потоковый ввод-вывод

Цель работы: изучить механизм ввода-вывода с помощью потоков

Краткие теоретические сведения

Большинство средств управления вводом-выводом сосредоточены в классе ios, который является базовым для ostream и istream. По сути здесь находится управление связью между istream или ostream и буфером,используемым для операций ввода-вывода. Именно класс ios контролирует, как символы попадают в буфер и как они выбираются оттуда. Так, в классеios есть член, содержащий информацию об используемой при чтении или записи целых чисел системы счисления (десятичная, восьмеричная или шестнадцатеричная), о точности вещественных чисел и т.п., а также функции для проверки и установки значений переменных, управляющих потоком.

```
class ios {
   //...
public:
   ostream* tie(ostream* s);   // связать input и output
   ostream* tie();             // возвратить "tie"
int width(int w);              // установить поле width
int width() const;
char fill(char);               // установить символ заполнения
char fill() const;             // вернуть символ заполнения
long flags(long f);
long flags() const;
long setf(long setbits, long field);
long setf(long);
long unsetf(long);
int precision(int);            // установить точность для float
int precision() const;
int rdstate(); const;          // состояния потоков, см. §10.3.2
int eof() const;
int fail() const;
int bad() const;
int good() const;
void clear(int i=0);
//...
        };
```

Поля вывода

Функция width() устанавливает минимальное число символов, использующееся в последующей операции вывода числа или строки. Так в результате следующих операций

```
cout.width(4);
cout << '(' << 12 << ')';
```

получим число 12 в поле размером 4 символа, т.е. (12)

Заполнение поля заданными символами или выравнивание можно установить с помощью функции fill(), например:

```
cout.width(4);
cout.fill('#');
cout << '(' << "ab" << ')';
```

напечатает

```
###(ab)
```

По умолчанию поле заполняется пробелами, а размер поля по умолчанию есть 0, что означает "столько символов, сколько нужно". Вернуть размеру поля стандартное значение можно с помощью вызова

```
cout.width(0); // ``столько символов, сколько надо"
```

Функция width() задает минимальное число символов. Если появится больше символов, они будут напечатаны все, поэтому

```
cout.width(4);
cout << '(' << "121212" << ")\n";
```

напечатает

```
(121212)
```

Причина, по которой разрешено переполнение поля, а не усечение вывода,в том, чтобы избежать зависания при выводе. Лучше получить правильную выдачу, выглядящую некрасиво, чем красивую выдачу, являющуюся неправильной.

Вызов width() влияет только на одну следующую за ним операцию вывода, поэтому

```
cout.width(4);
cout.fill('#');
cout << '(' << 12 << "),(" << '(' <<12 << ")\n";
```

напечатает

```
(##12),((12)
```

а не

```
(##12),(##12)
```

как можно было бы ожидать. Однако, заметьте, что если бы влияние распространялось на все операции вывода чисел и строк, получился бы еще более неожиданный результат:

```
(##12#),(##12#
)
```

С помощью стандартного манипулятора, показанного в 10.4.2.1, можно более элегантно задавать размера поля вывода.

Состояние формата

В классе ios содержится состояние формата, которое управляется функциями flags() и setf(). По сути эти функции нужны, чтобы установить или отменить следующие флаги:

```
class ios {
public:
    // управляющие форматом флаги:
  enum {
    skipws=01,        // пропуск обобщенных пробелов для input
    // поле выравнивания:
    left=02,          // добавление перед значением
    right=04,         // добавление после значения
    internal=010,     // добавление между знаком и значением
    // основание целого:
    dec=020,          // восьмеричное
    oct=040,          // десятичное
    hex=0100,         // шестнадцатеричное
    showbase=0200,    // показать основание целого
```

```
        showpoint=0400,     // выдать нули в конце
        uppercase=01000,    // 'E', 'X' , а не 'e', 'x'
        showpos=02000,      //  '+' для положительных чисел
        // запись числа типа float:
        scientific=04000,   // .dddddd Edd
        fixed=010000,       // dddd.dd
        // сброс в выходной поток:
        unitbuf=020000,     // после каждой операции
        stdio=040000        // после каждого символа
      };
      //...
};
```

Конкретные значения флагов зависят от реализации и даны здесь только для того, чтобы избежать синтаксически неверных конструкций. Определение интерфейса как набора флагов и операций для их установки или отмены - это оцененный временем, хотя и несколько устаревший прием. Основное его достоинство в том, что пользователь может собрать воедино набор флагов, например, так:

```
const int my_io_options =
  ios::left|ios::oct|ios::showpoint|ios::fixed;
```

Такое множество флагов можно задавать как параметр одной операции

```
cout.flags(my_io_options);
```

а также просто передавать между функциями одной программы:

```
void your_function(int ios_options);
void my_function()
{
  // ...
 your_function(my_io_options);
 // ...
}
```

Множество флагов можно установить с помощью функции flags(), например:

```
void your_function(int ios_options)
{
 int old_options = cout.flags(ios_options);
 // ...
 cout.flags(old_options);  // reset options
 }
```

Функция flags() возвращает старое значение множества флагов. Это позволяет переустановить значения всех флагов, как показано выше, а также задать значение отдельному флагу. Например вызов

```
myostream.flags(myostream.flags()|ios::showpos);
```

заставляет класс myostream выдавать положительные числа со знаком+ и, в то же время, не меняет значения других флагов. Получается старое значение множества флагов, к которому добавляется с помощью операции | флаг showpos. Функция setf() делает то же самое,поэтому эквивалентная запись имеет вид

```
myostream.setf(ios::showpos);
```

После установки флаг сохраняет значение до явной отмены.

Вывод целых

Прием задания нового значения множества флагов с помощью операции | и функций flags() и setf() работает только тогда, когда один бит определяет значение флага. Не такая ситуация при задании системы счисления целых или вида выдачи вещественных. Здесь значение, определяющее вид выдачи,нельзя задать одним битом или комбинацией отдельных битов.

Решение, принятое в <iostream.h>, сводится к использованию версии функции setf(), работающей со вторым "псевдопараметром", который показывает какой именно флаг мы хотим добавить к новому значению.

Поэтому обращения

```
cout.setf(ios::oct,ios::basefield); // восьмеричное
cout.setf(ios::dec,ios::basefield); // десятичное
cout.setf(ios::hex,ios::basefield); // шестнадцатеричное
```

установят систему счисления, не затрагивая других компонентов состояния потока. Если система счисления установлена, она используется до явной переустановки, поэтому

```
cout << 1234 << ' ';  // десятичное по умолчанию
cout << 1234 << ' ';
cout.setf(ios::oct,ios::basefield);  // восьмеричное
cout << 1234 << ' ';
cout << 1234 << ' ';
cout.setf(ios::hex,ios::basefield);  // шестнадцатеричное
cout << 1234 << ' ';
cout << 1234 << ' ';
```

напечатает

1234 1234 2322 2322 4d2 4d2

Если появится необходимость указывать систему счисления для каждого выдаваемого числа, следует установить флаг showbase. Поэтому, добавив перед приведенными выше обращениями

```
cout.setf(ios::showbase);
```

мы получим

1234 1234 02322 02322 0x4d2 0x4d2

Выравнивание полей

С помощью обращений к setf() можно управлять расположением символов в пределах поля:

```
cout.setf(ios::left,ios::adjustfield);     // влево
cout.setf(ios::right,ios::adjustfield);    // вправо
cout.setf(ios::internal,ios::adjustfield); // внутреннее
```

Будет установлено выравнивание в поле вывода, определяемом функциейios::width(), причем не затрагивая других компонентов состояния потока.

Выравнивание можно задать следующим образом:

```
cout.width(4);
cout << '(' << -12 << ")\n";
cout.width(4);
cout.setf(ios::left,ios::adjustfield);
cout << '(' << -12 << ")\n";
```

```
cout.width(4);
cout.setf(ios::internal,ios::adjustfield);
cout << '(' << -12 << "\n";
```

что выдаст

```
(-12)
( -12 )
(-12)
```

Если установлен флаг выравнивания internal (внутренний), то символы добавляются между знаком и величиной. Как видно, стандартным является выравнивание вправо.

Вывод плавающих чисел

Вывод вещественных величин также управляется с помощью функций, работающих с состоянием потока. В частности, обращения:

```
cout.setf(ios::scientific,ios::floatfield);
cout.setf(ios::fixed,ios::floatfield);
cout.setf(0,ios::floatfield);   // вернуться к стандартному
```

установят вид печати вещественных чисел без изменения других компонентов состояния потока.

Например:

```
cout << 1234.56789 << '\n';
cout.setf(ios::scientific,ios::floatfield);
cout << 1234.56789 << '\n';
cout.setf(ios::fixed,ios::floatfield);
cout << 1234.56789 << '\n';
```

напечатает

```
1234.57
1.234568e+03
1234.567890
```

После точки печатается n цифр, как задается в обращении

```
cout.precision(n)
```

По умолчанию n равно 6. Вызов функции precision влияет на все операции ввода-вывода с вещественными до следующего обращения к precision,поэтому

```
cout.precision(8);
cout << 1234.56789 << '\n';
cout << 1234.56789 << '\n';
cout.precision(4);
cout << 1234.56789 << '\n';
cout << 1234.56789 << '\n';
```

выдаст

```
1234.5679
1234.5679
1235
1235
```

Заметьте, что происходит округление, а не отбрасывание дробной части.

Стандартные манипуляторы, введенные в §10.4.2.1, предлагают более элегантный способ задания формата вывода вещественных.

Манипуляторы

К ним относятся разнообразные операции, которые приходится применять сразу перед или сразу после операции ввода-вывода. Например:

```
cout << x;
cout.flush();
cout << y;
cin.eatwhite();
cin >> x;
```

Если писать отдельные операторы как выше, то логическая связь между операторами неочевидна, а если утеряна логическая связь, программу труднее понять.

Идея манипуляторов позволяет такие операции как flush() илиeatwhite() прямо вставлять в список операций ввода-вывода. Рассмотрим операцию flush(). Можно определить класс с операцией operator<<(), в котором вызывается flush():

```
class Flushtype { };
ostream& operator<<(ostream& os, Flushtype)
{
  return flush(os);
}
```

определить объект такого типа

```
Flushtype FLUSH;
```

и добиться выдачи буфера, включив FLUSH в список объектов, подлежащих выводу:

```
cout << x << FLUSH << y << FLUSH ;
```

Теперь установлена явная связь между операциями вывода и сбрасывания буфера. Однако, довольно быстро надоест определять класс и объект для каждой операции, которую мы хотим применить к поточной операции вывода. Рассмотрим такую функцию:

```
typedef ostream& (*Omanip) (ostream&);
ostream& operator<<(ostream& os, Omanip f)
{
 return f(os);
}
```

Здесь операция вывода использует параметры типа "указатель на функцию, имеющую аргумент ostream& и возвращающую ostream&". Отметив, что flush()есть функция типа "функция с аргументом ostream& и возвращающаяostream&", мы можем писать

```
cout << x << flush << y << flush;
```

получив вызов функции flush(). На самом деле в файле <iostream.h>функция flush() описана как

```
ostream& flush(ostream&);
```

а в классе есть операция operator<<, которая использует указатель на функцию, как указано выше:

```
class ostream : public virtual ios {
  // ...
public:
 ostream& operator<<(ostream& ostream& (*)(ostream&));
 // ...
```

```
};
```

В приведенной ниже строке буфер выталкивается в поток cout дважды в подходящее время:

```
cout << x << flush << y << flush;
```

Похожие определения существуют и для класса istream:

```
istream& ws(istream& is ) { return is.eatwhite(); }
class istream : public virtual ios {
 // ...
public:
  istream& operator>>(istream&, istream& (*) (istream&));
  // ...
};
```

поэтому в строке

```
cin >> ws >> x;
```

действительно обобщенные пробелы будут убраны до попытки чтения в x.Однако, поскольку по умолчанию для операции >>пробелы "съедаются" и так, данное применение ws() избыточно.

Находят применение и манипуляторы с параметрами. Например,может появиться желание с помощью

```
cout << setprecision(4) << angle;
```

напечатать значение вещественной переменной angle с точностью до четырех знаков после точки.

Для этого нужно уметь вызывать функцию, которая установит значение переменной, управляющей в потоке точностью вещественных. Это достигается, если определить setprecision(4) как объект, который можно "выводить" с помощью operator<<():

```
class Omanip_int {
    int i;
    ostream& (*f) (ostream&,int);
public:
    Omanip_int(ostream& (*ff) (ostream&,int), int ii)
        : f(ff), i(ii) { }
    friend ostream& operator<<(ostream& os, Omanip& m)
        { return m.f(os,m.i); }
 };
```

Конструктор Omanip_int хранит свои аргументы в i и f, а с помощью operator<< вызывается f() с параметром i. Часто объекты таких классов называют объект-функция. Чтобы результат строки

```
cout << setprecision(4) << angle
```

был таким, как мы хотели, необходимо чтобы обращение setprecision(4)создавало безымянный объект класса Omanip_int, содержащий значение 4и указатель на функцию, которая устанавливает в потоке ostream значение переменной, задающей точность вещественных:

```
ostream& _set_precision(ostream&,int);
Omanip_int setprecision(int i)
{
```

```
    return Omanip_int(&_set_precision,i);
}
```

Учитывая сделанные определения, operator<<() приведет к вызовуprecision(i).

Утомительно определять классы наподобие Omanip_int для всех типов аргументов, поэтому определим шаблон типа:

```
template<class T> class OMANIP {
   T i;
   ostream& (*f) (ostream&,T);
public:
   OMANIP(ostream (*ff) (ostream&,T), T ii)
      : f(ff), i(ii) { }
friend ostream& operator<<(ostream& os, OMANIP& m)
   { return m.f(os,m.i) }
};
```

С помощью OMANIP пример с установкой точности можно сократить так:

```
ostream& precision(ostream& os,int)
{
   os.precision(i);
   return os;
}
OMANIP<int> setprecision(int i)
{
   return OMANIP<int>(&precision,i);
}
```

В файле <iomanip.h> можно найти шаблон типа OMANIP, его двойник дляistream - шаблон типа SMANIP, а SMANIP - двойник для ioss.Некоторые из стандартных манипуляторов, предлагаемых поточной библиотекой, описаны ниже. Отметим,что программист может определить новые необходимые ему манипуляторы, не затрагивая определений istream,ostream, OMANIP или SMANIP.

Идею манипуляторов предложил А. Кениг. Его вдохновили процедуры разметки (layout) системы ввода-вывода Алгола68. Такая техника имеет много интересных приложений помимо ввода-вывода. Суть ее в том, что создается объект, который можно передавать куда угодно и который используется как функция. Передача объекта является более гибким решением, поскольку детали выполнения частично определяются создателем объекта, а частично тем, кто к нему обращается.

Стандартные манипуляторы ввода-вывода

Это следующие манипуляторы:

```
// Simple manipulators:
ios& oct(ios&);  // в восьмеричной записи
ios& dec(ios&);  // в десятичной записи
ios& hex(ios&);  // в шестнадцатеричной записи
ostream& endl(ostream&);   // добавить '\n' и вывести
ostream& ends(ostream&);   // добавить '\0' и вывести
ostream& flush(ostream&);  // выдать поток
istream& ws(istream&);     // удалить обобщенные пробелы
// Манипуляторы имеют параметры:
SMANIP<int>  setbase(int b);
SMANIP<int>  setfill(int f);
SMANIP<int>  setprecision(int p);
```

```
SMANIP<int>  setw(int w);
SMANIP<long> resetiosflags(long b);
SMANIP<long> setiosflags(long b);
```

Например,

```
cout << 1234 << ' '
    << hex  << 1234 << ' '
    << oct  << 1234 << endl;
```

напечатает

```
1234 4d2 2322
```

и

```
cout << setw(4) << setfill('#') << '(' << 12 << ')\n";
cout << '(' << 12 << ')\n";
```

напечатает

```
(##12)
(12)
```

Не забудьте включить файл <iomanip.h>, если используете манипуляторы с параметрами.

Члены ostream

В классе ostream есть лишь несколько функций для управления выводом,большая часть таких функций находится в классе ios.

```
class ostream : public virtual ios {
   //...
public:
   ostream& flush();
   ostream&  seekp(streampos);
   ostream&  seekp(streamoff, seek_dir);
   streampos tellp();
   //...
};
```

Как мы уже говорили, функция flush() опустошает буфер в выходной поток.Остальные функции используются для позиционирования в ostream при записи. Окончание на букву p указывает, что именно позиция используется при выдаче символов в заданный поток. Конечно эти функции имеют смысл,только если поток присоединен к чему-либо, что допускает позиционирование, например файл. Тип streampos представляет позицию символа в файле, а тип streamoff представляет смещение относительно позиции,заданной seek_dir. Все они определены в классе ios:

```
class ios {
   //...
   enum seek_dir {
      beg=0,  // от начала файла
      cur=1,  // от текущей позиции в файле
      end=2   // от конца файла
   };
   //...
};
```

Позиции в потоке отсчитываются от 0, как если бы файл был массивом изn символов:

```
char file[n-1];
```

и если fout присоединено к file, то

```
fout.seek(10);
fout<<'#';
```

поместит # в file[10].

Файлы и потоки

Ниже приведена программа копирования одного файла в другой. Имена файлов берутся из командной строки программы:

```
#include <fstream.h>
#include <libc.h>
void error(char* s, char* s2 ="")
{
  cerr << s << ' ' << s2 << '\n';
  exit(1);
}
int main(int argc, char* argv[])
{
   if (argc != 3) error("wrong number of arguments");
   ifstream from(argv[1]);
   if (!from) error("cannot open input file",argv[1]);
   ostream to(argv[2]);
   if (!to) error("cannot open output file",argv[2]);
   char ch;
   while (from.get(ch)) to.put(ch);
   if (!from.eof() || to.bad())
     error("something strange happened");
   return 0;
 }
```

Для открытия выходного файла создается объект класса ofstream -выходной поток файла, использующий в качестве аргумента имя файла.Аналогично, для открытия входного файла создается объект классaifstream - входной файловый поток, также использующий в качестве аргумента имя файла. В обоих случаях следует проверить состояние созданного объекта, чтобы убедиться в успешном открытии файла, а если это не так, операции завершатся не успешно, но корректно.

По умолчанию ifstream всегда открывается на чтение, а ofstreamоткрывается на запись. В ostream и в istream можно использовать необязательный второй аргумент, указывающий иные режимы открытия:

```
class ios {
public:
   //...
   enum open_mode {
     in=1,          // открыть на чтение
     out=2,          // открыть как выходной
     ate=4,          // открыть и переместиться в конец файла
     app=010,        // добавить
     trunc=020,      // сократить файл до нулевой длины
     nocreate=040,   // неудача, если файл не существует
```

```
        noreplace=0100   // неудача, если файл существует
    };
    //...
};
```

Настоящие значения для open_mode и их смысл вероятно будут зависеть от реализации. Будьте добры, за деталями обратитесь к руководству по вашей библиотеке или экспериментируйте. Приведенные комментарии могут прояснить их назначение. Например, можно открыть файл с условием,что операция открытия не выполнится, если файл уже не существует:

```
void f()
{
    ofstream mystream(name,ios::out|ios::nocreate);
    if (ofstream.bad()) {
        //...
    }
    //...
}
```

Также можно открыть файл сразу на чтение и запись:

```
fstream dictionary("concordance", ios::in|ios::out);
```

Все операции, допустимые для ostream и ostream, можно применять кfstream. На самом деле, класс fstream является производным от iostream,который является, в свою очередь, производным от istream и ostream.Причина, по которой информация по буферизации и форматированию дляostream и istream находится в виртуальном базовом классе ios, в том,чтобы заставить действовать всю эту последовательность производных классов. По этой же причине операции позиционирования в istream iostream имеют разные имена - seekp() и seekg(). В iostream есть отдельные позиции для чтения и записи.

Закрытие потоков

Файл может быть закрыт явно, если вызвать close() для его потока:

```
mystream.close();
```

Но это неявно делает деструктор потока, так что явный вызов close()может понадобиться, если только файл нужно закрыть до достижения конца области определенности потока.

Для каждого программного файла определен свой объект с именем io_init.Конструктор для объектов io_init использует Io_init::count как первый признак того, что действительная инициализация глобальных объектов потоковой библиотеки ввода-вывода сделана в точности один раз:

```
Io_init::Io_init()
{
  if (count++ == 0) {
      // инициализировать cout
      // инициализировать cerr
      // инициализировать cin
      // и т.д.
  }
}
```

Обратно, деструктор для объектов io_init использует Io_count, как последнее указание на то, что все потоки закрыты:

```
Io_init::^Io_init()
{
  if (--count == 0) {
     // очистить cout (сброс, и т.д.)
     // очистить cerr (сброс, и т.д.)
     // очистить cin
     // и т.д.
  }
 }
```

Это общий прием работы с библиотеками, требующими инициализации и удаления глобальных объектов. Впервые в C++ его применил Д. Шварц. В системах, где при выполнении все программы размещаются в основной памяти, для этого приема нет помех. Если это не так, то накладные расходы, связанные с вызовом в память каждого программного файла для выполнения функций инициализации, будут заметны. Как всегда, лучше, по возможности, избегать глобальных объектов. Для классов, в которых каждая операция значительна по объему выполняемой работы,чтобы гарантировать инициализацию, было бы разумно проверять такие первые признаки (наподобие Io_init::count) при каждой операции. Однако, для потоков такой подход был бы излишне расточительным.

Строковые потоки

Как было показано, поток может быть привязан к файлу, т.е. массиву символов, хранящемуся не в основной памяти, а, например, на диске. Точно так же поток можно привязать к массиву символов в основной памяти.Например, можно воспользоваться выходным строковым потоком ostrstreamдля форматирования сообщений, не подлежащих немедленной печати:

```
char* p = new char[message_size];
ostrstream ost(p,message_size);
do_something(arguments,ost);
display(p);
```

С помощью стандартных операций вывода функция do_something может писать в поток ost, передавать ost подчиняющимся ей функциям и т.п. Контроль переполнения не нужен, поскольку ost знает свой размер и при заполнении перейдет в состояние, определяемое fail(). Затем функция display может послать сообщение в "настоящий" выходной поток. Такой прием наиболее подходит в тех случаях, когда окончательная операция вывода предназначена для записи на более сложное устройство, чем традиционное,ориентированное на последовательность строк, выводное устройство.Например, текст из ost может быть помещен в фиксированную область на экране.

Аналогично, istrstream является вводным строковым потоком,читающим из последовательности символов, заканчивающейся нулем:

```
void word_per_line(char v[], int sz)
/*
  печатать "v" размером "sz" по одному слову в строке
*/
{
  istrstream ist(v,sz);  // создать istream для v
  char b2[MAX];          // длиннее самого длинного слова
  while (ist>>b2) cout <<b2 << "\n";
 }
```

Завершающий нуль считается концом файла.

Строковые потоки описаны в файле <strstream.h>.

Буферизация

Все операции ввода-вывода были определены без всякой связи с типом файла, но нельзя одинаково работать со всеми устройствами без учета алгоритма буферизации. Очевидно, что потоку ostream, привязанному к строке символов, нужен не такой буфер, как ostream, привязанному к файлу. Такие вопросы решаются созданием во время инициализации разных буферов для потоков разных типов. Но существует только один набор операций над этими типами буферов, поэтому в ostream нет функций, код которых учитывает различие буферов. Однако, функции, следящие за переполнением и обращением к пустому буферу, являются виртуальными.Это хороший пример применения виртуальных функций для единообразной работы с эквивалентными логически, но различно реализованными структурами, и они вполне справляются с требуемыми алгоритмами буферизации.Описание буфера потока в файле <iostream.h> может выглядеть следующим образом:

```
class streambuf {     // управление буфером потока
protected:
   char* base;       // начало буфера
   char* pptr;       // следующий свободный байт
   char* gptr;       // следующий заполненный байт
   char* eptr;       // один из указателей на конец буфера
   char  alloc;      // буфер, размещенный с помощью "new"
   //...
   // Опустошить буфер:
   // Вернуть EOF при ошибке, 0 - удача
virtual int overflow(int c = EOF);
   // Заполнить буфер:
   // Вернуть EOF в случае ошибки или конца входного потока,
   // иначе вернуть очередной символ
 virtual int underflow();
 //...
 public:
    streambuf();
    streambuf(char* p, int l);
    virtual ~streambuf();
    int snextc()       // получить очередной символ
    {
       return (++gptr==pptr) ? underflow() : *gptr&0377;
    }
    int allocate();    // отвести память под буфер
    //...
};
```

Подробности реализации класса streambuf приведены здесь только для полноты представления. Не предполагается, что есть общедоступные реализации, использующие именно эти имена. Обратите внимание на определенные здесь указатели, управляющие буфером; с их помощью простые посимвольные операции с потоком можно определить максимально эффективно (и причем однократно) как функции-подстановки. Только функции overflow() и underflow() требует своей реализации для каждого алгоритма буферизации, например:

```
class filebuf : public streambuf {
protected:
```

```
    int  fd;            // дескриптор файла
    char opened;        // признак открытия файла
public:
    filebuf() { opened = 0; }
    filebuf(int nfd, char* p, int l)
        : streambuf(p,l) { /* ... */ }
    ~filebuf() { close(); }
    int overflow(int c=EOF);
    int underflow();
    filebuf* open(char *name, ios::open_mode om);
    int close() { /* ... */ }
    //...
};
int filebuf::underflow()    // заполнить буфер из "fd"
{
    if (!opened || allocate()==EOF) return EOF;
    int count = read(fd, base, eptr-base);
    if (count < 1) return EOF;
    gptr = base;
    pptr = base + count;
    return *gptr & 0377; // &0377 предотвращает размножение знака
 }
```

Лабораторная работа № 5
Обработка одномерных массивов

Цель работы: освоение средств языка С++ для изучения способов описания, ввода-вывода и обработки одномерных массивов.

Краткие теоретические сведения

Одномерный массив — массив, с одним параметром, характеризующим количество элементов одномерного массива. Фактически одномерный массив — это массив, у которого может быть только одна строка, и n-е количество столбцов. Столбцы в одномерном массиве — это элементы массива. На рисунке 1 показана структура целочисленного одномерного массива *a*. Размер этого массива — 16 ячеек.

5	-12	-12	9	10	0	-9	-12	-1	23	65	64	11	43	39	-15
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]	a[13]	a[14]	a[15]

Рисунок 1— Одномерный массив

Максимальный индекс одномерного массива *a* равен 15, но размер массива 16 ячеек, потому что нумерация ячеек массива всегда начинается с 0. Индекс ячейки - это целое неотрицательное число, по которому можно обращаться к каждой ячейке массива и выполнять какие-либо действия над ней (ячейкой). Синтаксис объявления одномерного массива в С++:

тип данных имя одномерного массива [размерность одномерного массива];
//пример объявления одномерного массива, изображенного на рисунке 1:
int a[16];

где, int — целочисленный тип данных; а — имя одномерного массива; 16 — размер одномерного массива, 16 ячеек.

Всегда сразу после имени массива идут квадратные скобочки, в которых задаётся размер одномерного массива, этим массив и отличается от всех остальных переменных.

Рассмотрим на примерах способы ввода-вывода одномерного массива:

1) размер массива можно не указывать только при его инициализации, при обычном объявлении массива обязательно нужно указывать размер массива

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
   cout << "obrabotka massiva" << endl;
// объявление и инициализация одномерного массива
   int array1[16] = { 5, -12, -12, 9, 10, 0, -9, -12, -1, 23, 65, 64, 11, 43, 39, -
   15 };
   cout << "indeks" << "\t\t" << "element massiva" << endl; // печать
   заголовков
    for (int counter = 0; counter < 16; counter++) //начало цикла
   {
   //вывод на экран индекса ячейки массива, а затем содержимого этой
   ячейки cout << "array1[" << counter << "]" << "\t\t" << array1[counter]
   << endl;
   }
```

```
    system("pause");
    return 0;
}
```

Результат работы представлен на рисунке 2.

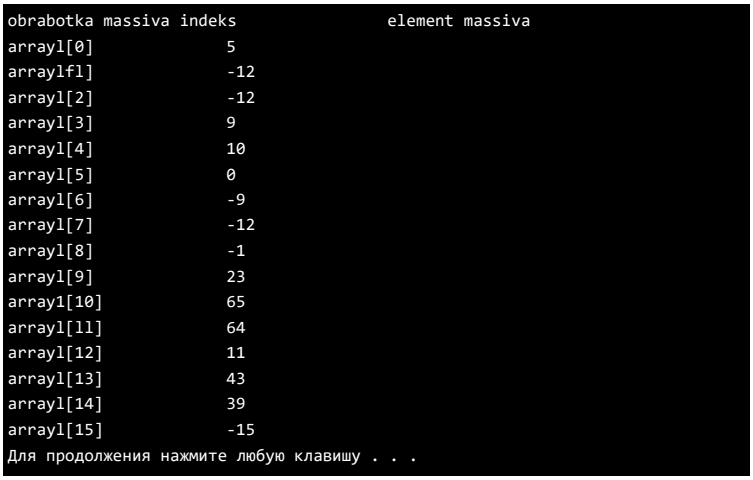


Рисунок 2

2) программа должна последовательно считывать десять введённых чисел с клавиатуры. Все введённые числа просуммировать, результат вывести на экран. #include "stdafx.h"

```
#include <iostream>
using namespace std;
int main(int argc, char* argv[])
{
    int array1[10]; // объявляем целочисленный массив
    cout << "Enter elementi massiva: " << endl;
    int sum = 0;
    for ( int counter = 0; counter < 10; counter++ ) // цикл для считывания чисел
        cin >> array1[counter]; // считываем вводимые с клавиатуры числа
    cout << "array1 = {";
for ( int counter = 0; counter < 10; counter++ ) // цикл для вывода элементов массива cout <<
array1[counter] << " "; // выводим элементы массива на стандартное
устройство вывода
    for ( int counter = 0; counter < 10; counter++ ) // цикл для суммирования чисел
массива
        sum += array1[counter]; // суммируем элементы массива
    cout << "}\nsum = " << sum << endl;
    system("pause");
    return 0;
}
```

Результат представлен на рисунке 3.

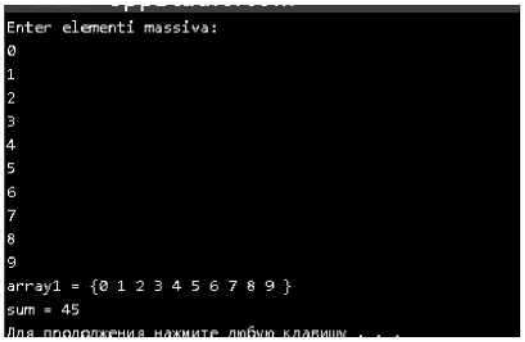


Рисунок 3.

Пример выполнения задания.

Задание: произвести следующую обработку 15 целых чисел: подсчитать сумму чисел, входящих в диапазон [-5..5] и количество нечетных чисел.

Схема программы к данному заданию с использованием одного цикла представлена на рисунке 4.

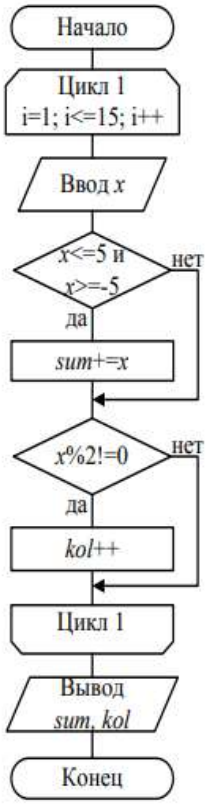


Рисунок 4

Текст программы без использования массива:

```
#include <iostream>

void main()
{
int x,sum=0,i,kol=0;
printf("Enter numbers\n");
for (i=1;i<=15;i++)
{
scanf("%d",&x);
if ((x>=-5)&&(x<=5)) sum+=x;
if (x%2!=0) kol++;
```

```
}
printf("Summa v diapazone [-5,5]=%d\n", sum);
printf("Kolichestvo nechetnih=%d", kol);
}
```

Схема программы с циклами для ввода и обработки массива (рисунок 5).

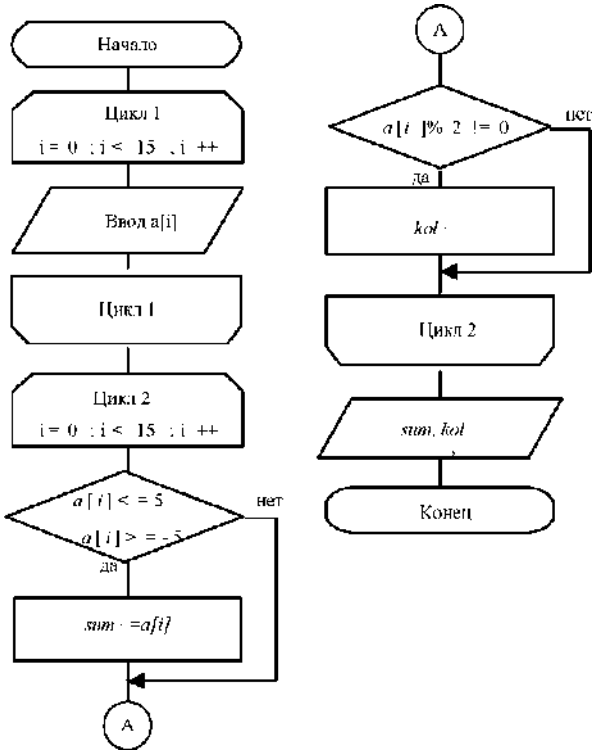


Рисунок 5

Пример программы с использованием одномерного массива

```
#include<iostream>
void main()
{
int a[15],s("E um=0,i,kol=0;
printf(“Enter numbers\n");
for (i=0;i<15;i++)
scanf("%d",&a[i]);
for (i=0;i<15;i++)
{
if ((a[i]>=-5)&&(a[i]<=5)) sum+=a[i];
if (a[i]%2!=0) kol++;
}
printf("Summa v diapazone [-5,5]=%d\n", sum);
printf("Kolichestvo nechetnih=%d", kol);
}
```

Контрольные вопросы

1. Массивы в языке С++: понятие массива в языке С++, описание массива в программе, представление элементов массива в памяти, обращение к элементам массива.

Варианты заданий

1. Произвести следующую обработку 15 целых чисел: найти количество отрицательных чисел, количество нулевых и подсчитать сумму положительных чисел.
2. Произвести следующую обработку 15 целых чисел: найти количество четных

чисел, а нечетные числа, входящие в диапазон [1..11] возвести в квадрат.

3. Произвести следующую обработку 15 вещественных чисел: найти количество отрицательных чисел, а числа, входящие в диапазон [0..10] возвести в квадрат.

4. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, больших или равных 1,5, и подсчитать сумму отрицательных чисел.

5. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, равных нулю, и найти сумму чисел, входящих в диапазон [15..15].

6. Произвести следующую обработку 15 целых чисел: найти количество отрицательных чисел и подсчитать разность положительных чисел.

7. Произвести следующую обработку 15 вещественных чисел: найти среднее арифметическое положительных чисел и подсчитать количество чисел, входящих в диапазон [-15..5].

8. Произвести следующую обработку 10 целых чисел: найти количество отрицательных чисел и подсчитать сумму положительных чисел, делящихся без остатка на 3.

9. Произвести следующую обработку 10 целых чисел: найти количество отрицательных чисел, а числа, входящие в диапазон [0..10], умножить на 10.

10. Произвести следующую обработку 10 целых чисел: найти количество отрицательных чисел, а числа, входящие в диапазон [0..10], умножить на 3.

11. Произвести следующую обработку 15 вещественных чисел: найти среднее арифметическое отрицательных чисел и подсчитать количество чисел, входящих в диапазон [0..5].

12. Произвести следующую обработку 15 вещественных чисел: найти среднее арифметическое нечетных чисел и подсчитать сумму чисел, входящих в диапазон [15..5].

13. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, равных нулю, и найти синус чисел, входящих в диапазон [-15..15].

14. Произвести следующую обработку 10 целых чисел: подсчитать сумму положительных чисел и определить номера отрицательных чисел.

15. Произвести следующую обработку 15 вещественных чисел: найти количество отрицательных чисел и номера нулевых чисел.

16. Произвести следующую обработку 12 целых чисел: подсчитать количество чисел, делящихся без остатка на 5, и сумму чисел, входящих в диапазон [-5..5].

17. Произвести следующую обработку 10 вещественных чисел: подсчитать количество чисел, отличающихся от числа 3 не более чем на 0.5, и сумму отрицательных чисел.

18. Произвести следующую обработку 15 целых чисел: подсчитать количество нулевых чисел и вычислить квадраты чисел, входящих в диапазон [-5..5].

19. Произвести следующую обработку 12 целых чисел: подсчитать количество нечетных чисел и сумму отрицательных чисел.

20. Произвести следующую обработку 15 вещественных чисел: подсчитать количество чисел, отличающихся от заданного не более чем на 0.5, и сумму положительных чисел.

21. . Произвести следующую обработку 10 вещественных чисел: найти количество отрицательных чисел, находящихся в диапазоне от -5 до 5 и подсчитать сумму положительных чисел.

22. Произвести следующую обработку 15 целых чисел: найти количество чисел, входящих в диапазон [1..11] и каждое число возвести в квадрат.

23. Произвести следующую обработку 15 вещественных чисел: найти количество чисел, равных 0, а числа, входящие в диапазон [-1..1] возвести в куб.

24. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, больших или равных 1,5, и подсчитать сумму отрицательных чисел, входящих в диапазон [-1..0].

25. Произвести следующую обработку 10 вещественных чисел: найти количество чисел, меньших 15, и найти произведение чисел, входящих в диапазон [10..15].

Лабораторная работа № 6
Работа со строками

Цель работы: изучение правил описания, ввода-вывода и основных функций обработки символьных данных.

Разработка программ с использованием строк **Краткие теоретические сведения**.

Строки в С++ позволяют работать с символьными данными. С помощью строк возможно осуществить чтение с клавиатуры текста, его обработка и вывод.

В С++ существует 2 типа строк:

1. Массив переменных типа char, заканчивающийся нуль-терминатором \0.

Символьные строки состоят из набора символьных констант заключённых в двойные кавычки. При объявлении строкового массива необходимо учитывать наличие в конце строки нуль-терминатора, и отводить дополнительный байт под него.

char string[10]; // string - имя строковой переменной, 10 - размер массива, (в строке может поместиться 9 символов, последнее место отводится под нуль-терминатор)

2. Специальный класс string

Для его работы необходимо в начале программы подключить заголовочный файл string: *#include <string>*

Для создания строки необходимо в начале программы написать

using namespace std;

Теперь чтоб создать строку достаточно написать: *string s;*

Для записи в строку можно использовать оператор =

s="Hello";

Доступ к i-му элементу строки **s** типа string осуществляется стандартным образом s[i]. Над строками типа string определены следующие операции:

1. присваивания, например s1=s2;
2. объединения строк (s1+=s2 или s1=s1+s2) — добавляет к строке si строку s2, результат храниться в строке si, пример объединения строк:
3. сравнения строк: si=s2, si!=s2, si<s2, si>s2, si<=s2, si>=s2 — результатом будет логическое значение.

Существует множество функций для работы со строками (таблица 1).

Таблица 1- Функции для работы со строками

Функция	Объяснение
s.append(str)	добавляет в конец строки строку str
s.assign(str)	присваивает строке s значение строки str
s.clear()	отчищает строку, т.е. удаляет все элементы в ней
s.compare(str)	сравнивает строку s со строкой str и возвращает 0 в случае совпадение
s.copy(C, I, N)	копирует из строки s в строку C(может быть как строка типа string, так и строка типа char) I символов, начиная с N-го символа
bool b=s.empty()	если строка пуста, возвращает true, иначе false
s.erase(I,N)	удаляет N элементов с I-го символа
s.find(str,I)	ищет строку str начиная с I- го символа
s.insert(pos, s1)	вставляет строку **si** в строку **s**, начиная с позиции **pos**
int len=s.length()	записывает в len длину строки
s.push back(symbol)	добавляет в конец строки символ
s.replace(index, n,str)	берет n первых символов из str и заменяет символы строки s на них, начиная с позиции index

Функция	Объяснение
str=s.substr(n,	возвращает m символов начиная с позиции n
s.swap(str)	меняет содержимое s и str местами.
s.size()	возвращает число элементов в строке.

Функции для работы со строками, прототипы которых описаны в заголовочном файле string.h:

1 Сравнение строк. Для сравнения строк используются функции strcmp и strncmp. Функция

int strcmp (const char *str1, const char *str2);

лексикографически сравнивает строки strl, str2 и возвращает -1, 0 или 1, если строка str1 соответственно меньше, равна или больше строки str2.

Функция

int strncmp (const char *str1, const char *str2, size_t n);

лексикографически сравнивает не более чем n первых символов из строк str1 и str2. Функция возвращает -1, 0 или 1, если первые n символов из строки str1 соответственно меньше, равны или больше первых n символов из строки str2.

Пример:

```
// пример сравнения строк
#include #include
int main() {
char str1[] = "aa bb";
char str2[] = "aa aa";
char str3[] = "aa bb cc";
printf("%d\n", strcmp(str1, str3)); // печатает: -1
printf("%d\n", strcmp(str1, str1)); // печатает: -0
printf("%d\n", strcmp(str1, str2)); // печатает: 1
printf("%d\n", strncmp(str1, str3, 5)); // печатает: 0 return 1;
}
```

2 Копирование строк. Для копирования строк используются функции strcpy и strncpy. Функция

char *strcpy (char *str1, const char *str2);

копирует строку str2 в строку str1. Строка str2 копируется полностью, включая завершающий нулевой байт. Функция возвращает указатель на str1. Если строки перекрываются, то результат непредсказуем.

Функция

char *strncpy (char *str1, const char *str2, size_t n);

копирует n символов из строки str2 в строку str1. Если строка str2 содержит меньше чем n символов, то последний нулевой байт копируется столько раз, сколько нужно для расширения строки str2 до n символов. Функция возвращает указатель на строку str1.

Пример:

```
char str1[80];
char str2 = "Copy string.";
strcpy ( str1, str2 );
printf ( str1 ); // печатает: Copy string.
```

3 Соединение строк. Для соединения строк в одну строку используются функции strcat и strncat. Функция

char* strcat (char *str1, const char *str2);

присоединяет строку str2 к строке str1, причем завершающий нулевой байт строки str1 стирается. Функция возвращает указатель на строку str1.

Функция

char* strncat (char *str1, const char *str2, size_t n);

присоединяет n символов из строки str2 к строке str1, причем завершающий нулевой байт строки str1 стирается. Функция возвращает указатель на строку str1. если длина строки

str2 меньше n, то присоединяются только символы, входящие в строку str2. После соединения строк к строке str1 всегда добавляется нулевой байт. Функция возвращает указатель на строку str1.

Пример:

```
#include <stdio.h>
#include <string.h>
int main ( )
{
char str1[80] = "String ";
char str2 = "catenation ";
char str3 = "Yes No";
strcat ( str1, str2 );
printf ("%s\n", str1 );                    // печатает: String catenation
strncat ( str1, str3, 3 );
printf ("%s\n", str1 );                    // печатает: String catenation Yes
return 1;
}
```

4 Поиск символа в строке. Для поиска символа в строке используются функции strchr, strrchr, strspn, strcspn и strpbrk. Функция char* strchr (const char *str, int c);

ищет первое вхождение символа, заданного параметром c, в строку str. В случае успеха функция возвращает указатель на первый найденный символ, а в случае неудачи - NULL.

Функция

char* strrchr (const char *str, int c);

ищет последнее вхождение символа, заданного параметром c, в строку str. В случае успеха функция возвращает указатель на последний найденный символ, а в случае неудачи - NULL.

Пример:

```
#include <stdio.h>
#include <string.h>
int main ( )
{
char str[80] = "Char search";
printf ("%s\n", strchr ( str, 'r' ));                    // печатает: r search
printf ("%s\n", strrchr ( str, 'r' )); // печатает: rch
return 1;
}
```

Функция

size_t strspn (const char *str1, const char *str2);

возвращает индекс первого символа из строки str1, который не входит в строку str2.

Функция

size_t strcspn (const char *str1, const char *str2);

возвращает индекс первого символа из строки str1, который входит в строку str2.

Пример:

```
int main ( )
{
char str[80] = "123 abc";
printf ("n = %d\n", strspn ( str, "321" ); // печатает: n = 3
printf ("n = %d\n", strcspn ( str, "cba" ); // печатает: n = 4 return 1;
}
```

Функция

char* strpbrk (const char *str1, const char *str2);

находит первый символ в строке str1, который равен одному из символов в строке str2. В случае успеха функция возвращает указатель на этот символ, а в случае неудачи - NULL.

Пример.

```
char str[80] = "123 abc";
```

```
printf ("%s\n", strpbrk ( str, "bca" )); // печатает: abc
```

5. Сравнение строк. Для сравнения строк используются функция strstr. Функция

```
char* strstr ( const char *str1, const char *str2 );
```

находит первое вхождение строки str2 (без конечного нулевого байта) в строку str1. В случае успеха функция возвращает указатель на найденную подстроку, а в случае неудачи - NULL. Если указатель str1 указывает на строку нулевой длины, то функция возвращает указатель str1.

Пример:

```
char str[80] = "123 abc 456;
printf ("%s\n", strstr ( str, "abc"); // печать: abc 456
```

6. Разбор строки на лексемы. Для разбора строки на лексемы используется функция strtok. Функция

```
char* strtok ( char *str1, const char *str2 );
```

возвращает указатель на следующую лексему (слово) в строке str1, в которой разделителями лексем являются символы из строки str2. В случае если лексемы закончились, то функция возвращает NULL. При первом вызове функции strtok параметр str1 должен указывать на строку, которая разбирается на лексемы, а при последующих вызовах этот параметр должен быть установлен в NULL. После нахождения лексемы функция strtok записывает после этой лексемы на место разделителя нулевой байт.

Пример.

```
#include <stdio.h>
#include <string.h>
int main( )
{
char str[ ] = "12 34 ab cd";
char *p;
p = strtok ( str, " " );
while (p)
{
printf ( "%s\n", p ); // печатает в столбик значения: 12 34 ab cd
p = strtok ( NULL, " " );
}
return 1;
}
```

7. Определение длины строки. Для определения длины строки используется функция strlen. Функция

```
size_t strlen ( const char *str);
```

возвращает длину строки, не учитывая последний нулевой байт. Например, char str[] = "123";

```
printf ("len = %d\n", strlen ( str )); // печатает: len = 3
```

Пример выполнения задания.

Задание. Найти слова во введенной с клавиатуры строке, вывести их на экран и подсчитать их количество.

Схема программы представлена на рисунке 1:

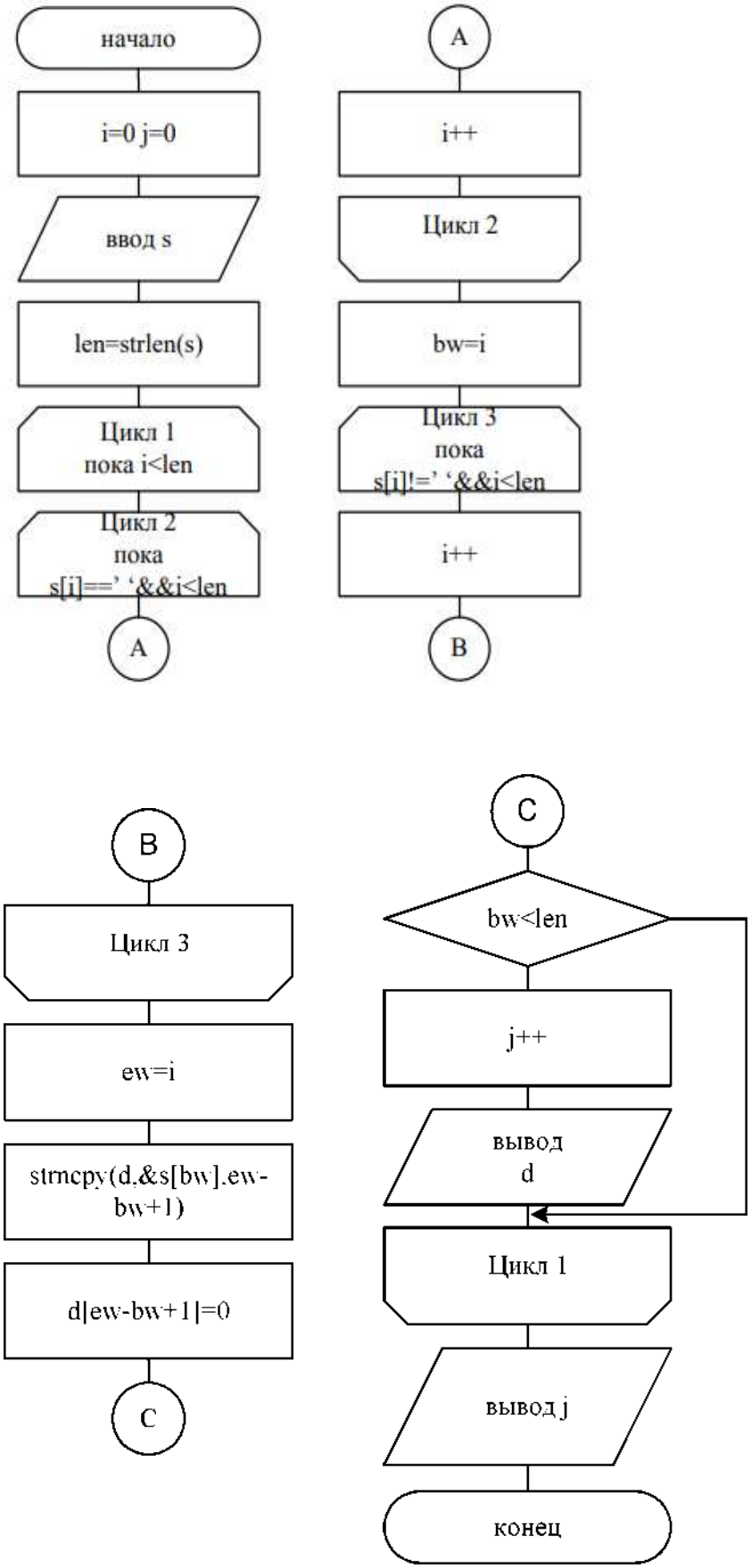

начало
i=0 j=0
ввод s
len=strlen(s)
Цикл 1
пока i<len
Цикл 2
пока
s[i]==' '&&i<len
A
A
i++
Цикл 2
bw=i
Цикл 3
пока
s[i]!=' '&&i<len
i++
B
B
Цикл 3
ew=i
strncpy(d,&s[bw],ew-bw+1)
d[ew-bw+1]=0
C
C
bw<len
j++
вывод
d
Цикл 1
вывод j
конец

Рисунок 1

Пример программы:

```
#include<stdio.h>
#include<string.h>
vid main()
{
 char s[100],d[100];
 int i=0,j=0,bw,ew,len;
 gets(s); len=strlen(s);
 while (i<len)
 {
   while((s[i]==' ')&&(i<len)) i++;
   bw=i;
   while((s[i]!=' ')&&(i<len)) i++;
   ew=i;
   strncpy(d,&s[bw],ew-bw+1);
   d[ew-bw+1]=0;
   if (bw<len)
      printf("%s\n",d);}
 }
printf("Vsego slov %d\n", j);
}
```

Контрольные вопросы

1. Строки в языке С++: понятие строки, описание строк в программе, обращение к элементам строки.
2. Три способа ввода строк в С++.
3. Три способа вывода строк в С++.
4. Способы инициализации строк (задание значений в программе).
5. Стандартные функции для обработки строк.

Варианты заданий

Дана последовательность символов S1, ..., SN. Группы символов, разделенные пробелом (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами.

1. Определить число символов в самом длинном слове строки. Слова отделяются знаком “/”.
2. В произвольном тексте выделить и отпечатать слова, начинающиеся с буквы А.
3. В произвольном тексте вставить между первым и вторым словом новое слово.
4. В произвольном тексте найти и отпечатать слова, содержащие букву Е.
5. Отпечатать второе и третье слова произвольного текста.
6. В произвольном тексте вставить между вторым и третьим словом новое слово.
7. В произвольном тексте найти и отпечатать все слова длиной 5 символов.
8. В произвольном тексте найти самое короткое слово.
9. В последовательности из 10 пятибуквенных слов найти и поменять местами пару слов, у которых первые три буквы одного совпадают с последними тремя буквами другого.
10. Упорядочить в алфавитном порядке последовательность из 10 пятибуквенных слов.
11. В строке из 50 символов отдельные слова разделены пробелом. Упорядочить строку так, чтобы каждое следующее слово было не короче предыдущего.
12. Расположить слова строки в порядке, обратным исходному.
13. Подсчитать количество букв ‘а’ в последнем слове строки.

14. Найти количество слов, у которых первый и последний символы совпадают между собой.

15. Исключить из строки слова, расположенные между скобками (,). Сами скобки должны быть исключены.

16. В произвольном тексте найти и отпечатать слова, содержащие букву А.

17. Отпечатать первое и второе слова произвольного текста.

18. В произвольном тексте вставить после первого слова новое слово.

19. В произвольном тексте найти и отпечатать все слова длиной 4 символа.

20. В произвольном тексте найти самое длинное слово.

21. Выполнить сравнение двух строк s и d. Результат вывести в виде сообщения «идентичны» или «не идентичны».

22. Добавить в конец строки новое слово, длинною 5 символов, иначе выдать сообщение об ошибке.

23. Добавить в начало строки новое слово, начинающееся с буквы а, иначе, если слово начинается с другой буквы вывести сообщение о невозможности добавления.

24. Посчитать какое количество раз встречается буква n (задается при каждом выполнении алгоритма).

25. Проанализировать массив символов, состоящий из n символов. Если массив состоит из n-5 символов, добавить в конец набор символов rrrrr.

Лабораторная работа № 7

Обработка двумерных массивов. Указатели

Цель работы: изучение способов описания, ввода-вывода и обработки двумерных массивов, использование указателей при работе с массивами.

Краткие теоретические сведения

Двумерный массив — это обычная таблица, со строками и столбцами. Фактически двумерный массив — это одномерный массив одномерных массивов. Структура двумерного массива, с именем *a*, размером *m* на *n* показана ниже (рисунок 6).

a[0][0]	a[0][1]	a[0][2]	a[0][3]	. . .	a[0][n]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	. . .	a[1][n]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	. . .	a[2][n]
. . .	. . .	. . .	. . .	. . .	. . .
a[m][0]	a[m][1]	a[m][2]	a[m][3]	. . .	a[m][n]

Рисунок 6 — Двумерный массивы в C++

где, *m* — количество строк двумерного массива; *n* — количество столбцов двумерного массива;

m x *n* — количество элементов массива.

Наиболее распространенный синтаксис объявления двумерного массива:

/*тип данных*/ /*имя массива*/[/*количество строк*/][/*количество столбцов*/];

Примеры инициализации двумерного массива:

— способ 1:

```
int arr[5][3];
```

— способ 2:

```
int arr[5][3] = { {4, 7, 8}, {9, 66, -1}, {5, -5, 0}, {3, -3, 30}, {1, 1, 1} };
```

В последнем случае представление массива и обращение к элементу массива имеет вид, показанный на рисунке 7,8.

Рисунок 7 — Двумерный массив в C++

Рисунок 8 — Обращение в элементу массива

Ввод массива с клавиатуры и его вывод на экран выполняется следующим образом:

```
int m,n,ij;
 cout<<"Введите размеры матрицы:"<<endl;
 cin>>m;
 cin>>n;
 int elem;
 int ar [m][n];
  for (i=0; i<m; i++)
 {for (j=0; j<n; j++){
  cout<<«"Введите элемент:"«endl;
```

```
  cin>>elem;
   ar [i][j]=elem;
   }
}
for (i=0; i<m; i++)
{for (j=0; j<n; j++){
  cout<<ar[i][j]«endl;
  }
}
system("pause");
return 0;
}
```

При работе с массивами можно использовать указатели. Указатель - переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию. Для этого нужно знать адрес переменной или функции. Так вот, чтобы узнать адрес конкретной переменной в C++ существует унарная операция взятия адреса &. Такая операция извлекает адрес объявленных переменных, для того, чтобы его присвоить указателю.

Указатели используются для передачи по ссылке данных, что намного ускоряет процесс обработки этих данных (в том случае, если объём данных большой), так как их не надо копировать, как при передаче по значению, то есть, используя имя переменной. В основном указатели используются для организации динамического распределения памяти, например при объявлении массива, не надо будет его ограничивать в размере. Ведь программист заранее не может знать, какого размера нужен массив тому или иному пользователю, в таком случае используется динамическое выделение памяти под массив. Любой указатель необходимо объявить перед использованием, как и любую переменную:

/*тип данных*/ * /*имя указателя*/;

Работу с указателями можно представить следующим образом:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main(int argc, char* argv[])
{
   int var = 123; // инициализация переменной var числом 123
   int *ptrvar = &var; // указатель на переменную var (присвоили адрес переменной указателю)
   cout << "&var = " << &var << endl;// адрес переменной var содержащийся в памяти, извлечённый операцией взятия адреса
   cout << "ptrvar = " << ptrvar << endl;// адрес переменной var, является значением указателя ptrvar
   cout << "var = " << var << endl; // значение в переменной var
   cout << "*ptrvar = " << *ptrvar << endl; // вывод значения содержащегося в переменной var через указатель, операцией разименования указателя
   system("pause");
   return 0;
}
```

Результат работы представлен ниже на рисунке 9.

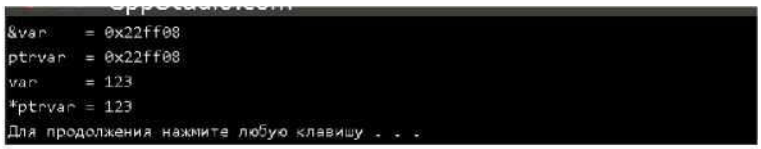


Рисунок 9

Пример выполнения задания.

Задание. Найти максимальную сумму элементов строк матрицы 3x5. Написать программы без использования указателей и с использованием указателей.

Схема программы без использования указателей представлена на рисунке 10:

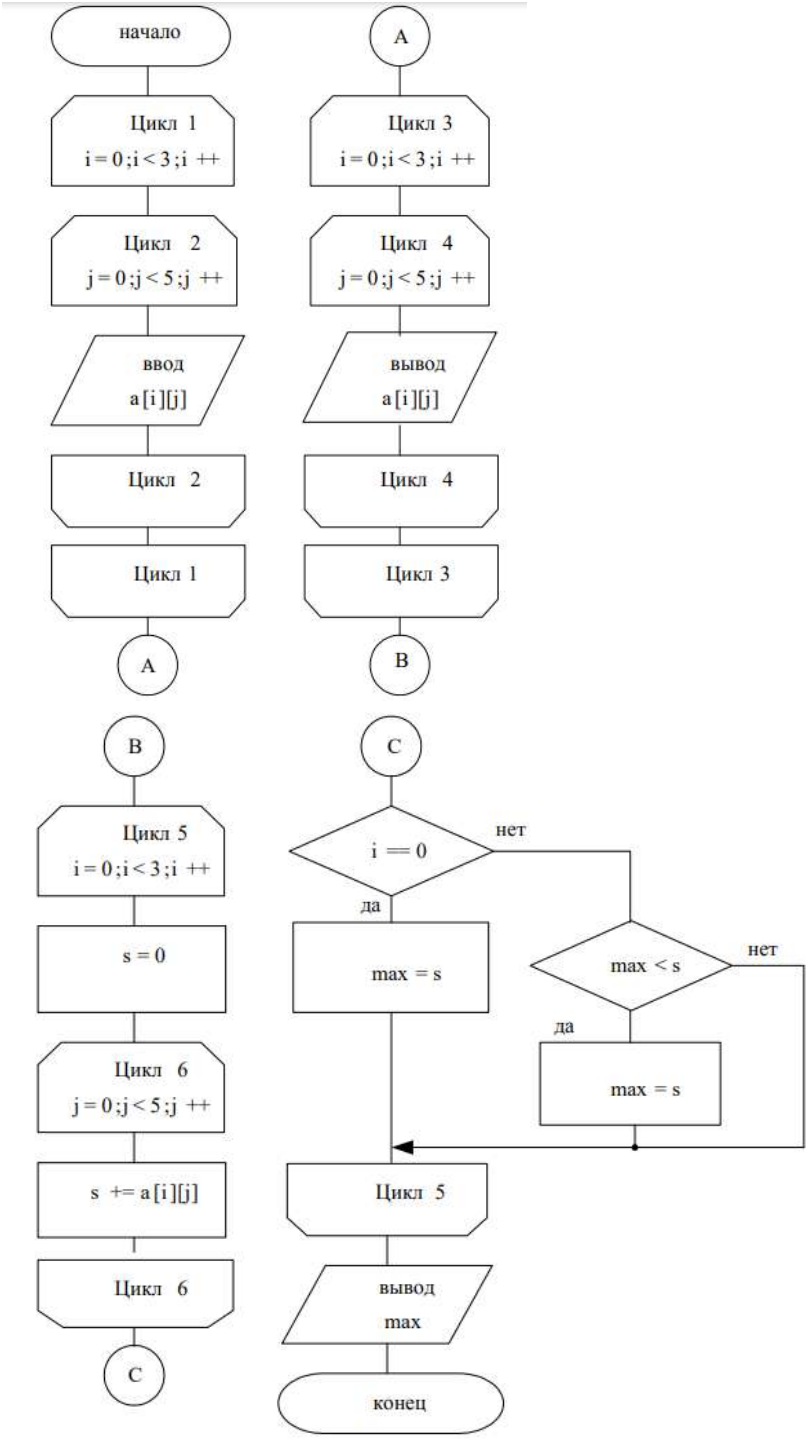
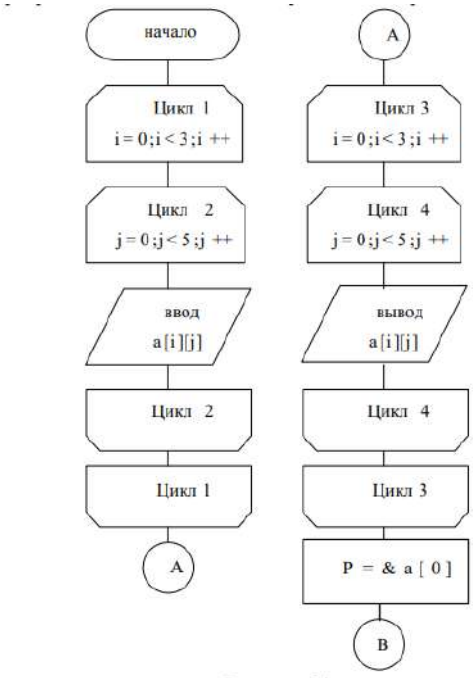


Рисунок 10

Текст программы:

```
#include <stdio.h>
void main()
{
int a[3][5], i, j, s, max;
printf (“Введите 3 строки по 5 чисел”);
for (i=0;i<3;i++)
  for (j=0;j<5;j++)
  scanf("%d",&a[i][j]);
   printf (“Матрица a :\n”);
      for (i=0; i<3; i++)
       {for (j=0; j<5; j++)
        printf (“%5d”, a[i][j]);
      printf (“\n”);
  }
 for(i=0;i<3;i++)
  {s=0;
      for (j=0;j<5;j++)
       s+=a[i][j];
    if (i==0) max=s;
    else if (max<s) max=s;}
printf ("Максимальная сумма строки = %d",max);
}
```

Схема программы с использованием указателей представлена на рисунке 11



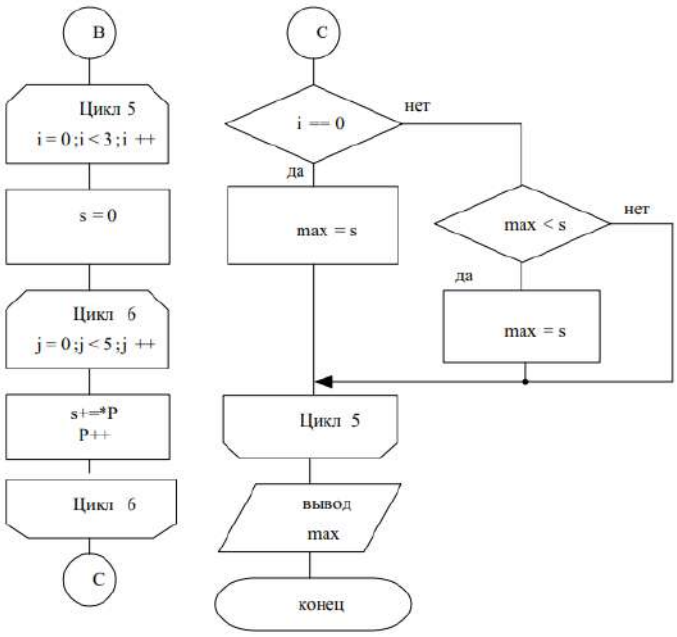


Рисунок 11

Пример программы с использованием указателей:

```
#include <stdio.h>
void main()
{
int a[3][5], *P, i, j, s, max;
printf (“Введите 3 строки по 5 чисел”);
for (i=0;i<3;i++)
 for (j=0;j<5;j++)
  scanf("%d",&a[i][j]);
printf (“Матрица a :\n”);
 for (i=0; i<3; i++)
  {for (j=0; j<5; j++)
    printf (“%5d”, a[i][j]);
    printf (“\n”);
  }
P=&a[0][0];
for(i=0;i<3;i++)
  {s=0;
    for (j=0;j<5;j++)
      {s+=*P;
       P++;
      }
    if (i==0) max=s;
    else if (max<s) max=s;
  }
printf ("Максимальная сумма строки = %d",max);
}
```

Контрольные вопросы

1. Особенности организации двумерных массивов: понятие массива в языке С++, описание массива в программе, представление элементов массива в памяти,

обращение к элементам массива.

2. Указатели в языке С++: понятие указателя, описание указателя в программе.
3. Операции над указателями.
4. Связь массивов и указателей.

Варианты заданий

1. Вычислить сумму положительных элементов каждого столбца матрицы A(mxn).
2. Из матрицы *X* (mxn) построить матрицу *Y,* поменяв местами строки и столбцы.
3. Найти наименьший элемент матрицы *X* (mxn) и записать нули в ту строку и столбец, где он находится.
4. Переписать первые элементы каждой строки матрицы *A* (mxn), большие *C*, в массив *B***.** Если в строке нет элемента, большего *C*, то записать ноль в массив *B*.
5. Дана действительная матрица размера *m*x*n*. Найти сумму наибольших значений элементов ее строк.
6. В данной действительной матрице размера mxn поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что такой элемент единственный.
7. В данной действительной квадратной матрице порядка *n* найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
8. Дана действительная матрица размера *m*x*n,* все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.
9. Дана целочисленная матрица размера mxn. Найти матрицу, получающуюся перестановкой столбцов (первого с последним, второго с предпоследним и т.д.).
10. Дана целочисленная матрица размера mxn. Найти матрицу, получающуюся перестановкой строк (первой с последней и т.д.).
11. Дана действительная матрица [*aij*], где i, j = 1..*n.* Получить действительную матрицу [*bij*], где i, j = 1..*n,* элемент *bij* которой равен сумме элементов данной матрицы, расположенных в области, определяемой индексами i, j (область заштрихована на рисунке 12):

Рисунок 12

12. Дана действительная квадратная матрица порядка *n*. Преобразовать матрицу по правилу: строку с номером *n* сделать столбцом с номером *n,* а столбец с номером *n* сделать строкой с номером *n.*
13. Просуммировать элементы матрицы X (4,5), сумма индексов которых равна заданной константе K.
14. Дана матрица M (4x5). Вычислить вектор D, компоненты которого равны сумме элементов строк матрицы.
15. Дана матрица M (6x6). Вычислить сумму элементов главной диагонали.
16. Дана матрица N (6x5). Найти столбец с минимальной суммой элементов.
17. Дана матрица M (4x5) и константа C. Вычислить матрицу D, равную произведению элементов матрицы M на константу.
18. Дана матрица M (4x6). Вычислить вектор D, компоненты которого равны сумме элементов столбцов матрицы.

19. Дана действительная квадратная матрица порядка *n,* все элементы которой различны. Найти наибольший элемент, среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.

20. Дана действительная квадратная матрица порядка n. Найти наибольшее из значений элементов, расположенных в заштрихованной части матрицы (рисунок 13):

Рисунок 13

21. Дана матрица M (2x5), определить максимальный и минимальный элементы. Поменять местами максимальный и минимальный элементы.

22. В матрице A(пхп) вычислить сумму элементов матрицы (n-2xn-2) и определить максимальный элемент в ней.

23. Дана матрица M (6x6). Вычислить произведение элементов главной диагонали с константой C.

24. Дана матрица N (6x5). Найти строку с минимальной суммой элементов, а элемент с номером nij возвести в квадрат.

25. Дана матрица вещественных чисел A (mxn). В сроке m определить максимальный элемент, а в столбце n количество элементом, меньших порога k.

Лабораторная работа № 8

Работа с функциями

Цель работы: Освоение методов определения функций, передачи аргументов, использования библиотек функций, основ нисходящей технологии программирования.

Краткие теоретические сведения

Функции разбивают большие вычислительные задачи на маленькие подзадачи и позволяют использовать в работе то, что уже сделано другими, а не начинать каждый раз с пустого места. Соответствующие функции часто могут скрывать в себе детали проводимых в разных частях программы операций, знать которые нет необходимости, проясняя тем самым всю программу, как целое, и уменьшая трудности при внесении изменений.
Широко используются «библиотечные» функции. Каждая функция языка С имеет имя и список аргументов (формальных параметров).

Функции могут возвращать значение. Это значение может быть использовано далее в программе. Так как функция может вернуть какое-нибудь значение, то обязательно нужно указать тип данных возвращаемого значения. Если тип не указан, то по умолчанию предполагается, что функция возвращает целое значение (тип int). После имени функции принято ставить круглые скобки (это касается вызова функции, её объявления и описания). В этих скобках перечисляются параметры функции, если они есть. Если у функции нет параметров, то при объявлении и при описании функции вместо <список параметров> в приведенном далее примере кода надо поставить void - пусто.
Основная форма описания функции имеет вид:

```
        тип <имя функции>(список параметров)
{
      тело функции
}
```

Объявление (прототип) функции имеет вид:

```
        тип <имя функции>(список параметров);
```

Обратите внимание на то, что при описании функции после заголовка функции точка с запятой не ставится, а при объявлении функции точка с запятой ставится.
Вызов функции делается следующим образом:

```
        <имя функции>(параметры);
```

или

```
        <переменная>=<имя функции>(параметры);
```

При вызове функции так же ставится точка с запятой.

Почему надо объявлять функцию до использования? Дело в том, что для правильной работы кода функции машине надо знать тип возвращаемого значения, количество и типы аргументов. При вызове какой-либо функции копии значений фактических параметров записываются в стек, в соответствии с типами указанными в ее прототипе. Затем происходит переход в вызываемую функцию.
Приведем пример вызова функции

```
        /* Используем свою функцию */
void function1(void);      // Объявление функции
void main(void)       // Точка входа в программу
{
    function1();             //  Вызов функции
}
/* Описание функции  */
void function1(void)          // Заголовок функции
{       // Начало тела функции


        }       //  Конец тела функции
```

Мы объявили функцию function1(), затем её вызвали. Необходимо обратить внимание ещё на то, что тип возвращаемого значения у функции void (пусто). Это значит, что функция не будет возвращать никакого значения.

Рассмотрим пример, в котором описаны две функции, для одной из которых указан тип возвращаемого значения - int.

```
        int x;// Объявляем переменную x (глобальная переменная)
        void main(void)
{
      void function1(void);      // Объявляем функцию
      int function2();           // function2() будет
         // возвращать значение типа int
               x = 10; // Присваиваем переменной x значение
               function1(); // Вызываем функцию function1()
               x = function2();// Вызываем функцию function2()
}

/* Описание функций */
void function1(void)
{

        }
int function2(void)
{
      int y; // Объявляем локальную переменную
      y = x + 10;
      return y; // Возвращаем значение y
}
```

Теперь давайте посмотрим текст программы. Объявляем глобальную переменную x. Так как x - глобальная переменная, то она будет видна всем функция программы. В теле main() объявляем две функции, одна из которых может возвращать значение типа int. Далее присваиваем переменной x значение 10, так как x это глобальная переменная, то эта переменная будет видна функции main() т.е. функция main() может использовать эту переменную.

Далее мы вызываем функцию function1(). В следующей строке x = function2(); переменная x принимает значение которое вернет функция function2(). Посмотрите на описание функции function2(). В теле этой функции объявляем переменную y, а дальше переменной y мы присваиваем значение переменной x + 10. Так как x - глобальная переменная (она видна для функции function2), все будет работать.

Далее идет строка return y; с помощью оператора return мы возвращаем значение переменной y. Запомните, что если функция возвращает значение, то в теле этой функции обязательно должен присутствовать оператор return (он может быть и не один). Ну так вот с помощью оператора return мы возвращаем значение локальной переменной y в вызывающую функцию main().

Функции языка C могут иметь параметры. Эти параметры передаются в функцию и там обрабатываются. Ещё раз рассмотрим основную форму описания функции:

```
        тип <имя функции>(список параметров)
{
      тело функции
}
```

В списке параметров для каждого параметра должен быть указан тип.
Пример правильного списка параметров:

function(int x, char a, float z)

Пример неправильного списка параметров:

function(int x, a, float z)

Рассмотрим все это на примере. Пусть будет функция, у которой присутствует один параметр x. Функция будет возвращать квадрат значения x.

```
int square(int x)
{
    x = x * x;
    return x;
}
```

Формальные параметры - это параметры которые объявляются в заголовке функции при описании.

Фактические параметры - это параметры которые подставляются при вызове функции.

```
void myfunc(int x); // Объявление функции
void main(void)
{
   int a;
   a=5;
   myfunc(a); // a- фактический параметр
}
// Описание функции
void myfunc(int x) // x - формальный параметр
{
   x = x + 10;
}
```

В языке С функция может возвращать несколько значений. Чтобы функция могла вернуть несколько значений необходимо пользоваться указателями. Этот механизм называется - передача параметров по ссылке.

В С++ при вызове функций можно опускать параметры. В таких случаях для опущенных параметров будут использоваться значения по умолчанию. Обеспечение значений по умолчанию для параметров упрощает возможность повторного использования функций (их использования несколькими программами).

Применение библиотечных функций сокращает объем программирования, который программист должен выполнить самостоятельно. Вместо этого программа просто вызывает функции библиотеки этапа выполнения. В зависимости от компилятора библиотека этапа выполнения может состоять из тысяч функций.

Ранее было сказано, что до того, как программа сможет вызвать функцию, компилятор С++ должен узнать определение или прототип функции. Поскольку функции библиотеки этапа выполнения не определены в основной программе, необходимо указать прототип для каждой библиотечной функции, которую намерен использовать программист. Для упрощения использования библиотечных функций компилятор С++ предоставляет заголовочные файлы, содержащие корректные прототипы. Таким образом, программам необходимо просто включить требуемый заголовочный файл с помощью оператора #include, а затем вызвать необходимую функцию.

3. Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант задания у преподавателя.
3. Выполнить задание.
4. Продемонстрировать выполнение работы преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу.

4. Требования к оформлению отчета

Отчет по лабораторной работе должен содержать следующие разделы:
титульный лист;
цель работы:
задание на лабораторную работу;
техническое описание выполненного задания;
блок-схему;
ответы на контрольные вопросы;
выводы по проделанной работе.

5. Варианты заданий

Во всех вариантах необходимо использовать пользовательские функции. Предусмотреть режим диалога.

1. Треугольник задан координатами своих вершин. Составить программу для вычисления его площади.
2. Составить программу для нахождения наибольшего общего делителя четырех натуральных чисел.
3. Составить программу для нахождения наименьшего общего кратного трех натуральных чисел.
4. Написать программу для нахождения суммы большего и меньшего из трех чисел.
5. Вычислить площадь правильного шестиугольника со стороной a, используя функцию вычисления площади треугольника.
6. На плоскости заданы своими координатами n точек. Составить программу, определяющую, между какими из пар точек самое большое расстояние. Указание. Координаты точек занести в массив.
7. Составить программу, которая в массиве A[10] находит второе по величине число (вывести на печать число, которое меньше максимального элемента массива, но больше всех других элементов).
8. Написать программу для вычисления суммы факториалов всех нечетных чисел от 1 до 9.
9. Даны две дроби A/B и C/D (A, B, C, D — натуральные числа). Составить программу для деления дроби на дробь. Результат должен быть несократимой дробью.
10. Задан массив D. Определить следующие суммы: D[1] + D[2] + D[3]; D[3] +D[4] +D[5]; D[4] + D[5] + D[6]. Пояснение. Составить подпрограмму для вычисления суммы трех последовательно расположенных элементов массива с номерами k до m.
11. На плоскости заданы своими координатами n точек. Создать массив размером n(n-1), элементами которого являются расстояния от каждой из точек до n -1 других.
12. Даны числа X, Y, Z, T — длины сторон четырехугольника. Вычислить его площадь, если угол между сторонами длиной X и Y— прямой.
13. Составить программу для вычисления суммы факториалов всех четных чисел от m до n.
14. Заменить отрицательные элементы линейного массива их модулями, не пользуясь стандартной функцией вычисления модуля. Подсчитать количество произведенных замен.
15. Дано простое число. Составить функцию, которая будет находить следующее за ним простое число.
16. Составить функцию для нахождения наименьшего нечетного натурального делителя k (k не равно 1) любого заданного натурального числа n.
17. Дано натуральное число N. Составить программу для формирования массива, элементами которого являются цифры числа N.
18. Составить программу, определяющую, в каком из данных двух чисел больше цифр.
19. Заменить данное натуральное число на число, которое получается из исходного записью его цифр в обратnov порядке.

20. Даны три квадратных матрицы A, B, C размерность 10. Вывести на печать ту из них, норма которой наименьшая. Пояснение. Нормой матрицы назовем максимум из абсолютных величин ее элементов.
21. Два натуральных числа называются «дружественными», если каждое из них равно сумме всех делителей (кроме его самого) другого числа (например, числа 220 и 284). Найти все пары «дружественных чисел», которые не больше данного числа N.
22. Два простых числа называются «близнецами», если они отличаются друг от друга на 2 (например, 41 и 43). Напечатать все пары «близнецов» из отрезка [n, 2n], где n — заданное натуральное число больше 2.

Контрольные вопросы

1. Зачем нужно использовать функции ?
2. Чем отличается прототип и определение функции?
3. Для чего нужны прототипы функций?
4. Что такое формальные и фактические параметры? Пример.

Лабораторная работа № 9
Работа с файлами

Цель работы: научиться выполнять ввод-вывод из файла с помощью стандартной библиотеки.

Краткие теоретические сведения

Частью стандартной библиотеки C++ является библиотека iostream – объектно-ориентированная иерархия классов, где используется и множественное, и виртуальное наследование. В ней реализована поддержка для файлового ввода/вывода данных встроенных типов. Кроме того, разработчики классов могут расширять эту библиотеку для чтения и записи новых типов данных.

Для использования библиотеки iostream в программе необходимо включить заголовочный файл

#include <iostream>

Операции ввода/вывода выполняются с помощью классов istream (потоковый ввод) и ostream (потоковый вывод). Третий класс, iostream, является производным от них и поддерживает двунаправленный ввод/вывод. Для удобства в библиотеке определены три стандартных объекта-потока:

- cin – объект класса istream, соответствующий стандартному вводу. В общем случае он позволяет читать данные с терминала пользователя;
- cout – объект класса ostream, соответствующий стандартному выводу. В общем случае он позволяет выводить данные на терминал пользователя;
- cerr – объект класса ostream, соответствующий стандартному выводу для ошибок. В этот поток мы направляем сообщения об ошибках программы.

Помимо чтения с терминала и записи на него, библиотека iostream поддерживает чтение и запись в файлы. Для этого предназначены следующие классы:

- ifstream, производный от istream, связывает ввод программы с файлом;
- ofstream, производный от ostream, связывает вывод программы с файлом;
- fstream, производный от iostream, связывает как ввод, так и вывод программы с файлом.

Чтобы использовать часть библиотеки iostream, связанную с файловым вводом/выводом, необходимо включить в программу заголовочный файл

#include <fstream>

Если файл будет использоваться только для вывода, мы определяем объект класса ofstream. Например:

```
ofstream outfile( "copy.out", ios::base::out );
```

Передаваемые конструктору аргументы задают имя открываемого файла и режим открытия. Файл типа ofstream может быть открыт либо – по умолчанию – в режиме вывода (ios_base::out), либо в режиме дозаписи (ios_base::app). Такое определение файла outfile2 эквивалентно приведенному выше:

```
// по умолчанию открывается в режиме вывода
ofstream outfile2( "copy.out" );
```

Если в режиме вывода открывается существующий файл, то все хранившиеся в нем данные пропадают. Если же мы хотим не заменить, а добавить данные, то следует открывать файл в режиме дозаписи: тогда новые данные помещаются в конец. Если указанный файл не существует, то он создается в любом режиме.

Прежде чем пытаться прочитать из файла или записать в него, нужно проверить, что файл был успешно открыт:

```
if ( ! outfile ) { // открыть файл не удалось
  cerr <<"не могу открыть "copy.out" для записи\n";
  exit( -1 );
}
```

Класс ofstream является производным от ostream. Все определенные в ostream операции применимы и к ofstream.

Следующая программа читает из стандартного ввода символы и копирует их в стандартный вывод:

```
#include <fstream>
int main()
{
   // открыть файл copy.out для вывода
   ofstream outFile( "copy.out" );
   if ( ! outFile ) {
      cerr << "Не могу открыть 'copy.out' для вывода\n";
      return -1;
   }
   char ch;
   while ( cin.get( ch ) )
   outFile.put( ch );
}
```

Чтобы открыть файл только для чтения, применяется объект класса ifstream, производного от istream. Следующая программа читает указанный пользователем файл и копирует его содержимое на стандартный вывод:

```
#include <fstream>
#include <string>
int main()
{
   cout << "filename: ";
   string file_name;
   cin>> file_name;
   // открыть файл для ввода
   ifstream inFile( file_name.c_str() );
   if ( !inFile ) {
       cerr < <"не могу открыть входной файл: "
           <<file_name << " -- аварийный останов!\n";
       return -1;
   }
   char ch;
   while ( inFile.get( ch ))
         cout.put( ch );
}
```

Объекты классов ofstream и ifstream разрешено определять и без указания имени файла. Позже к этому объекту можно присоединить файл с помощью функции-члена open():

```
ifstream curFile;
// ...
curFile.open( filename.c_str() );
if ( ! curFile )  // открытие успешно?
  // ...
```

Чтобы закрыть файл, вызываем функцию-член close():

```
.
.
.
.
      curFile.close();
```

```
    }
}
```

Объект класса fstream может также открывать файл одновременно для ввода и вывода. Например, приведенная инструкция открывает файл word.out для ввода и дозаписи:

```
fstream io( "word.out", ios_base::in|ios_base::app );
```

Для задания нескольких режимов используется оператор побитового ИЛИ.

Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант задания у преподавателя.
3. Выполнить задание.
4. Продемонстрировать выполнение работы преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу.

Требования к оформлению отчета

Отчет по лабораторной работе должен содержать следующие разделы:
титульный лист;
цель работы:
задание на лабораторную работу;
техническое описание выполненного задания;
блок-схему;
выводы по проделанной работе.

Варианты заданий

1. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g те компоненты файла f, которые являются четными.
2. Записать в файл 10 действительных чисел. Вычислить произведение компонентов файла и вывести на печать.
3. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g все компоненты файла f, которые делятся на 2.
4. Записать в файл 10 целых чисел, полученных с помощью генератора случайных чисел. Подсчитать количество пар противоположных чисел среди компонентов этого файла.
5. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Из файла f получить файл g, исключив повторные вхождения чисел.
6. Записать в файл 10 произвольных натуральных чисел. Переписать в другой файл те элементы, которые кратны К.
7. Заполнить файл 10 действительными числами, полученными с помощью датчика случайных чисел. Найти сумму минимального и максимального элементов этого файла.
8. Записать в файл 15 натуральных чисел: a_1, a_2,..., a_n (числа получить с помощью датчика случайных чисел). Сформировать новый файл, элементами которого являются числа a_1, a_1*a_2, $a_1*a_2*a_3$, ….
9. Записать в файл f 10 натуральных чисел. Получить в другом файле все компоненты файла f, кроме тех, которые кратны К.
10. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Найти количество удвоенных нечетных чисел среди компонентов файла.
11. Заполнить файл f натуральными числами, полученными с помощью генератора случайных чисел. Найти количество квадратов нечетных чисел среди компонентов.

12. Записать в файл 10 действительных чисел. Найти наибольшее из значений модулей компонентов с нечетными номерами.
13. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Из файла f получить файл g, исключив повторные вхождения чисел. Порядок следования чисел сохранить.
14. Записать в файл 10 действительных чисел. Найти разность первого и последнего компонентов файла.
15. Записать в файл f 10 целых чисел, полученных с помощью генератора случайных чисел. Заполнить файл g числами, которые являются произведениями соседних компонентов файла f.
16. Багаж пассажира характеризуется количеством вещей и их общим весом. Дан файл Bagazh, содержащий сведения о багаже нескольких пассажиров. Сведения о багаже каждого пассажира представляют собой запись с двумя полями: одно поле целого типа (количество вещей) и одно — действительного (вес в килограммах). Найти багаж, средний вес одной вещи в котором отличается не более чем на t кг от общего среднего веса одной вещи.
17. В условиях предыдущей задачи найти число пассажиров, имеющих более двух вещей, и число пассажиров, количество вещей которых превосходит среднее число вещей.
18. В условиях задачи 17 выяснить, имеется ли пассажир, багаж rоторого состоит из одной вещи весом менее t кг.
19. Дан файл Bibl, содержащий сведения о книгах. Сведения о каждой из книг — это фамилия автора, название и год издания. Найти названия книг данного автора, изданных начиная с 2000 г.
20. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Получить название игрушек, цена которых не превышает 1400 руб. и которые подходят детям 5 лет.
21. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет).
22. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Определить стоимость самого дорогого конструктора.
23. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Напечатать название наиболее дорогих игрушек (цена которых отличается от цены самой дорогой игрушки не более чем на 500 руб.);
24. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Получить названия игрушек, которые подходят детям как четырех, так и десяти лет.
25. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Получить сведения о том, можно ли подобрать игрушку, любую, кроме мяча, подходящую ребенку трех лет.
26. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка

может предназначаться для детей от двух до пяти лет). Получить название самой дешевой игрушки.

27. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Получить название самой дорогой игрушки для детей до четырех лет.
28. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Получить названия игрушек для детей четырех-пяти лет.
29. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Получить название самой дорогой игрушки, подходящей детям двух-трех лет.
30. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Определить стоимость самой дорогой куклы.
31. Дан файл Assort, содержащий сведения об игрушках: указываются название игрушки, ее стоимость в рублях и возрастные границы (например, игрушка может предназначаться для детей от двух до пяти лет). Определить суммарную стоимость кукол для детей шести лет.

Лабораторная работа № 10
Основы работы с интегрированной средой
Краткие теоретические сведения

Microsoft Visual Studio .NET 20xx. В Visual Studio .NET (далее VS .NET) каждый проект является частью того, что Microsoft называет решением (solution). Любой код, созданный в VS .NET IDE, относится к некоторому решению. Решение можно рассматривать как хранилище всей информации, необходимой для компиляции программы и ее перевода в форму, пригодную для исполнения. Таким образом, решение состоит из одного или нескольких проектов; различных вспомогательных файлов (графических изображений, ресурсных файлов, метаданных, то есть данных, описывающих другие данные, и т. д.); документации в формате XML. Решение позволяет легко выбрать файлы, задействованные в решении конкретной проблемы.

Новое решение создается командой File > New > Project. Далее требуется выбрать тип проекта, который будет первым в решении, имя решения, и каталог где оно будет находиться. Все новые проекты добавляются в решение лишь с одним отличием, при создании проекта в поле «решение» требуется выбрать «В текущее» (Add to Solution) вместо «в новое решение» (Create new solution). При помощи команды View в главном меню всегда можно вызвать нужное окно на передний план (и передать ему фокус). Все окна IDE свободно перетаскиваются мышью.

Основные окна IDE.

Редактор текста. Редактор обладает полным набором стандартных возможностей, поддерживаемых в редакторах такого рода (вырезание, вставка, поиск/замена и т. д.). Для работы с ними можно использовать стандартные комбинации клавиш Windows (Ctrl+X — вырезать, Ctrl+V — вставить и т. д.). Если вы предпочитаете работать с командами меню, к вашим услугам меню Edit и контекстное меню окна программы. Полный список сочетаний клавиш вызывается из меню Edit; кроме того, он приведен в разделе «Editing, shortcut keys» справочной системы. Например, комбинация Ctrl+I включает режим поиска с приращением.

В распоряжении разработчика имеется средство IntelliSense, выдающее информацию о методах заданного объекта или параметрах, передаваемых при вызове функции. Обычно IntelliSense вызывается автоматически, но его можно вызвать нажатием «Ctrl + Пробел».

Настройка большинства глобальных параметров редактора выполняется в диалоговом окне — выполните команду Tools > Options и выберите в списке строку Text Editor. Например, чтобы выбрать размер позиций табуляции, щелкните в строке Text Editor и выберите нужное значение для всех языков или только для C++. Здесь же выбирается режим создания отступов: None (отступы отсутствуют), Block (курсор выравнивается по началу предыдущей строки) или Smart (автоматическое создание отступов в теле цикла, как того требует хороший стиль программирования). Кстати говоря, устанавливать размер позиций табуляции и форматировать отступы можно в готовом тексте, для чего используются комбинации клавиш Ctrl+K, Ctrl+F (сочетания клавиш требуется нажать подряд, без длительной паузы) или команда Edit > Advanced > Format Selection.

Также изменять отступы выделенного блока можно используя Tab для увеличения отступа и Shift + Tab для уменьшения.

Редактор поддерживает и такую возможность, как свертка фрагментов программы и отображение на их месте заголовков (folding). Обратите внимание на значки «-» рядом с некоторыми строками. Если щелкнуть на таком значке, в листинге будет скрыта соответствующая область (region), а после первой строки кода из блока появится многоточие. Если задержать указатель мыши над многоточием, на экране будет показан свернутый код. Для управления сверткой используется подменю Edit > Outlining. Информация о редакторе IDE находится в разделе справки «Code and Text Editor».

Список задач. В Visual Studio поддерживается список задач (task list). Идея состоит в том, что в программу включаются комментарии с описанием действий, которые

предполагается выполнить в будущем; тип задачи определяется специальным ключевым словом, следующим после знака комментария. В настоящее время определены три встроенные категории задач — TODO, HACK и UNDONE. Комментарии с задачами выводятся в окне, вызываемом командой View > Other Windows > Task List (или комбинацией клавиш Ctrl+/, Ctrl + T).

Окно решения. В окне решения (Solution Explorer) выводится список файлов, входящих в решение. По умолчанию имя решения совпадает с именем первого созданного в нем проекта. Используя Solution Explorer, можно добавлять в проект различные файлы. Например, текст или исходный код. Для этого требуется щелкнуть правой кнопкой мыши по папке, куда необходимо добавить файл, в контекстном меню выбрать Add > New Item (Добавить > Новый элемент), чтобы добавить новый файл, или Add > Existent Item (Добавить > Существующий элемент), чтобы добавить существующий файл.

Окно свойств. Функции окна свойств в VS .NET уже не ограничиваются простым заданием свойств элементов управления. Содержимое окна зависит от того, что в настоящий момент выделено в IDE. Имя и тип выделенного элемента указаны в списке, находящемся в верхней части окна. Чтобы изменить значение свойства, щелкните в правой ячейке и начинайте вводить символы. В окне свойств действуют стандартные комбинации клавиш, используемые при редактировании в системе Windows.

Окно вывода и окно ошибок. В окне вывода (вызываемом командой View > Output или комбинацией клавиш Alt + 2) отображается текущая информация состояния. При построении решения в этом окне компилятор выводит сообщения как об успешном завершении, так и о возникших ошибках. В окне ошибок выводятся все ошибки или предупреждения возникшие во время компиляции программы. Оно вызывается сочетанием клавиш Ctrl + /, Ctrl + E.

Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант задания у преподавателя.
3. Выполнить задание.
4. Продемонстрировать выполнение работы преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу.

Требования к оформлению отчета

Отчет по лабораторной работе должен содержать следующие разделы:
титульный лист;
цель работы:
задание на лабораторную работу;
техническое описание выполненного задания;
выводы по проделанной работе.

Задание на работу

1. Создать решение, содержащее консольное приложение.
2. Добавить проекту файл с исходным кодом следующего содержания:

```
#include “stdafx.h”
int sum( int a, int b){
   return a + b;
}
```

3. В функцию main добавить следующий код:

```
printf(“Sum: %d”,sum(3, 5));
```

4. Запустить созданный проект (клавиша F5). Объяснить, что выведено в окно вывода.
5. Добавить заголовочный файл с прототипом функции sum:

```
int sum(int a, int b)
```

6. Добавить в файл с описанием функции main подключение вновь созданного заголовочного файла #include “имя_файла.h”.

7. Заново запустить программу. Объяснить, что произошло.
8. Добавить в одном из файлов с исходным кодом новые задачи. Добавьте новые задачи через список задач. В чем разница?

Контрольные вопросы

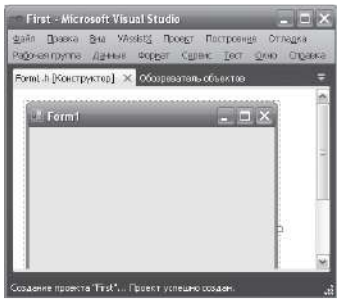
1. Что такое решение (solution) в Visual Studio .NET? Зачем они нужны?
2. Как создать решение? Как добавить туда проект?
3. Что такое IntelliSense? Как он вызывается?
4. Что такое список задач? Как можно добавить задачу?
5. Как осуществляется «сворачивание кода»? Какие области сворачиваются?
6. Зачем нужно окно свойств?
7. Зачем нужно окно вывода и окно ошибок? Чем они отличаются?
8. Зачем нужно окно решения?
9. Как настраиваются параметры форматирования текста в окне редактора?
10. Какие существуют команды форматирования? Как можно менять отступы блоков текста?
11. Как добавить в проект новый или существующий файл.

Лабораторная работа № 11

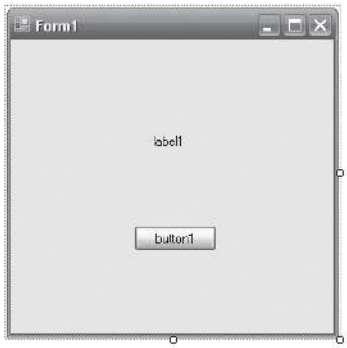
Создание интерфейса пользователя в среде разработки программ на языке С++.

Цель работы: научиться создавать простейшие программы с экранной формой и элементами управления.

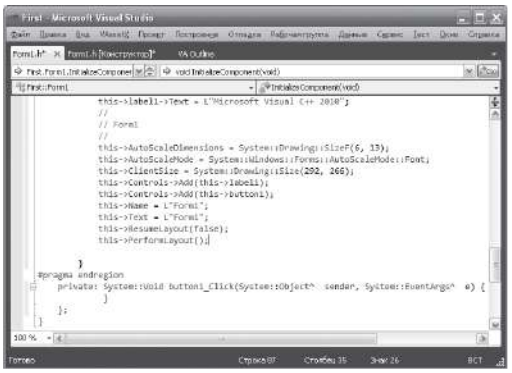
Краткие теоретические сведения



Для создания проекта выбираем File->New Project-> Windows Forms Application Visual C++ вводим имя проекта. Экранная форма — Form1, в которой можно расположить элементы управления. Например поля для ввода текста textBox, командные кнопки button, строчки текста label, которые не могут быть отредактированы пользователем. Визуальное программирование предполагает возможность перетаскивания элементов с помощью мыши из панели элементов Toolbox, где расположены всевозможные элементы управления, в форму. Панель Toolbox предоставляет доступ ко множеству элементов управления при создании Web- и Windows-форм. Она дает также доступ почти ко всему, что можно перетащить на визуальный конструктор, используемый для создания форм, XML, схем, классов и т. д.



Добавьте на форму метку label и кнопку button в форму, дважды щелкая на этих элементах на панели Toolbox или перетаскивая мышкой. У каждого объекта есть свойства. Свойств много, их можно увидеть, если щелкнуть правой кнопкой мыши в пределах формы и выбрать в контекстном меню команду Properties, при этом появится панель свойств Properties. Для объекта label1 выберем свойство Text и напишем напротив этого поля «Лабораторная работа № 9» (вместо текста label1). Для объекта button1 также в свойстве Text напишем «Ввод». Объекты не только имеют свойства, но и обрабатываются событиями. Событием, например, является клик на кнопке, загрузка (Load) формы в оперативную память и пр. Управляют событиями с помощью процедуры обработки события. Для получения пустого обработчика события щелчок на командной кнопке в свойствах кнопки button1 кликаем на значке молнии Events (события) и в списке всех возможных событий кнопки button1 выбираем двойным кликом событие Click. После этого мы попадаем на вкладку программного кода Form1.h.



На вкладке Form1.h мы увидим, что управляющая среда Visual C++ 2010 сгенерировала строки программного кода. Например, для свойства Text кнопки button управляющая среда назначила

this->button1->Text = L"Ввод";

Теперь напишем процедуру обработки события Click на кнопке button1. У нас есть обработчик события button1_Click:

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) { }. В фигурных скобках обработчика события напишем: *MessageBox::Show("Это моя программа!");*

Здесь вызывается метод Show объекта MessageBox с текстом «Это моя программа!» Оператор разрешения области действия (::) указывает системе найти метод Show среди методов объекта Messagebox. Теперь нажмем F5 и проверим работоспособность программы.

При работе с формой часто ввод данных организуют через элемент управления textBox . Добавляем в форму текстовое поле textBox. Изменим некоторые свойства элементов управления. Получим пустой обработчик загрузки формы (дважды кликаем по экранной форме). Задаем свойствам формы (к форме обращаемся посредством ссылки this), кнопке button1 и текстовому полю textBox1, метке label1 следующие значения:

```
this->Text = "Извлечение квадратного корня";
button1->Text = "Извлечь корень";
textBox1->Clear(); // Очистка текстового поля
label1->String::Empty;
```

Далее программируем событие button1.

```
private: System:: Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
Single X;
bool s1 = Single::TryParse(textBox1->Text,
System::Globalization::NumberStyles::Number,
System::Globalization::NumberFormatInfo::CurrentInfo, X);
if (s1 == false)
{
label1->Text = "Введите чило";
label1->ForeColor = Color::Red; // - цвет текста
return; }
Single Y = (Single)Math::Sqrt(X);
label1->ForeColor = Color::Black;
label1->Text = String::Format("Корень из {0} равен {1:F5}", X, Y);
}
```

При обработке события click проверяется, введено ли число в текстовом поле. Проверка осуществляется с помощью функции TryParse. Первым параметром метода TryParse является анализируемое поле textBox1->Text. Второй параметр — это разрешаемый для преобразования стиль числа. Третий параметр указывает, на какой основе формируется допустимый формат, CurrentInfo – это на основе текущего языка и региональных параметров. Четвертый параметр возвращает результат преобразования. Функция TryParse возвращает булеву переменную true или false, успешно ли выполнено преобразование. Если пользователь ввел число, то будет выполняться оператор извлечения квадратного корня Math::Sqrt(X). Математические функции Visual Studio 2010 являются методами класса Math. Функция Math::Sqrt(X)возвращает значение типа double (двойной точности с плавающей запятой), которое мы приводим с помощью неявного преобразования (Single) к переменной одинарной точности. Затем формируем строку label1->Text с использованием метода String::Format. Использованный формат ≪Корень из {0} равен {1:F5}≫ означает: взять нулевой выводимый элемент, то есть переменную X, и записать эту переменную вместо фигурных скобок; после чего взять первый

выводимый элемент, то есть Y, и записать его вместо вторых фигурных скобок в формате с фиксированной точкой и пятью десятичными знаками после запятой.

Рассмотрим пример работы с CheckBox (Флажком).

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
this->Text = "Флажок CheckBox";
checkBox1->Text = "Полужирный"; checkBox1->Focus();
label1->Text = "Выбери стиль шрифта";
label1->TextAlign = ContentAlignment::MiddleCenter;
label1->Font = gcnew System::Drawing::Font("Courier New",14.0F);
}
private: System::Void checkBox1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ // Изменение состояния флажка на противоположное
if (checkBox1->Checked == true) label1->Font = gcnew
System::Drawing::Font("Courier New", 14.0F,FontStyle::Bold);
if (checkBox1->Checked == false) label1->Font = gcnew
System::Drawing::Font("Courier New", 14.0F,FontStyle::Regular);
}
```

При обработке события загрузки формы задаем начальные значения некоторых свойств объектов Form1 (посредством ссылки this), label1 и checkBox1. Изменение состояния флажка соответствует событию CheckedChanged. если флажок установлен (то есть содержит «галочку») Checked = true, то для метки label1 устанавливается тот же шрифт Courier New, 14 пунктов, но Bold, то есть полужирный. Если флажок не установлен, то есть checkBox1.Checked = false, то шрифт устанавливается Regular, то есть обычный.

Элемент управления ComboBox служит для отображения вариантов выбора в раскрывающемся списке. Продемонстрируем работу этого элемента управления Из панели Toolbox перетащим в форму два текстовых поля TextBox, кнопку Button, метку Label и комбинированный список ComboBox.

```
private: System:: Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
this->Text = "Калькулятор"; label1->Text = "Равно: ";
button1->Text = "Выбор операции";
comboBox1->Text = "Выбор операции";
array<String^>^ Операции = {"Сложение", "Вычитание",
"Умножение", "Деление", "Очистить"};
comboBox1->Items->AddRange(Операции);
comboBox1->TabIndex = 2;
textBox1->Clear(); textBox1->TabIndex = 0;
textBox2->Clear(); textBox2->TabIndex = 1;
}
private: System::Void comboBox1_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e)
{
label1->Text = "Равно: ";
Single x, y, z= 0;
```

```
bool s1 = Single::TryParse(textBox1->Text,
System::Globalization::NumberStyles::Number,
System::Globalization::NumberFormatInfo::CurrentInfo, x);
bool s2 = Single::TryParse(textBox2-
>Text,System::Globalization::NumberStyles::Number,
System::Globalization::NumberFormatInfo::CurrentInfo, y);
if (s1 == false || s2 == false)
{
MessageBox::Show("Следует вводить числа!", "Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Error);
return;
}
switch (comboBox1->SelectedIndex)
{
case 0: z = x + y; break;
case 1: z = x - y; break;
case 2: z = x * y; break;
case 3: z = x / y; break;
case 4: textBox1->Clear(); textBox2->Clear(); label1->Text = "Равно: ";
 return;
}
label1->Text = String::Format("Равно {0:F5}", Z);
}
private: System::Void button1_Click(System::Object^ sender,System::EventArgs^ e)
{
comboBox1->DroppedDown = true;
}
};
}
```

При обработке события загрузки формы присваиваем начальные значения некоторым свойствам, задаем коллекцию элементов комбинированного списка, задаем табличные индексы TabIndex для текстовых полей и комбинированного списка. Табличный индекс определяет порядок обхода элементов. При обработке события comboBox1_SelectedIndexChanged с помощью функции TryParse проверяем, можно ли текстовые поля преобразовать в число. По умолчанию при инсталляции русифицированной версии Windows разделителем целой и дробной частей числа является запятая. Оператор switch осуществляет множественный выбор арифметической
операции в зависимости от индекса выбранного элемента списка SelectedIndex. Оператор switch передает управление той или иной метке case. Последний оператор в процедуре обработки события изменения индекса выбранного элемента осуществляет формирование строки с помощью метода String::Format для вывода ее на метку label1. Формат {0:F5} означает, что значение переменной z следует выводить по фиксированному формату с пятью знаками после запятой. Последняя процедура обработки события обеспечивает раскрытие комбинированного списка через нажатие на кнопку.

Рассмотрим пример, который позволяет пользователю вводить в текстовое поле

цифры, а также разделитель целой и дробной части числа.

```
String^  razd;
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
this->Text = "Введите число";
razd = Globalization::NumberFormatInfo::CurrentInfo->NumberDecimalSeparator;
}
private: System::Void textBox1_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e)
{
bool razd_find = false;
if (Char::IsDigit(e->KeyChar) == true) return;
if (e->KeyChar == (char)Keys::Back) return;
if (textBox1->Text->IndexOf(razd) != -1)
razd_find = true;
if (razd_find == true) { e->Handled = true; return; }
if (e->KeyChar.ToString() == razd) return;
e->Handled = true;
}
};
```

Выясняем, что установлено в данной системе в качестве разделителя целой дробной части — точка или запятая. Этот разделитель записываем в строковую переменную razd, которая видна из всех процедур данной программы, поскольку объявлена вне всех процедур. Далее в процедуре обработки события KeyPress разрешаем ввод десятичных цифр и нажатие клавиши Backspace путем обхода с помощью return последнего оператора процедуры e->Handled = true, запрещающего ввод символа в текстовое поле. В данной задаче разрешить ввод разделителя мы можем только один раз, но при этом надо помнить, что пользователь может его удалить и ввести в другом месте числовой строки. Каждый раз при очередном нажатии клавиши, разрешив ввод десятичных цифр, в текстовом поле ищем искомый разделитель. Если он найден, то запрещаем ввод любых нецифровых символов, включая злосчастный разделитель. А если не найден, то разрешаем его ввод.

Содержание работы.

1. Ознакомиться с теоретической частью методических указаний.
2. Получить вариант задания у преподавателя.
3. Решить поставленную задачу с использованием языка программирования C++.
4. Представить работающую программу преподавателю.

Варианты заданий

Разработать программу с использованием экранной формы и элементов управления по варианту.

1. Вычислить периметр и площадь прямоугольного треугольника по длинам *a* и *b* двух катетов.
2. Заданы координаты трех вершин треугольника. Найти его периметр и площадь.
3. Вычислить длину окружности и площадь круга одного и того же заданного радиуса *R*.

4. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
5. Вычислить расстояние между двумя точками с данными координатами.
6. Даны два действительных числа *хну.* Вычислить их сумму, разность, произведение и частное.
7. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
8. Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоту, радиусы вписанной и описанной окружностей.
9. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
10. Найти площадь кольца, внутренний радиус которого равен r, а внешний — *R (R> r).*
11. Треугольник задан величинами своих углов и радиусом описанной окружности. Найти стороны треугольника.
12. Найти площадь равнобедренной трапеции с основаниями *a* и *b* и углом а при большем основании *а.*
13. Вычислить корни квадратного уравнения *ах2+ Ьх + с* = 0 с заданными коэффициентами *a, b* и *c* (предполагается, что *а±0* и что дискриминант уравнения неотрицателен).
14. Найти площадь треугольника, две стороны которого равны *а* и b, а угол между этими сторонами *у.*
15. Написать программу, которая выводит на экран первые четыре степени числа л.
16. Найти сумму членов арифметической прогрессии, если известны ее первый член, знаменатель и число членов прогрессии.
17. Найти (в радианах в градусах) все углы треугольника со сторонами *a, b, c.*
18. Составить программу перевода радианной меры угла в градусы, минуты и секунды.
19. Составить программу для вычисления пути, пройденного лодкой, если ее скорость в стоячей воде *v* км/ч, скорость течения реки *v1* км/ч, время движения по озеру t1 ч, а против течения реки — *t2* ч.
20. Текущее показание электронных часов: *m* ч (0 < *m* < 23) *n* мин (0 < *n* < 59) *к* с (0 < *к* < 59). Какое время будут показывать часы через *р ч q* мин *r* с?
21. Вычислить высоты треугольника со сторонами *a, b, c.*
22. Полторы кошки за полтора часа съедают полторы мышки. Сколько мышек съедят А кошек за К часов?
23. Составить программу вычисления объема цилиндра и конуса, которые имеют одинаковую высоту и одинаковый радиус основания.
24. Дана величина *А,* выражающая объем информации в байтах. Перевести *А* в более крупные единицы измерения информации.
25. Окружность вписана в квадрат заданной площади. Найти площадь квадрата, вписанного в эту окружность. Во сколько раз площадь вписанного квадрата меньше площади заданного?

Содержание отчета.

1. Цель работы и задание.
2. Пункты, соответствующие порядку выполнения работы.
3. Выводы по работе.

Лабораторная работа № 12
Ввод-вывод в языке С++

Цель работы: изучение ввода-вывода в с++

Краткие теоретические сведения

Строгую типовую и единообразную работу как со встроенными, так и с пользовательскими типами можно обеспечить, если использовать единственное перегруженное имя функции для различных операций вывода.
Например:

```
put(cerr,"x = "); // cerr - выходной поток ошибок
put(cerr,x);
put(cerr,'\n');
```

Тип аргумента определяет какую функцию надо вызывать в каждом случае. Такой подход применяется в нескольких языках, однако, это слишком длинная запись. За счет перегрузки операции << , чтобы она означала"вывести" ("put to"), можно получить более простую запись и разрешить программисту выводить в одном операторе последовательность объектов,например так:

```
cerr << "x = " << x << '\n';
```

Здесь cerr обозначает стандартный поток ошибок. Так, если x типа intco значением 123, то приведенный оператор выдаст x = 123 и еще символ конца строки в стандартный поток ошибок. Аналогично, если x имеет пользовательский тип complex со значением (1,2.4), то указанный оператор выдаст x = (1,2.4)в поток cerr. Такой подход легко использовать пока x такого типа, для которого определена операция <<, а пользователь может просто доопределить << для новых типов.

Операции << и >> асиметричны, что позволяет приписывать им смысл "в" и "из". Они не относятся к числу наиболее часто используемых операций над встроенными типами, а приоритет << достаточно низкий, чтобы писать арифметические выражения в качестве операнда без скобок:

```
cout << "a*b+c=" << a*b+c << '\n';
```

Скобки нужны, если выражение содержит операции с более низким приоритетом:

```
cout << "a^b|c=" << (a^b|c) << '\n';
```

Операцию сдвига влево можно использовать в операции вывода, но, конечно,она должна быть в скобках:

```
cout << "a<<b=" << (a<<b) << '\n';
```

Состояния потока

С каждым потоком (istream или ostream) связано определенное состояние.Нестандартные ситуации и ошибки обрабатываются с помощью проверки и установки состояния подходящим образом.

Узнать состояние потока можно с помощью операций над классом ios:

```
class ios { //ios является базовым для ostream и istream
   //...
public:
   int eof() const;     // дошли до конца файла
   int fail() const;    // следующая операция будет неудачна
   int bad() const;     // поток испорчен
   int good() const;    // следующая операция будет успешной
   //...
};
```

Последняя операция ввода считается успешной, если состояние задаетсяgood() или eof(). Если состояние задается good(), то последующая операция ввода может быть успешной, в противном случае она будет неудачной. Применение операции ввода к потоку в состоянии, задаваемом не good(), считается пустой операцией. Если произошла неудача при попытке чтения в переменную v, то значение v не изменилось (оно не изменится, если v имеет тип, управляемый функциями члена из istreamили ostream). Различие между состояниями, задаваемыми как fail() или как bad() уловить трудно, и оно имеет смысл только для разработчиков операций ввода. Если состояние есть fail(), то считается, что поток не поврежден, и никакие символы не пропали; о состоянии bad() ничего сказать нельзя.

Значения, обозначающие эти состояния, определены в классе ios:

```
class ios {
   //...
public:
   enum io_state {
      goodbit=0,
      eofbit=1,
      filebit=2,
      badbit=4,
   };
   //...
};
```

Истинные значения состояний зависят от реализации, и указанные значения приведены только, чтобы избежать синтаксически неправильных конструкций.

Проверять состояние потока можно следующим образом:

```
switch (cin.rdstate()) {
case ios::goodbit:
   // последняя операция с cin была успешной
   break;
case ios::eofbit:
   // в конце файла
   break;
 case ios::filebit:
   // некоторый анализ ошибки
   // возможно неплохой
   break;
  case ios::badbit:
   //  cin возможно испорчен
   break;
  }
```

В более ранних реализациях для значений состояний использовались глобальные имена. Это приводило к нежелательному засорению пространства именования, поэтому новые имена доступны только в пределах класса ios. Если вам необходимо использовать старые имена в сочетании с новой библиотекой, можно воспользоваться следующими определениями:

```
const int _good = ios::goodbit;
const int _bad = ios::badbit;
const int _file = ios::filebit;
const int _eof = ios::eofbit;
```

typedef ios::io_state state_value ;

Разработчики библиотек должны заботиться о том, чтобы не добавлять новых имен к глобальному пространству именования. Если элементы перечисления входят в общий интерфейс библиотеки, они всегда должны использоваться в классе с префиксами, например, как ios::goodbitи ios::io_state.

Для переменной любого типа, для которого определены операции<< и >>, цикл копирования записывается следующим образом:

while (cin>>z) cout << z << '\n';

Если поток появляется в условии, то проверяется состояние потока, и условие выполняется (т.е. результат его не 0) только для состоянияgood(). Как раз в приведенном выше цикле проверяется состояние потокаistream, что является результатом операции cin>>z. Чтобы узнать,почему произошла неудача в цикле или условии, надо проверить состояние.Такая проверка для потока реализуется с помощью операции приведения (7.3.2).

Так, если z является символьным вектором, то в приведенном цикле читается стандартный ввод и выдается для каждой строки стандартного вывода по одному слову (т.е. последовательности символов, не являющихся обобщенными пробелами). Если z имеет тип complex, то в этом цикле с помощью операций, определенных в 10.2.2 и 10.2.3, будут копироваться комплексные числа. Шаблонную функцию копирования для потоков со значениями произвольного типа можно написать следующим образом:

```
complex z;
iocopy(z,cin,cout);    // копирование complex
double d;
iocopy(d,cin,cout);    // копирование double
char c;
iocopy(c,cin,cout);    // копирование char
```

Поскольку надоедает проверять на корректность каждую операцию ввода-вывода, то распространенным источником ошибок являются именно те места в программе, где такой контроль существенен. Обычно операции вывода не проверяют, но иногда они могут завершиться неудачно. Потоковый ввод-вывод разрабатывался из того принципа, чтобы сделать исключительные ситуации легкодоступными, и тем самым упростить обработку ошибок в процессе ввода-вывода.

Члены istream

Как и для ostream, большинство функций форматирования и управления вводом находится не в классе iostream, а в базовом классе ios.

```
class istream : public virtual ios {
   //...
public:
   int        peek()
   istream&   putback(char c);
   istream&   seekg(streampos);
   istream&   seekg(streamoff, seek_dir);
   streampos  tellg();
   //...
};
```

Функции позиционирования работают как и их двойники из ostream.Окончание на букву g показывает, что именно позиция используется при вводе символов из заданного потока. Буквы p и g нужны, поскольку мы можем создать производный класс iostreams из классов ostream иistream, и в нем необходимо следить за позициями ввода и вывода.

С помощью функции peek() программа может узнать следующий символ, подлежащий вводу, не затрагивая результата последующего чтения.

Порядок выполнения работы

1. Получить задание у преподавателя.
2. Разработать алгоритм решения задачи и написать программу, реализующую задание.
3. Проверить правильность ее работы.
4. Составить отчет и защитить работу.

Варианты заданий

1. Построить класс complex . Переопределить операторы ввода-вывода.
2. Построить класс matrix. Переопределить операторы ввода-вывода.
3. Построить класс vector . Переопределить операторы ввода-вывода.
4. Построить класс line . Переопределить операторы ввода-вывода.
5. Построить класс ellips . Переопределить операторы ввода-вывода.
6. Построить класс poligon . Переопределить операторы ввода-вывода.
7. Построить класс list . Переопределить операторы ввода-вывода.
8. Построить класс queue. Переопределить операторы ввода-вывода.
9. Построить класс tree . Переопределить операторы ввода-вывода.
10. Построить класс graph . Переопределить операторы ввода-вывода.
11. Построить класс polinom . Переопределить операторы ввода-вывода.
12. Построить класс string . Переопределить операторы ввода-вывода.

Лабораторная работа № 13
Классы языка C++

Цель работы: ознакомление с основными концепциями объектно-ориентированного программирования; изучение классов языка C++, способов их описания и использования, получение представления о перегрузке операторов и функций; получение навыков применения объектов в прикладных программах.

Краткие теоретические сведения

Объектно-ориентированный подход впитал в себя лучшие идеи структурированного и комбинирует их с новыми мощными концепциями, позволяющими увидеть задачу программирования в новом свете. Объектно-ориентированное программирование позволяет легко разложить сложную задачу на подзадачи, взаимодействующие друг с другом. Затем можно преобразовать эти подзадачи в единицы, называемые объектами. Все объектно-ориентированные языки имеют три общие концепции: инкапсуляцию, полиморфизм и наследование.

Инкапсуляция представляет собой механизм, который связывает вместе код и данные и который хранит их от внешнего воздействия и от неправильного использования. Более того, именно инкапсуляция позволяет создавать объекты. Попросту говоря, объект представляет собой логическое целое, включающее в себя данные и код для работы с этими данными. Мы можем определить часть кода и данных как собственность объекта, которая недоступна извне. На этом пути объект обеспечивает существенную защиту против случайной модификации или некорректного использования таких своих частных (*private*) членов. Во всех случаях объект представляет собой переменную, тип которой определяется пользователем. На первый взгляд может показаться странным представлять себе объект, который соединяет в себе вместе код и данные, как переменную. Тем не менее, в объектном программировании дело обстоит именно так. Когда создается объект, неявным образом создается новый тип переменной.

Объектно-ориентированные языки программирования поддерживают полиморфизм, который можно охарактеризовать следующей фразой: «один интерфейс – множество методов». Т.е. полиморфизм представляет собой механизм, который позволяет использовать один и тот же интерфейс при реализации целого набора различных действий. Выбор того, какое именно действие будет совершено, определяется конкретной ситуацией. Например, рассмотрим пример программы, которая определяет три различных типа списков. Один из них используется для целых чисел, другой – для символов, третий – для значений с плавающей запятой. Благодаря полиморфизму, можно создать три набора функций, имеющих одинаковое имя *push()* (поместить) и *pop()* (извлечь) – по одной на каждый тип данных. Общая концепция (интерфейс) заключается в том, чтобы вставлять и извлекать данные в список и из списка. Функции определяют специфические способы (методы), с помощью которых эти операции выполняются для каждого типа данных. Когда информация вставляется в список, автоматически вызывается та версия функции *push(),* которая соответствует типу обрабатываемых данных. Задача выбора специфического действия, т.е. метода, в зависимости от конкретной ситуации возлагается на компилятор.

Наследование представляет собой процесс, благодаря которому один объект может наследовать, приобретать свойства другого объекта. Это свойство поддерживает концепцию классификации, чем и обуславливается его важность. Например, красное яблоко представляет собой часть класса яблоко, который, в свою очередь, представляет собой часть класса фрукт, входящий в больший класс продукты питания. Без использования классификации каждый объект должен был бы определять все свои характеристики явным образом. На основе классификации объект нуждается только в определении таких качеств, которые отличают его от других объектов этого класса. Благодаря механизму наследования объект может характеризоваться в рамках классификации общего и частного.

Механизм объектов реализуется в языке программирования C++ с помощью классов. Общий вид описания класса следующий:

```
class <имя класса> {
    private:
        <частные данные и функции>;
protected:
        <защищенные данные и функции>;
    public:
        <публичные данные и функции>;
} <список объектов>;
```

Класс может содержать как частные (*private*) и защищенные (*protected*), так и публичные (*public*) данные. По умолчанию все члены класса являются частными, т.е. доступ к таким данным и функциям имеют только члены данного класса. Публичные элементы доступны для обращения из других мест программы. Защищенные члены аналогичны частным, отличия касаются только механизмов их наследования, которые будут рассматриваться в следующей лабораторной работе.

Как только определен класс, можно создать объект этого типа, используя имя класса. Фактически имя класса становится спецификатором нового типа данных. Рассмотрим, например, класс, описывающий окружность:

```
class circle {
        int x; // координаты центра
        int y;
        int r; // значение радиуса
    public:
        float get_length(void);
        float get_square(void);
        void set_params(int c_x, int c_y, int c_r);
        friend void func(void);
}
circle one, *two;
// создаем объект окружность one и указатель на объект two
```

Сравните: *int a, *b;*

Внутри объявления класса используются прототипы функций. Для написания реального кода функции-члена компилятору необходимо указать, к какому именно классу относится данная функция:

```
float circle::get_length(void)
{
   return 2*3.1415*r
}
```

```
float circle:: get_square(void)
{
   return 3.1415*r*r;
}
```

```
void circle:: set_params(int c_x, int c_y, int c_r)
{  x = c_x, y = c_y, r = c_r;
}
```

Последовательность символов :: называется оператором области видимости. Он показывает принадлежность функции конкретному классу. В C++ несколько различных классов могут использовать одинаковые имена функций. Компилятор понимает, какая из них принадлежит какому классу благодаря оператору области видимости и имени класса.

Функция, не являющаяся членом класса, может иметь доступ к его частным членам в случае, если она объявлена дружественной (*friend*) этому классу (см. пример выше). Одна из причин, почему C++ допускает существование таких функций, связана с той ситуацией, когда два класса для определенной цели должны использовать одну и ту же функцию.

Перед использовании объекта может потребоваться инициализировать некоторые его данные. Поскольку требования инициализации являются весьма распространенными, то C++ позволяет производить инициализацию объектов во время их создания. Такая автоматическая инициализация выполняется с помощью функции, называемой конструктором класса. Функция конструктор, являющаяся членом класса и имеющая имя, совпадающее с именем класса, представляет собой специальный тип функции. Конструктор

объекта вызывается автоматически при создании объекта. При инициализации глобальных или статических объектов конструктор вызывается только один раз, для локальных объектов конструктор вызывается каждый раз, когда встречается объявление объекта. Следует иметь в виду, что в C++ конструкторы не могут возвращать значений. Теперь в классе circle функция set_params() может быть заменена конструктором:

```
class circle {
        int x; // координаты центра
        int y;
        int r; // значение радиуса
    public:
        circle(int c_x, int c_y, int c_r);
        ~circle();
        float get_length(void);
        float get_square(void);
        friend void func(void);
}
circle::circle(int c_x, int c_y, int c_r)
  {
        x = c_x; y = c_y; r = c_r;
  }
```

Для передачи аргументов конструктору необходимо задать его значение при объявлении объекта. C++ поддерживает два способа решения этой задачи. В первом вызывается непосредственно с передачей ему значений переменных:

```
circle a = circle(1, 2, 3);
```

Во втором способе аргументы следуют непосредственно за объектом в круглых скобках:

```
circle a(1, 2, 3);
```

Дополнением конструктора является деструктор. Во многих случаях перед уничтожением объекта необходимо выполнить определенные действия. Локальные объекты создаются при входе в блок кода и уничтожаются при выходе из него. Глобальные объекты уничтожаются при завершении работы программы. Имеется много причин, чтобы существовал деструктор. Например, может потребоваться освободить память, которая была ранее зарезервирована. В C++ за дезактивацию отвечает деструктор. Он должен иметь то же имя, что и конструктор только к нему добавляется значок ~: *~circle()*.

Если рассматривать часть программы, которая не входит в состав класса, то для доступа к его членам необходимо использовать следующую конструкцию:

<имя объекта>.<имя члена класса>, если мы работаем непосредственно с объектом или

<имя указателя>-><имя члена класса>, если мы работаем с указателем на объект. Например:

```
circle one, *two;
float a, b;
a = one.get_length();
b = two->get_square();
```

Напомним, что созданные объекты в C++ являются обычными переменными, поэтому они могут передаваться в функции в качестве аргументов, а также служить типом возвращаемого функцией значения. Помимо этого, возможно создавать массивы объектов и получать указатели на объекты:

```
circle function(circle a, circle* b)
void main(void) {
circle one, *two, three[10], *four, *five, six;
```

```
two = &one; four = three; five = &three[2];
six = function(one, five);
}
```

Для реализации полиморфизма в C++ существует механизм, называемый перегрузкой функций. Смысл его заключается в том, что две и более функции могут иметь одинаковое имя, если они отличаются набором параметров в интерфейсе. Компилятор определяет, какую функцию можно использовать в конкретной ситуации, благодаря типу аргумента. По существу перегрузка функций позволяет создавать единое имя для операции, а компилятор устанавливает, какую именно функцию следует использовать в конкретной ситуации для выполнения данной операции. Например, опишем функции для ввода символов, целых и вещественных чисел, которые могут выдавать подсказку пользователю:

```
#include<iostream.h>
void promt(char *str, char *ch)
    {
        cout<<str;
        cin>>*ch;
    }
void promt(char *str, int *i)
    {
        cout<<str;
        cin>>*i;
    }
void promt(char *str; float *f);
    {
        cout<<str;
        cin>>*f;
    }
void main(void)
    {
        char a; int b; float c;
        promt(“Enter a char: ”, &a);
        promt(“Enter an integer: ”, &b);
        promt(“Enter a float: ”, &c);
        cout<<a<<” “<<b<<” “<<c;
    }
```

Другим способом реализации полиморфизма в языке C++ служит перегрузка операторов. Например, можно перегрузить операторы +, = и т.д. для выполнения специфичных действий. В общем случае можно перегружать операторы C++, определяя, что они означают применительно к определенному классу. Можно перегрузить оператор + по отношению к объектам типа <множество> таким образом, что он производит объединение двух произвольных множеств. Другой класс может использовать + совершенно другим образом. Например, для добавления элементов в список. Для того, чтобы перегрузить оператор, необходимо определить, что именно означает оператор по отношению к тому классу, к которому он применяется. Для этого определяется функция-оператор, задающая действие оператора. Общая форма записи функции оператора для случая, когда она является членом класса, имеет вид:

```
<тип> <имя класса>::operator#(<список аргументов>)
    {
    <действия, выполняемые оператором>;
    }
```

Здесь перегружаемый оператор подставляется вместо символа #, а тип задает тип значений возвращаемых оператором. Для того, чтобы упростить использование перегруженного оператора в сложных выражениях, в качестве возвращаемого значения часто выбирают тот же самый тип, что и класс, для которого перегружен оператор. Например, перегрузим операторы + и = для класса *circle*.

```
class circle {
        int x; // координаты центра
        int y;
        int r; // значение радиуса
    public:
        circle(int c_x, int c_y, int c_r);
        ~circle();
        circle operator+(circle);
        circle operator=(circle);
}
circle::circle(int c_x, int c_y, int c_r)
  {
        x = c_x; y = c_y; r = c_r;
  }
circle circle::operator+(circle a)
// смысл оператора определяет сами – среднее //арифметическое двух окружностей
    {
        circle temp;
        temp.x = (x+a.x)/2;
        temp.y = (y + a.y)/2;
        temp.r = (r + a.r)/2;
        return temp;
    }
circle circle::operator=(circle a)
    {
        x = a.x; y = a.y; r = a.r;
        return *this;
    }
void main(void)
    {
        circle one(1, 1, 1), two(2, 2, 2), three(3, 3, 3);
        one = two + three;
    }
```

Необходимо отметить два момента. Во-первых, обе функции-оператора имеют по одному параметру, хотя и перегружают бинарные операторы + и =. Во-вторых, мы использовали ключевое слово языка C++ *this*. Всякий раз, когда вызывается функция-член класса, в нее неявно передается указатель на объект, вызывающий данную функцию. Чтобы получить доступ к этому указателю, и используется *this*.

Теперь вернемся к рассматриваемому примеру. Во всех случаях именно объект, стоящий слева от знака операции вызывает функцию-оператор. К нему можно обратиться через *this*. Объект, стоящий справа от знака операции, передается функции как аргумент. Поэтому строка *x = a.x* эквивалентна *this->x = a.x*, а *this* в операторе присваивания также используется для возвращения значения.

При перегрузке унарных операций функции-операторы не будут иметь параметров, а при перегрузке бинарных операторов, функции-операторы имеют по одному аргументу.

Порядок выполнения работы

1. Получить задание у преподавателя.
2. Разработать алгоритм решения задачи и написать программу, реализующую задание.
3. Проверить правильность ее работы.
4. Составить отчет и защитить работу.

Варианты заданий

1. Разработать класс, позволяющий выполнять действия над матрицами произвольных размерностей (сложение, вычитание, умножение, транспонирование, нахождение определителя). Интерфейс класса должен включать перегруженные операторы. Предусмотреть возможность загрузки данных из файла.

2. Разработать класс, позволяющий выполнять действия над векторами произвольной длины (сложение, вычитание, скалярное и векторное умножение, нахождение модуля и направляющих косинусов). Интерфейс класса должен включать перегруженные операторы. Предусмотреть возможность загрузки данных из файла.

3. Разработать класс, позволяющий оперировать с комплексными числами (сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, перевод из одной формы записи в другую). Интерфейс класса должен включать перегруженные операторы. Предусмотреть возможность загрузки данных из файла.

4. Разработать класс, позволяющий оперировать с числами произвольной длины (сложение, вычитание, умножение). Интерфейс класса должен включать перегруженные операторы. Длинными называются числа, для операций с которыми нельзя использовать стандартные типы данных. Например, 1234567890987654321. Предусмотреть возможность загрузки данных из файла.

5. Разработать класс, позволяющий оперировать с множествами целых чисел (находить объединение, пересечение, инверсию и мощность множеств). Интерфейс класса должен включать перегруженные операторы. Предусмотреть возможность загрузки данных из файла.

6. Разработать класс, позволяющий хранить и обрабатывать информацию о студентах группы (добавление информации, сортировку по различным критериям, поиск по различным критериям, вывод информации об успеваемости, выбор всех отличников, хорошистов, троечников, выбор всех студентов, имеющих средний балл не ниже заданного). Интерфейс класса должен включать перегруженные операторы. Предусмотреть возможность загрузки данных из файла.

7. Разработать класс для хранения текста произвольного объема. Предусмотреть ввод и поиск информации, добавление и удаление данных в произвольное место текста, объединение нескольких текстов в один и разбиение текста на части. Интерфейс класса должен включать перегруженные операторы. Предусмотреть возможность загрузки данных из файла.

Контрольные вопросы

1. В чем заключается отличие объектно-ориентированного программирования от структурированного? Перечислите достоинства и недостатки объектно-ориентированного подхода.
2. В чем заключается смысл инкапсуляции?
3. В чем заключается смысл полиморфизма?
4. В чем заключается смысл наследования?
5. Классы в C++. Их объявление, описание и использование.
6. Что такое конструкторы и деструкторы? Для чего они используются?
7. Каким образом можно передавать объекты в функции? Может ли функция возвращать объекты?
8. Что такое перегрузка функций? Для чего она используется?
9. Что такое перегрузка операторов? Для чего она используется?

10. Что означает ключевое слово *this* в тексте программы на C++?

11. Объясните работу перегруженного оператора присваивания в примере, предложенном в теоретических положениях. Почему он возвращает значение?

Лабораторная работа № 14
Массивы и указатели в программах на языке С++

Цель работы: Познакомиться с принципами использования массивов, указателей и адресной арифметики в программах на языках С и С++.

Краткие теоретические сведения

В языке С между указателями и массивами существует тесная связь. Например, когда объявляется массив в виде int array[25], то этим определяется не только выделение памяти для двадцати пяти элементов массива, но и для указателя с именем array. Значение array равно адресу первого по счету (нулевого) элемента массива, т.е. сам массив остается безымянным, а доступ к элементам массива осуществляется через указатель с именем array. С точки зрения синтаксиса языка указатель array является константой, значение которой можно использовать в выражениях, но изменить это значение нельзя.

Поскольку имя массива является указателем, допустимо, например, такое присваивание:

```
int array[25];
int *ptr;
ptr = array;
```

Здесь указатель ptr устанавливается на адрес первого элемента массива, причем присваивание ptr = array можно записать в эквивалентной форме ptr = &array[0].

Для доступа к элементам массива существует два способа. Первый способ связан с использованием обычных индексных выражений в квадратных скобках, например, array[16] = 3 или array[i+2] = 7. При таком способе доступа записываются два выражения, причем второе выражение заключается в квадратные скобки. Одно из этих выражений должно быть указателем, а второе - выражением целого типа. Последовательность записи этих выражений может быть любой, но в квадратных скобках записывается выражение следующее вторым. Поэтому записи array[16] и 16[array] будут эквивалентными и обозначают элемент массива с номером шестнадцать. Указатель используемый в индексном выражении не обязательно должен быть константой, указывающей на какой-либо массив, это может быть и переменная. В частности после выполнения присваивания ptr=array доступ к шестнадцатому элементу массива можно получить с помощью указателя ptr в форме ptr[16] или 16[ptr].

Второй способ доступа к элементам массива связан с использованием адресных выражений и операции разадресации в форме *(array+16)=3 или *(array+i+2)=7. При таком способе доступа адресное выражение равное адресу шестнадцатого элемента массива тоже может быть записано разными способами *(array+16) или *(16+array).

При реализации на компьютере первый способ приводится ко второму, т.е. индексное выражение преобразуется к адресному. Для приведенных примеров array[16] и 16[array] преобразуются в *(array+16).

Для доступа к начальному элементу массива (т.е. к элементу с нулевым индексом) можно использовать просто значение указателя array или ptr. Любое из присваиваний

```
*array = 2;
array[0] = 2;
*(array+0) = 2;
*ptr = 2;
ptr[0] = 2;
*(ptr+0) = 2;
```

присваивает начальному элементу массива значение 2, но быстрее всего выполнятся присваивания *array=2 и *ptr=2, так как в них не требуется выполнять операции сложения.

Указатели на многомерные массивы в языке С - это массивы массивов, т.е. такие массивы, элементами которых являются массивы. При объявлении таких массивов в памяти компьютера создается несколько различных объектов. Например при выполнении объявления двумерного массива int arr2[4][3] в памяти выделяется участок для хранения

значения переменной arr, которая является указателем на массив из четырех указателей. Для этого массива из четырех указателей тоже выделяется память. Каждый из этих четырех указателей содержит адрес массива из трех элементов типа int, и, следовательно, в памяти компьютера выделяется четыре участка для хранения четырех массивов чисел типа int, каждый из которых состоит из трех элементов. Такое выделение памяти показано на схеме на рис. 1.

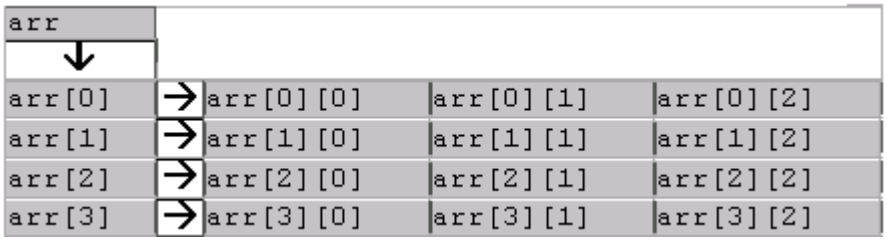


Рис. 1. Распределение памяти для двумерного массива

Таким образом, объявление arr2[4][3] порождает в программе три разных объекта: указатель с идентификатором arr, безымянный массив из четырех указателей и безымянный массив из двенадцати чисел типа int. Для доступа к безымянным массивам используются адресные выражения с указателем arr. Доступ к элементам массива указателей осуществляется с указанием одного индексного выражения в форме arr2[2] или *(arr2+2). Для доступа к элементам двумерного массива чисел типа int должны быть использованы два индексных выражения в форме arr2[1][2] или эквивалентных ей *(*(arr2+1)+2) и (*(arr2+1))[2]. Следует учитывать, что с точки зрения синтаксиса языка С указатель arr и указатели arr[0], arr[1], arr[2], arr[3] являются константами и их значения нельзя изменять во время выполнения программы.

Размещение трехмерного массива происходит аналогично и объявление float arr3[3][4][5] порождает в программе кроме самого трехмерного массива из шестидесяти чисел типа float массив из четырех указателей на тип float, массив из трех указателей на массив указателей на float, и указатель на массив массивов указателей на float.

При размещении элементов многомерных массивов они располагаются в памяти подряд по строкам, т.е. быстрее всего изменяется последний индекс, а медленнее - первый. Такой порядок дает возможность обращаться к любому элементу многомерного массива, используя адрес его начального элемента и только одно индексное выражение.

Например, обращение к элементу arr2[1][2] можно осуществить с помощью указателя ptr2, объявленного в форме int *ptr2=arr2[0] как обращение ptr2[1*4+2] (здесь 1 и 2 это индексы используемого элемента, а 4 это число элементов в строке) или как ptr2[6]. Заметим, что внешне похожее обращение arr2[6] выполнить невозможно так как указателя с индексом 6 не существует.

Для обращения к элементу arr3[2][3][4] из трехмерного массива тоже можно использовать указатель, описанный как float *ptr3=arr3[0][0] с одним индексным выражением в форме ptr3[3*2+4*3+4] или ptr3[22].

Над указателями можно выполнять унарные операции: инкремент и декремент. При выполнении операций ++ и -- значение указателя увеличивается или уменьшается на длину типа, на который ссылается используемый указатель.

Пример:

```
int *ptr, a[10];
ptr=&a[5];
ptr++;      /* равно адресу элемента a[6] */
ptr--;      /* равно адресу элемента a[5] */
```

В бинарных операциях сложения и вычитания могут участвовать указатель и величина типа int. При этом результатом операции будет указатель на исходный тип, а его значение будет на указанное число элементов больше или меньше исходного.

Управление динамической памятью с помощью new и delete

При динамическом распределении памяти для массивов следует описать соответствующий указатель и присваивать ему значение при помощи функции new.

Для создания двумерного массива нужно распределить память для массива указателей на одномерные массивы, а затем распределять память для одномерных массивов.

В C++ выделение динамической памяти для массивов и создаваемых динамически объектов осуществляется с помощью оператора **new**:

```
int n = 10;
int *array = new int[n]; // память под 10 элементов
if(array == NULL)
        // обработка ошибки выделения памяти
```

Во второй строке описан указатель на целую величину, которому присваивается адрес начала непрерывной области динамической памяти, выделенной с помощью оператора new. Выделяется столько памяти, сколько необходимо для хранения n величин типа int. При этом величина n может быть переменной.

Следует помнить, что обнуления памяти при выделении не происходит, и инициализировать динамический массив нельзя. Таким образом, после выделения он окажется заполненным случайным «мусором».

Если динамический массив в какой-то момент перестает быть нужным, то память необходимо освободить с помощью оператора **delete[]:**

```
delete[] array;
```

Размерность массива при этом не указывается.

В данном случае использование квадратных скобок обязательно, так как оператор delete без квадратных скобок используется для освобождения памяти, используемой единичными динамически создаваемыми объектами, что может привести к неопределенному поведению программы.

Следует помнить, что локальная переменная при выходе из блока, в котором она описана, «теряется». Если эта переменная является указателем, и в ней хранится адрес выделенной динамически памяти, при выходе из блока эта память перестает быть доступной, однако не помечается как свободная, поэтому не может быть использована в дальнейшем. Это называется утечкой памяти и является распространенной ошибкой:

```
{
        // пример утечки памяти
        int n; cin >> n;
        int *pmas = new int[n];
        ...
}       // после выхода из блока указатель pmas недоступен
...
```

Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант задания у преподавателя.
3. Выполнить задание.
4. Продемонстрировать выполнение работы преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу.

Варианты заданий

1. Дано целое число N (> 0). Сформировать и вывести целочисленный массив размера N, содержащий степени двойки от первой до N-й: 2, 4, 8, 16, … .

2. Дано целое число N (> 1), а также первый член A и разность D *арифме-тической прогрессии*. Сформировать и вывести массив размера N, содер-жащий N первых членов данной прогрессии:

$A, A + D, A + 2 \cdot D, A + 3 \cdot D, \ldots$.

3. Дано целое число N (> 1), а также первый член A и знаменатель Q *гео-метрической прогрессии*. Сформировать и вывести массив размера N, со-держащий N первых членов данной прогрессии:

$A, A \cdot D, A \cdot D^2, A \cdot D^3, \ldots$.

4. Дано целое число N (> 2). Сформировать и вывести целочисленный массив размера N, содержащий N первых элементов последовательности *чисел Фибоначчи* F_K:

$F_1 = 1, F_2 = 1, F_K = F_{K-2} + F_{K-1}, K = 3, 4, \ldots$.

5. Даны целые числа N (> 2), A и B. Сформировать и вывести целочисленный массив размера N, первый элемент которого равен A, второй равен B, а каждый последующий элемент равен сумме всех предыдущих.

6. Дан массив размера N. Вывести его элементы в обратном порядке.

7. Дан целочисленный массив размера N. Вывести все содержащиеся в данном массиве нечетные числа в порядке возрастания их индексов, а также их количество K.

8. Дан целочисленный массив размера N. Вывести все содержащиеся в данном массиве четные числа в порядке убывания их индексов, а также их количество K.

9. Дан целочисленный массив размера N. Вывести вначале все содержащиеся в данном массиве четные числа в порядке возрастания их индексов, а затем — все нечетные числа в порядке убывания их индексов.

10. Дан массив A размера N и целое число K ($1 \le K \le N$). Вывести элементы массива с порядковыми номерами, кратными K: A_K, $A_{2 \cdot K}$, $A_{3 \cdot K}$, … . Условный оператор не использовать.

11. Дан массив A ненулевых целых чисел размера 10. Вывести значение первого из тех его элементов A_K, которые удовлетворяют неравенству $A_K < A_{10}$. Если таких элементов нет, то вывести 0.

12. Дан массив размера N и целые числа K и L ($1 \le K \le L \le N$). Найти сумму элементов массива с номерами от K до L включительно.

13. Дан массив размера N и целые числа K и L ($1 < K \le L \le N$). Найти среднее арифметическое всех элементов массива, кроме элементов с номерами от K до L включительно.

14. Дан целочисленный массив размера N. Проверить, чередуются ли в нем четные и нечетные числа. Если чередуются, то вывести 0, если нет, то вывести порядковый номер первого элемента, нарушающего закономерность.

15. Дан массив A размера N. Найти минимальный элемент из его элементов с четными номерами: $A_2, A_4, A_6, \ldots$.

Контрольные вопросы

1. Какая связь между массивами и указателями в языке C++?
2. Каким образом можно инициализировать массив данными одновременно с его объявлением?
3. Каким образом динамически выделяется память под массив?
4. Какие арифметические операции допускается использовать с указателями?
5. Каким образом происходит работа с многомерными массивами?

Лабораторная работа № 15
Функции в программах на С++

Цель работы: Научиться решать задачи, используя рекурсивные функции.

Краткие теоретические сведения

Суть рекурсивных методов—сведение задачи к самой себе. Вы уже знаете, что в С существует возможность рекурсивного определения функций. Эта возможность представляет собой способ программной реализации рекурсивных алгоритмов. Однако увидеть рекурсивный путь решения задач и (рекурсивный алгоритм) часто очень непросто.

Классическая задача - «Ханойская башня». На площадке (назовем ее А) находится пирамида, составленная из дисков уменьшающегося от основания к вершине размера. Эту пирамиду в том же виде требуется переместить на площадку В. При выполнении этой работы необходимо соблюдать следующие ограничения:

•перекладывать можно только по одному диску, взятому сверху пирамиды;

•класть диск можно либо только на основание площадки, либо на диск большего размера;

•в качестве вспомогательной можно использовать площадку С.

Название «Ханойская башня» связано с легендой, согласно которой в давние времена монахи одного ханойского храма взялись переместить по этим правилам башню, состоящую из 64 дисков. С завершением их работы наступит конец света. Монах и все еще работают и, надеемся, еще долго будут работать!

Нетрудно решить эту задачу для двух дисков. Обозначая перемещения диска, например, с площадки А на В так: А—>В, напишем алгоритм для этого случая

A-)C;A->B;C->B.

Функция Сложения двух натуральных чисел ($Sum(a,\ b) = a + b$) может быть рассмотрена в качестве рекурсивной функции двух переменных

$Sum(x,\ y+1) = F(x,\ y,\ Sum(x,\ y))$;

Умножение двух натуральных чисел ($Mul(a,\ b) = a \times b$) может быть рассмотрено в качестве рекурсивной функции двух переменных

$Mul(x,\ y+1) = G(x,\ y,\ Mul(x,y))$;

Рекурсивной называется функция, которая вызывает саму себя. Такая рекурсия называется прямой. При косвенной рекурсии, две или более функций вызывают друг друга. При рекурсивном вызове функции в стеке создается копия значений ее параметров, как и при вызове обычной функции, после чего управление передается первому исполняемому оператору функции. При повторном вызове этот процесс повторяется. Ясно, что для завершения вычислений каждая рекурсивная функция должна содержать хотя бы одну нерекурсивную ветвь алгоритма, заканчивающуюся оператором возврата.

При завершении функции соответствующая часть стека освобождается, и управление передается вызывающей функции, выполнение которой продолжается с точки, следующей за рекурсивным вызовом.

Приведем пример рекурсивной функции, вычисляющей факториал.

```
long fact(long n){
if (n==0 || n==1) return 1;
return (n * fact(n - 1);
}
```

Содержание работы.

1. Ознакомиться с теоретической частью методических указаний.
2. Получить вариант задания у преподавателя.
3. Решить поставленную задачу с использованием языка программирования С++.
4. Представить работающую программу преподавателю.

Варианты заданий

Решить следующие задачи, используя рекурсивную функции.

1. Найти сумму цифр заданного натурального числа.
2. Подсчитать количество цифр в заданном натуральном числе.
3. Составить программу для вычисления наибольшего общего делителя двух натуральных чисел.
4. Составить программу для нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке. Например, для числа1234 получаем результат 4321.
5. Составить программу для перевода данного натурального числа в 7 -ричную систему счисления.
6. Дана символьная строка, представляющая собой запись натурального числа в 7-ичнойсистемесчисления(2<p<9).Составить программу для перевода этого числа в десятичную систему счисления.
7. Составить программу для вычисления суммы: 2! + 4! + 6! +

+ ... + «! (п< 16, п — четное).

Примечание. Тип результата значения функции — Longlnt.

8. Дано п различных натуральных чисел. Напечатать все перестановки этих чисел.
9. Логическая функция возвращает True, если ее аргумент — простое число.
10. Описать функцию, которая удаляет из строки все лишние пробелы. Пробелы считаются лишними, если их подряд идет более двух, если они стоят в конце строки после последней точки, если стоят после открывающегося парного знака препинания.

Содержание отчета.

4. Цель работы и задание.
5. Пункты, соответствующие порядку выполнения работы.
6. Выводы по работе.

Лабораторная работа № 16
Наследование классов в С++

Цель работы: Ознакомление с механизмом наследования классов и возможностями, которые он предоставляет; получение навыков использования наследования в прикладных программах.

Краткие теоретические сведения

Наследование и полиморфизм являются одними из основных элементов объектно-ориентированного подхода в программировании. С помощью наследования можно создать общий класс, определяющий общие черты совокупности объектов. Этот класс может наследоваться другими более специфическими классами, каждый из которых может добавить свои элементы в описании объекта. Есть еще одна причина, по которой наследованию придается большая важность: с его помощью поддерживается полиморфизм во время выполнения программы. В С++ полиморфизм поддерживается как на этапе выполнения программы, так и на этапе ее компиляции. В качестве примера полиморфизма на этапе компиляции можно указать перегрузку операторов и функций. Но, к сожалению, при ее использовании нельзя решить всех задач, возникающих в объектно-ориентированном программировании. Поэтому в С++ используется также полиморфизм времени исполнения программы, для чего используются производные классы и виртуальные функции.

Класс, который наследуется другим классом, называется базовым классом. Иногда его также называют родительским классом. Класс, выполняющий наследование, называется производным классом или потомком.

Когда один класс наследует другой, то все публичные члены базового класса доступны в производном классе. В противоположность этому частные члены базового класса не доступны внутри производного класса. Если необходимо оставить член частным и вместе с тем позволить использовать его производным классам, то для этих целей и используется ключевое слово *protected*. Защищенный член подобен частному за исключением механизма наследования. При наследовании защищенного члена производный класс также имеет к нему доступ. Таким образом, указав спецификатор доступа *protected*, можно позволить использовать член внутри иерархии классов и запретить доступ к нему извне этой иерархии. Общая форма наследования классов имеет следующий вид:

```
class <имя производного класса>: <доступ> <имя базового класса> {
……
<описание производного класса>;
……
};
```

Здесь доступ определяет, каким способом наследуется базовый класс. Спецификатор доступа может принимать три значения: *private*, *public* и *protected*. В случае, если спецификатор доступа опущен, то по умолчанию подразумевается на его месте спецификатор *public*. При этом все публичные и защищенные члены базового класса становятся соответственно публичными и защищенными членами производного класса. Если спецификатор доступа имеет значение *private*, то все публичные и защищенные члены базового класса становятся частными членами производного класса. И, наконец, если спецификатор доступа принимает значение *protected*, то все публичные и защищенные члены базового класса становятся защищенными членами производного класса. Например:

```
class X {
        protected:
                int i, j;
        public:
                void get_ij() {cin>>i>>j;}
                void put_ij() {cout<<i<<” “<<j<<endl;}
```

```
};
// В классе Y i и j класса X становятся частными членами
class Y: private X {
        protected:
                int k;
        public:
                int get_k() {return k;}
                void make_k() {k = i * j;}
};
/* Класс Z имеет доступ к k класса Y, т.к. он защищенный, но не имеет доступа к i и
j, т.к. они частные */
class Z : public Y {
public:
                void f();
};
void Z::f()
        {
        k = 5; // корректно
                i = j = 10; // не корректно, т.к. нет доступа
        }
```

При использовании производных классов важно представлять себе, каким образом и когда выполняются конструкторы и деструкторы базового и производного классов. Конструкторы вызываются в том же самом порядке, в каком классы следуют друг за другом в иерархии классов. Поскольку базовый класс ничего не знает про свои производные классы, то его инициализация может быть отделена от инициализации производных классов и производится до их создания, так что конструктор базового класса вызывается перед вызовом конструктора производного класса. В противоположность этому деструктор производного класса вызывается перед деструктором базового класса.

Когда базовый класс имеет конструктор с аргументами, производные классы должны явным образом обрабатывать эту ситуацию путем передачи базовому классу необходимых аргументов. Для этого используется расширенная форма конструкторов производных классов, в которые передаются аргументы конструкторам базовых классов:

<порожденный конструктор>(<список аргументов>): <базовый конструктор> (<список аргументов>)

```
        {
                ……
        }
```

Список аргументов, ассоциированный с базовым классом, может состоять из констант, глобальных переменных или параметров конструктора производного класса. Поскольку инициализация объекта происходит во время выполнения программы, можно использовать в качестве аргумента любой идентификатор, определенный в области видимости класса.

```
class X {
        protected:
                int a, b;
        public:
                X(int i, int j) {a = i; b = j;}
};
class Y : public X {
        private:
                int c;
        public:
```

```
        Z(int l, int m, int n): X(l, m) { c = n;}
}
void main(void)
    {
        Z ob(0,1,2);
    }
```

Также один класс может наследовать атрибуты двух и более классов одновременно. Для этого используется список базовых классов, в котором каждый из базовых классов отделен друг от друга запятой. Например:

```
class X {
    protected:
        int a;
public:
    void make_a(int p) { a = p; }
};
class Y {
    protected:
        int b;
public:
    void make_b(int q) { a = q; }
}
class Z: public X, public Y {
public:
    int make_ab() { return a*b; }
}
```

Поскольку класс *Z* наследует оба класса *X* и *Y*, то он имеет доступ к публичным и защищенным данным обоих классов. При множественном наследовании конструкторы базовых классов вызываются слева направо по порядку их следования в описании производного класса. Деструкторы же вызываются в обратном порядке – справа налево.

Имеется одна особенность использования указателей на классы при наследовании. В общем случае указатель одного типа не может указывать на объект другого типа. Из этого правила есть исключение, которое относится только к производным классам. В C++ указатель на базовый класс может указывать на объект производного класса, полученный из этого базового класса. Пусть имеем некий базовый класс *B_class* и его производный класс *D_class*. В C++ любой указатель типа *B_class** может также указывать на объект типа *D_class*. Например, если имеются следующие объявления переменных:

```
B_class *p;
B_class b_ob;
D_class d_ob;
```

то следующие присвоения абсолютно законны:

```
p = &b_ob;
p = &d_ob;
```

Используя указатель *p*, можно получить доступ ко всем членам *d_ob*, которые наследованы от *b_ob*. Однако специфические члены *d_ob* не могут быть получены с использованием указателя *p* (по крайней мере до тех пор, пока не будет осуществлено преобразование типов). Это является следствием того, что указатель «знает» только о членах базового типа и не знает ничего о специфических членах производных типов. Если необходимо получить доступ к элементам производного класса с помощью указателя, имеющего тип указателя на базовый класс, необходимо воспользоваться приведением типов. Например: *((D_class*)p)->function();*

Хотя указатель, имеющий тип указателя на базовый класс, может использоваться в качестве указателя на производный объект, обратное не имеет место. Это означает, что

указатель на производный класс не может использоваться для доступа к объектам базового типа.

Полиморфизм времени исполнения обеспечивается не только за счет использования производных классов, но и применением виртуальных функций. Виртуальная функция – это функция, объявленная с ключевым словом *virtua*l в базовом классе и переопределенная в одном или нескольких производных классах. Виртуальные функции являются особыми функциями, потому что при вызове объекта производного класса с помощью указателя или ссылки на него C++ генерирует код, который определяет во время исполнения программы, какую функцию вызвать, основываясь на типе объекта. Для разных объектов вызываются разные версии одной и той же виртуальной функции. Класс, содержащий одну или более виртуальных функций, называется полиморфным классом. Если функция объявлена как виртуальная, то она остается таковой вне зависимости от количества уровней в иерархии классов, через которые она прошла. Например, рассмотрим классы, содержащие информацию о геометрических фигурах:

```
class figure {
        protected:
                double x, y;
        public:
                void set(double a, double b) { x = a; y = b; }
                virtual double area() { cout<<”No area”; return 0; }
}
class triangle: public figure {
        public:
                double area() { return 0.5*x*y;}
}
class square: public figure {
        public:
                double area() { return x*y; }
}
class circle: public figure {
        protected:
                double r;
        public:
                circle(double a) { r = a;}
                double area() { return 3.14*r*r};
}
void main(void)
        {       double res;
                figure *p; // создание указателя базового типа
                triangle t;     square s;       circle c(10); // объекты производных типов
                p = &t;         p->set(1, 1);
                res = p->area(); // находим площадь треугольника
                p = &s;         p->set(2, 2);
                res = p->area(); // находим площадь четырехугольника
                p = &c;         p->set(20, 30);
                res = p->area(); // находим площадь окружности
        }
```

Ключевым моментом в использовании виртуальных функций для обеспечения полиморфизма времени исполнения служит то, что используется указатель именно на базовый класс. Полиморфизм времени исполнения достигается только при вызове виртуальной функции с использованием указателя или ссылки на базовый класс. Ничего не мешает использовать виртуальные функции, как и любые другие нормальные функции,

однако достичь полиморфизма времени исполнения на этом пути не удастся. Также стоит отметить, что виртуальным может быть и деструктор класса, хотя конструктор виртуальным быть не может.

Когда виртуальная функция не переопределена в производном классе, то при вызове ее в объекте производного класса вызывается версия из базового класса. Однако во многих случаях невозможно ввести содержательное определение виртуальной функции в базовом классе. Например, при объявлении класса *figure* в предыдущем примере реализация функции *area()* не несет никакого смысла. Могут быть также такие виртуальные функции, которые обязательно должны быть переопределены в производных классах, без чего эти классы не будут иметь никакого значения. В таких случаях необходим метод, гарантирующий, что производные классы действительно определят все необходимые функции. Язык C++ предлагает в качестве решения этой проблемы чисто виртуальные функции. Это такие функции, которые объявлены в базовом классе, но не имеют в нем определения. Т.к. они не имеют определений, т.е. тел в этом базовом классе, то всякий производный класс обязан иметь свою собственную версию реализации. Для объявления чистой виртуальной функции используется следующая общая форма: virtual <тип возвращаемого значения> <имя функции> (<список параметров>) = 0;

Например, для определения чисто виртуальной функции area() в классе figure необходимо написать следующее: *virtual double area() = 0;*

Если какой-либо класс имеет хотя бы одну чисто виртуальную функцию, то такой класс называется абстрактным. Важной особенностью абстрактных классов является то, что нельзя создать ни одного объекта данного класса. Вместо этого абстрактный класс служит в качестве базового для других производных классов. Причина, по которой абстрактный класс не может быть использован для объявления объектов, заключается в том, что одна или несколько его функций-членов не имеют определения. Тем не менее, даже если базовый класс является абстрактным, все равно можно объявлять указатели или ссылки на него, с помощью которых затем поддерживается полиморфизм времени исполнения.

Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант задания у преподавателя.
3. Выполнить задание.
4. Продемонстрировать выполнение работы преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу.

Варианты заданий

1. Разработать иерархию классов, которые позволяют оперировать с геометрическими фигурами: точка, линия, фигура, окружность, треугольник, четырехугольник, многоугольник, пирамида и призма. Реализация должна допускать создание объектов с различными параметрами, вычисление различных геометрических характеристик фигур, сопоставление одноименных фигур друг с другом. Интерфейс должен содержать виртуальные функции.

2. Разработать иерархию классов, которые позволяют создавать картотеку с информацией по предметам различного типа (книги, мебель, бытовая техника, компьютеры, продукты питания и т.д.). Реализация должна допускать добавление информации в картотеку, поиск информации, отбор по различным критериям, сортировку и хранение наиболее часто просматриваемых карточек. Интерфейс должен содержать виртуальные функции.

3. Разработать иерархию классов, которые позволяют обрабатывать информацию о работниках различного уровня на предприятии (уборщики, рабочие, мастера и инженеры различных подразделений, экономисты, дирекция и т.д.). Реализация должна допускать прием на работу и увольнение сотрудников, их перевод из одного подразделения в другое,

начисление ежемесячной зарплаты, учет отпусков, поиск и сортировку информации по различным критериям. Интерфейс должен содержать виртуальные функции.

4. Разработать иерархию классов, позволяющих моделировать работу системного блока персонального компьютера. Реализация должна позволять добавлять различные устройства в ПК, эмулировать их взаимодействие между собой, производить апгрейд аппаратуры. Интерфейс должен содержать виртуальные функции.

5. Разработать иерархию классов, описывающих существующее на рынке программное обеспечение. Реализация должна допускать классификацию ПО по различным критериям, добавление и удаление ПО в базу данных и из нее, выработку аргументированных (по каким критериям осуществлен выбор) рекомендаций пользователю по покупке программ для решения определенных задач. Интерфейс должен содержать виртуальные функции.

Контрольные вопросы

1. Что такое полиморфизм? Как он поддерживается в C++?
2. Что такое наследование классов? Для чего оно применяется?
3. В чем заключаются особенности описания конструкторов и деструкторов при наследовании? В каком порядке они вызываются при создании и удалении объектов производного типа?
4. В чем заключаются особенности использования указателей на базовые и производные классы?
5. Что такое виртуальные функции? Для чего они применяются?
6. Какие Вы знаете стандартные классы языка C++ ?

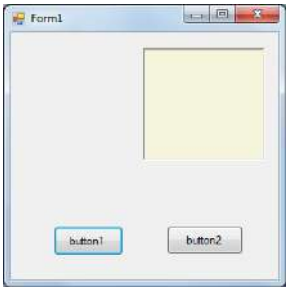
Лабораторная работа № 17
Работа с графикой в C++. Рисование графических примитивов
Краткие теоретические сведения

Необходимо спроектировать приложение, интерфейс которого включает форму, элемент вывода графики PictureBox, кнопки button1 и button2.

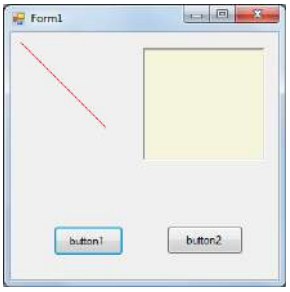
При клике по кнопке button1 на форме рисуется отрезок прямой красного цвета.

При клике по кнопке button2 в элементе PictureBox рисуется отрезок прямой синего цвета.

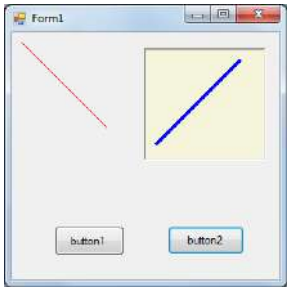
Разработаем интерфейс



а) после запуска



а) после нажатия button1



а) после нажатия button1

Порядок создания элементов управления

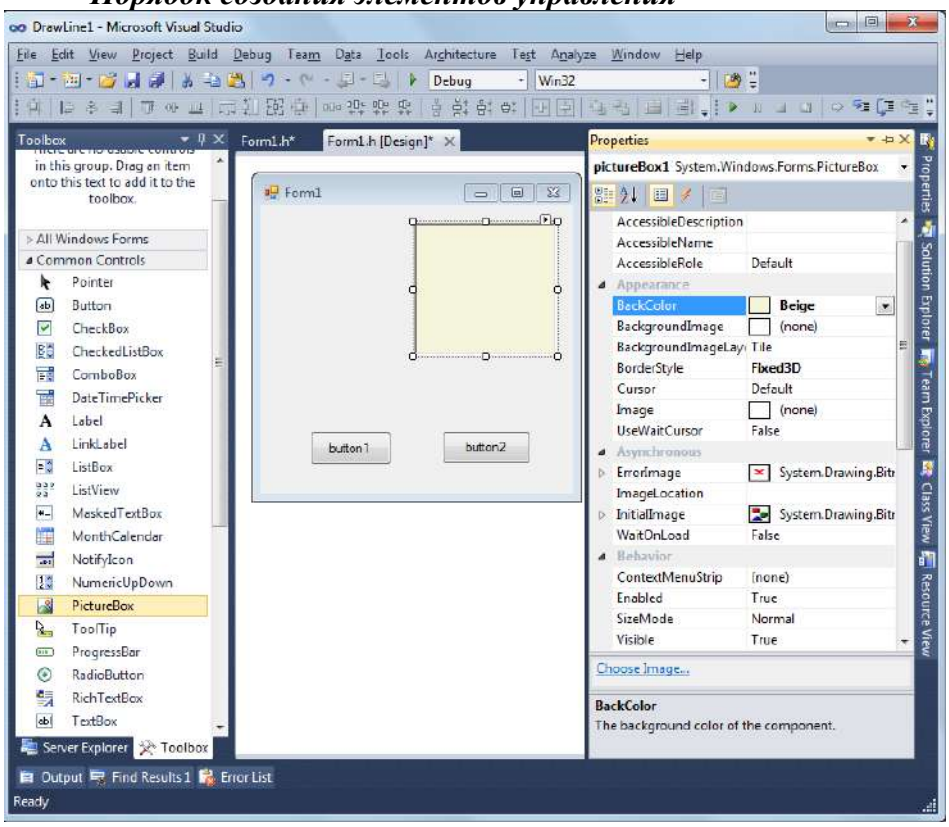


Рис. Окно дизайнера формы

Класс формы

```
#pragma once
namespace DrawLine1 {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
		Form1(void)
		{
			InitializeComponent();
			//TODO: Add the constructor code here
		}

	protected:
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:
	private: System::Windows::Forms::Button^  button2;
	private: System::Windows::Forms::PictureBox^  pictureBox1;

	private:
		System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
            void InitializeComponent(void)
            {
                  this->button1 = (gcnew System::Windows::Forms::Button());
                  this->button2 = (gcnew System::Windows::Forms::Button());
                  this->pictureBox1 = (gcnew
System::Windows::Forms::PictureBox());
                  (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->pictureBox1))->BeginInit();
                  this->SuspendLayout();
                  // button1
                  this->button1->Location = System::Drawing::Point(45, 205);
                  this->button1->Name = L"button1";
                  this->button1->Size = System::Drawing::Size(74, 31);
                  this->button1->TabIndex = 0;
                  this->button1->Text = L"button1";
                  this->button1->UseVisualStyleBackColor = true;
                  this->button1->Click += gcnew System::EventHandler(this,
&Form1::button1_Click);
                  // button2
                  this->button2->Location = System::Drawing::Point(165, 205);
                  this->button2->Name = L"button2";
                  this->button2->Size = System::Drawing::Size(81, 30);
                  this->button2->TabIndex = 1;
                  this->button2->Text = L"button2";
                  this->button2->UseVisualStyleBackColor = true;
                  this->button2->Click += gcnew System::EventHandler(this,
&Form1::button2_Click);
                  // pictureBox1
                  this->pictureBox1->BackColor = System::Drawing::Color::Beige;
                  this->pictureBox1->BorderStyle =
System::Windows::Forms::BorderStyle::Fixed3D;
                  this->pictureBox1->Location = System::Drawing::Point(140, 16);
                  this->pictureBox1->Name = L"pictureBox1";
                  this->pictureBox1->Size = System::Drawing::Size(129, 120);
                  this->pictureBox1->TabIndex = 2;
                  this->pictureBox1->TabStop = false;
                  // Form1
                  this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                  this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                  this->ClientSize = System::Drawing::Size(284, 262);
                  this->Controls->Add(this->pictureBox1);
                  this->Controls->Add(this->button2);
                  this->Controls->Add(this->button1);
                  this->Name = L"Form1";
                  this->Text = L"Form1";
                  (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->pictureBox1))->EndInit();
                  this->ResumeLayout(false);

            }
#pragma endregion
      private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
      Color ^col=gcnew Color;
      Pen ^pen= gcnew Pen(col->Red);
      Graphics ^im  =this->CreateGraphics();
      im->DrawLine(pen,10,10,100,100);
      }
      private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {
      Color ^col=gcnew Color;
      Pen ^pen= gcnew Pen(col->Blue,4);
      Graphics ^im  =pictureBox1->CreateGraphics();
      im->DrawLine(pen,10,100,100,10);
      }
```

```
};
}
```

Порядок выполнения работы

1. Получить задание у преподавателя.
2. Разработать алгоритм решения задачи и написать программу, реализующую задание.
3. Проверить правильность ее работы.
2. Составить отчет и защитить работу.

Варианты заданий

1. Спроектировать приложение для рисования отрезка прямой линии на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии.
2. Спроектировать приложение для рисования эллипса на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии.
3. Спроектировать приложение для рисования окружности на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии.
4. Спроектировать приложение для рисования сектора эллипса на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии.
5. Спроектировать приложение для рисования хорды эллипса на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии.
6. Спроектировать приложение для рисования под произвольным углом эллипса на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии, значение угла поворота.
7. Спроектировать приложение для рисования графика функции на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линии графика,таблица значений функции.
8. Спроектировать приложение для рисования изометрии пирамиды на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линий,таблица значений координат вершин пирамиды.
9. Спроектировать приложение для рисования изометрии параллелепипеда на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линий,таблица значений координат вершин параллелепипеда.
10. Спроектировать приложение для рисования изометрии параллелограмма в декартовой системе координат на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода. В поле ввода задаются параметры линий, таблица значений координат вершин параллелограмма.

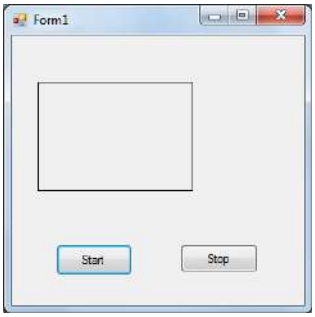
Лабораторная работа №18

Работа с графикой в C++. Рисование анимированных объектов

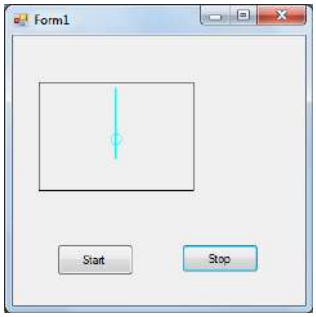
Краткие теоретические сведения

Спроектируем приложение, интерфейс которого включает форму, кнопку Start, кнопку Stop. При инициализации формы на ней рисуется прямоугольный контур таймера. При клике по кнопке Start в на форме рисуется секундная стрелка синим цветом каждую нечетную секунду и красным цветом – каждую четную.

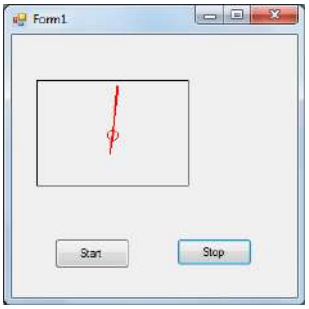
Разработка интерфейса



а) после запуска

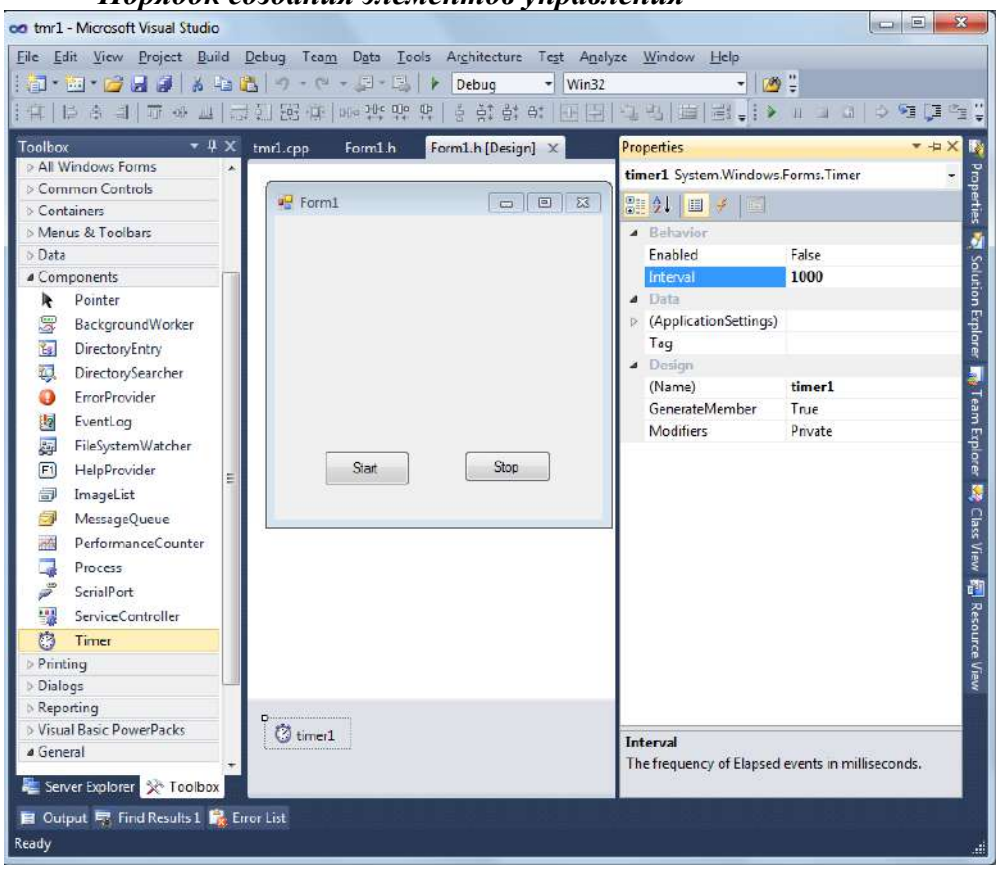


а) после нажатия start



а) после истеченияодной секунды

Порядок создания элементов управления



Класс формы

```
#pragma once
#include <math.h>
#include <stdlib.h>
using namespace std;
namespace tmr1 {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
	Form1(void)
	{
		InitializeComponent();
		//TODO: Add the constructor code here
		im=this->CreateGraphics();
		col=gcnew Color;
		pen =gcnew Pen(col->Red);
		// Build the rectangles from points and size
		Drawing::Point point1 = Drawing::Point(25,45);
		Drawing::Size size = Drawing::Size(150, 105);
		rect1 = Drawing::Rectangle(point1, size);
```

```
        }
        protected:
        ~Form1()
        {
                if (components)
                {
                delete components;
                }
        }
        private: System::Windows::Forms::Button^  button1;
        protected:
        private: System::Windows::Forms::Button^  button2;
        private: System::Windows::Forms::Timer^  timer1;
        private: System::ComponentModel::IContainer^  components;
        private:
         Color ^col;
         Graphics ^im;
         Pen ^ pen ;
         Drawing::Rectangle rect1;
#pragma region Windows Form Designer generated code
        void InitializeComponent(void)
        {
                this->components = (gcnew System::ComponentModel::Container());
                this->button1 = (gcnew System::Windows::Forms::Button());
                this->button2 = (gcnew System::Windows::Forms::Button());
                this->timer1 = (gcnew System::Windows::Forms::Timer(this->components));
                this->SuspendLayout();
                this->Paint +=
                gcnew System::Windows::Forms::PaintEventHandler(this,
                &Form1::Form1_Paint);
                // button1
                this->button1->Location = System::Drawing::Point(43, 203);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(74, 30);
                this->button1->TabIndex = 0;
                this->button1->Text = L"Start";
                this->button1->UseVisualStyleBackColor = true;
this->button1->Click += gcnewSystem::EventHandler(this,&Form1::button1_Click);
                // button2
                this->button2->Location = System::Drawing::Point(164, 203);
                this->button2->Name = L"button2";
                this->button2->Size = System::Drawing::Size(75, 27);
                this->button2->TabIndex = 1;
                this->button2->Text = L"Stop";
                this->button2->UseVisualStyleBackColor = true;
                this->button2->Click += gcnew System::EventHandler(this, &Form1::button2_Click);
                // timer1
                this->timer1->Interval = 1000;
                this->timer1->Tick += gcnew System::EventHandler(this, &Form1::timer1_Tick);
                // Form1
                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
```

```
            this->ClientSize = System::Drawing::Size(284, 262);
            this->Controls->Add(this->button2);
            this->Controls->Add(this->button1);
            this->Name = L"Form1";
            this->Text = L"Form1";
            this->ResumeLayout(false);

        }
#pragma endregion
//----------My code--------------
        private:
        System::Void Form1_Paint(System::Object^ sender,
System::Windows::Forms::PaintEventArgs^ e)
            {
            // Draw a rectangle
            e->Graphics->DrawRectangle(Pens::Black, rect1);
            }

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
            if(timer1->Enabled);else timer1->Enabled=true;
             im->DrawRectangle(Pens::Black, rect1);}
        private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
             if(!timer1->Enabled);else timer1->Enabled=false; }
        private: System::Void timer1_Tick(System::Object^  sender, System::EventArgs^  e) {
            static float x=100,y=100,fi=0,hfi=3.14159/30,fil,pi=3.14159 ,
            x1=100,y1=100,xl,yl,x1l,y1l,cx=100,
            cy=100,l1=20,l=50,ts=0;
            x1l=x1;y1l=y1;xl=x;yl=y;
        static int t;
        t++;
        y=100-l*sin(fi+pi/2);
        x=100-l*cos(fi+pi/2);
        x1=cx+l1*cos(fi+pi/2);
        y1=cy+l1*sin(fi+pi/2);
        fi=fi+hfi;
        pen->Color=this->BackColor;
        pen->Width=2;
        im->DrawLine(pen,int(xl),int(yl),100,100);
        im->DrawLine(pen,int(x1l),int(y1l),int(cx),int(cy));
        if(t%2)pen->Color=col->Cyan;else pen->Color=col->Red;
        im->DrawLine(pen,int(x1),int(y1),int(cx),int(cy));
        im->DrawLine(pen,int(x),int(y),100,100);
        im->DrawEllipse((pen->Width=1,pen),int(cx-5),int(cy-5),10,10);
             }
        };
}
```

Порядок выполнения работы

1. Получить задание у преподавателя.
2. Разработать алгоритм решения задачи и написать программу, реализующую задание.
3. Проверить правильность ее работы.

4. Составить отчет и защитить работу.

Варианты заданий

1. Спроектировать приложение для рисования изометрии отрезка прямой линии в трехмерной системе координат на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются также параметры линии.
2. Спроектировать приложение для рисования для рисования изометрии эллипса в трехмерной системе координат на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются также параметры линии.
3. Спроектировать приложение для рисования для рисования изометрии окружности в трехмерной системе координат на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются также параметры линии.
4. Спроектировать приложение для рисования для рисования изометрии сектора эллипса в трехмерной системе координат на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются также параметры линии.
5. Спроектировать приложение для рисования для рисования изометрии и хорды эллипса в трехмерной системе координат на форме. На форме предусмотреть кнопку для ввода данных,кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются также параметры линии.
6. Спроектировать приложение для рисования изометрии графика функции на форме. На форме предусмотреть кнопку для ввода данных, кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот плоскости графика функции осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются параметры линии графика,таблица значений функции.
7. Спроектировать приложение для рисования изометрии пирамиды на форме. На форме предусмотреть кнопку для ввода данных, кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот изображения объекта осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются координаты вершин пирамиды.
8. Спроектировать приложение для рисования изометрии параллелепипеда на форме. На форме предусмотреть кнопку для ввода данных, кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот изображения объекта осуществляется вокруг оси, параметры которой задаются в поле ввода. В поле ввода задаются параметры линий,таблица значений координат вершин параллелепипеда.
9. Спроектировать приложение для рисования изометрии параллелограмма в декартовой системе координат на форме. На форме предусмотреть кнопку для ввода данных, кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот изображения объекта осуществляется вокруг оси, параметры которой задаются в поле ввода В поле ввода задаются параметры линий, таблица значений координат вершин параллелограмма.
10. Спроектировать приложение для рисования изометрии ограниченного параболоида вращения в декартовой системе координат на форме. На форме

предусмотреть кнопку для ввода данных, кнопу для рисования, поле ввода, кнопку для однократного поворота объекта. Поворот изображения объекта осуществляется вокруг оси, параметры которой задаются в поле ввода В поле ввода задаются параметры линий, таблица значений параметров параболоида.

Минобрнауки России
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНО-ПРАКТИЧЕСКИХ РАБОТ

по дисциплине «**Численные методы»**

специальности 09.02.07 Информационные системы и программирование

Тула 2021

УТВЕРЖДЕНЫ

Цикловой комиссией информационных технологий

Протокол от «14» октября 2021 г. № 3

Председатель цикловой комиссии И.В. Миляева

Составитель Воронцова Н.В.

СОДЕРЖАНИЕ

Тема 1: Элементы теории погрешностей

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 1

Тема работы: Вычисление погрешности приближенного значения величины. Вычисление погрешности арифметических действий

Цель работы: *уметь:*

Вычислять относительную и абсолютною погрешности чисел, определять верные цифры чисел, уметь работать с табличном процессором и проводить расчеты и определять погрешность результата.

Материально-техническое оснащение: ПК, операционная система Windows
Количество часов: 2 часа.

I.Теоретическая часть

Различают два вида погрешностей – *абсолютную* и *относительную* погрешности.

Абсолютная погрешность некоторого числа равна разности между его истинным значением и приближённым значением, полученным в результате вычисления или измерения.

$$\Delta x = x - a,$$

где Δx – абсолютная погрешность;

x – истинное значение числа;

a – приближённое значение числа.

Относительная погрешность – это отношение абсолютной погрешности к приближённому значению числа.

$$\delta x = \Delta x / a,$$

где δx – относительная погрешность.

К сожалению, истинное значение величины x обычно неизвестно. Поэтому, приведённые выражения для погрешностей практически не могут быть использованы. На практике обычно известно приближённое значение a (т.е. снимаемое с измерительных приборов или датчиков). Встаёт задача определения *предельной погрешности* Δa, являющейся верхней оценкой модуля абсолютной погрешности Δx.

$$|\Delta x| \le \Delta a,$$

где Δa – абсолютная погрешность приближённого числа a (граница погрешности Δx).

В этом случае истинное значение x находится в интервале:

$$a - \Delta a \le x \le a + \Delta a.$$

Для приближённого числа, полученного в результате округления, абсолютная погрешность, Δa принимается равной половине единицы

последнего разряда числа. Например, значение - **a**=0.986 могло быть получено округлением чисел – 0.98644, 0.98555. При этом $|\Delta x| \le 0.0005$ и $\Delta a = 0.0005$.

На практике оперируют и с предельным значением относительной погрешности - **δa**.

$$\delta a = \Delta a / |a|,$$

где **δa** – предельное значение относительной погрешности (граница относительной погрешности).

Заметим, что погрешность всегда округляется в сторону увеличения. Например, $\delta \approx 0.022 \approx 0.03$. Значащую цифру называют верной, если абсолютная погрешность числа не превосходит единицы разряда, соответствующего этой цифре.

II. Практическая часть

Задача №1 1) Определить, какое равенство точнее.

2) Округлить сомнительные цифры числа, оставив верные знаки: а) в узком смысле; б) в широком смысле. Определить абсолютную погрешность результата.

3) Найти предельные абсолютные и относительные погрешности чисел, если они имеют только верные цифры: а)в узком смысле; б) в широком смысле.

Образец выполнения задания

1) $9/11 = 0{,}818$; $\sqrt{18} = 4{,}24$; 2) а) $72{,}353\ (\pm 0{,}026)$; б) $2{,}3544$; $\delta = 0{,}2\%$; 3) а) $0{,}4357$; б)$12{,}384$.

---

1) Находим значения данных выражений с большим числом десятичных знаков: $a_1 = 9/11 = 0{,}81818\ldots$, $a_2 = \sqrt{18} = 4{,}2426\ldots$. Затем вычисляем предельные абсолютные погрешности, округляя их с избытком;

$\alpha_{a_1} = |0{,}81818 - 0{,}818| \le 0{,}00019$. $\alpha_{a_2} = |4.2426 - 4.24| \le 0{,}0027$,

Предельные относительные погрешности составляют

$$\delta_{a_1} = \frac{\alpha_{a_1}}{a_1} = \frac{0.00019}{0.818} = 0.00024 = 0.024\%$$

$$\delta_{a_1} = \frac{\alpha_{a_2}}{a_2} = \frac{0.0027}{4.24} = 0.00064 = 0.064\%$$

Так как $\delta_{a_1} < \delta_{a_2}$, то равенство $9/11 = 0.818$ является более точным.

2) а) Пусть $72.353\ (\pm 0.026) = a$. Согласно условию, погрешность $\alpha_a = 0.026 < 0.05$; это означает что в числе 72,353 верными в узком смысле являются цифры 7, 2, 3. По правилам округления найдем приближенное значение числа, сохранив десятые доли:

$a_1 = 72.4$; $\alpha_{a_1} = \alpha_a = \Delta_{\text{окр}} = 0{,}026 + 0{,}047 = 0{,}073$

Полученная погрешность больше 0,05; значит, нужно уменьшить число цифр в приближенном числе до двух:

$a_2 = 72$; $\alpha_{a_2} = \alpha_a = \Delta_{окр} = 0{,}026 + 0{,}353 = 0{,}379$

Так как $\alpha_{a_2} < 0{,}5$, то обе оставшиеся цифры верны в узком смысле.

б) Пусть $a = 2.3544$; $\delta_a = 0.2\%$; тогда $\alpha_a = a * \delta_a = 0.00471$. В данном числе верными в широком смысле являются три цифры, по этому округляем его, сохраняя эти цифры:

$a_1 = 2.35$;$\alpha_{a_1} = 0.0044 + 0.00471 = 0.00911 < 0{,}01$.

Значит, и в округленном числе 2,35 все три цифры верны в широком смысле.

3) а) Так как все четыре числа $a = 0.4357$ верны в узком смысле, то абсолютная погрешность $\alpha_{a=}0{,}00005$, а относительная погрешность $\delta_a = 1/(2 * 4 * 10^3) = 0.000125 = 0.0125\%$.

Б) Так как все пять цифр числа $a = 12.384$ верны в широком смысле, то $\alpha_a = 0{,}001$; $\delta_a = 1/(1 * 10^4) = 0.0001 = 0.01\%$.

Задача № 2

Вычислить и определить погрешность результата

$$x = \frac{m^2 n^3}{\sqrt{k}}$$

Где m=28.3(±0.02)

N=7.45(±0.01)

K=0.678(±0.003)

1. Находим m^2=800.89≈800.9

N^3=413.49≈413.5

√k=0.823407≈0.8234

$$x = \frac{800.9 * 413.5}{0.8234} = 402200 = 4.02 * 10^5 \pm 0.003 * 10^5$$

2. Определяем предельные относительные погрешности

$$\delta_m = \frac{0.02}{28.3} = 0.00071$$

$$\delta_n = \frac{0.01}{7.45} = 0.00135$$

$$\delta_k = \frac{0.0003}{0.678} = 0.00443$$

Следовательно,

$\delta_x = 2\delta_m + 3\delta_n + 0.5\delta_k = 0.00142 + 0.00405 + 0.00222 = 0.00769 = 0.77\%$ коэффициент ставим по степени x,m,n

$$\Delta x = \delta_x * |x| = 4.02 * 10^5 * 0.0077 = 3.1 * 10^3$$

Ответ $x = 4{,}02 * 10^5$ $\delta_x = 0{,}77\%$

Задача №3

Вычислить результат выражения и определить его относительную погрешность

$$N = \frac{(n-1)(m+n)}{(m-n)^2}$$

Где m=5,72(±0.02)

N=3,0567(±0.0001)

1. Находим n-1=3.0567(±0.0001)

M+n=5,72(±0.02)+ 3.0567(±0.0001)=8.777(±0.0201)

m-n=5,72(±0.02)- 3.0567(±0.0001)=2.663(±0.0201)

$$N = \frac{2.0567 * 8.777}{2.663^2} = 2.54509449 \approx 2.546$$

2. Определяем предельные относительные погрешности

$$\delta_n = \frac{0.0001}{2.0567} + \frac{0.0201}{8.777} + 2\frac{0.0201}{2.663} = 0.0175 = 1.75\%$$

$$\Delta N = 2.546 * 0.0175 = 0.045$$

N=2.55±0.05

Ответ N=2.55±0.05 δ_n=1.74%

Задача № 4

Вычислить результат выражения и определить его абсолютную и относительную погрешности

$$X = \frac{(a-b)c}{\sqrt{m+n}}$$

Где m=12,375(±0.004)

N=86,2(±0.05)

A=27.16(±0.006)

B=5.03(±0.01)

C=3.6(±0.02)

1. Находим n-1=3.0567(±0.0001)

a-b=27.16(±0.006)- 5.03(±0.01)=22.13(±0.016)

m+n=12,375(±0.004)+ 86,2(±0.05)=98.575(±0.054)

$$X = \frac{22.13 * 3.6}{\sqrt{98.575}} = \frac{79.668}{9.9285} = 8.02417 \approx 8.02$$

2. Определяем предельные относительные погрешности

$$\delta_x = \frac{0.016}{22.13} + \frac{0.02}{3.6} + 0.5\frac{0.054}{2.98575} = 0.00655 = 0.66\%$$

$$\Delta x = 8.02 * 0.0655 = 0.0526$$

Ответ x=8.02(±0.053) δ_x=0.66%

II. Порядок выполнения работы

1. Загрузить табличный процессор
2. Произвести расчеты.
3. Результаты работы показать преподавателю.

4. Сохранить работу на диске в своем каталоге.
5. Выйти из табличного процессора.
6. Оформить отчет.

III Задания для самостоятельной работы

С помощью правил определения погрешностей арифметических операций вычислите значения выражении и оценить погрешность

1. $\dfrac{\sqrt{a}+\sqrt{b}}{b+\sqrt{a^2+b^2}}$ $a = 3.23 \pm 0.02$ $b = 2.45 \pm 0.01$

2. $\dfrac{\sqrt{a}+\sqrt{ab}+b}{(\sqrt{a}+b)^2}$ $a = 3.23 \pm 0.02$ $b = 2.45 \pm 0.01$

Контрольные вопросы

1. Как вычислить абсолютную погрешность?
2. Как относительную погрешность?
3. Какие цифры числа считаются верными?
4. Как записать формулу в табличном процессоре?
5. Как установить формат ячейки?

Оформление отчета

Отчет должен содержать название работы, задание на работу, исходные данные, вид формы данных, промежуточные и сводные данные.

Тема 2. Приближённые решения алгебраических и трансцендентных уравнений

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема работы: Решение уравнений приближенными методами
Цель работы: *уметь:*

Решать уравнения методами хорд, касательных и комбинированным методом, работать с программой СИ.

Материально-техническое оснащение: ПК, операционная система Windows
Количество часов: 4 часа.

I.Теоретическая часть

При выполнении вычислений необходимо придерживаться простых правил, выработанных практикой. Соблюдение правил экономит труд вычислителя и позволяет рационально использовать вычислительную технику. При вычислениях необходимо:

1) Уяснить постановку задачи. Проанализировать исходные данные на полноту представления и точность.
2) Выбрать методы вычисления, соответствующие условиям задачи.
3) Разработать алгоритм решения задачи (составить вычислительную схему).

Под алгоритмом решения задачи понимают однозначную последовательность операций получения конечного результата по исходным данным. Алгоритм разрабатывается в соответствии с выбранными методами решения задачи и учётом возможностей и особенности, применяемой вычислительной техники. В силу наглядности алгоритмы чаще всего представляют в виде блок-схемы. Каждый её элемент – блок отображает определённую совокупность вычислительных операций и отображается прямоугольником. Кроме них в блок – схему вводят логические блоки (блоки проверки условий), изображаемые в форме ромба. Все блоки имеют сквозную нумерацию. На рис. 1.1 дан пример блок – схемы метода табуляции функции y=f(x) на отрезке [A, B] с шагом H .

4) Программирование–запись алгоритма на языке, воспринимаемом машиной. Подготовленную программу и исходную информацию записывают на носитель. После этого следует отладка программы и расчёт по ней.

Метод табуляции

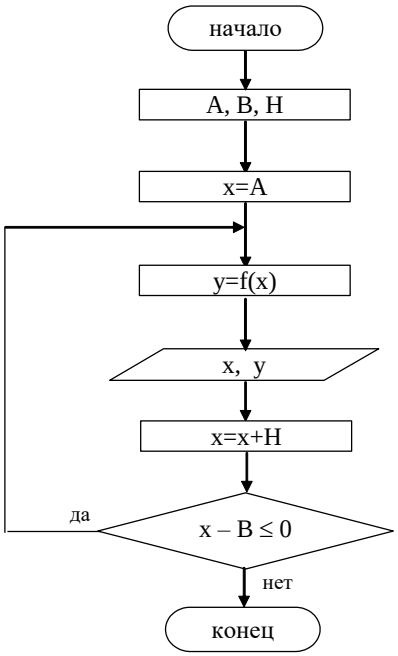


Рис. 1 Схема алгоритма метода табуляции функции y=f(x) на отрезке [A, B] с шагом H.

Пример записи программы на алгоритмическом языке СИ для табулирования функции приводится ниже.

Пример 1 Табулировать функцию $y=x^2+4*x-e^x$ на отрезке [-5, 5] с шагом H=0.5.

Решение: Составим программу табуляции заданной функции на языке СИ и выполним расчёты по ней :

```
#include<stdio.h>
#include<math.h>
main()
{float x,y,a=-5,b=5,h=0.5;
 clrscr();
 x=a;
 do{
    y=x*x+4*x-exp(x);
    printf("\n x=%f\ty=%f\t",x,y);
    x=x+h;
   }while((x-b)<=0);
}
```

```
x=-5.000000   y=4.993262          x=0.500000   y=0.601279
x=-4.500000   y=2.238891          x=1.000000   y=2.281718
x=-4.000000   y=-0.018316         x=1.500000   y=3.768311
x=-3.500000   y=-1.780197         x=2.000000   y=4.610944
x=-3.000000   y=-3.049787         x=2.500000   y=4.067506
x=-2.500000   y=-3.832085         x=3.000000   y=0.914463
x=-2.000000   y=-4.135335         x=3.500000   y=-6.865452
x=-1.500000   y=-3.973129         x=4.000000   y=-22.598150
x=-1.000000   y=-3.367879         x=4.500000   y=-51.767132
```

x=-0.500000 y=-2.356531 x=5.000000 y=-103.413162
x= 0.000000 y=-1.000000

При ручном счёте аналогом программы служит расчётная таблица, в которую заносятся все исходные и промежуточные данные. В каждый столбец таблицы заносят результаты определённой операции, которые используются для последующих вычислительных операций. Форма таблицы зависит от используемой вычислительной техники и выбранного способа расчёта.

Для рассмотренного ранее примера 1 табулирования функции и вычисления функции e^x с помощью ряда рабочая таблица будет иметь вид:

Таблица 1

n	x	x^2	x^2+4x	e^x	Δe^x	y	Δy
0	-5,0	25,00	5,00	0,007	0,0004	4,993	0,0004
1	-4,5	20,25	2,25	0,011	0,0002	2,239	0,0002
2	-4,0	16,00	0,00	0,018	0,0004	-0,018	0,0004
3	-3,5	12,25	-1,75	0,030	0,0002	-1,780	0,0002
4	-3,0	9,00	-3,00	0,050	0,0005	-3,050	0,0005
5	-2,5	6,25	-3,75	0,082	0,0001	-3,832	0,0001
6	-2,0	4,00	-4,00	0,135	0,0004	-4,135	0,0004
7	-1,5	2,25	-3,75	0,223	0,0002	-3,973	0,0002
8	-1,0	1,00	-3,00	0,368	0,0002	-3,368	0,0002
9	-0,5	0,25	-1,75	0,607	0,0005	-2,357	0,0005
10	0,0	0,00	0,00	1,000	0,0000	-1,000	0,0000

5) При решении задачи необходимо организовать контроль вычислений. Он подразделяется на текущий и заключительный. При текущем контроле проверяется правильность выполнения отдельных этапов вычислений. После выполнения расчётов проверяется правильность окончательного результата. Применяются различные методы контроля: подстановка полученных результатов, получение результатов различными методами и т. п .

6) Оценка точности полученного результата. Результат вычислений не может считаться таковым, если не даётся объективная оценка его погрешности.

Метод хорд

Программа, выполненная на языке “Си” для уточнения корня методом хорд .

```
#include<stdio.h>
#include<math.h>
 double y(double x)
 {return(x*x+4*x-exp(x));}
 double y1(double x)
 {return(2*x+4-exp(x));}
  main()
```

```
 {
 double a=0,b=0.5,c=100,ex=0.00001,ey=0.000001;
   clrscr();
l1 : if(y1(a)*y(a)>=0)
   {
   c=a-y(a)*(b-a)/(y(b)-y(a));
   if(fabs(c-a)<=ex && fabs(y(c))<=ey)
   printf("x=%f +/-%f\tn=%f",c,fabs(c-a));
   else{ a=c;goto l1;}
   ;}else
    {
   c=b-y(b)*(b-a)/(y(b)-y(a));
   if(fabs(b-c)<=ex && fabs(y(c))<=ey){
   printf("Ответ:\n");
   printf("x=%f +/-%f\n",c,fabs(b-c));}
   else{ b=c;goto l1;}
    }
   }
```

Ответ:

x= 0,318384 ± 0,000000 .

Метод касательных

Уточнение корня на отрезках [0 ; 0,5] и [3 ; 3,5] выполним на ПК.

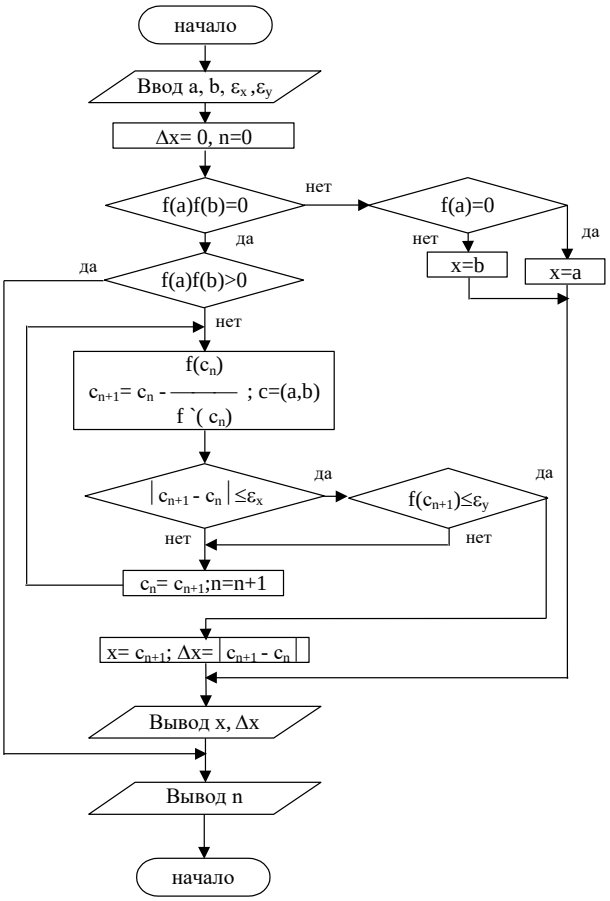


Рис.2. Схема алгоритма уточнения корня уравнения методом касательных с предварительным определением подвижного конца отрезка

Ниже приводится программа, выполненная на языке Си .

```
#include<stdio.h>
#include<math.h>
 double y(double x)
 {return(x*x+4*x-exp(x));}
 double y1(double x)
 {return(2*x+4-exp(x));}
  main()
 {
 double b=0.5,pog,per,ex=0.00001,ey=0.00001;
 int n=0;
  clrscr();
// в b – координата b отрезка отделения .
 do{
 per=b-y(b)/y1(b);
 pog=fabs(per-b);
 b=per;
 n++;
 }
 while(pog>=ex && fabs(y(per))>=ey);
   printf("Ответ:\n");
   printf("x=%f +/-%f\n",b,pog);
   printf("Шагов: %d\n",n);
   scanf("%c");
  }
```

Ответ:

x= 0,318384 ± 0,002198 .

Шагов: 2

Уточненный метод Ньютона (комбинированный метод)

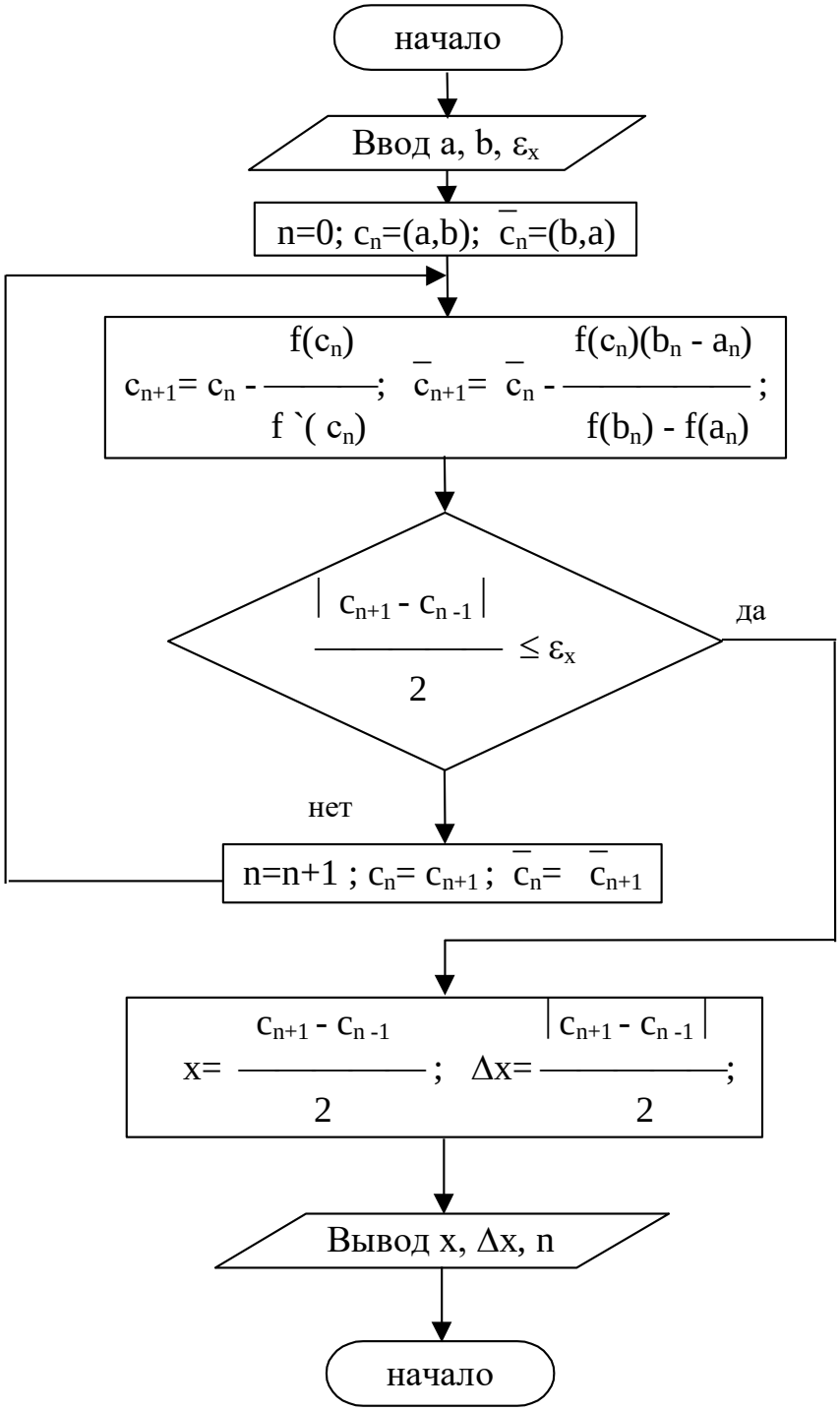


Рис. 3 Упрощённая схема алгоритма уточнения корня уравнения на отрезке отделения совмещённым методом хорд и касательных

Распечатка программы и результаты расчётов приведены ниже .

```
#include<stdio.h>
#include<math.h>
 double y(double x)
 {return(x*x+4*x-exp(x));}
 double y1(double x)
 {return(2*x+4-exp(x));}
```

```
 main(){
double b=0.5,pog,per,y0,ex=0.00001,ey=0.00001;
int n=0;
 clrscr();
y0=y1(0);
 do{
 per=b-y(b)/y0;
 pog=fabs(per-b);
 b=per;
 n++;}
 while(pog>=ex && fabs(y(per))>=ey);
   printf("Ответ:\n");
   printf("x=%f +/-%f\n",b,pog);
   printf("Шагов: %d\n",n); scanf("%c");  }
```

Ответ :

x= 0,318383 ± 0,000013

Шагов 5 .

Программа уточнения корня совмещённым методом хорд и касательных:

```
#include<stdio.h>
#include<math.h>
double y (double x){return x*x+4*x-exp(x);}
double y1 (double x){return 2*x+4-exp(x);}
double y2 (double x){return 2-exp(x);}
main()
{
 double a=0, b=0.5, c,k, ex=0.00001, ey=0.00001, x2;
 char fo;
 int n=0;
 int f=1;
 clrscr();
 x2=(a+b)/2;
 if(y(a)*y(b)<=0)
  do
{
   if(y2(x2)*y(a)>ey)
   {
     c=a-((b-a)*y(a)/(y(b)-y(a)));
     k=b-(y(b)/y1(b));
     n++;
    }
    else
   {
     c=a-(y(a)/y1(a));
     k=b-((b-a)*y(b)/(y(b)-y(a)));
     n++;}
    if(fabs(k-c)/2 <= ex)
      if(fabs(y((c+k)/2)) <= ey)
```

```
        { f=0;}
      a=c;
      b=k;}
 while(f==1);
 printf("Ответ:\n");
 printf("x=%f+/-%f\n",(a+b)/2, fabs(b-a)/2);
 printf("Шагов: %d\n",n);
 scanf("%c",&fo);
 }
```

:

$0{,}318381 \pm 0{,}000006$.

Шагов: 2 .

Для ручного счёта составим таблицу, положив в ней:

отрезок уточнения [-4,4 ; -4] .

$$h_{an} = -\frac{f(a_n)}{f\,`(a_n)}, \quad h_{bn} = -\frac{f(b_n)(b_n - a_n)}{f(b_n) - f(a_n)}$$

$a_{n+1} = a_n + h_{an}$; $b_{n+1} = b_n + h_{bn}$

Таблица2.5.

N	A_n	b_n	$f(a_n)$	$f\,`(a_n)$	$f(b_n)$	h_{an}	h_{bn}
0	-4,5000	-4,0000	2,2389	-5,0111	-0,0183	0,4468	-0,0028
1	-4,0532	-4,0028	0,1983	-4,1238	-0,0071	0,0481	-0,0003
2	-4,0051	-4,0031	0,0022	-4,0272	-0,0058	0,0005	-0,0000
3	-4,0046	-4,0031	0,0002	-4,0274	-0,0058	0,0000	0,0000

Результат расчёта :

$\xi_{1к} = -4{,}004 \pm 0{,}001$.

Для расчёта потребовалось три итерационных цикла.

II. Порядок выполнения работы

1. Загрузить программу СИ
2. Получить вариант работы у преподавателя.
3. Произвести расчеты в соответствии с вариантом.
4. Результаты работы показать преподавателю.
5. Сохранить работу на диске в своем каталоге.
6. Выйти из программы.
7. Оформить отчет.

=

III. Контрольные вопросы

1. Как производится решение уравнения методом табуляции?
2. Как производится решение уравнения методом хорд?
3. Как производится решение уравнения методом касательных?
4. Как производится решение уравнения методом комбинированным методом хорд и касательных?

5. Как производится решение уравнения методом простой итерации?

IV. Оформление отчета

Отчет должен содержать название лабораторной работы, задание на работу, исходные данные, вид формы данных, промежуточные и сводные данные.

Тема 3. Решение систем линейных алгебраических уравнений

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА№ 3

Тема работы: Решение систем линейных уравнений приближенными методами

Цель работы: *уметь:*

- Решать системы уравнений по схеме Гаусса; пользоваться электронной таблицей

Материально-техническое оснащение: ПК, операционная система Windows.
Количество часов: 4 часа.

I.Теоретическая часть

Наиболее распространенным методом решения систем линейных уравнений является **метод последовательного исключения неизвестных или метод Гаусса.**

Элементарными преобразованиями называются следующие три типа преобразований систем линейных уравнений:

1) перестановка двух уравнений системы;

2) умножение обеих частей уравнения системы на любое отличное от нуля число;

3) прибавление (вычитание) к обеим частям одного уравнения соответствующих частей другого уравнения, умноженных на любое отличное от нуля число.

Элементарные преобразования переводят данную систему линейных уравнений в эквивалентную.

Метод последовательного исключения неизвестных [метод Гаусса) рассмотрим на примере системы четырех уравнений с четырьмя неизвестными.

Метод Гаусса применим при том условии, что все ведущие коэффициенты отличны от нуля.

Пусть дана система

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = a_{15}, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = a_{25}, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = a_{35}, \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = a_{45}. \end{cases} \quad (1)$$

Будем исключать неизвестное x_1 из всех уравнений системы (1), кроме первого. Назовем x_1 *ведущим неизвестным,* а коэффициент a_{11} — *ведущим коэффициентом.* Разделим первое уравнение на a_{11}, получим

$$x_1 + \frac{a_{12}}{a_{11}}x_2 + \frac{a_{13}}{a_{11}}x_3 + \frac{a_{14}}{a_{11}}x_4 = \frac{a_{15}}{a_{11}}.$$

обозначим

$b_{ij} = a_{ij}/a_{11}$ $(j > 1)$. Тогда рассматриваемое уравнение примет вид

$$x_1 + b_{12}x_2 + b_{13}x_3 + b_{14}x_4 = b_{15}, \quad (2)$$

$$x_1 = b_{15} - b_{12}x_2 - b_{13}x_3 - b_{14}x_4.$$

Для исключения неизвестного x_1 из уравнений системы (1) проведем следующие преобразования.

1) Из каждого *i-го* (i>2) уравнения системы (1) вычтем уравнение (2), умноженное на a_{i1}:

Обозначим $a_{ij}^{(1)} = a_{ij} - a_{i1} b_{1j}$

В результате проведенных элементарных преобразований имеем систему трех уравнений с тремя неизвестными, эквивалентную системе (I):

$$\begin{cases} a_{22}^{(1)} x_2 + a_{23}^{(1)} x_3 + a_{24}^{(1)} x_4 = a_{25}^{(1)}, \\ a_{32}^{(1)} x_2 + a_{33}^{(1)} x_3 + a_{34}^{(1)} x_4 = a_{35}^{(1)}, \\ a_{42}^{(1)} x_2 + a_{43}^{(1)} x_3 + a_{44}^{(1)} x_4 = a_{45}^{(1)}, \end{cases} \qquad (1')$$

Разделив, далее, коэффициенты первого уравнения системы (1') на ведущий коэффициент $a_{22}^{(1)} \neq 0$, получим первое уравнение системы в виде

$$x_2 + \frac{a_{23}^{(1)}}{a_{22}^{(1)}} x_3 + \frac{a_{24}^{(1)}}{a_{22}^{(1)}} x_4 = \frac{a_{25}^{(1)}}{a_{22}^{(1)}}.$$

Обозначим $b_{2j}^{(1)} = \frac{a_{2j}^{(1)}}{a_{22}^{(1)}}$, (j > 2). Тогда первое уравнение системы (1') примет вид

$$x_2 + b_{23}^{(1)} x_3 + b_{24}^{(1)} x_4 = b_{25}^{(1)}, \qquad (2')$$

$$x_2 = b_{25}^{(1)} - b_{23}^{(1)} x_3 - b_{24}^{(1)} x_4.$$

Исключая теперь x_2 из всех уравнений системы (1') , кроме первого, таким же способом, какими мы исключали x_1, придем к следующей системе из двух уравнений с двумя неизвестными:

$$\begin{cases} a_{33}^{(2)} x_3 + a_{34}^{(2)} x_4 = a_{35}^{(2)}, \\ a_{43}^{(2)} x_3 + a_{44}^{(2)} x_4 = a_{45}^{(2)}, \end{cases}$$

где ($i,j \neq 3$). Разделив коэффициенты первого уравнения системы (1'') на ведущий коэффициент $a_{ij}^{(2)} \neq 0$, получим

$$x_3 + b_{34}^{(2)} x_4 = b_{35}^{(2)}, \qquad (2'')$$

где $b_{3j}^{(2)} = \frac{a_{3j}^{(2)}}{a_{33}^{(2)}}$ $(j>3)$

$$x_3 = b_{35}^{(2)} - b_{34}^{(2)} x_4.$$

Исключив теперь x_3 аналогичным путем из системы *(1")*, находим

$a_{44}^{(3)} x_4 = a_{45}^{(3)}$ *(1''')*,

где $a_{ij}^{(3)} = a_{ij}^{(2)} - a_{i3}^{(2)} b_{3j}^{(2)}$. Отсюда

$$x_4 = \frac{a_{45}^{(3)}}{a_{44}^{(3)}}$$

Остальные неизвестные системы последовательно определяются из уравнений (2"), (2') и (2):

$$x_3 = b_{35}^{(2)} - b_{34}^{(2)} x_4,$$

$$x_2 = b_{25}^{(1)} - b_{24}^{(1)} x_4 - b_{23}^{(1)} x_3,$$

$$x_1 = b_{15} - b_{14} x_4 - b_{13} x_3 - b_{12} x_2.$$

Таким образом, процесс решения системы линейных уравнений по методу Гаусса сводится к построению эквивалентной системы уравнений (2), (2'), (2") (2''').

Для удобства вычисления производятся по схеме, называемой *схемой единственного деления.* Вычисление элементов b_{ij} называется *прямым ходом,* вычисление значений неизвестных — *обратным ходом,* так как сначала определяется значение последнего неизвестного.

Схема единственного деления (схема Гаусса) составляется следующим образом.

В раздел I схемы (см. табл.) записываются коэффициенты при неизвестных (в столбцах соответствующих неизвестных), свободные члены и для каждой строки подсчитанные «контрольные суммы» (столбец 2), равные сумме элементов a_{ij} в данной строке (здесь (i == 1, 2, 3, 4; j = 1, 2, 3, 4, 5); последняя строка раздела **I,** состоящая из 1 и элементов *bij,* получается делением первой строки раздела на ведущий коэффициент a_{11}.

Элементы раздела **II** схемы равны соответствующим элементам раздела I минус произведение $a_{i1}b_{1j}$ *(i, j•□ 2).* Последняя строка раздела II, состоящая из 1 и элементов $b^{(1)}_{2j}$, получается делением первой строки раздела на ведущий коэффициент $a^{(1)}_{22}$.

Аналогично вычисляются элементы III и IV разделов схемы. 1 II III и IV разделы, заканчивающиеся вычислением элементов $b^{(i-1)}_{ij}$ (i = 1, 2, 3, 4; j = 2, 3, 4, 5) составляют прямой ход вычислений схемы.

Обратный ход начинается с вычисления последнего неизвестного системы линейных уравнений x_4 и заканчивается вычислением первого неизвестного x_1. При обратном ходе используются лишь строки прямого хода, содержащие единицы и соответствующие элементы b_{ij} (назовем эти строки «отмеченными»).

Элемент $b^{(3)}_{45}$ последней «отмеченной» строки и столбца свободных членов дает значение x_4. Далее, остальные неизвестные x_3, x_2, x_1 находятся вычитанием из свободного члена «отмеченной» строки суммы произведений ее коэффициентов на соответствующие значения ранее найденных неизвестных.

Значения неизвестных последовательно выписываются в V раздел. Расставленные там единицы помогают находить для *xj* соответствующие коэффициенты в «отмеченных» строках.

Для контроля вычислений используются так называемые контрольные суммы:

Над контрольными суммами в каждой строке проделываются те же операции, что и над остальными элементами этой строки. При отсутствии ошибок в вычислениях элементы столбца □ равны суммам элементов соответствующих преобразованных строк. Таким образом, контролируется прямой ход схемы.

Для контроля обратного хода $\bar{x}_4$ находится в последней «отмеченной» строке столбца , т. е. $\bar{x}_4$ == $b^{(3)}_{46}$, а остальные неизвестные этого столбца $\bar{x}_j$ *(j* = 3, 2, 1) подсчитываются в тех же строках и по тем же формулам, что и

неизвестные x_j, только в формулы подставляются соответствующие $\bar{x}_j$. В итоге числа $\bar{x}_j$ должны совпадать с числами $x_j + 1$.

2 Практическая часть

Задание 1. По схеме единственного деления решить систему

$$\begin{cases} 2x_1 + 2x_2 - x_3 + x_4 = 4, \\ 4x_1 + 3x_2 - x_3 + 2x_4 = 6, \\ 8x_1 + 5x_2 - 3x_3 + 4x_4 = 12, \\ 3x_1 + 3x_2 - 2x_3 + 2x_4 = 6. \end{cases}$$

Решение. В раздел I таблицы 1 вписываем матрицу системы, ее свободные члены и контрольные суммы. Затем подсчитываем «отмеченную» строку этого раздела, разделив первую строку на $a_{11} = 2$. Например, $b_{12} = a_{12}/a_{11} = 2/2 = 1$

Таблица 1

x_1	x_2	x_3	x_4	Свободные члены	S	Разделы схемы
2	2	−1	1	4	8	I
4	3	−1	2	6	14	
8	5	−3	4	12	26	
3	3	−2	2	6	12	
1	1	−0,5	0,5	2	4	
	−1	1	0	−2	−2	II
	−3	1	0	−4	−6	
	0	−0,5	0,5	0	0	
	1	−1	0	2	2	
		−2	0	2	0	III
		−0,5	0,5	0	0	
		1	0	−1	0	
			0,5	−0,5	0	IV
			1	−1	0	
		1	1	$x_4 = -1$	$\bar{x}_4 = 0$	V
	1			$x_3 = -1$	$\bar{x}_3 = 0$	
1				$x_2 = 1$	$\bar{x}_2 = 2$	
				$x_1 = 1$	$\bar{x}_1 = 2$	

Элементы раздела **II** вычисляем по следующему правилу: каждый элемент этого раздела равен соответствующему элементу раздела I минус произведение первого элемента его строки на элемент «отмеченной» строки в его столбце. Полученный результат записываем на соответствующее место в разделе II. Например,

$$a_{23}^{(1)} = a_{23} - a_{21}\, b_{13} = -1 - 4\,(-0,5) = 1,$$
$$a_{33}^{(1)} = a_{33} - a_{31}\, b_{13} = -3 - 8\,(-0,5) = 1.$$

Элементы «отмеченной» строки раздела **II** получим, разделив его первую строку на ведущий коэффициент $a_{22}^{(1)} = -1$. Например, $b_{23}^{(1)} = a_{23}^{(1)} / a_{22}^{(1)} = 1/(-1) = -1$.

Аналогично вычисляются элементы III и IV разделов. Например:

$$a_{14}^{(2)} = a_{14}^{(1)} - a_{12}^{(1)} b_{24}^{(1)} = 2 - 3 \cdot 0{,}5 = 0{,}5,$$
$$a_{45}^{(3)} = a_{45}^{(2)} - a_{43}^{(2)} b_{35}^{(2)} = 0 - (-0{,}5)(-1) = -0{,}5.$$

Для вычисления элементов раздела V, т. е. для нахождения неизвестных используем «отмеченные» строки, начиная с последней.

Неизвестное x_4 представляет собой свободный член последней «отмеченной» строки:

$x_4 = b_{45}^{(3)} = 1$, а остальные неизвестные x_3, x_2 и x_1 получаются последовательно в результате вычитания из свободных членов «отмеченных» строк суммы произведений соответствующих коэффициентов $b_{ij}^{(i-1)}$ на ранее найденные значения неизвестных.

Контроль осуществляется с помощью столбца □, над которым производятся те же действия, что и над остальными столбцами и в итоге сумма элементов каждой строки схемы (кроме столбца □) должна быть равна элементу этой строки из столбца □. Корни x_j принадлежащие столбцу □,- должны быть равны $1 + x_j$ для каждой строки раздела V.

В результате получаем $x_1 =$ **1,** $x_2 = 1$, $x_3 = -1$, $x_4 = -1$.

Задание 2. Используя схему Гаусса, решить систему уравнений с точностью до 0,001.

$$\begin{cases} 0{,}68x_1 + 0{,}05x_2 - 0{,}11x_3 + 0{,}08x_4 = 2{,}15, \\ 0{,}21x_1 - 0{,}13x_2 + 0{,}27x_3 - 0{,}8x_4 = 0{,}44, \\ -0{,}11x_1 - 0{,}84x_2 + 0{,}28x_3 + 0{,}06x_4 = -0{,}83, \\ -0{,}08x_1 + 0{,}15x_2 - 0{,}5x_3 - 0{,}12x_4 = 1{,}16. \end{cases}$$

Вычисления производим по схеме единственного деления:

Коэффициенты при неизвестных				Свободные члены	Контрольные суммы Σ	Строчные суммы Σ'
x_1	x_2	x_3	x_4			
0,68	0,05	−0,11	0,08	2,15	2,85	2,85
0,21	−0,13	0,27	−0,8	0,44	−0,01	−0,01
−0,11	−0,84	0,28	0,06	−0,83	−1,44	−1,44
−0,08	0,15	−0,5	−0,12	1,16	0,61	0,61
1	0,0735	−0,1618	0,1176	3,1618	4,1912	4,1912
	−0,1454	0,30398	−0,8247	−0,22398	−0,89015	−0,8901
	−0,8319	0,2622	0,0729	−0,4822	−0,97897	−0,97896
	0,1559	−0,5129	−0,1106	1,4129	0,9453	0,9453
	1	−2,0906	5,6719	1,5404	6,1221	6,1217
		−1,47697	4,79139	0,7992	4,1140	4,1136
		−0,18697	−0,9948	1,1723	−0,00913	−0,0095
		1	−3,2441	−0,5411	−2,7854	−2,7851
			−1,6013	1,0711	−0,5299	−0,5302
			1	−0,6689	0,3309	0,3311
2,8264	−0,3337	−2,7110	−0,6689			
3,8263	0,6664	−1,7119	0,3309			

Ответ: $x_1 = 2{,}826$; $x_2 = -0{,}334$; $x_3 = -2{,}711$; $x_4 = -0{,}669$.

Задание для самостоятельной работы дома:

1 Решите систему уравнений методом единственного решения.

№ 1.
$$\begin{cases} 4{,}4x_1 - 2{,}5x_2 + 19{,}2x_3 - 10{,}8x_4 = 4{,}3, \\ 5{,}5x_1 - 9{,}3x_2 - 14{,}2x_3 + 13{,}2x_4 = 6{,}8, \\ 7{,}1x_1 - 11{,}5x_2 + 5{,}3x_3 - 6{,}7x_4 = -1{,}8, \\ 14{,}2x_1 + 23{,}4x_2 - 8{,}8x_3 + 5{,}3x_4 = 7{,}2. \end{cases}$$

№ 2.
$$\begin{cases} 8{,}2x_1 - 3{,}2x_2 + 14{,}2x_3 + 14{,}8x_4 = -8{,}4, \\ 5{,}6x_1 - 12x_2 + 15x_3 - 6{,}4x_4 = 4{,}5, \\ 5{,}7x_1 + 3{,}6x_2 - 12{,}4x_3 - 2{,}3x_4 = 3{,}3, \\ 6{,}8x_1 + 13{,}2x_2 - 6{,}3x_3 - 8{,}7x_4 = 14{,}3. \end{cases}$$

№ 3.
$$\begin{cases} 5{,}7x_1 - 7{,}8x_2 - 5{,}6x_3 - 8{,}3x_4 = 2{,}7, \\ 6{,}6x_1 + 13{,}1x_2 - 6{,}3x_3 + 4{,}3x_4 = -5{,}5, \\ 14{,}7x_1 - 2{,}8x_2 + 5{,}6x_3 - 12{,}1x_4 = 8{,}6, \\ 8{,}5x_1 + 12{,}7x_2 - 23{,}7x_3 + 5{,}7x_4 = 14{,}7. \end{cases}$$

№ 4.
$$\begin{cases} 3{,}8x_1 + 14{,}2x_2 + 6{,}3x_3 - 15{,}5x_4 = 2{,}8, \\ 8{,}3x_1 - 6{,}6x_2 + 5{,}8x_3 + 12{,}2x_4 = -4{,}7, \\ 6{,}4x_1 - 8{,}5x_2 - 4{,}3x_3 + 8{,}8x_4 = 7{,}7, \\ 17{,}1x_1 - 8{,}3x_2 + 14{,}4x_3 - 7{,}2x_4 = 13{,}5. \end{cases}$$

№ 5.
$$\begin{cases} 15{,}7x_1 + 6{,}6x_2 - 5{,}7x_3 + 11{,}5x_4 = -2{,}4, \\ 8{,}8x_1 - 6{,}7x_2 + 5{,}5x_3 - 4{,}5x_4 = 5{,}6, \\ 6{,}3x_1 - 5{,}7x_2 - 23{,}4x_3 + 6{,}6x_4 = 7{,}7, \\ 14{,}3x_1 + 8{,}7x_2 - 15{,}7x_3 - 5{,}8x_4 = 23{,}4. \end{cases}$$

№ 6.
$$\begin{cases} 4{,}3x_1 - 12{,}1x_2 + 23{,}2x_3 - 14{,}1x_4 = 15{,}5, \\ 2{,}4x_1 - 4{,}4x_2 + 3{,}5x_3 + 5{,}5x_4 = 2{,}5, \\ 5{,}4x_1 + 8{,}3x_2 - 7{,}4x_3 - 12{,}7x_4 = 8{,}6, \\ 6{,}3x_1 - 7{,}6x_2 + 1{,}34x_3 + 3{,}7x_4 = 12{,}1. \end{cases}$$

№ 7.
$$\begin{cases} 14{,}4x_1 - 5{,}3x_2 + 14{,}3x_3 - 12{,}7x_4 = -14{,}4, \\ 23{,}4x_1 - 14{,}2x_2 - 5{,}4x_3 + 2{,}1x_4 = 6{,}6, \\ 6{,}3x_1 - 13{,}2x_2 - 6{,}5x_3 + 14{,}3x_4 = 9{,}4, \\ 5{,}6x_1 + 8{,}8x_2 - 6{,}7x_3 - 23{,}8x_4 = 7{,}3. \end{cases}$$

№ 8.
$$\begin{cases} 1{,}7x_1 + 10x_2 - 1{,}3x_3 + 2{,}1x_4 = 3{,}1, \\ 3{,}1x_1 + 1{,}7x_2 - 2{,}1x_3 + 5{,}4x_4 = 2{,}1, \\ 3{,}3x_1 - 7{,}7x_2 + 4{,}4x_3 - 5{,}1x_4 = 1{,}9, \\ 10x_1 - 20{,}1x_2 + 20{,}4x_3 + 1{,}7x_4 = 1{,}8. \end{cases}$$

№ 9.
$$\begin{cases} 1{,}7x_1 - 1{,}8x_2 + 1{,}9x_3 - 57{,}4x_4 = 10, \\ 1{,}1x_1 - 4{,}3x_2 + 1{,}5x_3 - 1{,}7x_4 = 19, \\ 1{,}2x_1 + 1{,}4x_2 + 1{,}6x_3 + 1{,}8x_4 = 20, \\ 7{,}1x_1 - 1{,}3x_2 - 4{,}1x_3 + 5{,}2x_4 = 10. \end{cases}$$

№ 10.
$$\begin{cases} 6{,}1x_1 + 6{,}2x_2 - 6{,}3x_3 + 6{,}4x_4 = 6{,}5, \\ 1{,}1x_1 - 1{,}5x_2 + 2{,}2x_3 - 3{,}8x_4 = 4{,}2, \\ 5{,}1x_1 - 5{,}0x_2 + 4{,}9x_3 - 4{,}8x_4 = 4{,}7, \\ 1{,}8x_1 + 1{,}9x_2 + 2{,}0x_3 - 2{,}1x_4 = 2{,}2. \end{cases}$$

№ 11.

$$\begin{cases} 2.2x_1 - 3.1x_2 + 4.2x_3 - 5.1x_4 = 6.01, \\ 1.3x_1 + 2.2x_2 - 1.4x_3 + 1.5x_4 = 10, \\ 6.2x_1 - 7.4x_2 + 8.5x_3 - 9.6x_4 = 1.1, \\ 1.2x_1 + 1.3x_2 + 1.4x_3 + 4.5x_4 = 1.6. \end{cases}$$

№ 12.

$$\begin{cases} 35.8x_1 + 2.1x_2 - 34.5x_3 - 11.8x_4 = 0.5, \\ 27.1x_1 - 7.5x_2 + 11.7x_3 - 23.5x_4 = 12.8, \\ 11.7x_1 + 1.8x_2 - 6.5x_3 + 7.1x_4 = 1.7, \\ 6.3x_1 + 10x_2 + 7.1x_3 + 3.4x_4 = 20.8. \end{cases}$$

№ 13.

$$\begin{cases} 35.1x_1 + 1.7x_2 + 37.5x_3 - 2.8x_4 = 7.5, \\ 45.2x_1 + 21.1x_2 - 1.1x_3 - 1.2x_4 = 11.1, \\ -21.1x_1 + 31.7x_2 + 1.2x_3 - 1.5x_4 = 2.1, \\ 31.7x_1 + 18.1x_2 - 31.7x_3 + 2.2x_4 = 0.5. \end{cases}$$

№ 14.

$$\begin{cases} 1.1x_1 + 11.2x_2 + 11.1x_3 - 13.1x_4 = 1.3, \\ -3.3x_1 + 1.1x_2 + 30.1x_3 - 20.1x_4 = 1.1, \\ 7.5x_1 + 1.3x_2 + 1.1x_3 + 10x_4 = 20, \\ 1.7x_1 + 7.5x_2 - 1.8x_3 + 2.1x_4 = 1.1. \end{cases}$$

№ 15.

$$\begin{cases} 7.5x_1 + 1.8x_2 - 2.1x_3 - 7.7x_4 = 1.1, \\ -10x_1 + 1.3x_2 - 20x_3 - 1.4x_4 = 1.5, \\ 2.8x_1 - 1.7x_2 + 3.9x_3 + 4.8x_4 = 1.2, \\ 10x_1 + 31.4x_2 - 2.1x_3 - 10x_4 = -1.1. \end{cases}$$

№ 16.

$$\begin{cases} 30.1x_1 - 1.4x_2 + 10x_3 - 1.5x_4 = 10, \\ -17.5x_1 + 11.1x_2 + 1.3x_3 - 7.5x_4 = 1.3, \\ 1.7x_1 - 21.1x_2 + 7.1x_3 - 17.1x_4 = 10, \\ 2.1x_1 + 2.1x_2 + 3.5x_3 + 3.3x_4 = 1.7. \end{cases}$$

№ 17.

$$\begin{cases} 7.3x_1 - 8.1x_2 + 12.7x_3 - 6.7x_4 = 8.8, \\ 11.5x_1 + 6.2x_2 - 8.3x_3 + 9.2x_4 = 21.5, \\ 8.2x_1 - 5.4x_2 + 4.3x_3 - 2.5x_4 = 6.2, \\ 2.4x_1 + 11.5x_2 - 3.3x_3 + 14.2x_4 = -6.2. \end{cases}$$

№ 18.

$$\begin{cases} 4.8x_1 + 12.5x_2 - 6.3x_3 - 9.7x_4 = 3.5, \\ 22x_1 - 31.7x_2 + 12.4x_3 - 8.7x_4 = 4.6, \\ 15x_1 + 21.1x_2 - 4.5x_3 + 14.4x_4 = 15, \\ 8.6x_1 - 14.4x_2 + 6.2x_3 + 2.8x_4 = -1.2. \end{cases}$$

№ 19.

$$\begin{cases} 6.4x_1 + 7.2x_2 - 8.3x_3 + 42x_4 = 2.23, \\ 5.8x_1 - 8.3x_2 + 14.3x_3 - 6.2x_4 = 17.1, \\ 8.6x_1 + 7.7x_2 - 18.3x_3 + 8.8x_4 = -5.4, \\ 13.2x_1 - 5.2x_2 - 6.5x_3 + 12.2x_4 = 6.5. \end{cases}$$

№ 20.

$$\begin{cases} 14.2x_1 + 3.2x_2 - 4.2x_3 + 8.5x_4 = 13.2, \\ 6.3x_1 - 4.3x_2 + 12.7x_3 - 5.8x_4 = -4.4, \\ 8.4x_1 - 22.3x_2 - 5.2x_3 + 4.7x_4 = 6.4, \\ 2.7x_1 + 13.7x_2 + 6.4x_3 - 12.7x_4 = 8.5. \end{cases}$$

№ 21.

$$\begin{cases} 7.3x_1 + 12.4x_2 - 3.8x_3 - 14.3x_4 = 5.8, \\ 10.7x_1 - 7.7x_2 + 12.5x_3 + 6.6x_4 = -6.6, \\ 15.6x_1 + 6.6x_2 + 14.4x_3 - 8.7x_4 = 12.4, \\ 7.5x_1 + 12.2x_2 - 8.3x_3 + 3.7x_4 = 9.2. \end{cases}$$

№ 22.

$$\begin{cases} 13.2x_1 - 8.3x_2 - 4.4x_3 + 6.2x_4 = 6.8, \\ 8.3x_1 + 4.2x_2 - 5.6x_3 + 7.7x_4 = 12.4, \\ 5.8x_1 - 3.7x_2 + 12.4x_3 - 6.2x_4 = 8.7, \\ 3.5x_1 + 6.6x_2 - 13.8x_3 - 9.3x_4 = -10.8. \end{cases}$$

№ 23.

$$\begin{cases} 8.1x_1 + 1.2x_2 - 9.1x_3 + 1.7x_4 = 10, \\ 1.1x_1 - 1.7x_2 + 7.2x_3 - 3.4x_4 = 1.7, \\ 1.7x_1 - 1.8x_2 + 10x_3 + 2.3x_4 = 2.1, \\ 1.3x_1 + 1.7x_2 - 9.9x_3 + 3.5x_4 = 27.1. \end{cases}$$

№ 24.

$$\begin{cases} 3.3x_1 - 2.2x_2 - 10x_3 + 1.7x_4 = 1.1, \\ 1.8x_1 + 21.1x_2 + 1.3x_3 - 2.2x_4 = 2.2, \\ -10x_1 + 1.1x_2 + 20x_3 - 4.5x_4 = 10, \\ 70x_1 - 1.7x_2 - 2.2x_3 + 3.3x_4 = 2.1. \end{cases}$$

№ 25.

$$\begin{cases} 1.7x_1 + 9.9x_2 - 20x_3 - 1.7x_4 = 1.7, \\ 20x_1 + 0.5x_2 - 30.1x_3 - 1.1x_4 = 2.1, \\ 10x_1 - 20x_2 + 30.2x_3 + 0.5x_4 = 1.8, \\ 3.3x_1 - 0.7x_2 + 3.3x_3 + 20x_4 = -1.7. \end{cases}$$

№ 26.

$$\begin{cases} 1.7x_1 - 1.3x_2 - 1.1x_3 - 1.2x_4 = 2.2, \\ 10x_1 - 10x_2 - 1.3x_3 + 1.3x_4 = 1.1, \\ 3.5x_1 + 3.3x_2 + 1.2x_3 + 1.3x_4 = 1.2, \\ 1.3x_1 + 1.1x_2 - 1.3x_3 - 1.1x_4 = 10. \end{cases}$$

№ 27.

$$\begin{cases} 1.1x_1 + 11.3x_2 - 1.7x_3 + 1.8x_4 = 10, \\ 1.3x_1 - 11.7x_2 + 1.8x_3 + 1.4x_4 = 1.3, \\ 1.1x_1 - 10.5x_2 - 1.7x_3 - 1.5x_4 = 1.1, \\ 1.5x_1 - 0.5x_2 + 1.8x_3 - 1.1x_4 = 10. \end{cases}$$

№ 28.

$$\begin{cases} 1.4x_1 + 2.1x_2 - 3.3x_3 + 1.1x_4 = 10, \\ 10x_1 - 1.7x_2 + 1.1x_3 - 1.5x_4 = 1.7, \\ 2.2x_1 + 34.4x_2 - 1.1x_3 - 1.2x_4 = 20, \\ 1.1x_1 + 1.3x_2 + 1.2x_3 + 1.4x_4 = 1.3. \end{cases}$$

№ 29.

$$\begin{cases} 1.3x_1 - 1.7x_2 + 3.3x_3 + 1.7x_4 = 1.1, \\ 10x_1 + 5.5x_2 - 1.3x_3 + 3.4x_4 = 1.3, \\ 1.1x_1 + 1.8x_2 - 2.2x_3 - 1.1x_4 = 10, \\ 1.3x_1 - 1.2x_2 + 2.1x_3 + 2.2x_4 = 1.8. \end{cases}$$

№ 30.

$$\begin{cases} 1.2x_1 + 1.8x_2 - 2.2x_3 - 4.1x_4 = 1.3, \\ 10x_1 - 5.1x_2 + 1.2x_3 + 5.5x_4 = 1.2, \\ 2.2x_1 - 30.1x_2 + 3.1x_3 + 5.8x_4 = 10, \\ 10x_1 + 2.4x_2 - 30.5x_3 - 2.2x_4 = 34.1. \end{cases}$$

III. Контрольные вопросы

1. Опишите прямой ход схемы Гаусса.
2. Опишите обратный ход схемы Гаусса.
3. Как работать с формулами в электронной таблице?

IV. Оформление отчёта

Отчет по лабораторно-практической работе составляется по следующей структуре:

1. Наименование лабораторно-практической работы.

2. Цель работы.
3. Ответы на контрольные вопросы.
4. Вывод по работе.

Результаты работы в виде файлов представьте преподавателю.

Тема 4: Интерполирование и экстраполирование функций

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема работы: Составление интерполяционных формул Лагранжа, Ньютона

Цель работы: *уметь:*

- Составлять интерполяционные формулы Лагранжа, Ньютона

Материально-техническое оснащение ПК, операционная система Windows.
Количество часов: 2 часа.

I.Теоретическая часть

Интерполяция

В вычислительной математике нередки случаи, когда одну функцию приходится заменять другой, более простой и удобной для дальнейшей работы. Такую задачу называют аппроксимацией функций.

Поводом для аппроксимации функции может послужить, в частности, табличный способ ее задания. Предположим, что в . результате некоторого эксперимента для конечного набора значений xi величины x из отрезка [a; b]

$a = x_0 < x_1 < ... < x_i < \ldots < x_n = b$

получен набор значений yi величины y. Если допустить, что между x и y существует функциональная зависимость y= F(x), можно поставить вопрос о поиске аналитического представления функции F (очевидно, что в такой общей постановке эта задача решается неоднозначно). Точки x_0 , x_1 , ..., x_n в этом случае называются узлами. Если расстояние $h = x_{i+1} - x_i$, является постоянным • (т.е. независящим от i), то сетка значений, представленная в таблице 1, называется равномерной.

Таблица 1

x	x_0	x_1	x_2	…	x_i	…	x_n
$F(x)$	y_0	y_1	y_2	…	y_i	…	y_n

Повод для аппроксимации может возникнуть даже тогда, когда аналитическое выражение для некоторой функции $y= F(x)$ имеется, однако оно оказывается мало пригодным для решения поставленной задачи, потому что операция, которую требуется осуществить над этой функцией, трудновыполнима. Элементарный пример — вычисление значения трансцендентной функции «вручную». Действительно, чтобы вычислить, например, ln 3,2756, проще всего воспользоваться степенным разложением функции, т. е. заменить трансцендентную функцию степенной. При этом получится, разумеется, приближенное значение функции, но если мы умеем контролировать погрешность, то можно считать, что мы получили интересующий нас результат (хотя бы потому, что в реальности все равно приходится ограничиваться приближенным представлением значений логарифмической функции).

Другая ситуация, когда может потребоваться аппроксимация аналитически заданной функции, — вычисление определенных интегралов. Задача эта, как правило, весьма сложная, часто элементарными приемами невыполнимая. Например, Как вычислить интеграл $\int_1^2 \frac{\sin x}{x} dx$? Он, несомненно, существует, но по формуле Ньютона—Лейбница вычислен быть практически не может, так как первообразная $\int \frac{\sin x}{x} dx$ не выражается в элементарных функциях. Аппроксимация подынтегральной функции — один из возможных приемов (и важно отметить, что цель аппроксимации налагает отпечаток на ее способ).

Классический подход к численному решению подобных задач заключается в том, чтобы, опираясь на информацию о функции *F,* по некоторому алгоритму подобрать аппроксимирующую функцию *G,* в определенном смысле «близкую» к *F*.

Чаще всего задача аппроксимации решается с помощью многочленов. Вычисления значений многочлена легко автоматизировать, производная и интеграл от многочлена, в свою очередь, также являются многочленами.

Интерполяционный многочлен Лагранжа

Пусть функция *F(x)* задана таблицей 1. Построим многочлен $L_n(x)$, степень которого не выше, чем *n,* и для которого выполнены условия интерполяции

$$L_n(x_0) = y_0,\ L_n(x_1) = y_1,\ \ldots,\ L_n(x_n) = y_n. \qquad (1)$$

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0) \cdot \ldots \cdot (x - x_{i-1})(x - x_{i+1}) \cdot \ldots \cdot (x - x_n)}{(x_i - x_0) \cdot \ldots \cdot (x_i - x_{i-1})(x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n)}. \qquad (2)$$

Это и есть *интерполяционный многочлен Лагранжа.* По таблице исходной функции *F* формула (2) позволяет довольно просто составить «внешний вид» многочлена.

Конечные разности

Пусть функция задана таблицей вида табл. 1 с постоянным шагом.

Разности между значениями функции в соседних узлах интерполяции называются *конечными разностями первого порядка:*

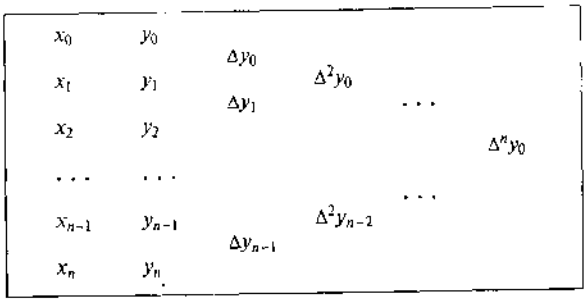
$$\Delta y_i = y_{i+1} - y_i \ \ (i = 0,\ 1,\ \ldots,\ n-1).$$

Из конечных разностей первого порядка образуются *конечные разности второго порядка,*

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i \ \ (i = 0,\ 1,\ \ldots,\ n-2).$$

Продолжая этот процесс, можно по заданной таблице функции составить таблицу конечных разностей (табл. 2).

Таблица 2



Первая интерполяционная формула Ньютона

Пусть для функции, заданной таблицей с постоянным шагом, составлена таблица конечных разностей (табл. 2). Будем искать интерполяционный многочлен в виде

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \\ + \ldots + a_n(x - x_0) \cdot \ldots \cdot (x - x_{n-1}). \qquad (1)$$

Это - многочлен *n-ой* степени. Значения коэффициентов $a_0, a_1, \ldots, a_n$ найдем из условия совпадения значений исходной функции и многочлена в узлах.

Полагая $x=x_o$, из (1) находим $y_o = P_n(xo) = a_0$, откуда $a_o=y_o$.

Далее, полагая $x=x_1$, получаем

$$y_1 = P_n(x_1) = a_0 + a_1(x_1 - x_0), \text{ откуда } a_1 = \Delta y_0 / h.$$

При $x=x_2$ имеем

$$y_2 = P_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1),$$

т.е. $y_2 - 2\Delta y_0 - y_0 = 2h^2 a_2$, или $y_2 - 2y_1 + y_0 = 2h^2 a_2$, откуда

$$a_2 = \Delta^2 y_0 / 2h^2.$$

Проведя аналогичные выкладки, можно получить $a_3 = \square^3 y_o / 3h^3$. Исходя из этих формул, методом полной математической индукции можно доказать, что в общем случае выражение для a_k будет иметь вид

$$a_k = \frac{\Delta^k y_0}{k! h^k}. \qquad (2)$$

Подставим теперь (2) в выражение для многочлена (1)

$$P_n(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x - x_0)(x - x_1) + \ldots + \frac{\Delta^n y_0}{n!h^n}(x - x_0) \cdot \ldots \cdot (x - x_{n-1}). \quad (3)$$

Формула (3) называется первой интерполяционной формулой Ньютона.

Эта формула традиционно применяется для интерполирования в начале отрезка интерполяции. Первую интерполяционную формулу Ньютона называют по этой причине ***формулой для интерполирования вперед***.

Вторая интерполяционная формула Ньютона

Когда значение аргумента находится ближе к концу отрезка интерполяции, применять первую интерполяционную формулу становится невыгодно.

В этом случае применяется ***формула для интерполирования назад*** - вторая интерполяционная формула Ньютона, которая ищется в виде

$$P_n(x) = a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) + \ldots + a_n(x - x_n) \cdot \ldots \cdot (x - x_1). \quad (4)$$

Как и для первой формулы Ньютона, коэффициенты a_0, a_1,…,a_n находятся из условия совпадения значений функции и интерполяционного многочлена в узлах

$$a_k = \frac{\Delta^k y_{n-k}}{k!h^k}. \quad (5)$$

Вторая интерполяционная формула Ньютона имеет следующий вид:

$P_n(x)$ =

$y_n +$  (6)

Пример 1. Построить интерполяционный многочлен Лагранжа для функции, заданной таблицей значений:

x	1	3	4
$f(x)$	12	4	6

Из таблицы следует, что $n = 2$ (т. е. степень многочлена будет не выше 2); здесь $x_0 = 1$, $x_1 = 3$, $x_2 = 4$. Используя формулу (5), получаем

$$L_2(x) = 12\frac{(x-3)(x-4)}{(1-3)(1-4)} + 4\frac{(x-1)(x-4)}{(3-1)(3-4)} + 6\frac{(x-1)(x-3)}{(4-1)(4-3)} =$$
$$= 2(x^2 - 7x + 12) - 2(x^2 - 5x + 4) + 2(x^2 - 4x + 3) =$$
$$= 2x^2 - 12x + 22.$$

Непосредственное применение формулы Лагранжа приводит к большому числу однотипных вычислений. Организация вычислений существенно улучшается, если пользоваться специальной вычислительной схемой.

В таблице 2. показано построение такой схемы для 4 узлов (i=0,1,2,3). Таблица составляется заново для каждого нового значения аргумента *x.*

Заполнение таблицы начинается с того, что вычисляются и заносятся в соответствующие клетки все элементарные разности. Вслед за этим вычисляются произведения P_i разностей по строкам:

$$P_0 = (x - x_0)(x_0 - x_1)(x_0 - x_2)(x_0 - x_3);$$
$$P_1 = (x_1 - x_0)(x - x_1)(x_1 - x_2)(x_1 - x_3) \text{ и т. д.}$$

Таблица 2

x	x_0	x_1	x_2	x_3	P_i	y_i	y_i/P_i
x_0	$x - x_0$	$x_0 - x_1$	$x_0 - x_2$	$x_0 - x_3$			
x_1	$x_1 - x_0$	$x - x_1$	$x_1 - x_2$	$x_1 - x_3$			
x_2	$x_2 - x_0$	$x_2 - x_1$	$x - x_2$	$x_2 - x_3$			
x_3	$x_3 - x_0$	$x_3 - x_1$	$x_3 - x_2$	$x - x_3$			
							$S=(y_i/P_i)$

Легко видеть, что использованное в таблице 2 обозначение *P,* — это знаменатель в формуле Лагранжа (2),

Все необходимые значения последовательно получаются в таблице. Сумма S образуется сложением элементов последнего столбца. Для получения окончательного значениями $L_n(x)$ достаточно умножить *S* на произведение диагональных разностей таблицы).

Пример 2. Построить интерполяционный многочлен Ньютона по следующим данным:

x	0,5	1	1,5	2	2,5
y	1,715	2,348	3,127	5,289	8,914

Построим таблицу разностей:

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0,5	1,715				
		0,633			
1	2,348		0,146		
		0,779		1,237	
1,5	3,127		1,383		-1,157
		2,162		0,080	
2	5,289		1,463		
		3,625			
2,5	8,914				

Таким образом, многочлен Ньютона, представленный в форме (3), имеет вид

$$P_4(x) = 1,715 + \frac{0,853}{0,5}(x-0,5) + \frac{0,140}{2!\cdot 0,5^2}(x-0,5)(x-1) +$$

$$+\frac{1,237}{3!\cdot 0,5^3}(x-0,5)(x-1)(x-1,5) - \frac{1,157}{4!\cdot 0,5^4}(x-0,5)(x-1)(x-1,5)(x-2) =$$

$$= 1,715 + 1,266(x-0,5) + 0,292(x-0,5)(x-1) +$$

$$+1,649(x-0,5)(x-1)(x-1,5) - 0,771(x-0,5)(x-1)(x-1,5)(x-2).$$

2 Практическая часть

Задание 1. Имеется таблица значений некоторой функции:

x	$f(x)$	x	$f(x)$
0,41	2,63	2,67	4,87
1,55	3,75	3,84	5,03

Требуется получить значение этой функции в точке x= 1,91, пользуясь интерполяционным многочленом Лагранжа.

Вычисления составим таблицу 3 с применением табличного процессора. Для нахождения окончательного результата сумма значений последнего столбца умножается на произведение диагональных разностей:

$$f(1,91) = 0,792 \cdot 5,241 \approx 4,15.$$

Таблица 3

$x = 1,91$	x_0	x_1	x_2	x_3	P_i	y_i	y_i / P_i
x_0	1,50	−1,14	−2,26	−3,43	−13,26	2,63	−0,198
x_1	1,14	0,36	−1,12	−2,29	1,05	3,75	3,561
x_2	2,26	1,12	−0,76	−1,17	2,25	4,87	2,163
x_3	3,43	2,29	1,17	−1,93	−17,74	5,03	−0,284
							5,242

Задание 2. Для таблично заданной функции

x	***f(x)***	***x***	***f(x)***
1,62	8,14	1,65	7,21
1,63	8,02	1,66	6,54
1,64	7,93	1,67	5,01

Вычислите конечные разности до третьего порядка включительно, составьте интерполяционные формулы Ньютона (x_0=1,62, x_n=1,67). Вычислите для контроля значения интерполяционных многочленов соответственно в точках 1,63 и 1,66; сопоставьте полученные результаты.

Задание для самостоятельной работы:

1 Найти приближенное значение функции при данном значении аргумента с помощью интерполяционного многочлена Лагранжа, интерполяционных формул Ньютона.

Таблица 1

x	y	№ варианта	x
1.375	5.04192	1	1.3832
1.380	5.17744	7	1.3926
1.385	5.32016	13	1.3862
1.390	5.47069	19	1.3934
1.395	5.62968	25	1.3866
1.400	5.79788		

Таблица 2

x	y	№ варианта	x
0.115	8.65729	2	0.1264
0.120	8.29329	8	0.1315
0.125	7.95829	14	0.1232
0.130	7.64893	20	0.1334
0.135	7.36235	26	0.1285
0.140	7.09613		

Таблица 3

x	y	№ варианта	x
0.150	6.61659	3	0.1521
0.155	6.39989	9	0.1611
0.160	6.19658	15	0.1662
0.165	6.00551	21	0.1542
0.170	5.82558	27	0.1625
0.175	5.65583		

Таблица 4

x	y	№ варианта	x
0.180	5.61543	4	0.1838
0.185	5.46693	10	0.1875
0.190	5.32634	16	0.1944
0.195	5.19304	22	0.1976
0.200	5.06649	28	0.2038
0.205	4.94619		

Таблица 5

x	y	№ варианта	x
0.210	4.83170	5	0.2121
0.215	4.72261	11	0.2165
0.220	4.61855	17	0.2232
0.225	4.51919	23	0.2263
0.230	4.42422	29	0.2244
0.235	4.33337		

Таблица 6

x	y	№ варианта	x
1.415	0.888551	6	1.4179
1.420	0.889599	12	1.4258
1.425	0.890637	18	1.4396
1.430	0.891667	24	1.4236
1.435	0.892687	30	1.4315
1.440	0.893698		

II. Порядок выполнения работы

1. Загрузить табличный процессор
2. Произвести расчеты.
3. Результаты работы показать преподавателю.
4. Сохранить работу на диске в своем каталоге.
5. Выйти из табличного процессора. Оформить отчет.

Тема 5: Численное интегрирование

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема работы: Приближенное вычисление интегралов с помощью формул Ньютона-Кортеса
Цель работы: *уметь:*
Проводить вычисления интегралов с помощью формул Ньютона-Кортеса

Материально-техническое оснащение ПК, операционная система Windows.
Количество часов: 2 часа.

I.Теоретическая часть

Квадратурные формулы Ньютона – Котеса.

Пусть для данной функции y=f(x) требуется вычислить интеграл $\int_a^b ydx$.

Выберем шаг

$$h=\frac{b-a}{n} \qquad (1)$$

и разобьем отрезок [a,b] с помощью равноотстоящих точек x_0=a, $x_i=x_0+ih$, i=1,2,...,n-1, x_n=b на n равных частей. Выполним вычисления $У_i=f(x_i)$. Таким образом, произведена табуляция аналитически заданной функции f(x) и получен ее табличный образ. Если подынтегральная функция уже задана таблицей, то необходимость в ее табуляции отпадет.

Заменяя функцию y интерполяционным полиномом Лагранжа L_n(x) получим приближенную квадратурную формулу

$$\int_{x_0=a}^{x_n=b} ydx=\int_{x_0=a}^{x_n=b} L_n(x)dx=\sum_{i=0}^{n} A_i y_i, \qquad (2)$$

где A_i- некоторые постоянные коэффициенты. Получим их явные выражения. Полином Лагранжа получен ранее.

$$L_n(x)=\sum_{i=0}^{n}(-1)^{n-i}\cdot\frac{q(q-1)(q-2)...(q-n)}{i!(n-i)!(q-i)}\cdot y_i=\sum_{i=0}^{n}(-1)^{n-i}\cdot\frac{q^{[n+1]}}{i!(n-i)!(q-i)}\cdot y_i. \qquad (3)$$

Подставляя (3) в (2), получим:

$$\int_{x_0}^{x_n} ydx=\int_{x_0}^{x_n}\sum_{i=0}^{n}\frac{(-1)^{n-i}}{i!(n-i)!}\cdot\frac{q^{[n+1]}}{q-i}\cdot y_i dx, \qquad (4)$$

интеграл обладает свойством аддитивности:

$$\int_{x_0}^{x_n}\sum_{i=0}^{n}\frac{(-1)^{n-i}}{i!(n-i)!}\cdot\frac{q^{[n+1]}}{q-i}\cdot y_i dx. \qquad (5)$$

Так как $q=\frac{x-x_0}{h}, dx=hdq.$

При x=x_0 q=0,при x=x_n $q=\frac{x_n-x_0}{h}=\frac{x_0+nh-x_0}{h}=n.$

Перейдем на новые пределы интегрирования и вынесем за интеграл величины не зависимые от q:

$$\int_{x_0}^{x_n} y dx = \sum_{i=0}^{n} \frac{(-1)^{n-i}}{i!(n-i)!} h y_i \int_0^n \frac{q^{[n+1]}}{q-i} dq. \qquad (6)$$

Заменим h имеем:

$$\int_{x_0}^{x_n} y dx = (b-a)\frac{1}{n} \sum_{i=0}^{n} \frac{(-1)^{n-i}}{i!(n-i)!} y_i \int_0^n \frac{q^{[n+1]}}{q-i} dq. \qquad (7)$$

Обозначим:

$$H_i = \frac{1}{n} \frac{(-1)^{n-i}}{i!(n-i)!} \int_0^n \frac{q^{[n+1]}}{q-i} dq. \qquad (8)$$

H_i - постоянные коэффициенты, называемые коэффициентами Котеса.
Тогда $A_i=(b-a)H_i$.
Окончательный вид квадратурной формулы

$$\int_a^b y dx = (b-a)\sum_{i=0}^{n} H_i y_i, \qquad (9)$$

где $h = \frac{b-a}{n}$, $y_i = f_i(a+ih)$, i=0,1,…,n.

Коэффициенты H_i называются коэффициентами Ньютона–Котеса. Эти коэффициенты не зависят от вида функции F(x), а являются функцией только от n (количества узлов интерполяции). Поэтому коэффициенты Ньютона–Котеса можно вычислить заранее для различного числа узлов интерполяции и свести в таблицу.

n=1	$H_0=H_1=1/2$
n=2	$H_0=H_2=1/6$ $H_1=2/3$
n=3	$H_0=H_3=1/8$ $H_1=H_2=3/8$
n=4	$H_0=H_4=7/90$ $H_1=H_3=16/45$ $H_2=2/15$
n=5	$H_0=H_5=19/288$ $H_1=H_4=25/96$ $H_2=H_3=25/144$
n=6	$H_0=H_6=41/840$ $H_1=H_5=9/35$ $H_2=H_4=9/280$ $H_3=34/105$
n=7	$H_0=H_7=751/17280$ $H_1=H_6=3577/17280$ $H_2=H_5=1323/17280$ $H_2=H_3=2989/17280$

Пример Вычислить по формуле Ньютона–Котеса $\int_0^{\frac{\pi}{2}} \frac{\cos x}{1+x} dx$ выбрав n=4

Решение: Формула Ньютона–Котеса для n=4 имеет вид $\int_0^{\frac{\pi}{2}} \frac{\cos x}{1+x} dx = \frac{\pi}{2} \sum_{i=0}^{4} H_i y_i$

Определим шаг $h=(b-a)/4 \approx 0.4$. Далее найдем значения подынтегральной функции $y=\frac{\cos x}{1+x}$ в точках x_0, x_1,x_2, x_3,x_4.

$$x_0 = 0 \; y_0 = \frac{\cos 0}{1+0} = 1 \quad x_1 = 0.4 \; y_1 = \frac{\cos 22.5^\circ}{1+0.4} = 0.659 \quad x_2 = 0.8 \; y_2 = \frac{\cos 45^\circ}{1+0.8} = 0.393$$

$$x_3 = 1.2 \; y_3 = \frac{\cos 67.5^\circ}{1+1.2} = 0.174 \; x_4 = 1.6 \; y_4 = \frac{\cos 90^\circ}{1+1.6} = 0$$

Пользуясь таблицей, находим коэффициенты Ньютона–Котеса $H_0=H_4=7/90$ $H_1=H_3=16/45$ $H_2=2/15$

Подставив найденные значения в формулу получим:

$$\int_0^{\frac{\pi}{2}} \frac{\cos x}{1+x}dx = \frac{\pi}{2}\left[\frac{7}{90}(1+0)+\frac{16}{45}(0.659+0.174)+\frac{2}{15}0.393\right] = 0.670$$

II. Порядок выполнения работы

Задание. Выберите вариант и напишите программу вычисления интеграл по формулам Ньютона – Котеса при n=5 и n=7на языке Си.

№ 1. 1) $\int_{0.6}^{1.4} \frac{\sqrt{x^2+5}\,dx}{2x+\sqrt{x^2+0.5}}$;

№ 2. 1) $\int_{0.4}^{1.2} \frac{\sqrt{0.5x+2}\,dx}{\sqrt{2x^2+1}+0.8}$;

№ 3. 1) $\int_{0.8}^{1.8} \frac{\sqrt{0.8x^2+1}\,dx}{x+\sqrt{1.5x^2+2}}$;

№ 4. 1) $\int_{1.0}^{2.2} \frac{\sqrt{1.5x+0.6}\,dx}{1.6+\sqrt{0.8x^2+2}}$;

№ 5. 1) $\int_{1.2}^{2.0} \frac{\sqrt{2x^2+1.6}\,dx}{2x+\sqrt{0.5x^3+3}}$;

№ 6. 1) $\int_{1.3}^{2.5} \frac{\sqrt{x^2+0.6}\,dx}{1.4+\sqrt{0.8x^2+1.}}$

№ 7. 1) $\int_{1.2}^{2.6} \frac{\sqrt{0.4x+1.7}\,dx}{1.5x+\sqrt{x^2+1.3}}$;

№ 8. 1) $\int_{0.8}^{1.6} \frac{\sqrt{0.3x^2+2.3}\,dx}{1.8+\sqrt{2x+1.6}}$;

№ 9. 1) $\int_{1.2}^{2} \frac{\sqrt{0.6x+1.7}\,dx}{2.1x+\sqrt{0.7x^2+1}}$;

№ 10. 1) $\int_{0.8}^{2.4} \frac{\sqrt{0.4x^2+1.5}\,dx}{2.5+\sqrt{2x+0.8}}$;

№ 11. 1) $\int_{1.2}^{2.8} \frac{\sqrt{1.2x+0.7}\,dx}{1.4x+\sqrt{1.3x^2+0.5}}$;

№ 12. 1) $\int_{0.6}^{2.4} \frac{\sqrt{1.1x^2+0.9}\,dx}{1.6+\sqrt{0.8x^2+1.4}}$;

№ 13. 1) $\int_{0.7}^{2.1} \frac{\sqrt{0.6x+1.5}\,dx}{2x+\sqrt{x^2+3}}$;

№ 14. 1) $\int_{0.8}^{2.4} \frac{\sqrt{1.5x+2.3}\,dx}{3+\sqrt{0.3x+1}}$;

№ 15. 1) $\int_{1.9}^{2.6} \frac{\sqrt{2x+1.7}\,dx}{2.4+\sqrt{1.2x^2+0.6}}$;

№ 16. 1) $\int_{0.5}^{1.9} \frac{\sqrt{0.7x^2+2.3}\,dx}{3.2+\sqrt{0.8x+1.4}}$;

№ 17. 1) $\int_{1}^{2.6} \frac{\sqrt{0.4x+3}\,dx}{0.7x+\sqrt{2x^2+0.5}}$;

№ 18. 1) $\int_{0.7}^{2.1} \frac{\sqrt{1.7x^2+0.5}\,dx}{1.4+\sqrt{1.2x+1.3}}$;

№ 19. 1) $\int_{0.6}^{2.2} \frac{\sqrt{1.5x+1}\,dx}{1.2x+\sqrt{x^2+1.8}}$;

№ 20. 1) $\int_{1.2}^{3} \frac{\sqrt{2x^2+0.7}\,dx}{1.5+\sqrt{0.8x+1}}$;

№ 21. 1) $\int_{1.3}^{2.7} \frac{\sqrt{1.3x^2+0.8}\,dx}{1.7x+\sqrt{2x+0.5}}$;

№ 22. 1) $\int_{0.6}^{1.4} \frac{\sqrt{x^2+0.5}\,dx}{2x+\sqrt{x^2+2.5}}$;

№ 23. 1) $\int_{0.4}^{1.2} \frac{\sqrt{2x^2+1}\,dx}{0.8x+\sqrt{0.5x+2}}$;

№ 24. 1) $\int_{0.8}^{1.8} \frac{\sqrt{1.5x^2+2}\,dx}{x+\sqrt{0.8x^2+1}}$;

№ 25. 1) $\int_{1}^{2.2} \frac{\sqrt{0.8x^2+2}\,dx}{1.6+\sqrt{1.5x+0.6}}$;

№ 26. 1) $\int_{1.2}^{2.0} \frac{\sqrt{0.5x^2+3}\,dx}{2x+\sqrt{2x^2+1.6}}$;

№ 27. 1) $\int_{1.3}^{2.5} \frac{\sqrt{0.8x^2+1.3}\,dx}{1.4+\sqrt{x^2+0.6}}$;

№ 28. 1) $\int_{1.2}^{2.6} \frac{\sqrt{x^2+1.3}\,dx}{1.5x+\sqrt{0.4x+1.7}}$;

№ 29. 1) $\int_{0.8}^{1.6} \frac{\sqrt{2x+1.6}\,dx}{1.8+\sqrt{0.3x^2+2.3}}$;

№ 30. 1) $\int_{1.2}^{2} \frac{\sqrt{0.7x^2+1}\,dx}{2.1x+\sqrt{0.6x+1.7}}$;

Тема 5: Численное интегрирование

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема работы: Приближенное вычисление интегралов с помощью формул Гаусса
Цель работы: *уметь:*
Проводить вычисления интегралов с помощью формул Гаусса

Материально-техническое оснащение ПК, операционная система Windows.
Количество часов: 2 часа.

I. Теоретическая часть

Существует иной, связанный с именем Гаусса, подход к построению квадратурных формул, в котором центральное место играет выбор узлов для интерполирования подынтегральной функции. Ознакомимся с этим подходом в его простейшей возможной реализации.

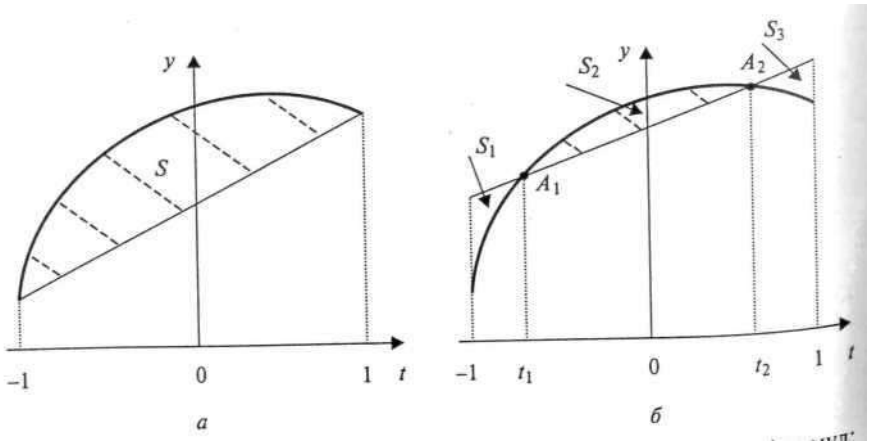


Рисунок 1 - Иллюстрация подходов к построению квадратурных формул:
а — фиксированные узлы линейной интерполяции подынтегральной функции (метод трапеций); погрешность интегрирования характеризуется величиной площади S,
б — подвижные узлы интерполяции (метод Гаусса); величина погрешности зависит от степени несовпадения площадей $S_1 + S_3$ и S_2

Для разъяснения существа метода Гаусса обратимся к рисунку 1. Будем использовать простейшую (т.е. линейную) интерполяцию подынтегральной функции. Если в качестве узлов интерполяции взять концы отрезка [-1; 1], то различие в площадях криволинейной трапеции, ограниченной сверху кривой $y = \varphi(x)$, и «обычной» трапеции, ограниченной сверху прямой, проведенной через концы указанной кривой, фиксировано видом функции $y = \varphi(x)$ (рисунок 1, а). Однако если сделать узлы интерполяции «подвижными» (рисунок 1, б), то можно выбрать их таким образом, чтобы разность между площадями криволинейной и «обычной» трапеции была значительно меньше, чем в случае а. Более того, можно сделать эти площади равными ($S_1 + S_3 = S_2$), т.е. аппроксимировать интеграл точно.

Если взять узлами линейной интерполяции числа

$t_1 = -\frac{1}{\sqrt{3}},\ t_2 = +\frac{1}{\sqrt{3}},$

, то получим требуемое условие.

Применительно к исходному виду интеграла, при делении отрезка на равные части, практически используемые формулы Гаусса для вычисления интеграла и оценки погрешности имеют вид

$$I_G = \frac{h}{2}\sum_{i=0}^{n-1}\left[f\left(x_i + \frac{h}{2} - \frac{h}{2\sqrt{3}}\right) + f\left(x_i + \frac{h}{2} + \frac{h}{2\sqrt{3}}\right)\right]; \quad (5)$$

$$|I - I_G| \le \frac{1}{4320 \cdot n^4}(b-a)^5 M_4.$$

II. Практическая часть

Задание 1. Вычислить интеграл $I = \int_0^1 x^2 \sin x dx$ по формуле Гауса, разделив отрезок [0; 1] на 10 равных частей, и оценить погрешность вычислений.

Для оценки остаточного члена найдем производную четвертого порядка от подынтегральной функции f(x) = x^2sinx

$f^{IV}(x) = (x^2 - 12)\sin x - 8x\cos x.$

Значение $|f^{VI}(x)|$ на отрезке [0; 1] ограничено числом 14. Используя формулу (5), получаем оценку: $|I - I_G| = \frac{1}{4320 \cdot 10^4} \cdot 0{,}1^5 \cdot 14 = 0{,}0000000000032$

Составим таблицу значений функции в точках, входящих в формулу (5) (таблица 3).

Таблица 5.3

x_i	$x_i + \frac{h}{2} - \frac{h}{2\sqrt{3}}$	$x_i + \frac{h}{2} + \frac{h}{2\sqrt{3}}$	y_i
0	0,02113249	0,078868	0,00000944 0,00049005
0,1	0,12113249	0,178868	0,00177304 0,00569215
0,2	0,22113249	0,278868	0,01072537 0,02140672
0,3	0,32113249	0,378868	0,03255086 0,05309115
0,4	0,42113249	0,478868	0,07250071 0,10566206
0,5	0,52113249	0,578868	0,13520907 0,18331848
0,6	0,62113249	0,678868	0,22452206 0,28938023
0,7	0,72113249	0,778868	0,34334373 0,42614496
0,8	0,82113249	0,878868	0,49350196 0,59476723
0,9	0,92113249	0,978868	0,67563779 0,79516236
1,0			
			$\sum = 4,46488894$

Таким образом, I_G= 0,05*4,46488894 = 0,22324447.

Задание для самостоятельной работы

Вычислить интеграл по формуле Гаусса при n=10,.

№ 1. 1) $\int_{0.6}^{1.4} \frac{\sqrt{x^2+5}\,dx}{2x+\sqrt{x^2+0.5}}$;

№ 2. 1) $\int_{0.4}^{1.2} \frac{\sqrt{0.5x+2}\,dx}{\sqrt{2x^2+1}+0.8}$;

№ 3. 1) $\int_{0.8}^{1.8} \frac{\sqrt{0.8x^2+1}\,dx}{x+\sqrt{1.5x^2+2}}$;

№ 4. 1) $\int_{1.0}^{2.2} \frac{\sqrt{1.5x+0.6}\,dx}{1.6+\sqrt{0.8x^2+2}}$;

№ 5. 1) $\int_{1.2}^{2.0} \frac{\sqrt{2x^2+1.6}\,dx}{2x+\sqrt{0.5x^2+3}}$;

№ 6. 1) $\int_{1.3}^{2.5} \frac{\sqrt{x^2+0.6}\,dx}{1.4+\sqrt{0.8x^2+1.3}}$;

№ 7. 1) $\int_{1.2}^{2.6} \frac{\sqrt{0.4x+1.7}\,dx}{1.5x+\sqrt{x^2+1.3}}$;

№ 8. 1) $\int_{0.8}^{1.6} \frac{\sqrt{0.3x^2+2.3}\,dx}{1.8+\sqrt{2x+1.6}}$;

№ 9. 1) $\int_{1.2}^{2} \frac{\sqrt{0.6x+1.7}\,dx}{2.1x+\sqrt{0.7x^2+1}}$;

№ 10. 1) $\int_{0.8}^{2.4} \frac{\sqrt{0.4x^2+1.5}\,dx}{2.5+\sqrt{2x+0.8}}$;

№ 11. 1) $\int_{1.2}^{2.8} \frac{\sqrt{1.2x+0.7}\,dx}{1.4x+\sqrt{1.3x^2+0.5}}$;

№ 12. 1) $\int_{0.6}^{2.4} \frac{\sqrt{1.1x^2+0.9}\,dx}{1.6+\sqrt{0.8x^2+1.4}}$;

№ 13. 1) $\int_{0.7}^{2.1} \frac{\sqrt{0.6x+1.5}\,dx}{2x+\sqrt{x^2+3}}$;

№ 14. 1) $\int_{0.8}^{2.4} \frac{\sqrt{1.5x+2.3}\,dx}{3+\sqrt{0.3x+1}}$;

№ 15. 1) $\int_{1.9}^{2.6} \frac{\sqrt{2x+1.7}\,dx}{2.4+\sqrt{1.2x^2+0.6}}$;

№ 16. 1) $\int_{0.5}^{1.9} \frac{\sqrt{0.7x^2+2.3}\,dx}{3.2+\sqrt{0.8x+1.4}}$;

№ 17. 1) $\int_{1}^{2.6} \frac{\sqrt{0.4x+3}\,dx}{0.7x+\sqrt{2x^2+0.5}}$;

№ 18. 1) $\int_{0.1}^{2.1} \frac{\sqrt{1.7x^2+0.5}\,dx}{1.4+\sqrt{1.2x+1.3}}$;

№ 19. 1) $\int_{0.6}^{2.2} \frac{\sqrt{1.5x+1}\,dx}{1.2x+\sqrt{x^2+1.8}}$;

№ 20. 1) $\int_{1.2}^{3} \frac{\sqrt{2x^2+0.7}\,dx}{1.5+\sqrt{0.8x+1}}$;

№ 21. 1) $\int_{1.3}^{2.7} \frac{\sqrt{1.3x^2+0.8}\,dx}{1.7x+\sqrt{2x+0.5}}$;

№ 22. 1) $\int_{0.6}^{1.4} \frac{\sqrt{x^2+0.5}\,dx}{2x+\sqrt{x^2+2.5}}$;

№ 23. 1) $\int_{0.4}^{1.2} \frac{\sqrt{2x^2+1}\,dx}{0.8x+\sqrt{0.5x+2}}$;

№ 24. 1) $\int_{0.8}^{1.8} \frac{\sqrt{1.5x^2+2}\,dx}{x+\sqrt{0.8x^2+1}}$;

№ 25. 1) $\int_{1}^{2.2} \frac{\sqrt{0.8x^2+2}\,dx}{1.6+\sqrt{1.5x+0.6}}$;

№ 26. 1) $\int_{1.2}^{2.0} \frac{\sqrt{0.5x^2+3}\,dx}{2x+\sqrt{2x^2+1.6}}$;

№ 27. 1) $\int_{1.3}^{2.5} \frac{\sqrt{0.8x^2+1.3}\,dx}{1.4+\sqrt{x^2+0.6}}$;

№ 28. 1) $\int_{1.2}^{2.6} \frac{\sqrt{x^2+1.3}\,dx}{1.5x+\sqrt{0.4x+1.7}}$;

№ 29. 1) $\int_{0.8}^{1.6} \frac{\sqrt{2x+1.6}\,dx}{1.8+\sqrt{0.3x^2+2.3}}$;

№ 30. 1) $\int_{1.2}^{2} \frac{\sqrt{0.7x^2+1}\,dx}{2.1x+\sqrt{0.6x+1.7}}$;

III. Контрольные вопросы

1. Как осуществляется вычисление интеграла по формулам Ньютона- Котеса?
2. Как осуществляется вычисление интеграла по формулам Гаусса?

IV. Оформление отчёта

Отчет по лабораторно-практической работе составляется по следующей структуре:

1. Наименование лабораторной работы.
2. Цель работы.
3. Ответы на контрольные вопросы.
4. Вывод по работе.

Результаты работы в виде файлов представьте преподавателю.

Тема 6: Численное решение обыкновенных дифференциальных уравнений

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема работы: Решение обыкновенных дифференциальных уравнений при помощи формул Эйлера

Цель работы: *уметь:*

Решать обыкновенные дифференциальные уравнения при помощи формул Эйлера

Материально-техническое оснащение ПК, операционная система Windows.

Количество часов: 2 часа.

I. Теоретическая часть

Пусть дано уравнение (6) с начальным условием (7) (т.е. поставлена задача Коши).

$$y' = f(x,y) \qquad (6)$$

найти решение в виде функции y(x), удовлетворяющей начальному условию

$$y(x_0) = y_0 \qquad (7)$$

Вначале найдем простейшим способом приближенное значение решения в некоторой точке $x_1 = x_0 + h$, где h достаточно малый шаг. Заметим, что уравнение (6) совместно с начальным условием (7) задают направление касательной к искомой интегральной кривой в точке $M_0(x_0, y_0)$- Двигаясь вдоль этой касательной (рисунок 4), получим приближенное значение решения в точке x_1:

$$y_1 = y_0 + hf(x_0, y_0). \qquad (8)$$

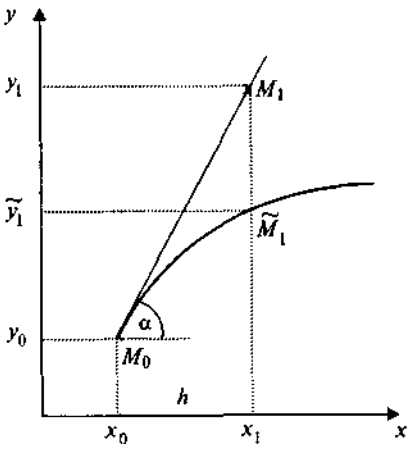


Рисунок 4 - Иллюстрация первого шага метода Эйлера

Располагая приближенным решением в точке $M_1(x_1,y_1)$, можно повторить описанную выше процедуру: построить прямую, проходящую через эту точку под углом, определяемым условием $\operatorname{tg}\beta = f(x_1, y_1)$, и по ней найти приближенное значение решения в точке $x_2 = x_1 + h$. Заметим, что, в отличие от ситуации, изображенной на рисунке 4, эта прямая не есть касательная к реальной интегральной кривой, поскольку точка $\tilde{M}_1$ нам недоступна. Однако представляется интуитивно ясным, что если h достаточно мало, то получаемые приближения будут близки к точным значениям решения.

Продолжая эту идею, построим систему равноотстоящих точек $x_i = x_0 + ih$ (i =0, 1,2,..., n). Получение таблицы значений искомой функции $y(x)$ по методу Эйлера заключается в циклическом применении пары формул:

$$\Delta y_i = hf(x_i, y_i); \quad y_{i+1} = y_i + \Delta y_i \quad (i = 0, 1, 2, \ldots, n). \tag{9}$$

Наиболее используемым эмпирическим методом оценки точности как метода Эйлера, так и других пошаговых методов приближенного численного интегрирования обыкновенных дифференциальных уравнений является способ двойного прохождения заданного отрезка — с шагом h и с шагом h/2. Совпадение соответствующих десятичных знаков в полученных двумя способами результатах дает эмпирические основание считать их верными (хотя полной уверенности в этом быть не может).

Метод Эйлера – Коши с уточнением по итерационной процедуре является более точным, чем ранее рассмотренные методы Эйлера. Сущность метода заключается в том, что каждое значение решения y_i уточняется по итерационной процедуре. Вначале выбирается грубое приближение

$$y_{i+1}^{(0)} = y_i + hf(x_i, y_i). \tag{10}$$

Затем строится итерационный процесс

$$y_{i+1}^{(k)} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f\left(x_{i+1}, y_{i+1}^{(k-1)}\right)\right]. \tag{11}$$

Итерации продолжаются до тех пор, пока два последовательных приближения $y_{i+1}^{(k)}$ и $y_{i+1}^{(k+1)}$ будут отличаться друг от друга не более, чем на наперед заданную малую величину E

$$\left|y_{i+1}^{(k+1)} - y_{i+1}^{(k)}\right| \le \mathrm{E}. \tag{12}$$

Если после трех-четырех итераций при выбранном значении h неравенство (12) не выполняется, то следует уменьшить шаг расчета. При выполнении неравенства (6.3) на первой-второй итерации, шаг интегрирования следует увеличить.

Программа

```
# include<stdio.h>
# include<math.h>
  float F(float x,float y)
   {return(x+y);}
    main()
     { float x,x1,y,h,b,e,z1,d,z2,z3;
      clrscr();
    printf("  Решение дифференциального уравнения усовершенствованным
метедом Эйлера-Коши");
      printf("                    с итерационным уточнением");
      printf("\n      Данные\n");
      scanf("%f %f %f %f %f",&x,&y,&h,&b,&e);
```

```
      printf("\t Результаты счета\n");
do { z1=y+h*F(x,y);
      printf("z1=%f\n",z1);
      x1=x+h;
      z3=z1;
  do {z2=z3;
       z3=y+h/2*(F(x,y)+F(x1,z2));
       d=fabs(z3-z2);
       printf("z3=%f d=%f\n",z3,d);
      }while(d>e);
     x=x1;y=z3;
     printf("x=%f y=%f d=%f\n",x,y,d);
    }while(x<b);
   }
```

II. Порядок выполнения работы

Задание 1. Решить методом Эйлера дифференциальное уравнение $y' = \cos y + 3x$ с начальным значением $y(0) = 1{,}3$ на отрезке [0; I], приняв шаг $h = 0{,}2$.

Результаты вычислений с двумя знаками после запятой приведены в таблице.

k	x_k	y_k	$\Delta y_k = 0,2(\cos y_k + 3x_k)$
0	0	1,3	0,05
1	0,2	1,35	0,16
2	0,4	1,52	0,25
3	0,6	1,77	0,32
4	0,8	2,09	0,38
5	1,0	2,47	

Порядок вычислений вполне очевиден: вначале находим

$\Delta y_0 = h(\cos y_0 + 3x_0)$, затем $y_1 = y_0 + \Delta y_0$ и т.д.

Задания: Выбрать вариант работы и составить программу решения.

1. Методом Эйлера решить дифференциальное уравнение $y' = 0{,}5xy$ при начальном условии $y(0)=1$ на отрезке [0; 0,6], приняв $h=0{,}2$. Вычислять погрешность интегрирования на каждом шаге и накопление погрешности, используя повторный расчет с половинным шагом $h_1=0{,}1$.

Ответ:

i	0	1	2
x	0	0,2	0,4
y	1	1,220±0,020	1,528±0,044

2. Применяя усовершенствованный метод Эйлера, найти на отрезке [0; 1] таблицу решения дифференциального уравнения $y'=y - \frac{2x}{y}$ при начальном условии y(0)=1, приняв h=0,2.

Ответ: $y_0=1$; $y_1=1,1836$; $y_2=1,3426$; $y_3=1,4850$; $y_4=1,6152$; $y_5=1,7362$.

3. Применяя усовершенствованный метод Эйлера-Коши, решить дифференциальное уравнение из упр.2.

Ответ: $y_0=1$; $y_1=1,1867$; $y_2=1,3484$; $y_3=1,4938$; $y_4=1,6272$; $y_5=1,7542$.

4. Применяя усовершенствованный метод Эйлера-Коши с итерационным уточнением, решить дифференциальное уравнение $y'=x + \sin\frac{y}{2,25}$, $y_0(1,4)=2,2$, $x\in[1,4; 1,6]$. Все вычисления вести с четырьмя десятичными знаками. Принять h=0,1, E=0,0005.

Ответ:

i	0	1	2
x	1,4	1,5	1,6
y	2,2	2,4306±0,0001	2,6761±0,0002

III. Контрольные вопросы

1. Понятие дифференциального уравнения и его решения.
2. Классификация методов приближенного решения обыкновенных дифференциальных уравнений.
3. Сущность аналитических методов решения дифференциальных уравнений. Достоинства и недостатки аналитических методов приближенного решения дифференциальных уравнений.
4. Метод Эйлера и его модификации.

IV. Оформление отчёта

Отчет по лабораторно-практической работе составляется по следующей структуре:

1. Наименование лабораторной работы.
2. Цель работы.
3. Общее описание используемых функций
4. Ответы на контрольные вопросы.
5. Вывод по работе.

Результаты работы в виде файлов представьте преподавателю.

Тема 6: Численное решение обыкновенных дифференциальных уравнений

ЛАБОРАТОРНО-ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема работы: Решение обыкновенных дифференциальных уравнений методом Рунге-Кутта

Цель работы: *уметь:*

- обеспечивать достоверность информации в процессе автоматизированной обработки данных;

Материально-техническое оснащение ПК, операционная система Windows.
Количество часов: 2 часа.

I.Теоретическая часть

Задача формулируется как и прежде. Пусть задано уравнение y′=f(x, y) с краевым условием $y(x_0)$. Требуется найти решение исходного уравнения.

Получение расчетных формул по методу Рунге – Кутта основано на следующих допущениях. Фиксируют некоторые числа:

$$\alpha_2,\dots,\alpha_q,\dots,\ p_1,\dots,\ p_q, b_{ij}, \qquad 0 < j < i \le q.$$

Последовательно полагаем:

$$\begin{aligned} &k_1(h) = hf(x, y),\\ &k_2(h) = hf\left(x+\alpha_2 h, y + p_{2,1}k_1(h)\right),\\ &\dots\dots\dots\dots\dots\dots\\ &k_q(h) = hf\left(x+\alpha_q h, y + p_{q,1}k_1(h) + \dots + \beta_{q,q-1}k_{q-1}(h)\right), \end{aligned} \tag{1}$$

а также:

$$y(x+h) = z(h) = y - \sum_{i=1}^{q} p_i k_i(h). \tag{2}$$

Рассмотрим вопрос о выборе параметров $\alpha_i, p_i, \beta_{i,j}$. Обозначим $R(h) = y(x+h) - z(h)$. Если f(x, y) – достаточно гладкие функции своих аргументов, то $k_1(h),\dots,k_q(h)$ и φ(h) – гладкие функции параметра h. Предположим также, что

$$\begin{aligned} &R(0) = R'(0) = \dots = R^{(s)}(0) = 0,\\ &R^{(s+1)}(0) \ne 0. \end{aligned} \tag{3а}$$

Согласно формуле Тейлора, справедливо равенство

$$R(h) = \sum_{i=0}^{s} \frac{R^{(i)}(0)}{i!} h^i + \frac{R^{(s+1)}(\theta h)}{(s+1)!} h^{(s+1)} = \frac{R^{(s+1)}(\theta h)}{(s+1)!} h^{(s+1)}, \tag{3}$$

где 0<θ<1. Величина R(h) называется погрешностью метода на шаге, а s – порядком погрешности метода. При q=1 имеем:

$$\begin{aligned} &R(h) = y(x+h) - y - p_1 hf(x, y);\\ &R(0) = 0;\\ &R'(0) = \left[y'(x+h) - p_1 f(x, y)\right]_{h=0} = f(x, y)(1 - p_1);\\ &R''(h) = y''(x+h). \end{aligned} \tag{4}$$

Здесь и далее y=y(x). Очевидно, что равенство R′(0)=0 выполняется при всех f(x, y) лишь в случае $p_1=1$. Этому значению p_1 соответствует метод Эйлера. Для погрешности этого метода, согласно (3) получаем:

$$R(h)=\frac{y''(x+\theta h)h^2}{2}. \qquad (5)$$

Рассмотрим случай q=2. Тогда по (2) имеем:

$$R(h)=y(x+h)-y-p_1hf(x,y)-p_2hf(\tilde{x},\tilde{y}), \qquad (6)$$

где $x=x+\alpha_2 h, \qquad \tilde{y}=y+\beta_{21}h\cdot f(x,y).$

Вычислим производные функции R(h)

$$R'(h)=y'(x+h)-p_1\cdot f(x,y)-p_2 f(\tilde{x},\tilde{y})-p_2h\left(\alpha_2\cdot f_x(\tilde{x},\tilde{y})+\beta_{21}\cdot f_y(\tilde{x},\tilde{y})\cdot f(x,y)\right), \qquad (7)$$

$$R''(h)=y''(x+h)-2p_2\left(\alpha_2\cdot f_x(\tilde{x},\tilde{y})+\beta_{21}\cdot f_y(\tilde{x},\tilde{y})\cdot f(x,y)-p_2h\left(\alpha_2^2\cdot f_{xx}(\tilde{x},\tilde{y})+\right.\right.$$
$$\left.\left.+2\alpha_2\beta_{21}f_{xy}(\tilde{x},\tilde{y})\cdot f(x,y)+\beta_{21}^2\cdot f_{yy}(\tilde{x},\tilde{y})\cdot(f(x,y))^2\right)\right), \qquad (8)$$

$$R'''(h)=y'''(x+h)-3p_2\left(\alpha_2^2\cdot f_{xx}(\tilde{x},\tilde{y})+2\alpha_2\beta_{21}f_{xy}(\tilde{x},\tilde{y})\cdot f(x,y)+\right.$$
$$\left.+\beta_{21}^2\cdot f_{yy}(\tilde{x},\tilde{y})\cdot(f(x,y))^2\right)+0(h), \qquad (9)$$

согласно исходному дифференциальному уравнению:

$$y'=f,\ y''=f_x+f_yf,\ y'''=f_{xx}+2f_{xy}f+f_{yy}f^2+f_yy''. \qquad (10)$$

Подставим в формулы (7 – 9) значение h=0 и воспользуемся для упрощения формулы (10). В результате получим:

$$R(0)=y-y=0,$$

$$R'(0)=(1-p_1-p_2)f(x,y), \qquad (11)$$

$$R''(0)=(1-2p_2\alpha_2)f_x(x,y)+(1-2p_2\beta_{21})f_y(x,y)f(x,y), \qquad (12)$$

$$R'''(0)=(1-3p_2\alpha_2^2)f_{xx}(x,y)+(2-6p_2\alpha_2\beta_{21})f_{xy}(x,y)f(x,y)+$$
$$+(1-3p_2\beta_{21}^2)\cdot f_{yy}(x,y)(f(x,y))^2+f_y(x,y)y''(x). \qquad (13)$$

Соотношение R′(0)=0 выполняется при всех f, если:

$$1-p_1-p_2=0. \qquad (14)$$

Соотношение R″ (0)=0, если

$$1-2p_2\alpha_2=0 \text{ и } 1-2p_2\beta_{21}=0. \qquad (15)$$

Таким образом R (0)=R′(0)=R″ (0)=0 при всех f(x, y), если выполнены три соотношения (14) и (15) относительно четырех параметров. Произвольно задавая один из них, получим различные формулы для решения исходного уравнения с погрешностью второго порядка малости по h. Например, при $p_1=\frac{1}{2}$ получаем $p_2=\frac{1}{2},\ \alpha_2=1,\ \beta_{21}=1,$

При $p_1=0$ получаем $p_2=1,\ \alpha_2=\frac{1}{2},\ \beta_{21}=\frac{1}{2},$

Для методов с q=s=2 главная часть погрешности на шаге есть величина $\frac{f''(0)}{6}h^3$. Пользуясь явным выражением $R'''(0)$, в ряде случаев удается уменьшить эту величину за счет удачного подбора параметров метода. Если для рассматриваемого класса уравнений величина f_yy'' мала, то разумно распорядиться этими параметрами так, чтобы в уравнении (13) обращались в

нуль первые три слагаемые. Нетрудно проверить, что при $p_1 = \frac{1}{4}$, $p_1 = \frac{3}{4}$, $\alpha_2 = \beta_{21} = \frac{2}{3}$ выполняются равенства (14) и (15) и одновременно $R'''(0) = f_y(x,y) \cdot y''(x)$. Этой совокупности параметров отвечают расчетные формулы:

$$k_1 = h \cdot f(x,y),$$
$$k_2 = h \cdot f\left(x + \frac{2}{3}h, y + \frac{2}{3}k_1\right), \qquad (16)$$
$$\Delta y = z(h) - y = \frac{1}{4}(k_1 + 3k_2).$$

Приведем результаты аналогичных выкладок для случая q=3. Расчетных формул, соответствующих значению s=4, не существует. Чтобы s равнялось трем, необходимо выполнение соотношений

$$\alpha_2 = \beta_{21}, \quad \alpha_3 = \beta_{31} + \beta_{32}, \quad \alpha_3(a_3 - a_2) - \beta_{32}\alpha_2 \cdot (2 - 3\alpha_2) = 0,$$
$$p_3\beta_{32}\alpha_2 = \frac{1}{6}, \quad p_2\alpha_2 + p_3\alpha_3 = \frac{1}{2}, \quad p_1 + p_2 + p_3 = 1. \qquad (17)$$

Эта система шести уравнений с восемью неизвестными имеет бесчисленное множество решений. Чаще других применяется совокупность формул

$$k_1 = hf(x,y),$$
$$k_2 = hf\left(x + \frac{h}{2}, y + \frac{k_1}{2}\right),$$
$$k_3 = hf(x + h, y - k_1 + 2k_2), \qquad (18)$$
$$\Delta y = \frac{1}{6}(k_1 + 4k_2 + k_3).$$

Если правая часть исходного уравнения не зависит от y, иначе f_y=0, то эта расчетная формула превращается в формулу Симпсона:

$$y(x+h) - y(x) = \frac{h}{6}\left(f(x) + 4f\left(x + \frac{h}{2}\right) + f(x+h)\right). \qquad (19)$$

Согласно оценке погрешности формулы Симпсона, погрешность этого приближения имеет порядок $0(h^5)$. Поэтому аналогично (16) формулы (18) следует применять при малых значениях f_y.

При q=4 не удается построить формулы по значениям s=5. При q=s=4 имеется двухпараметрическое множество расчетных формул. Для примера приведем одно параметрическое семейство таких формул

$$k_1 = hf(x,y),$$
$$k_2 = hf\left(x + \frac{h}{2}, y + \frac{k_1}{2}\right),$$
$$k_3 = hf\left(x + \frac{h}{2}, y + \left(\frac{1}{2} + \frac{1}{2t}\right)k_1 + \frac{1}{2t}k_2\right), \qquad (20)$$
$$k_4 = hf(x + h, y + (1-t)k_2 + tk_3),$$
$$\Delta y = \frac{1}{6}(k_1 + (4 - 2t)k_2 + 2tk_3 + k_4).$$

Наиболее применима совокупность формул этого семейства, соответствующая t=1:

$$
\begin{aligned}
k_1 &= hf(x, y),\\
k_2 &= hf\left(x + \frac{h}{2}, y + \frac{k_1}{2}\right),\\
k_3 &= hf\left(x + \frac{h}{2}, y + \frac{k_2}{2}\right),\\
k_4 &= hf(x + h, y + k_3),\\
\Delta y &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
\qquad (21)
$$

Среди других совокупностей формул со значениями q=s=4 отметим еще одну совокупность:

$$
\begin{aligned}
k_1 &= hf(x, y),\\
k_2 &= hf\left(x + \frac{h}{3}, y + \frac{k_1}{3}\right),\\
k_3 &= hf\left(x + \frac{2h}{3}, y - \frac{k_1}{3} + k_2\right),\\
k_4 &= hf(x + h, y + k_1 - k_2 + k_3),\\
\Delta y &= \frac{1}{8}(k_1 + 3k_2 + 3k_3 + k_4).
\end{aligned}
\qquad (22)
$$

Формулы (21) предпочтительнее формул (22) при $f_y<0$ и гладких решениях, поскольку они допускают интегрирование с более крупным шагом без потери слабой чувствительности метода к влиянию вычислительной погрешности. При малых значениях hf_y предпочтение нужно сделать в пользу формулы (22), как дающей более точный результат. Оценка погрешности формул (21), (22)

$$R_{10,57} = -\frac{f^{(4)}(x)h^5}{2880} + 0(h^6), \qquad (23)$$

$$R_{10,58} = -\frac{f^{(4)}(x)h^5}{6480} + 0(h^6). \qquad (24)$$

Однако формулы (23), (24) мало применимы и оценка погрешности производится по методу двойных расчетов как и ранее

$$|\hat{y}_i - y(x_i)| \le \frac{1}{15}|\hat{y}_i - y_i|, \qquad (25)$$

где $y(x_i)$ - точное решение исходного уравнения в точке x_i, а $\hat{y}_i$ и y_i - приближенные значения, полученные с шагом $\frac{h}{2}$ и h.

Если E - заданная точность решения, то шаг интегрирования выбирается таким образом, чтобы

$$h^4 < \mathrm{E}. \qquad (26)$$

Шаг расчета можно менять при переходе от одной точки к другой (чем меньше шагов, тем меньше ошибка накопления). Для оценки правильности выбора шага h используют равенство

$$q = \left|\frac{k_2^{(i)} - k_3^{(i)}}{k_1^{(i)} - k_2^{(i)}}\right|, \qquad (27)$$

где q должно быть равно нескольким сотым, в противном случае шаг h уменьшают.

Пример. Найти решение дифференциального уравнения $y' = \varphi(x, y) = x^2 + 4x - e^x - y$ на отрезке [0;1,2] методом Рунге-Кутта при начальных условиях $x_0 = 0$, $y_0 = -1$ с точностью $E_y = 0{,}001$.

Решение. В примере составлена программа, предназначенная для решения систем обыкновенных дифференциальных уравнений первого порядка следующего вида:

$$\left.\begin{aligned} \frac{dy_1}{dx} &= \varphi_1(x, y_1, y_2, \ldots, y_n) \\ \frac{dy_2}{dx} &= \varphi_2(x, y_1, y_2, \ldots, y_n) \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{dy_n}{dx} &= \varphi_n(x, y_1, y_2, \ldots, y_n) \end{aligned}\right\} \text{(A)}$$

с начальными условиями:

$x = x_0$, $y_1(x_0) = y_1$, $y_2(x_0) = y_2, \ldots, y_n(x_0) = y_n$.

В программе предусмотрено $1 \le n \le 10$. Интегрирование производится на отрезке $[x_0, x_k]$ с автоматическим выбором шага интегрирования по заданной погрешности решения E. Шаг печати результатов - $h_{печ}$ должен быть задан.
Конкретный вид правых частей в системе (A) задается в подпрограмме DIFUR.
Исходные данные, вводимые в программу, должны быть расположены в следующем порядке:

Номер строки ввода	I	2			
Величина	H	$y_{1,0}$	$y_{2,0}$	…	$y_{6,0}$
Формат записи	I2	E 11.4			

Номер строки ввода	3				4			
Величина	$y_{7,0}$	$y_{8,0}$	$y_{9,0}$	$y_{10,0}$	x_0	$x_к$	$h_{печ}$	E
Формат записи	E 11.4				E 11.4			

В результате вычислений по программе печатаются таблицы функций решения.

Распечатка программы и результаты расчетов по ней представлены ниже.

Программа .

```
# include<stdio.h>
# include<math.h>
  float runge();
  float difur(float x,float y[10],float f[10])
```

```
    {
        f[0]=x*x+4*x-exp(x)-y[0];return;}
   main()
    { float y0[10],y[10];
      float x0,xk,hp,eps,h,xh,xp;
      int i,n
      clrscr();
      printf("Программа решения системы обыкновенных дифференциальных
уравнений  \n");
      printf(" методом Рунге – Кутта четвертого порядка с автоматическим
выбором шага \n");
      scanf("%d",&n);
      printf("\n\t Исходные данные \n");
      for(i=0;i<n;i++)
        scanf("%f",&y0[i]);
      scanf("x0=%f xk=%f hp=%f eps=%f",x0,xk,hp,eps);
      for(i=0;i<n;i++)
        y[i]=y0[i];
      xh=x0;
      xp=x0+hp;
      h=(xk-x0)/100;
    d:runge;
      printf("\t\n Результаты счета \n");
      for(i=0;i<n;i++)
        printf("x=%f y[%d]=%f\n",xp,i,y[i]);
      if(xk>xp) xp=xh;xp=xp+hp;goto d;
    }
 float runge(int n,float h,float x0,float xk,float eps)
  { float y[10],z[10],f[10],ea[10],a[4],r[10][4],ak[10][4],x,am;
    int i,j;
    x=x0;
    a[0]=0;
 l:  a[1]=h/2;
    a[2]=h/2 ;
    a[3]=h;
    for(j=0;j<4;j++)
      { for(i=0;i<n;i++)
          { r[i][j]=0;return;
           r[i][j]=h*ak[i][j-1]/2;return;
           r[i][j]=h*ak[i][j-1]/2;return;
           r[i][j]=h*ak[i][j-1];return;
          }
          for(i=0;i<n;i++) z[i]=y[i]+r[i][j];
          difur;
          for(i=0;i<n;i++)
            ak[i][j]=f[i];
      }
```

```
     for(i=0;i<0;i++)
      ea[i]=fabs((ak[i][2]-ak[i][3])/(ak[i][1]-ak[i][2]));
    am=ea[0];
    for(i=0;i<0;i++)
     { if(ea[i]<=am) return;am=ea[i];}
    if(am<=eps) x=x+h;
      else goto k;
    for(i=0;i<n;i++)
     y[i]=y[i]+h*(ak[i][0]+2*ak[i][1]+2*ak[i][2]+ak[i][3])/6;
     if(am<eps/87) h=1.5*h ;
         else  if(x>=xk) goto r;
          else goto l;
    k:h=h/3; goto l;
    r:return;
  }
```

Результат.

Программа решения системы обыкновенных дифференциальных уравнений

методом Рунге – Кутта четвертого порядка с автоматическим выбором шага

n=1
y0[1]=-1.000000
x_0=0.000000 x_k=1.200000 h_p=0.100000 eps=0.001000

Результаты счета

x=0.100000 y[i]=-0.985300
x=0.200000 y[i]=-0.941900
x=0.300000 y[i]=-0.871900
x=0.400000 y[i]=-0.777400
x=0.500000 y[i]=-0.660200
x=0.600000 y[i]=-0.523900
x=0.700000 y[i]=-0.368400
x=0.800000 y[i]=-0.195500
x=0.900000 y[i]=-0.007000
x=1.000000 y[i]= 0.195300
x=1.100000 y[i]= 0.409600
x=1.200000 y[i]= 0.633900

II. Порядок выполнения работы

Методом Рунге-Кутта, приняв h=0,1, найти решение дифференциального уравнения $y'=x+y^2$, если y(1)=0, $x\in[1;\ 1,5]$.Составить программу решения данного примера.

Ответ: $y_0=0$; $y_1=1,10536$; $y_2=1,223314$; $y_3=1,35660$; $y_4=1,51042$; $y_5=1,69150$.

III. Контрольные вопросы

1. Понятие дифференциального уравнения и его решения.
2. Классификация методов приближенного решения обыкновенных дифференциальных уравнений.
3. Сущность аналитических методов решения дифференциальных уравнений.

4. Достоинства и недостатки аналитических методов приближенного решения дифференциальных уравнений.
5. Методы Рунге-Кутта.

.

IV. Оформление отчёта

Отчет по лабораторно-практической работе составляется по следующей структуре:

1. Наименование лабораторной работы.
2. Цель работы.
3. Общее описание используемых функций
4. Ответы на контрольные вопросы.
5. Вывод по работе.

Результаты работы в виде файлов представьте преподавателю.

Минобрнауки России
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению самостоятельных работ

по дисциплине
ЧИСЛЕННЫЕ МЕТОДЫ

специальности СПО
09.02.07 Информационные системы и программирование

Тула 2021

УТВЕРЖДЕНЫ
Цикловой комиссий информационных технологий
Протокол от «14» октября 2021 г. № 3
Председатель цикловой комиссии И.В. Миляева

Авторы: Воронцова Н.В., канд.техн.наук.

Тема 1. Элементы теории погрешностей

Работа 1

Задание. 1) Определить, какое равенство точнее.

2) Округлить сомнительные цифры числа, оставив верные знаки: а) в узком смысле; б) в широком смысле. Определить абсолютную погрешность результата.

3) Найти предельные абсолютные и относительные погрешности чисел, если они имеют только верные цифры: а)в узком смысле; б) в широком смысле.

№ 1. 1)$\sqrt{44} = 6{,}63$; $19/41 = 0{,}463$.
2) а)$22{,}553\ (\pm 0{,}016)$
б)$2{,}8546$; $\delta = 0{,}3\%$.
3) а) $0{,}2387$; б) $42{,}884$.

№ 2. 1)$7/15 = 0{,}467$; $\sqrt{30} = 5{,}48$.
2) а)$17{,}2834$; $\delta = 0{,}3\%$.
б)$6{,}4257\ (\pm 0{,}0024)$.
3) а) $3{,}751$; б) $0{,}537$.

№ 3. 1)$\sqrt{10{,}5} = 3{,}24$; $4/17$=0325
2) а) $34{,}834$; $\delta = 0{,}1\%$.
б)$0{,}5748\ (\pm 0{,}0034)$.
3) а)$11{,}445$; б)$2{,}043$.

№ 4. 1)$15/7 = 2{,}14$;$\sqrt{10} = 3{,}16$.
2) а)$2{,}3485\ (\pm 0{,}0042)$;
б)$0{,}34484$; $\delta = 0{,}4\%$.
3) а)$2{,}3445$; б)$0{,}745$.

№ 5. 1) $6/7 = 0{,}857$; $\sqrt{4{,}8} = 2{,}19$.
2) а) $5{,}435\ (\pm 0{,}0028)$;
б) $10{,}8441$; $\delta = 0{,}5\%$.
3) а)$8{,}345$ б) $0{,}288$.

№ 6. 1) $12/11 = 1{,}091$; $\sqrt{6{,}8} = 2{,}61$.
2) а) $8{,}24163$; $\delta = 0{,}2\%$
б) $0{,}12356\ (\pm 0{,}00036)$.
3) а)$12{,}45$; б) $3{,}4453$.

№ 7. 1)$2/21 = 0{,}095$; $\sqrt{22} = 4{,}69$.
2) а)$2{,}4543\ (\pm 0{,}0032)$;
б)$24{,}5643$; $\delta = 0{,}1\%$.
3) а)$0{,}374$; б) $4{,}348$.

№ 8. 1) $23/15 = 1{,}53$; $\sqrt{9{,}8} = 3{,}13$.
2) а) $23{,}574$; $\delta = 0{,}2\%$;
б) $8{,}3445\ (\pm 0{,}0022)$.
3) а)$20{,}43$; б)$0{,}576$.

№ 9. 1)$6/11 = 0{,}545$; $\sqrt{83} = 9{,}11$
2) а)$21{,}68563$; $\delta = 0{,}3\%$
б)$3{,}7834\ (\pm 0{,}0041)$.
3) а)$41{,}72$; б)$0{,}678$.

№ 10. 1)$17/19 = 0{,}895$; $\sqrt{52} = 7{,}21$.
2) а)$13{,}537\ (\pm 0{,}0026)$;
б)$7{,}521$; $\delta = 0{,}12\%$.
3) а)$5{,}634$; б)$0{,}0748$.

№ 11. 1)$21/29 = 0{,}723$;$\sqrt{44} = 6{,}63$.
2) а)$0{,}3567$; $\delta = 0{,}042\%$;
б)$13{,}6253\ (\pm 0{,}0021)$.
3) а)$18{,}357$; б)$2{,}16$.

№ 12. 1)$50/19 = 2{,}63$; $\sqrt{27} = 5{,}19$.
2) а)$1{,}784\ (\pm 0{,}0063)$;
б)$0{,}85637$; $\delta = 0{,}21\%$.
3) а)$0{,}5746$; б)$236{,}58$.

№ 13. 1)$13/17 = 0{,}764$; $\sqrt{31} = 5{,}56$.
2) а)$3{,}6878\ (\pm 0{,}0013)$;
б)$15{,}873$; $\delta = 0{,}42\%$;
3) а)$14{,}862$; б)$8{,}73$.

№ 14. 1)$7/22 = 0{,}318$; $\sqrt{13} = 3{,}60$.
2) а)$27{,}1548\ (\pm 0{,}0016)$;
б)$0{,}3945$; $\delta = 0{,}16\%$.
3) а)$0{,}3648$; б)$21{,}7$.

№ 15. 1)$17{,}11 = 1{,}545$; $\sqrt{18} = 4{,}24$.

№ 16. 1)$5/3 = 1{,}667$; $\sqrt{38} = 6{,}16$.

2) а)0,8647 ($\pm$0,0013);
б)24,3618; $\delta = 0{,}22\%$.
3) а)2,4516; б)0,863.

№ 17. 1)49,13 = 3,77; $\sqrt{14} = 3{,}74$.
2) а)83,736; $\delta = 0{,}085\%$.
б)5,6483 ($\pm$0,0017).
3) а)5,6432; б)0,00858.

№ 19. 1)19/12 = 1,58; $\sqrt{12} = 3{,}46$.
2) а)4,88445 ($\pm$0,00052);
б)0,096835; $\delta = 0{,}32\%$.
3) а)12,688; б)4,636.

№ 21. 1)18/7 = 2,57; $\sqrt{22} = 4{,}69$.
2) а)0,39642 ($\pm$0,00022);
б)46,453; $\delta = 0{,}15\%$.
3) а)15,644; б)6,125.

№ 23. 1)16/7 = 2,28; $\sqrt{11} = 3{,}32$.
2) а)24,3872; $\delta = 0{,}34\%$.
б)0,75244 ($\pm$0,00013).
3) а)16,383; б)5,734.

№ 25. 1)12/7 = 1,71; $\sqrt{47} = 6{,}86$.
2) а)72,354; $\delta = 0{,}24\%$;
б)0,38725 ($\pm$0,00112).
3) а)18,275; б)0,00644.

№ 27. 1)23/9 = 2,56; $\sqrt{87} = 9{,}33$.
2) а)23,7564; $\delta = 0{,}44\%$;
б)4,57633 ($\pm$0,00042).
3) а)3,75; б)6,8343.

№ 29. 1)7/3 = 2,33; $\sqrt{58} = 7{,}61$.
2) а)3,87683; $\delta = 0{,}33\%$.
б)13,5726 ($\pm$0,0072).
3) а)26,3; б)4,8556.

2) а)3,7542; $\delta = 0{,}32\%$.
б)0,98351 ($\pm$0,00042).
3) а)62,74; б)0,389.

№ 18. 1)13/7 = 1,857; $\sqrt{7} = 2{,}64$.
2) а)2,8867; $\delta = 0{,}43\%$.
б)32,7486 ($\pm$0,0012).
3) а)0,0384; б)63,745.

№ 20. 1)51/11 = 4,64; $\sqrt{35} = 5{,}91$.
2) а)38,4258 ($\pm$0,0014);
б)0,66385; $\delta = 0{,}34\%$.
3) а)6,743; б)0,543.

№ 22. 1)19/9 = 2,11; $\sqrt{17} = 4{,}12$.
2) а)5,8425; $\delta = 0{,}23\%$.
б)0,66385 ($\pm$0,00042).
3) а)0,3825; б)24,6.

№ 24. 1)20/13 = 1,54; $\sqrt{63} = 7{,}94$.
2) а)2,3684 ($\pm$0,0017);
б)45,7832; $\delta = 0{,}18\%$.
3) а)0,573; б)3,6761.

№ 26. 1)6/7 = 0,857; $\sqrt{41} = 6{,}40$.
2) а)0,36127 ($\pm$0,00034);
б)46,7843; $\delta = 0{,}32\%$.
3) а)3,425; б)7,38.

№ 28. 1)27/31 = 0,872; $\sqrt{42} = 6{,}48$.
2) а)15,8372 ($\pm$0,0026);
б)0,088748; $\delta = 0{,}56\%$.
3) а)3,643; б)72,385.

№ 30. 1)14/17 = 0,823; $\sqrt{53} = 7{,}28$.
2) а)0,66835 ($\pm$0,00115);
б)23,3748; $\delta = 0{,}27\%$.
3) а)43,813; б)0,645.

Образец выполнения задания

1) 9/11 = 0,818; $\sqrt{18} = 4{,}24$; 2) а) 72,353 ($\pm$0,026); б) 2,3544; $\delta = 0{,}2\%$; 3) а) 0,4357; б)12,384.

---

1) Находим значения данных выражений с большим числом десятичных знаков: $a_1 = 9/11 = 0{,}81818\ldots$, $a_2 = \sqrt{18} = 4{,}2426\ldots$. Затем вычисляем предельные абсолютные погрешности, округляя их с избытком;

$$\alpha_{a_1} = |0{,}81818 - 0{,}818| \leq 0{,}00019.\ \alpha_{a_2} = |4.2426 - 4.24| \leq 0{,}0027,$$

Предельные относительные погрешности составляют

$$\delta_{a_1} = \frac{\alpha_{a_1}}{a_1} = \frac{0.00019}{0.818} = 0.00024 = 0.024\%$$

$$\delta_{a_1} = \frac{\alpha_{a_2}}{a_2} = \frac{0.0027}{4.24} = 0.00064 = 0.064\%$$

Так как $\delta_{a_1} < \delta_{a_2}$, то равенство $9/11 = 0.818$ является более точным.

2) а) Пусть $72.353\ (\pm 0.026) = a$. Согласно условию, погрешность $\alpha_a = 0.026 < 0.05$; это означает что в числе 72,353 верными в узком смысле являются цифры 7, 2, 3. По правилам округления найдем приближенное значение числа, сохранив десятые доли:

$$a_1 = 72.4;\ \alpha_{a_1} = \alpha_a = \Delta_{\text{окр}} = 0{,}026 + 0{,}047 = 0{,}073$$

Полученная погрешность больше 0,05; значит, нужно уменьшить число цифр в приближенном числе до двух:

$$a_2 = 72;\ \alpha_{a_2} = \alpha_a = \Delta_{\text{окр}} = 0{,}026 + 0{,}353 = 0{,}379$$

Так как $\alpha_{a_2} < 0{,}5$, то обе оставшиеся цифры верны в узком смысле.

б) Пусть $a = 2.3544$; $\delta_a = 0.2\%$; тогда $\alpha_a = a * \delta_a = 0.00471$. В данном числе верными в широком смысле являются три цифры, по этому округляем его, сохраняя эти цифры:

$$a_1 = 2.35; \alpha_{a_1} = 0.0044 + 0.00471 = 0.00911 < 0{,}01.$$

Значит, и в округленном числе 2,35 все три цифры верны в широком смысле.

3) а) Так как все четыре числа $a = 0.4357$ верны в узком смысле, то абсолютная погрешность $\alpha_{a=}0{,}00005$, а относительная погрешность $\delta_a = 1/(2 * 4 * 10^3) = 0.000125 = 0.0125\%$.

Б) Так как все пять цифр числа $a = 12.384$ верны в широком смысле, то $\alpha_a = 0{,}001$; $\delta_a = 1/(1 * 10^4) = 0.0001 = 0.01\%$.

Работа 2

Задание. 1) Вычислить и определить погрешности результата.
2) Вычислить и определить погрешности результата.
3) Вычислить, пользуясь правилами подсчета цифр.

№ 1. 1) $X = \frac{ab}{\sqrt[3]{c}}$

	а	б	в
a	3.85(±0.01)	4.16 (±0.005)	7,27 (±0,01)
b	2.0435(±0.0004)	12.163 (±0.002)	5,205 (±0,002)
c	962,6(±0.1)	55.18 (±0,01)	87,32(±0,03)

2) $X = \left[\frac{(a+b)c}{m-n}\right]^2$

	а	б	в
a	4,3 (±0,05)	5,2 (±0,04)	2,13(±0,01)
b	17,21 (±0,02)	15,32 (±0,01)	22,16(±0,03)
c	8,2 (±0,05)	7,5 (±0,05)	6,3(±0,04)
m	12,417 (±0,003)	21,823 (±0,002)	16,825(±0,004)
n	8,37 (±0,005)	7,56(±0,003)	8,13(±0,002)

3) $S = \frac{h^2}{18} : \frac{a^2+4ab+b^2}{(a+b)^2}$

	а	б	в
a	1,141	2,234	5,813
b	3,156	4,518	1,315
h	1,14	4,48	2,56

№ 2. 1)$X = \frac{\sqrt{a} * b}{c}$

	а	б	в
a	228,6(±0,06)	315,6(±0,05)	186,7(±0,04)
b	86,4(±0,02)	72,5(±0,03)	66,6(±0,02)
c	68,7(±0,05)	53,8(±0,04)	72,3(±0,03)

2) $X = \frac{m^3(a+b)}{c-d}$

	а	б	в
a	13,5(±0,02)	18,5(±0,03)	11,8(±0,02)
b	3,7(±0,02)	5,6(±0,02)	7,4(±0,03)
m	4,22(±0,004)	3,42(±0,003)	5,82(±0,005)
c	34,5(±0,02)	26,3(±0,01)	26,7(±0,03)
d	23,725(±0,005)	14,782(±0,006)	11,234(±0,004)

3) $M = \frac{(a+b)h^3}{4} + \frac{(a+b)h}{12}$

	а	б	в
a	8,53	6,44	9,05
b	6,271	5,323	3,244
h	12,48	15,44	20,18

№ 3. 1) $X = \frac{\sqrt{ab}}{c}$

	а	б	в
a	3,845(±0,004)	4,632(±0,003)	2,312(±0,004)
h	16,2(±0,05)	23,3(±0,04)	18,4(±0,03)
c	10,8(±0,1)	11,3(±0,06)	20,2(±0,08)

2) $X = \frac{(a+b)m}{(c-d)^2}$

	а	б	в
a	2,754(±0,001)	3,236(±0,002)	4,523(±0,003)
b	11,7(±0,04)	15,8(±0,03)	10,8(±0,02)
m	0,56(±0,005)	0,64(±0,004)	0,85(±0,003)
c	10,536(±0,002)	12,415(±0,003)	9,318(±0,002)
d	6,32(±0,008)	7,18(±0,006)	4,17(±0,004)

3) $N = \frac{(a+b)^2}{2h} + \frac{(a^2+b^2)h}{5}$

	а	б	в
a	0,562	0,834	0,445
b	0,2518	0,3523	0,4834
h	0,68	0,74	0,87

№ 4. 1)$X = \frac{a^2b}{c}$

	а	б	в
a	3,456(±0,002)	1,245(±0,001)	0,327(±0,005)
b	0,642(±0,0005)	0,121(±0,0002)	3,147(±0,0001)
c	7,12(±0,004)	2,34(±0,003)	1,78(±0,001)

2) $X = \frac{(a+b)m}{\sqrt{c-d}}$

	а	б	в
a	23,16(±0,02)	17,41(±0,01)	32,37(±0,03)
b	8,23(±0,005)	1,27(±0,002)	2,35(±0,001)
c	145,5(±0,08)	342(±0,04)	128,7(±0,02)
d	28,6(±0,1)	11,7(±01)	27,3(±0,04)
m	0,28(±0,006)	0,71(±0,003)	0,93(±0,001)

3) $V = \frac{h}{3} * S\left(1 + \frac{a}{A} + \frac{a^2}{A^2}\right)$

	а	б	в

a	8,51	5,71	7,28
A	23,42	32,17	11,71
S	45,8	51,7	21,8
h	3,81	2,42	5,31

№ 5. 1)$X = \frac{ab^3}{c}$

	а	б	в
a	0,643(±0,0005)	0,142(±0,0003)	0,258(±0,0002)
b	2,17(±0,002)	1,71(±0,002)	3,45(±0,001)
c	5,843(±0,001)	3,727(±0,001)	7,221(±0,003)

2) $X = \frac{(a-b)c}{\sqrt{m+n}}$

	а	б	в
a	27,16(±0,006)	15,71(±0,005)	12,31(±0,004)
b	5,03(±0,01)	3,28(±0,02)	1,73(±0,03)
c	3,6(±0,02)	7,2(±0,01)	3,7(±0,02)
m	12,375(±0,004)	13,752(±0,001)	17,428(±0,003)
n	86,2(±0,05)	33,7(±0,03)	41,7(±0,01)

3) $S = \frac{h^2}{18} * \frac{a^2+4ab+b^3}{(a+b)^2}$

	а	б	в
h	21,1	17,8	32,5
a	22,08	32,47	27,51
b	31,11	11,42	21,78

№ 6. 1) $X = \frac{ab}{c^2}$

	а	б	в
a	0,3575(±0,0002)	0,1756(±0,0001)	0,2731(±0,0003)
b	2,63(±0,01)	3,71(±0,03)	5,12(±0,02)
c	0,854(±0,0005)	0,285(±0,0002)	0,374(±0,0001)

2) $X = \frac{a+b}{\sqrt{(c-d)m}}$

	а	б	в
a	16,342(±0,001)	12,751(±0,001)	31,456(±0,002)
b	2,5(±0,03)	3,7(±0,02)	7,3(±0,01)
c	38,17(±0,002)	23,76(±0,003)	33,28(±0,003)
d	9,14(±0,005)	8,12(±0,004)	6,71(±0,001)
m	3,6(±0,04)	1,7(±0,01)	5,8(±0,02)

3) $V = \frac{1}{6}\pi h(3a^2 + h^2)$

	а	б	в
a	2,456	7,751	5,441
h	1,76	3,35	6,17

№ 7. 1) $V = \frac{\pi^2}{4} Dd^2$

	а	б	в
π	3,14	3,14	3,14
D	54(±0,5)	72(±0,3)	31(±0,01)
d	8,235(±0,001)	3,274(±0,002)	7,345(±0,001)

2) $S = \frac{1}{64}\pi\sqrt{D^4 - d^4}$

	а	б	в
D	36,5(±0,1)	41,4(±0,2)	52,6(±0,01)
d	26,35(±0,005)	31,75(±0,003)	48,39(±0,001)
π	3,14	3,14	3,14

3) $a = c^2\left(1 + \frac{2\beta}{c} + \frac{\gamma^2}{c^2}\right)$

	а	б	в
c’	2,435	7,834	4,539
β	0,15	0,21	0,34
γ	1,27	3,71	5,93

№ 8. 1) $\gamma = \frac{m^2 n}{c^3}$

	а	б	в
m	1,6531(±0,0003)	2,348(±0,002)	3,804(±0,003)
n	3,78(±0,002)	4,37(±0,004)	4,05(±0,003)
c	0,158(±0,0005)	0,235(±0,0003)	0,318(±0,0002)

2) $X = \frac{m\sqrt{a-b}}{c+d}$

	а	б	в
a	9,542(±0,001)	8,357(±0,003)	4,218(±0,001)
b	3,128(±0,002)	2,48(±0,004)	1,57(±0,006)
m	2,8(±0,03)	3,17(±0,01)	2,32(±0,02)
c	0,172(±0,001)	1,315(±0,0004)	2,418(±0,004)
d	5,4(±0,02)	2,4(±0,02)	1,8(±0,01)

3) $V = \frac{1}{15}\pi h(2D^2 + Dd + 0{,}75d^2)$

	а	б	в
h	84,2	76	45
D	28,3	17,2	48,3
d	42,08	9,344	32,14

№ 9. 1) $X = \sqrt{\frac{cd}{b}}$

	а	б	в
c	0,7568(±0,0002)	0,8345(±0,0004)	0,6384(±0,0002)
d	21,7(±0,02)	13,8(±0,03)	32,7(±0,04)
b	2,65(±0,01)	1,84(±0,006)	4,88(±0,03

2) $x = \frac{\sqrt[3]{a-b}}{m(n-a)}$

	а	б	в
a	10,82(±0,03)	9,37(±0,004)	11,45(±0,01)
b	2,786(±0,0006)	3,108(±0,0003)	4,431(±0,002)
m	0,28(±0,006)	0,46(±0,002)	0,75(±0,003)
n	14,7(±0,06)	15,2(±0,04)	16,7(±0,05)

3) $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a+b+c)/2$

	а	б	в
a	46,3	10,5	2,48
b	29,72	34,18	5,344
c	37,654	27,327	6,0218

№ 10. 1) $f = \frac{Qc^3}{48E}$

	а	б	в
Q	54,8(±0,02)	38,5(±0,01)	17,3(±0,07)
e	2,45(±0,01)	3,35(±0,02)	5,73(±0,01)
E	0,863(±0,004)	0,734(±0,001)	0,956(±0,004)

2) $Q = \frac{(2n-1)^2(x+y)}{x-y}$

	а	б	в
n	2,0435(±0,0001)	1,1753(±0,0002)	4,5681(±0,0001)
x	4,2(±0,05)	5,8(±0,01)	6,3(±0,02)
y	0,82(±0,01)	0,65(±0,02)	0,42(±0,03)

3)$\gamma = \frac{ab-\beta a}{b^2} - \frac{\beta(ab-\beta a)}{b^2(b+\beta)}$

	а	б	в
α	5,27	7,31	3,28
β	0,0562	0,0761	0,0545
a	158,35	234,36	341,17
b	61,21	87,26	52,34

Образец выполнения задания

1) $X = \frac{m^2 n^3}{\sqrt{k}}$, где $m = 28.3(\pm 0.02)$, $n = 7.45(\pm 0.01)$, $k = 0.678(\pm 0.02)$

2) $N = \frac{(n-1)(m+n)}{(m-n)^2}$, где $n = 3.0567(\pm 0{,}0001)$, $m = 5.72(\pm 0{,}02)$

3) $V = \pi h^2 \left(R - \frac{h}{3}\right)$, где $h = 11.8$, $R = 23.67$.

1) Находим $m^2 = 800{,}9$; $n^3 = 413{,}5$; $\sqrt{k} = 0{,}8234$; $X = \frac{800{,}9 * 413.5}{0{,}8234} = 402{,}200 = 4{,}02 * 10^5$.

Далее, имеем $\delta_m = 0{,}02/28{,}3 = 0{,}00071$; $\delta_n = 0{,}01/7{,}45 = 0{,}00135$; $\delta_k = 0{,}003/0{,}678 = 0{,}00443$, откуда

$\delta_X = 2\delta_m + 3\delta_n + 0{,}5\delta_k = 0{,}00142 + 0{,}00405 + 0{,}00222 = 0{,}00769 = 0{,}77\%$;

$a_X = 4{,}02 * 10^5 * 0{,}0077 = 3{,}1 * 10^3$.

Ответ: $X = 4{,}02 * 10^5 (\pm 3{,}1 * 10^3)$; $\delta_X = 0{,}77\%$.

2) Имеем $n - 1 = 2{,}0567(\pm 0{,}0001)$;

$m + n = 3{,}057(\pm 0{,}0004) + 5{,}72(\pm 0{,}02) = 8{,}777(\pm 0{,}0204)$;

$m - n = 5{,}72(\pm 0{,}02) - 3{,}057(\pm 0m0004) = 2{,}663(\pm 0{,}0204)$;

$N = \frac{2{,}0567 * 8{,}777}{2{,}663^2} = \frac{2{,}0567 * 8{,}777}{7{,}092} = 2{,}545 \approx 2{,}55$;

$\delta_N = \frac{0{,}0001}{2{,}0567} + \frac{0{,}0204}{8{,}777} + 2\frac{0{,}0204}{2{,}663} = 0{,}000049 + 0{,}00233 + 2 * 0{,}00766 = 0{,}00238 + 0{,}01532 = 0{,}017 = 1{,}77\%$

; $a_N = 2{,}55 * 0{,}0177 = 0{,}046$.

Ответ: $N \approx 2{,}55(\pm 0{,}046)$; $\delta_N = 1.77\%$

3) Находим

$V = 3{,}142 * 11{,}8^2(23{,}67 - 3{,}933) = 3{,}142 * 11{,}8^2 * 19{,}737 = 3{,}142 * 139{,}2 * 19{,}737$
$= 437{,}37 * 19{,}737 = 8630 \approx 8{,}63 * 10^5$

Ответ: $V \approx 8{,}63 * 10^5$.

Тема 2. Приближённые решения алгебраических и трансцендентных уравнений

Работа 1

Задание: 1) Отделить корни аналитически.

2) Отделить корни аналитически и уточнить одни из них методом проб с точностью до 0,01.

3) Отделить корни графически.

4) Отделить корни графически и уточнить одни из них методом проб с точностью до 0,01

№ 1. 1) $2^x + 5x = 3 - 0$;

2) $3x^4 + 4x^3 - 12x^2 - 5 = 0$;

3) $0.5^x + 1 = (x - 2)^2$;

4) $(x - 3)\cos x = 1.5 - 2\pi \le x \le 2\pi$.

№ 2. 1) $\operatorname{arctg} x - \frac{1}{3x^3} = 0$;

2) $2x^3 - 9x^2 - 60x + 1 = 0$;

3) $[\log_2(-x)] * (x + 2) = -1$;

4) $\sin\left(x + \frac{\pi}{3}\right) - 0.5x = 0$.

№ 3. 1) $5^x + 3x = 0$;

2) $x^4 - x - 1 = 0$;

3) $x^2 - 2 + 0.5^x = 0$;

4) $(x - 1)^2 * \lg(x + 11) = 1$.

№ 4. 1) $2e^x = 5x + 2$;

2) $2x^4 - x^2 - 10 = 0$;

3) $x * \log_3(x + 1) = 1$;

4) $\cos(x + 5) = x^3$.

№ 5. 1) $3^{x-1} - 2 - x = 0$;

№ 6. 1) $2\operatorname{arctg} x - \frac{1}{2x^3} = 0$;

2) $3x^4 + 8x^3 + 6x^2 - 10 = 0$;
3) $(x-4)^2 * \log_{0.5}(x-3) = -1$;
4) $5\sin x = x$.

№ 7. 1) $e^{-2x} - 2x + 1 = 0$;
2) $x^4 + 4x^3 - 12x^2 - 17 = 0$;
3) $0.5^x - 1 = (x+2)^2$;
4) $x^2 \cos 2x = -1$.

№ 8. 1) $5^x - 6x - 3 = 0$;
2) $x^4 - x^3 - 2x^2 + 3x - 3 = 0$;
3) $2x^2 - 0.5^x - 3 = 0$;
4) $x\lg(x+1) = 1$.

№ 9. 1) $\operatorname{arctg}(x-1) + 2x = 0$;
2) $3x^4 + 4x^3 - 12x^2 + 1 = 0$;
3) $(x-2)^2 2^x = 1$;
4) $x^2 - 20\sin x = 0$.

№ 10. 1) $2\operatorname{arcctg} x - x + 3 = 0$;
2) $3x^4 - 8x^3 - 18x^2 + 2 = 0$;
3) $2\sin\left(x + \frac{\pi}{3}\right) = 0.5x^2 - 1$;
4) $2\lg x - \frac{x}{2} + 1 = 0$.

№ 11. 1) $3^x + 2x - 2 = 0$;
2) $2x^4 - 8x^3 + 8x^2 - 1 = 0$;
3) $[9x-2)^2 - 1]2^x = 1$;
4) $(x-2)\cos x = 1. -2\pi \le x \le 2\pi$.

№ 12. 1) $2\operatorname{argtg} x - 3x + 2 = 0$;
2) $2x^4 + 8x^3 + 8x^2 - 1 = 0$;
3) $[\log_2(x+2)](x-1) = 1$;
4) $\sin(x-0.5) - x + 0.8 = 0$.

№ 13. 1) $3^x + 2x - 5 = 0$;
2) $x^4 - 4x^3 - 8x^2 + 1 = 0$;
3) $x^3 - 3 + 0.5^x = 0$;
4) $(x-2)^2 \lg(x+11) = 1$.

№ 14. 1) $2e^x + 3x + 1 = 0$;
2) $3x^4 + 4x^3 - 12x^2 - 5 = 0$;
3) $x\log_3(x+1) = 2$;
4) $\cos(x+0.3) = x^2$.

№ 15. 1) $3^{x-1} - 4 - x = 0$;
2) $2x^3 - 9x^2 - 60x + 1 = 0$;
3) $(x-3)^2 \log_{0.5}(x-2) = -1$;
4) $5\sin x = x - 1$.

№ 16. 1) $\operatorname{arctg} x - \frac{1}{3x^3} = 0$;
2) $x^4 - x - 1 = 0$;
3) $(x-1)^2 2^x = 1$;
4) $\operatorname{tg}^3 x = x - 1. -\pi/2 \le x \le \pi/2$.

№ 17. 1) $e^x + x + 1 = 0$;
2) $2x^4 - x^2 - 10 = 0$;
3) $0.5^x - 3 = (x+2)^2$;
4) $x^2 \cos 2x = -1, -2\pi \le x \le 2\pi$.

№ 18. 1) $3^x - 2x + 5 = 0$;
2) $3x^4 + 8x^3 + 6x^2 - 10 = 0$;
3) $2x^2 - 0.5^x - 2 = 0$;
4) $x\lg(x+1) = 1$.

№ 19. 1) $\operatorname{arctg}(x-1) + 3x - 2 = 0$;
2) $x^4 - 18x^2 + 6 = 0$;
3) $(x-2)^2 2^x = 1$;

№ 20. 1) $2\operatorname{arcctg} x - x + 3 = 0$;
2) $x^4 + 4x^3 - 8x^2 - 17 = 0$;
3) $2\sin\left(x + \frac{\pi}{3}\right) = x^2 - 0{,}5$;

4) $x^2 - 20\sin x = 0$.

4) $2\lg x - \frac{x}{2} + 1 = 0$.

№ 21. 1) $2^x - 3x - 2 = 0$;
2) $x^4 - x^3 - 2x^2 + 3x - 3 = 0$;
3) $(0{,}5)^x + 1 = (x-2)^2$;
4) $(x-3)\cos x = 1$.

№ 22. 1) $\operatorname{arcctg} x + 2x - 1 = 0$;
2) $3x^4 + 4x^3 - 12x^2 + 1 = 0$;
3) $(x+2)\log_2(x) = 1$;
4) $\sin(x+1) = 0{,}5x$.

№ 23. 1) $3^x + 2x - 3 = 0$;
2) $3x^4 - 8x^3 - 18x^2 + 2 = 0$;
3) $x^2 - 4 + 0.5^x = 0$;
4) $(x-2)^2 \lg(x+11) = 1$.

№ 24. 1) $2e^x - 2x - 3 = 0$;
2) $3x^4 + 4x^3 - 12x^2 - 5 = 0$;
3) $x\log_3(x+1) = 1$;
4) $\cos(x+0{,}5) = x^3$.

№ 25. 1) $3^x + 2 + x = 0$;
2) $2x^3 - 8x^3 - 60x + 1 = 0$;
3) $(x-4)^2 \log_{0.5}(x-3) = -1$;
4) $5\sin x = x - 0{,}5$.

№ 26. 1) $\operatorname{arcctg}(x-1) + 2x - 3 = 0$;
2) $x^4 - x - 1 = 0$;
3) $(x-1)^2 2^x = 1$;
4) $\operatorname{tg}^3 x = x + 1, -\pi/2 \le x \le \pi/2$.

№ 27. 1) $e^{-2x} - 2x + 1 = 0$;
2) $2x^4 - x^2 - 10 = 0$;
3) $0.5^x - 3 = -(x+1)^2$;
4) $x^2 \cos 2x = -1$.

№ 28. 1) $3^x - 2x - 5 = 0$;
2) $3x^4 + 8x^3 + 6x^2 - 10 = 0$;
3) $2x^2 - 0.5^x - 3 = 0$;
4) $x\lg(x+1) = 1$.

№ 29. 1) $\operatorname{arctg}(x-1) + 2x = 0$;
2) $x^4 - 18x^2 + 6 = 0$;
3) $(x-2)^2 2^x = 1$;
4) $x^2 - 10\sin x = 0$.

№ 30. 1) $3^x + 5x - 2 = 0$;
2) $3x^4 + 4x^3 - 12x^2 + 1 = 0$;
3) $0.5^x + 1 = (x-2)^2$;
4) $(x+3)\cos x = 1, -2\pi \le x \le 2\pi$.

Образец выполнения задания

1) $5^x - 6x - 3 = 0$; 2) $x^4 - x^3 - 2x^2 + 3x - 3 = 0$;
3) $2\cos\left(x + \frac{\pi}{6}\right) + x^2 = 3x - 2$; 4) $x^2 \log_{0.5}(x+1) = 1$.

---

1) Обозначим $f(x) = 5^x - 6x - 3$. Находим $f'(x) = 5^x \ln 5 - 6$. Вычислим корень производной:

$$5^x \lg 5 - 6 = 0;\ 5^x = \frac{6}{\ln 5};\ x \lg 5 = \lg 6 - \lg(\ln 5);$$
$$x = \frac{\lg 6 - \lg(\ln 5)}{\lg 5} = \frac{0.7782 - 0.2065}{0.6990} = \frac{0.5717}{0.6990} \approx 0.82$$

Составим таблицу знаков функций f(x), полагая x равным: а) критическим значениям функций (корням производной) или близким к ним.
б) граничным значениям (исходя из области допустимых значений неизвестного):

x	$-\infty$	1	$+\infty$
signf(x)	+	−	+

Так как происходят две перемены знака функций, то уравнение имеет два действительных корня. Чтобы завершить операцию отделения корней, следует уменьшить промежутки, содержащие корни, так чтобы их длинна была не больше 1. Для этого составим новую таблицу знаков функций f(x):

x	−1	0	1	2
signf(x)	+	−	−	+

Отсюда видно, что корни заключены в следующих промежутках $x_1 \in [-1, 0]$; $x_2 \in [1, 2]$.
2) Полагая $f(x) = x^4 - x^3 - 2x^2 + 3x - 3$, имеем $f'(x) = 4x^3 - 3x^2 - 4x + 3$. Найдем корни производной: $4x^3 - 3x^2 - 4x + 3 = 0$; $4x(x^2 - 1) - 3(x^2 - 1) = 0$; $(x^2 - 1)(4x - 3) = 0$; $x_1 = -1$; $x_2 = 1$; $x_3 = 3/4$.

Составим таблицу знаков функций f(x):

x	$-\infty$	1	3/4	1	$+\infty$
signf(x)	+	-	-	-	+

Из таблицы видно, что уравнение имеет два действительных корня:$x_1 \in [-\infty, -1]$; $x_2 \in [1, +\infty]$.

Уменьшим промежутки, в которых находятся корни:

x	-2	-1	1	2
signf(x)	+	-	-	+

Следовательно, $x_1 \in [-2, -1]$; $x_2 \in [1, 2]$.

Уточним один из корней, например $x_1 \in [-2, -1]$, методом проб до сотых долей. Все вычисления удобно производить, используя следующую таблицу:

n	a_n^+	b_n^-	$x_n = \frac{a_n + b_n}{2}$	x_n^4	$-x_n^3$	$-2x_n^2$	$3x_n$	$f(x_n)$
0	-2	-1	-1,5	5,6025	3,375	-4,5	-4,5	-3,56
1	-2	-1,5	-1,75	9,3789	5,3594	-6,125	-5,25	0,36
2	-1,75	-1,5	-1,63	7,0591	4,3307	-5,3138	-4,89	-1,81
3	-1,75	-1,63	-1,69	8,1579	4,8268	-5,7122	-5,07	-0,79
4	-1,75	-1,69	-1,72	8,7521	5,0884	-5,9168	-5,16	-0,23
5	-1,75	-1,72	-1,73	8,9575	5,1777	-5,9858	-5,19	-0,04
6	-1,75	-1,73	-1,74	9,1664	5,2680	-6,0552	-5,22	0,15
7	-1,74	-1,73						

Ответ: $x_1 \approx -1{,}73$

3) Перепишем уравнение в виде $2\cos\left(x+\frac{\pi}{6}\right) = -x^2+3x-2$. Обозначим $y_1 = 2\cos\left(x+\frac{\pi}{6}\right)$, $y_2 = -x^2+3x-2$, построим графики функций (рис. 1).

Из графика видно, что уравнение имеет два корня: $x_1 \approx 1$; $x_2 \approx 2{,}9$.

4) Перепишем уравнение в виде $\log_{0.5}(x+1) = 1/x^2$. Обозначим $y_1 = \log_{0.5}(x+1)$, $y_2 = 1/x^2$, построим графики этих функций (рис. 2). Из графика видно, что уравнение имеет один корень $x_1 \approx -0{,}8$.

Для уточнения этого корня методом проб выберем промежуток, в концах которого функция $f(x) = x^2\log_{0.5}(x+1) - 1$ имеет разные значения. Составим таблицу:

x	−0, .5	−0.8
signf(x)	−	+

Для удобства расчетов перейдем к десятичным логарифмам:

$$f(x) = x^2\frac{\lg(x+1)}{\lg 0.5} - 1 = x^2\frac{\lg(x+1)}{-0.301} - 1$$

Дальнейшие вычисления производим в таблице:

n	a_n^+	b_n^-	$x_n = \frac{a_n+b_n}{2}$	x_n^2	$\lg(x_n+1)$	$f(x_n)$
0	-0,8	-0,5	-0,65	0,4225	-0,4559	-0,360
1	-0,8	-0,65	-0,73	0,5329	-0,5686	0,0067
2	-0,73	-0,65	-0,69	0,4761	-0,5086	-0,196
3	-0,73	-0,96	-0,71	0,5041	-0,5376	-0,099
4	-0,73	-0,71	-0,72	0,5184	-0,5528	-0,048
5	-0,73	-0,72				

Ответ: $x \approx -0{,}73$.

Работа 2

Задание: 1) Отделить корни уравнения графически и уточнить один из них методом хорд с точностью до 0,001.

2) Отделить корни уравнения аналитически и уточнить один из них методом хорд с точностью до 0,001.

№ 1. 1)$x - \sin x = 0{,}25$; 2)$x^3 - 3x^2 + 9x - 8 = 0$.

№ 2. 1)$\operatorname{tg}(0.58x + 0.1) = x^2$; 2)$x^3 - 6x - 8 = 0$.

№ 3. 1)$\sqrt{x} - \cos(0{,}387x) = 0$; 2)$x^3 - 3x^2 + 6x + 3 = 0$.

№ 4. 1)$\text{tg}(0{,}4x+0{,}4)=x^2$; 2)$x^3-0{,}1x^2+0{,}4x-1{,}5=0$.

№ 5. 1)$\lg x-\frac{7}{2x+6}=0$; 2)$x^3-3x^2+9x+2=0$.

№ 6. 1)$\text{tg}(0{,}5x+0{,}2)=x^2$; 2)$x^3+x-5=0$.

№ 7. 1)$3x-\cos x-1=0$; 2)$x^3+0{,}2x^2+0{,}5x-1{,}2=0$.

№ 8. 1)$x+\lg x=0{,}5$; 2)$x^3+3x+1=0$.

№ 9. 1)$\text{tg}(0{,}5x+0{,}1)=x^2$; 2)$x^3+0{,}2x^2+0{,}5x-2=0$.

№ 10. 1)$x^2+4\sin x=0$; 2)$x^3-3x^2+12x-9=0$.

№ 11. 1) $\text{ctg}1{,}05x-x^2=0$; 2)$x^3-0{,}2x^2+0{,}3-1{,}2=0$.

№ 12. 1) $\text{tg}(0{,}5x+0{,}3)=x^2$; 2)$x^3-3x^2+6x-2=0$.

№ 13. 1)$x\lg x-1{,}2=0$; 2)$x^3-0{,}1x^2+0{,}4x-1{,}5=0$.

№ 14. 1)$1{,}8x^2-\sin 10x=0$; 2)$x^3+3x^2+6x-1=0$.

№ 15. 1)$\text{ctg}x-\frac{x}{4}=0$; 2)$x^3+0{,}1x^2+0{,}4x-1{,}2=0$.

№ 16. 1)$\text{tg}(0{,}3x+0{,}4)=x^2$; 2)$x^3+4x-6=0$.

№ 17. 1)$x^2-20\sin x=0$; 2)$x^3+0{,}2x^2+0{,}5x+0{,}8=0$.

№ 18. 1)$\text{ctg}x-\frac{x}{3}=0$; 2)$x^3-3x^3+12x-12=0$.

№ 19. 1)$\text{tg}(0{,}47x+0{,}2)=x^2$; 2)$x^3-0{,}2x^2+0{,}3x+1{,}2=0$.

№ 20. 1)$x^2+4\sin x=0$; 2)$x^3-2x+4=0$.

№ 21. 1)$\text{ctg}x-\frac{x}{2}=0$; 2)$x^3-0{,}2x^2+0{,}5-1{,}4=0$.

№ 22. 1)$2x-\lg x-7=0$; 2)$x^3-3x^2+6x-5=0$.

№ 23. 1)$\text{tg}(0{,}44x+0{,}3)=x^2$; 2)$x^3-0{,}1x^2+0{,}4x+1{,}2=0$.

№ 24. 1)$3x-\cos x-1=0$; 2)$x^3-0{,}1x^2+0{,}5x-1=0$.

№ 25. 1)$\text{ctg}x-\frac{x}{10}=0$; 2)$x^3+3x^2+12x+3=0$.

№ 26. 1)$x^2+4\sin x=0$; 2)$x^3-0{,}1x^2+0{,}4+2=0$.

№ 27. 1)$\text{tg}(0{,}36x+0{,}4)=x^2$; 2)$x^3-0{,}2x^2+0{,}4-1{,}4=0$.

№ 28. 1)$x+\lg x=0{,}5$; 2)$x^3+0{,}4x^2+0{,}6x-1{,}6=0$.

№ 29. 1)$\text{ctg}x-\frac{x}{5}=0$; 2)$x^3+x-3=0$.

№ 30. 1)$2\lg x-\frac{x}{2}+1=0$; 2)$x^3-0{,}2x^2+0{,}5x+1{,}4=0$.

Образец выполнения задания

1) $\text{tg}(0{,}55x+0.1)=x^2$; 2) $x^3-0.2x^2+0.5x+1.5=0$.

---

1) Отделим корень графически. Построим графики функций $y_1=\text{tg}(0.55x+0.1)$ и $y_2=x^2$ (рис. 3), составив таблицы значений этих функций:

x	0	0,2	0,4	0,6	0,8	1
$y_2=x^2$	0	0,04	0,16	0,36	0,64	1
0,55x	0	0,11	0,22	0,33	0,44	0,55
y_1	0,1	0,21	0,33	0,46	0,60	0,76

Таким образом, положительный корень уравнения заключен в промежутке [0,6; 0,8].

Чтобы уточнить корень методом хорд, определим знаки функций $f(x) = tg(0.55x + 0.1) - x^2$ на концах промежутка [0,6; 0,8] и знак со второй производной в этом промежутке: $f(0.6) = tg0{,}43 - 0{,}36 = 0{,}4586 - 0{,}36 = 0{,}0986$; $f(0{,}8) = tg0{,}54 - 0{,}64 = 0{,}5994 - 0.64 = -0{,}0406$;

$$f'(x)\frac{0{,}55}{\cos^2(0{,}55x + 0{,}1)} - 2x$$

$$f''(x) * 2\cos^3(0{,}55x + 0{,}1)\sin(0{,}55x + 0{,}1)\,0{,}55 - 2 = \frac{0{,}605\sin(0{,}55x+0{,}1)}{\cos^3(0{,}55x+0{,}1)} - 2 < 0 \text{ при}$$

$$x \in [0{,}6; 0{,}8].$$

Для вычислений применяем формулу

$x_{n+1} = x_n - \frac{f(x_n)}{f(b)-f(x_n)}$, где $b = 0{,}8$; $x_0 = 0{,}6$.

Вычисления удобно располагать в таблице:

n	x_n	$0{,}8 - x_n$	$0{,}55x_n + 0{,}1$	$tg(0{,}55x_n + 0{,}1)$
0	0,6	0,2	0,43	0,4586
1	0,742	0,058	0,5081	0,5570
2	0,750	0,50	0,5125	0,5627
3	0,7502	0,0498	0,5126	0,5628

n	x_n^2	$f(x_n)$	$f(0{,}8) - f(x_n)$	$h = \frac{f(x_n)}{f(0{,}8) - f(x_n) * * (b - x_n)}$
0	0,36	0,0986	-0,1392	-0,142
1	0,5506	0,0064	-0,0470	-0,008
2	0.5625	0,0002	-0,0408	-0,0002
3	0,5628	0		

Ответ: $x = 0{,}750$

2)Отделим корни аналитически. Находим $f(x) = x^3 + 0{,}2x^2 + 0{,}5x + 1{,}5$; $f'(x) = 3x^2 - 0{,}4x$+0,5; $D = 0{,}16 - 6 < 0$.

Составим таблицу знаков функций $f(x)$:

x	$-\infty$	-1	0	$+\infty$
signf(x)	−	−	+	+

Уравнение имеет один действительный корень, лежащий в промежутке $[-1, 0]$.

Чтобы уточнить корень, находим вторую производную $f''(x) = 6x - 0{,}4$; в промежутке $[-1, 0]$ выполняется неравенство $f''(x) < 0$.

Для вычислений применяем формулу

$$x_{n+1} = a - \frac{f(a)}{f(x_n)-f(a)}(x_n - a,$$

где $a = -1$; $x_0 = 0$; $f(a) = f(-1) - 1 - 0{,}2 - 0{,}5 + 1{,}5 = -0{,}2$.

Вычисления располагаем в таблице:

n	x_n	x_n^3	x_n^2	$0{,}2x_n^2$	$0{,}5x_2$
0	0	0	0	0	0
1	-0,882	-0,6861	0,7779	0,1556	-0,441
2	-0.943	-0,8386	0,8892	0,1778	-0,4715
3	-0,946	-0,8466	0,8646	0,1790	-0,473
4	-0,946				

n	$f(x_n)$	$f(x_n) + 0.2$	$x_n - a$	$\frac{f(a)(x_n - a)}{f(x_n) - f(a)}$
0	1,5	1,7	1	-0,118
1	0,2173	0,4173	0,118	-0,057
2	0,0121	0,2121	0,057	-0,054
3	0,0014	0.2014	0,054	-0,054

Ответ: $x \approx -0{,}946$.

Работа 3

Задание: 1) Отделить корни уравнения графически и уточнить один из них методом касательных с точностью до 0,001.

2) Отделить корни уравнения аналитически и уточнить один из них с точностью до 0,001 методом касательных.

Воспользоваться вариантами работы 2.

Образец выполнения задания

1)$\operatorname{tg}(0{,}55x + 0{,}1) = x^2$; 2)$x^3 - 0{,}2x^2 + 0{,}5x + 1{,}5 = 0$

1) Выше мы отделили один из корней этого уравнения и установили, что он заключен в промежутке $[0{,}6; 0{,}8]$. Уточним этот корень методом касательных. Так как $f(0{,}6) > 0$; $f(0{,}8) < 0$ и $f''(x) < 0$ то за начальное приближение пример $x_0 = 0{,}8$.

Вычисления производим по формуле

$$x_{n+1} - \frac{f(x_n)}{f'(x_{0)}}$$

Предварительно найдем

$$f'(0{,}8) = \frac{0{,}55}{\cos^2(0{,}44+0{,}1)} - 2*0{,}8 = \frac{0{,}55}{0{,}85772} - 1{,}6 = \frac{0{,}55}{0{,}7356} - 1{,}6 = 0{,}7477 - 1{,}6 = -0{,}8523$$

Составим таблицу:

n	x_n	x_n^2	$0{,}55x_n + 0{,}1$	$tg(0{,}55x_n + 0{,}1)$	$f(x_n)$	$\frac{f(x_n)}{-0{,}8523}$
0	0,8	0,64	0,54	0,5994	-0,0406	0,0476
1	0,7524	0,5661	0,5138	0,5643	-0,0018	0,0021
2	0,7503	0,5630	0,5127	0,5630	-0,0000	0

Ответ: $x \approx 0{,}750$.

2) Выше мы установили, что уравнение имеет действительный корень, принадлежащей промежутку $[-1,0]$. Уточним этот корень методом касательных. Так как $f(-1) < 0$, $f(0) > 0$ и $f''(x) < 0$, то за начальное приближение принимаем $x_0 = -1$.

Для вычислений применяем формулу

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Находим $f(x) = x^3 - 0{,}2x^2 + 0{,}5x + 1{,}5;\ f'(x) = 3x^2 - 0{,}4x + 0{,}5$. Для вычислений используем таблицу:

	x_n	x_n^2	x_n^3	$f(x_n)$	$f'(x_n)$	$h = \frac{f(x_n)}{f'(x_n)}$
0	-1	1	-1	-0,2	3,9	-0,051
1	-0,949	0,9006	-1,8547	-0,0093	3,5814	-0,0026
2	-0,9464	0,8957	-0,8477	-0,0004	3,5657	-0,00001

Ответ: $x \approx -0.946$.

Тема 3. Решение систем линейных алгебраических уравнений

Задание 1

Решить системы уравнений, пользуясь методом Гаусса

$$1.A = \begin{vmatrix} 3.2 & -1.5 & 0.5 \\ 1.6 & 2.5 & -1.0 \\ 1.0 & 4.1 & -1.5 \end{vmatrix} \quad b = \begin{vmatrix} 0.90 \\ 1.55 \\ 2.08 \end{vmatrix} \qquad 2.a = \begin{vmatrix} 1.5 & -0.2 & 0.1 \\ -0.1 & 1.5 & -0.1 \\ -0.3 & 0.2 & -0.5 \end{vmatrix} \quad b = \begin{vmatrix} 0.4 \\ 0.8 \\ 0.2 \end{vmatrix}$$

$$3.\ A = \begin{vmatrix} 8.64 & 1.71 & 5.42 \\ -6.39 & 4.25 & 1.84 \\ 4.21 & 7.92 & -3.41 \end{vmatrix} \quad b = \begin{vmatrix} 10.21 \\ 3.41 \\ 12.29 \end{vmatrix} \qquad 4.a = \begin{vmatrix} 8.30 & 2.62 & 4.10 & 1.9 \\ 3.92 & 8.45 & 7.78 & 2.46 \\ 3.77 & 7.21 & 8.04 & 2.28 \\ 2.21 & 3.65 & 1.69 & 6.99 \end{vmatrix} \quad b = \begin{vmatrix} -10.65 \\ 12.21 \\ 15.45 \\ -8.35 \end{vmatrix}$$

5. $A = \begin{vmatrix} 8.14 & 1.71 & 5.42 \\ -6.39 & 4.25 & 2.34 \\ 4.21 & 7.42 & -3.41 \end{vmatrix}$ $b = \begin{vmatrix} 9.71 \\ 3.91 \\ 12.29 \end{vmatrix}$ 6. $a = \begin{vmatrix} 8.30 & 2.82 & 4.10 & 1.9 \\ 3.92 & 8.45 & 7.58 & 2.46 \\ 3.77 & 7.41 & 8.04 & 2.28 \\ 2.21 & 3.45 & 1.69 & 6.99 \end{vmatrix}$ $b = \begin{vmatrix} -10.45 \\ 12.21 \\ 15.25 \\ -8.35 \end{vmatrix}$

7. $A = \begin{vmatrix} 7.64 & 1.71 & 5.42 \\ -6.39 & 4.25 & 2.84 \\ 4.21 & 6.92 & -3.41 \end{vmatrix}$ $b = \begin{vmatrix} 9.21 \\ 4.41 \\ 12.29 \end{vmatrix}$ 8. $a = \begin{vmatrix} 8.30 & 3.02 & 4.10 & 1.9 \\ 3.92 & 8.45 & 7.38 & 2.46 \\ 3.77 & 7.61 & 8.04 & 2.28 \\ 2.21 & 3.25 & 1.69 & 6.99 \end{vmatrix}$ $b = \begin{vmatrix} -10.25 \\ 12.21 \\ 15.05 \\ -8.35 \end{vmatrix}$

9. $A = \begin{vmatrix} 7.14 & 1.71 & 5.42 \\ -6.39 & 4.25 & 3.34 \\ 4.21 & 6.42 & -3.41 \end{vmatrix}$ $b = \begin{vmatrix} 8.71 \\ 4.91 \\ 12.29 \end{vmatrix}$ 10. $a = \begin{vmatrix} 8.30 & 3.22 & 4.10 & 1.9 \\ 3.92 & 8.45 & 7.18 & 2.46 \\ 3.77 & 7.81 & 8.04 & 2.28 \\ 2.21 & 3.05 & 1.69 & 6.99 \end{vmatrix}$ $b = \begin{vmatrix} -10.05 \\ 12.21 \\ 14.85 \\ -8.35 \end{vmatrix}$

11. $A = \begin{vmatrix} 6.64 & 1.71 & 5.42 \\ -6.39 & 4.25 & 3.84 \\ 4.21 & 5.92 & -3.41 \end{vmatrix}$ $b = \begin{vmatrix} 8.21 \\ 5.41 \\ 12.29 \end{vmatrix}$ 12. $a = \begin{vmatrix} 8.30 & 3.52 & 4.10 & 1.9 \\ 3.92 & 8.45 & 6.98 & 2.46 \\ 3.77 & 8.01 & 8.04 & 2.28 \\ 2.21 & 2.95 & 1.69 & 6.99 \end{vmatrix}$ $b = \begin{vmatrix} -10.25 \\ 12.21 \\ 14.65 \\ -8.35 \end{vmatrix}$

13. $A = \begin{vmatrix} 6.14 & 1.71 & 5.42 \\ -6.39 & 4.25 & 4.34 \\ 4.21 & 5.42 & -3.41 \end{vmatrix}$ $b = \begin{vmatrix} 7.71 \\ 5.91 \\ 12.29 \end{vmatrix}$ 14. $a = \begin{vmatrix} 8.30 & 3.72 & 4.10 & 1.9 \\ 3.92 & 8.45 & 6.78 & 2.46 \\ 3.77 & 8.21 & 8.04 & 2.28 \\ 2.21 & 2.75 & 1.69 & 6.99 \end{vmatrix}$ $b = \begin{vmatrix} -10.05 \\ 12.21 \\ 14.45 \\ -8.35 \end{vmatrix}$

15. $A = \begin{vmatrix} 0,15 & 2,11 & 30,75 \\ 0,64 & 1,21 & 2,05 \\ 3,21 & 1,53 & 1,04 \end{vmatrix}$ $в = \begin{vmatrix} -26,38 \\ 1,01 \\ 5,23 \end{vmatrix}$ 16. $a = \begin{vmatrix} 1,15 & 0,42 & 100,71 \\ 1,19 & 0,55 & 0,32 \\ 1,00 & 0,35 & 3,00 \end{vmatrix}$ $в = \begin{vmatrix} -198,70 \\ 2,29 \\ -0,65 \end{vmatrix}$

17. $A = \begin{vmatrix} 2 & 6 & -1 \\ 5 & -1 & 2 \\ -3 & -4 & 1 \end{vmatrix}$ $в = \begin{vmatrix} -12 \\ 29 \\ 5 \end{vmatrix}$ 18. $a = \begin{vmatrix} 2,10 & -4,50 & -2,00 \\ 3,00 & 2,50 & 4,30 \\ -6,00 & 3,50 & 2,50 \end{vmatrix}$ $в = \begin{vmatrix} -19,07 \\ 3,21 \\ -18,25 \end{vmatrix}$

$$19.A=\begin{vmatrix} 21{,}457 & -95{,}510 & -96{,}121 \\ 10{,}223 & -91{,}065 & -7{,}343 \\ 51{,}218 & 12{,}269 & 86{,}457 \end{vmatrix} \quad \text{в}=\begin{vmatrix} -49{,}930 \\ -12{,}465 \\ 60{,}812 \end{vmatrix} \qquad 20.\,\text{а}=\begin{vmatrix} 2{,}6 & -4{,}5 & -2{,}0 \\ 3{,}0 & 3{,}0 & 4{,}3 \\ -6{,}0 & 3{,}5 & 3{,}0 \end{vmatrix} \quad \text{в}=\begin{vmatrix} -19{,}07 \\ 3{,}21 \\ -18{,}25 \end{vmatrix}$$

Образец выполнения задания

$$\begin{cases} 0{,}68x_1+0{,}05x_2-0{,}11x_3+0{,}08x_4=2{,}15, \\ 0{,}21x_1-0{,}13x_2+0{,}27x_3-0{,}8x_4=0{,}44, \\ -0{,}11x_1-0{,}84x_2+0{,}28x_3+0{,}06x_4=-0{,}83, \\ -0{,}08x_1+0{,}15x_2-0{,}5x_3-0{,}12x_4=1{,}16. \end{cases}$$

Вычисления производим по схеме единственного деления:

Коэффициенты при неизвестных				Свободные члены	Контрольные суммы Σ	Строчные суммы Σ'
x_1	x_2	x_3	x_4			
0,68	0,05	−0,11	0,08	2.15	2,85	2,85
0,21	−0,13	0,27	−0,8	0.44	−0,01	−0,01
−0,11	−0,84	0,28	0,06	−0.83	−1,44	−1,44
−0,08	0,15	−0,5	−0,12	1.16	0.61	0,61
1	0,0735	−0,1618	0,1176	3,1618	4.1912	4,1912
	−0,1454	0,30398	−0,8247	−0,22398	−0.89015	−0,8901
	−0,8319	0,2622	0,0729	−0,4822	−0.97897	−0,97896
	0,1559	−0,5129	−0,1106	1,4129	0.9453	0,9453
	1	−2,0906	5,6719	1,5404	6.1221	6,1217
		−1,47697	4,79139	0,7992	4.1140	4,1136
		−0,18697	−0,9948	1,1723	−0.00913	−0,0095
		1	−3,2441	−0,5411	−2.7854	−2,7851
			−1,6013	1,0711	−0.5299	−0,5302
			1	−0,6689	0.3309	0,3311
2,8264	−0,3337	−2,7110	−0,6689			
3,8263	0,6664	−1.7119	0,3309			

Ответ: $x_1=2{,}826$; $x_2=-0{,}334$; $x_3=-2{,}711$; $x_4=-0.669$.

Задание. 1) Обратить матрицу по схеме единственного деления. Все расчеты вести с четырьмя десятичными знаками. Ответ округлить до трех десятичных знаков.

2) Вычислить определитель по схеме Гаусса с точностью до 0,0001.

№ 1.

1) $\begin{pmatrix} 1{,}00 & 0{,}47 & -0{,}11 & 0{,}55 \\ 0{,}42 & 1{,}00 & 0{,}35 & 0{,}17 \\ -0{,}25 & 0{,}67 & 1{,}00 & 0{,}36 \\ 0{,}54 & -0{,}32 & -0{,}74 & 1{,}00 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \end{vmatrix}$.

№ 2.

1) $\begin{pmatrix} 0{,}15 & 0{,}23 & 0{,}12 & 0{,}44 \\ -0{,}52 & 0{,}35 & 0{,}21 & -0{,}72 \\ 0{,}35 & 0{,}42 & 0{,}38 & -0{,}63 \\ 0{,}74 & -0{,}25 & 0{,}37 & 0{,}55 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}00 & 0{,}17 & -0{,}25 & 0{,}54 \\ 0{,}47 & 1{,}00 & 0{,}67 & -0{,}32 \\ -0{,}11 & 0{,}35 & 1{,}00 & -0{,}74 \\ 0{,}55 & 0{,}43 & 0{,}36 & 1{,}00 \end{vmatrix}$.

№ 3.

1) $\begin{pmatrix} 0{,}75 & 0{,}16 & 0{,}27 & 0{,}83 \\ 0{,}55 & 0{,}22 & -0{,}12 & 0{,}32 \\ 1{,}00 & 0{,}42 & 0{,}35 & 0{,}18 \\ -0{,}37 & 0{,}23 & 0{,}15 & 0{,}28 \end{pmatrix}$;

2) $\begin{vmatrix} 8{,}2 & 1{,}4 & -2{,}3 & 0{,}2 \\ -1{,}6 & 5{,}4 & -7{,}7 & 3{,}1 \\ 0{,}7 & 1{,}9 & -8{,}5 & 4{,}8 \\ 5{,}3 & -5{,}9 & 2{,}7 & -7{,}9 \end{vmatrix}$.

№ 4.

1) $\begin{pmatrix} 1{,}5 & 2{,}7 & -1{,}3 & 5{,}2 \\ 2{,}7 & -3{,}4 & 1{,}8 & 2{,}2 \\ -1{,}3 & 0{,}16 & 0{,}82 & 1{,}05 \\ 5{,}2 & 2{,}2 & 1{,}05 & 3{,}4 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \end{vmatrix}$.

№ 5.

1) $\begin{pmatrix} 1{,}17 & 2{,}13 & 0{,}32 & 0{,}56 \\ 2{,}13 & 0{,}82 & -0{,}72 & 1{,}10 \\ 0{,}32 & 0{,}25 & -0{,}42 & 0{,}16 \\ 0{,}56 & 1{,}10 & -0{,}25 & -0{,}44 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}47 & 1{,}00 & 0{,}67 & -0{,}32 \\ 1{,}00 & 0{,}17 & -0{,}25 & 0{,}54 \\ 0{,}55 & 0{,}43 & 0{,}36 & 1{,}00 \\ -0{,}11 & 0{,}35 & 1{,}00 & -0{,}74 \end{vmatrix}$.

№ 6.

1) $\begin{pmatrix} 1{,}2 & 3{,}2 & -1{,}5 & 2{,}7 \\ -5{,}3 & 4{,}1 & 3{,}8 & 1{,}7 \\ 0{,}3 & 1{,}5 & -1{,}6 & 4{,}2 \\ 1{,}6 & 4{,}5 & 6{,}3 & -1{,}2 \end{pmatrix}$;

2) $\begin{vmatrix} -1{,}6 & 5{,}4 & -7{,}7 & 3{,}1 \\ 8{,}2 & 1{,}4 & -2{,}3 & 0{,}2 \\ 5{,}3 & -5{,}9 & 2{,}7 & -7{,}9 \\ 0{,}7 & 1{,}9 & -8{,}5 & 4{,}8 \end{vmatrix}$.

№ 7.

1) $\begin{pmatrix} 0{,}62 & 0{,}73 & -0{,}43 & -0{,}23 \\ 0{,}73 & 1{,}00 & 0{,}25 & 0{,}64 \\ -0{,}41 & 0{,}62 & 0{,}21 & 0{,}44 \\ 0{,}84 & 0{,}32 & 0{,}18 & -0{,}47 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \end{vmatrix}$.

№ 8.

1) $\begin{pmatrix} 1{,}13 & 2{,}15 & 0{,}83 & 0{,}77 \\ 0{,}64 & -0{,}43 & 0{,}62 & -0{,}32 \\ 2{,}32 & 1{,}15 & 1{,}84 & 0{,}68 \\ -0{,}72 & 0{,}53 & 0{,}64 & -0{,}57 \end{pmatrix}$;

2) $\begin{vmatrix} -1{,}6 & 5{,}4 & -7{,}7 & 3{,}1 \\ 8{,}2 & 1{,}4 & -2{,}3 & 0{,}2 \\ 5{,}3 & -5{,}9 & 2{,}7 & -7{,}9 \\ 0{,}7 & 1{,}9 & -8{,}5 & 4{,}8 \end{vmatrix}$.

№ 9.

1) $\begin{pmatrix} 0{,}42 & 0{,}26 & 0{,}33 & -0{,}22 \\ 0{,}74 & -0{,}55 & 0{,}28 & -0{,}65 \\ 0{,}88 & 0{,}42 & -0{,}33 & 0{,}75 \\ 0{,}92 & 0{,}82 & -0{,}62 & 0{,}75 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}47 & 1{,}00 & 0{,}67 & -0{,}32 \\ 1{,}00 & 0{,}17 & -0{,}25 & 0{,}54 \\ 0{,}55 & 0{,}43 & 0{,}36 & 1{,}00 \\ -0{,}11 & 0{,}35 & 1{,}00 & -0{,}74 \end{vmatrix}$.

№ 10.

1) $\begin{pmatrix} 0{,}75 & 0{,}18 & 0{,}63 & -0{,}32 \\ 0{,}92 & 0{,}38 & -0{,}14 & 0{,}56 \\ 0{,}63 & -0{,}42 & 0{,}18 & 0{,}37 \\ -0{,}65 & 0{,}52 & 0{,}47 & 0{,}27 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \end{vmatrix}$.

№ 11. 1) $\begin{pmatrix} -2{,}41 & 7{,}55 & 0{,}82 & 0{,}33 \\ 0{,}28 & -3{,}44 & 0{,}75 & 0{,}23 \\ 0{,}17 & 0{,}28 & 0{,}05 & 3{,}48 \\ -1{,}00 & 0{,}23 & 2{,}00 & 7{,}00 \end{pmatrix}$; 2) $\begin{vmatrix} 1{,}13 & 0{,}15 & 0{,}26 & -0{,}43 \\ 0{,}45 & 0{,}62 & -0{,}80 & 0{,}74 \\ 0{,}62 & -1{,}12 & 0{,}64 & 0{,}78 \\ -0{,}13 & 0{,}73 & 0{,}16 & -0{,}36 \end{vmatrix}$.

№ 12. 1) $\begin{pmatrix} -1{,}09 & 7{,}56 & 3{,}45 & 0{,}78 \\ 3{,}33 & 4{,}45 & -0{,}21 & 3{,}44 \\ 2{,}33 & -4{,}45 & 0{,}17 & 2{,}21 \\ 4{,}03 & 1{,}00 & 3{,}05 & 0{,}11 \end{pmatrix}$; 2) $\begin{vmatrix} 0{,}84 & 1{,}32 & 0{,}48 & -1{,}13 \\ 1{,}16 & -0{,}46 & 0{,}64 & -0{,}13 \\ 0{,}44 & 0{,}83 & -1{,}12 & 0{,}44 \\ 0{,}16 & 0{,}32 & 0{,}08 & -0{,}57 \end{vmatrix}$.

№ 13. 1) $\begin{pmatrix} 4{,}5 & 4{,}8 & -3{,}7 & 2{,}1 \\ 4{,}5 & -3{,}7 & 5{,}6 & 3{,}3 \\ 4{,}8 & 7{,}5 & 8{,}3 & 9{,}2 \\ -1{,}5 & 2{,}3 & 4{,}8 & 3{,}1 \end{pmatrix}$; 2) $\begin{vmatrix} 0{,}52 & 0{,}83 & -1{,}2 & 0{,}32 \\ 0{,}63 & -0{,}42 & 0{,}57 & 1{,}15 \\ 0{,}44 & 0{,}52 & 0{,}44 & 0{,}18 \\ 0{,}62 & -0{,}12 & 0{,}08 & 0{,}42 \end{vmatrix}$.

№ 14. 1) $\begin{pmatrix} 5{,}5 & 3{,}7 & -8{,}3 & 9{,}1 \\ -4{,}5 & 6{,}8 & 7{,}2 & 3{,}4 \\ 7{,}5 & -4{,}9 & 3{,}5 & 7{,}1 \\ 5{,}6 & -4{,}8 & 7{,}3 & 5{,}3 \end{pmatrix}$; 2) $\begin{vmatrix} 1{,}5 & 0{,}84 & 0{,}63 & -0{,}18 \\ 0{,}15 & 0{,}36 & -0{,}16 & 0{,}88 \\ -0{,}27 & 0{,}45 & 0{,}64 & -0{,}38 \\ 0{,}41 & -0{,}83 & 0{,}62 & 0{,}27 \end{vmatrix}$.

№ 15. 1) $\begin{pmatrix} 1{,}8 & 1{,}02 & 1{,}03 & 1{,}05 \\ 7{,}03 & 8{,}04 & 9{,}05 & 6{,}08 \\ 1{,}11 & -2{,}02 & 2{,}03 & -3{,}04 \\ -3{,}41 & 4{,}52 & 7{,}28 & 5{,}51 \end{pmatrix}$; 2) $\begin{vmatrix} 0{,}8 & 1{,}3 & -0{,}12 & 0{,}25 \\ -1{,}2 & 0{,}18 & 0{,}72 & 0{,}13 \\ 1{,}6 & 0{,}2 & 0{,}12 & -0{,}11 \\ 1{,}4 & 0{,}15 & -0{,}83 & 0{,}41 \end{vmatrix}$.

№ 16. 1) $\begin{pmatrix} 1{,}71 & 3{,}56 & -0{,}33 & 0{,}17 \\ 2{,}81 & 3{,}45 & 0{,}17 & -0{,}22 \\ -0{,}34 & 0{,}75 & 0{,}33 & 0{,}22 \\ 7{,}03 & -3{,}45 & 0{,}32 & 0{,}17 \end{pmatrix}$; 2) $\begin{vmatrix} 2{,}5 & 0{,}35 & 0{,}4 & -0{,}8 \\ 0{,}2 & -1{,}5 & 0{,}61 & 2{,}3 \\ 0{,}16 & -0{,}42 & 0{,}57 & 0{,}63 \\ 0{,}23 & 0{,}15 & -0{,}08 & 3{,}1 \end{vmatrix}$.

№ 17. 1) $\begin{pmatrix} 0{,}17 & -0{,}13 & 0{,}45 & 0{,}66 \\ 0{,}18 & 0{,}22 & -0{,}11 & 0{,}71 \\ 0{,}82 & 0{,}33 & 0{,}18 & -0{,}63 \\ -0{,}28 & 0{,}41 & 0{,}28 & 0{,}33 \end{pmatrix}$; 2) $\begin{vmatrix} 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \end{vmatrix}$.

№ 18. 1) $\begin{pmatrix} 1{,}41 & 2{,}42 & 3{,}53 & 4{,}48 \\ 1{,}28 & -3{,}04 & 1{,}09 & 1{,}05 \\ 7{,}01 & 8{,}03 & 9{,}01 & 7{,}04 \\ 3{,}15 & 4{,}18 & -8{,}11 & 7{,}12 \end{pmatrix}$; 2) $\begin{vmatrix} 1{,}00 & 0{,}17 & -0{,}25 & 0{,}54 \\ 0{,}47 & 1{,}00 & 0{,}67 & -0{,}32 \\ -0{,}11 & 0{,}35 & 1{,}00 & -0{,}74 \\ 0{,}55 & 0{,}43 & 0{,}36 & 1{,}00 \end{vmatrix}$.

№ 19. 1) $\begin{pmatrix} 0{,}28 & 0{,}33 & 0{,}42 & 0{,}51 \\ 0{,}17 & 0{,}88 & 0{,}19 & 0{,}22 \\ -0{,}23 & 0{,}18 & 0{,}11 & -0{,}13 \\ 0{,}51 & 0{,}15 & 0{,}72 & -0{,}14 \end{pmatrix}$; 2) $\begin{vmatrix} 8{,}2 & 1{,}4 & -2{,}3 & 0{,}2 \\ -1{,}6 & 5{,}4 & -7{,}7 & 3{,}1 \\ 0{,}7 & 1{,}9 & -8{,}5 & 4{,}8 \\ 5{,}3 & -5{,}9 & 2{,}7 & -7{,}9 \end{vmatrix}$.

№ 20. 1) $\begin{pmatrix} 1{,}17 & 4{,}12 & 1{,}08 & 3{,}05 \\ 2{,}01 & -1{,}02 & 1{,}03 & 1{,}00 \\ 1{,}00 & 2{,}00 & 1{,}00 & 3{,}00 \\ 7{,}05 & 8{,}03 & -4{,}04 & 5{,}55 \end{pmatrix}$; 2) $\begin{vmatrix} 1{,}6 & 5{,}4 & -7{,}7 & -3{,}1 \\ 8{,}2 & 1{,}4 & -2{,}3 & 0{,}2 \\ 5{,}3 & -5{,}9 & 2{,}7 & -7{,}9 \\ 0{,}7 & 1{,}9 & -8{,}5 & 4{,}8 \end{vmatrix}$.

№ 21. 1) $\begin{pmatrix} 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \end{pmatrix}$; 2) $\begin{vmatrix} 0{,}47 & 1{,}00 & 0{,}67 & -0{,}32 \\ 1{,}00 & 0{,}17 & -0{,}25 & 0{,}54 \\ 0{,}55 & 0{,}43 & 0{,}36 & 1{,}00 \\ -0{,}11 & 0{,}35 & 1{,}00 & -0{,}74 \end{vmatrix}$.

№ 22.

1) $\begin{pmatrix} 2{,}11 & 3{,}01 & 4{,}02 & 0{,}22 \\ 0{,}18 & 3{,}41 & 0{,}15 & 1{,}43 \\ 2{,}14 & 0{,}17 & 0{,}26 & 0{,}18 \\ 1{,}28 & 0{,}42 & 0{,}54 & 1{,}00 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \\ 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \end{vmatrix}$.

№ 23.

1) $\begin{pmatrix} 7{,}13 & 8{,}21 & 4{,}47 & -2{,}11 \\ 3{,}25 & 1{,}54 & 2{,}91 & 5{,}43 \\ -6{,}34 & -8{,}17 & -10{,}2 & 3{,}93 \\ 4{,}52 & 6{,}73 & 1{,}37 & -9{,}89 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}2 & -7{,}7 & 3{,}1 & -1{,}6 \\ 0{,}7 & 1{,}4 & -8{,}5 & 4{,}8 \\ 8{,}2 & -2{,}3 & 0{,}3 & -7{,}9 \\ 0{,}55 & 1{,}00 & 0{,}32 & 0{,}4 \end{vmatrix}$.

№ 24.

1) $\begin{pmatrix} -2{,}00 & 3{,}01 & 0{,}12 & -0{,}11 \\ 2{,}92 & -0{,}17 & 0{,}11 & 0{,}22 \\ 0{,}66 & 0{,}52 & 3{,}17 & 2{,}11 \\ 3{,}01 & 0{,}42 & -0{,}27 & -0{,}15 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}25 & 0{,}16 & 0{,}35 & 0{,}18 \\ 1{,}2 & -0{,}8 & 0{,}62 & 0{,}34 \\ 0{,}83 & 0{,}48 & -0{,}18 & 0{,}72 \\ 0{,}43 & 0{,}57 & 0{,}62 & -0{,}13 \end{vmatrix}$.

№ 25.

1) $\begin{pmatrix} 3{,}41 & -0{,}18 & 2{,}34 & 7{,}08 \\ 0{,}21 & 0{,}17 & -0{,}51 & -0{,}44 \\ 0{,}33 & 3{,}42 & -5{,}17 & 0{,}66 \\ 0{,}77 & 3{,}68 & 0{,}22 & -0{,}19 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}0 & 2{,}14 & 0{,}42 & -1{,}13 \\ 0{,}23 & 0{,}42 & -1{,}5 & 0{,}16 \\ 0{,}34 & -0{,}12 & 0{,}18 & 0{,}57 \\ 0{,}83 & -0{,}17 & 0{,}62 & -0{,}83 \end{vmatrix}$.

№ 26.

1) $\begin{pmatrix} 4{,}20 & 0{,}32 & 0{,}11 & 0{,}13 \\ 0{,}17 & 0{,}25 & 0{,}48 & 0{,}52 \\ 0{,}12 & 0{,}08 & 0{,}72 & 0{,}61 \\ 0{,}54 & 0{,}13 & 0{,}81 & 0{,}17 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}92 & 0{,}16 & -0{,}23 & 0{,}8 \\ 0{,}16 & 0{,}12 & 0{,}15 & 0{,}72 \\ -0{,}23 & 0{,}15 & 0{,}88 & 0{,}16 \\ 0{,}8 & 0{,}72 & -0{,}13 & 0{,}72 \end{vmatrix}$.

№ 27.

1) $\begin{pmatrix} 2{,}00 & 0{,}17 & 3{,}02 & 0{,}11 \\ 0{,}28 & 0{,}13 & 0{,}54 & 3{,}12 \\ 0{,}54 & 0{,}18 & 2{,}11 & 3{,}08 \\ 2{,}33 & 0{,}11 & 0{,}22 & 2{,}22 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}00 & 0{,}13 & 0{,}25 & 0{,}82 \\ -0{,}15 & 0{,}27 & 0{,}35 & -0{,}44 \\ 0{,}83 & 0{,}11 & 0{,}72 & -0{,}32 \\ 0{,}94 & 0{,}08 & 0{,}32 & 0{,}12 \end{vmatrix}$.

№ 28.

1) $\begin{pmatrix} 0{,}54 & 0{,}32 & 1{,}00 & 0{,}22 \\ 0{,}66 & 0{,}44 & 0{,}22 & 1{,}00 \\ 1{,}00 & 0{,}42 & 0{,}54 & 0{,}66 \\ 0{,}42 & 1{,}00 & 0{,}32 & 0{,}44 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}03 & 0{,}88 & 0{,}64 & 0{,}12 \\ 0{,}12 & 0{,}62 & -0{,}13 & 0{,}32 \\ 0{,}18 & 0\,25 & 0{,}42 & 0{,}82 \\ 0{,}32 & 0{,}43 & 0{,}85 & 0{,}93 \end{vmatrix}$.

№ 29.

1) $\begin{pmatrix} -0{,}33 & 0{,}42 & 0{,}51 & -0{,}11 \\ 2{,}71 & -0{,}92 & -2{,}17 & 0{,}81 \\ 0{,}75 & 0{,}68 & 0{,}33 & 0{,}17 \\ 0{,}28 & -3{,}71 & 2{,}17 & 0{,}16 \end{pmatrix}$;

2) $\begin{vmatrix} 1{,}00 & 0{,}27 & 0{,}64 & 0{,}83 \\ 0{,}27 & 0{,}35 & -0{,}81 & 0{,}16 \\ 0{,}64 & -0{,}81 & -0{,}14 & 0{,}15 \\ 0{,}83 & -0{,}14 & 0{,}25 & 0{,}37 \end{vmatrix}$.

№ 30.

1) $\begin{pmatrix} 0{,}72 & 3{,}54 & 7{,}28 & 0{,}33 \\ -0{,}28 & -0{,}72 & 3{,}04 & 0.22 \\ 1{,}00 & 0{,}35 & -0{,}78 & 1{,}00 \\ 7{,}03 & -5{,}04 & -3{,}75 & 3{,}41 \end{pmatrix}$;

2) $\begin{vmatrix} 0{,}52 & 0{,}42 & 0{,}36 & 0{,}84 \\ 0{,}42 & 0{,}56 & 0{,}83 & -0{,}73 \\ 0{,}36 & 0{,}83 & -0{,}13 & 0{,}28 \\ 0{,}84 & 0{,}24 & -0{,}38 & 0{,}49 \end{vmatrix}$.

Образец выполнения задания

1) $A = \begin{pmatrix} 0{,}32 & 0{,}52 & -0{,}42 & 0{,}23 \\ 0{,}44 & -0{,}25 & 0{,}36 & -0{,}51 \\ -1{,}06 & 0{,}74 & -0{,}83 & 0{,}48 \\ 0{,}96 & 0{,}82 & 0{,}55 & 0{,}36 \end{pmatrix}$; 2) $\Delta = \begin{vmatrix} 0{,}32 & 0{,}54 & 0{,}67 & -0{,}82 \\ 0{,}84 & 0{,}88 & -0{,}35 & 0{,}71 \\ 1{,}02 & 0{,}32 & 0{,}48 & 0{,}57 \\ -0{,}18 & 0{,}64 & -0{,}24 & 0{,}43 \end{vmatrix}$.

1) Вычисление обратной матрицы производим в следующей таблице:

Элементы данной матрицы				Элементы единичной матрицы				Контрольные суммы	Строчные суммы
j_1	j_2	j_3	j_4	j_1	j_2	j_3	j_4	$\sum$	$\sum'$
0,32	0,52	−0,42	0,23	1	0	0	0	1,65	1,65
0,44	−0,25	0,36	−0,51	0	1	0	0	1,04	1,04
−1,06	0,74	−0,83	0,48	0	0	1	0	0,33	0,33
0,96	0,82	0,55	0,36	0	0	0	1	3,69	3,69
1	1,625	−1,3125	0,7188	3,125	0	0	0	5,1562	5,1562
	−0,965	0,9375	−0,8262	−1,375	1	0	0	−1,2288	−1,2287
	2,4625	−2,2212	1,2419	3,3125	0	1	0	5,7956	5,7956
	−0,74	1,81	−0,33	3,000	0	0	1	−1,26	−1,26
	1	−0,9715	0,8562	1,4249	−1,0363	0	0	1,2733	1,2733
		0,1711	−0,8666	−0,1962	2,5518	1	0	2,6601	2,6601
		1,0911	0,3036	−1,9456	−0,7668	0	1	−0,3177	−0,3177
			5,8306	−0,6940	−17,0425	−6,3781	1	−17,2837	−17,284
			1	−0,1190	−2,9229	−1,0939	0,1715	−2,9643	−2,9643
			1	−0,1190	−2,9229	−1,0939	0,1715	−2,9643	
		1		−1,7500	0,1105	0,3044	0,8688	0,5336	
	1			−0,1734	1,5737	1,2323	0,6972	4,3298	
1				1,1954	−0,3114	−0,8168	−0,1159	0,9512	

Ответ:

$$A^{-1}=\begin{pmatrix} 1,1954 & -0,3114 & -0,8168 & -0,1159 \\ -0,1734 & 1,5737 & 1,2323 & 0,6972 \\ -1,7500 & 0,1105 & 0,3044 & 0,8688 \\ -0,1190 & -2,9229 & -1,0939 & 0,1715 \end{pmatrix}$$

2) Вычисление определителя производим в таблице:

Элементы определителя				Контрольные суммы Σ
[0,32]	0,54	0,67	−0,82	0,71
0,84	0,88	−0,35	0,71	2,08
1,02	0,32	0,48	0,57	2,39
−0,18	0,64	−0,24	0,43	0,65
1	1,6875	2,09375	−2,5625	2,21875
	[−0,5375]	−2,10875	2,8625	0,21625
	−1,40125	−1,65562	3,18375	0,12688
	0,94375	0,13688	−0,03125	1,04938
	1	3,92326	−5,32558	−0,40232
		[3,84184]	−4,27872	−0,43687
		−3,56570	4,99477	1,42907
		1	−1,11372	−0,11371
			[1,02358]	1,0236
0,32	−0,5375	3,84184	1,02358	

$$\Delta = 0{,}32 \cdot (-0{,}5375) \cdot 3{,}84184 \cdot 1{,}02358 = -0{,}676378.$$

Ответ: $\Delta \approx -0{,}6764$.

Задание Методом итераций решить систему линейных уравнений с точностью до 0,001, предварительно оценив число необходимых для этого шагов

№ 1.
$$\begin{cases} x_1 = 0{,}23x_1 - 0{,}04x_2 + 0{,}21x_3 - 0{,}18x_4 + 1{,}24; \\ x_2 = 0{,}45x_1 - 0{,}23x_2 + 0{,}06x_3 - 0{,}88; \\ x_3 = 0{,}26x_1 + 0{,}34x_2 - 0{,}11x_3 + 0{,}62; \\ x_4 = 0{,}05x_1 - 0{,}26x_2 + 0{,}34x_3 - 0{,}12x_4 - 1{,}17. \end{cases}$$

№ 2.
$$\begin{cases} x_1 = 0{,}21x_1 + 0{,}12x_2 - 0{,}34x_3 - 0{,}16x_4 - 0{,}64; \\ x_2 = 0{,}34x_1 - 0{,}08x_2 + 0{,}17x_3 - 0{,}18x_4 + 1{,}42; \\ x_3 = 0{,}16x_1 + 0{,}34x_2 + 0{,}15x_3 - 0{,}31x_4 - 0{,}42; \\ x_4 = 0{,}12x_1 - 0{,}26x_2 - 0{,}08x_3 + 0{,}25x_4 + 0{,}83. \end{cases}$$

№ 3.
$$\begin{cases} x_1 = 0{,}32x_1 - 0{,}18x_2 + 0{,}02x_3 + 0{,}21x_4 + 1{,}83; \\ x_2 = 0{,}16x_1 + 0{,}12x_2 - 0{,}14x_3 + 0{,}27x_4 - 0{,}65; \\ x_3 = 0{,}37x_1 + 0{,}27x_2 - 0{,}02x_3 - 0{,}24x_4 + 2{,}23; \\ x_4 = 0{,}12x_1 + 0{,}21x_2 - 0{,}18x_3 + 0{,}25x_4 - 1{,}13. \end{cases}$$

№ 4.
$$\begin{cases} x_1 = 0{,}42x_1 - 0{,}32x_2 + 0{,}03x_3 + 0{,}44; \\ x_2 = 0{,}11x_1 - 0{,}26x_2 - 0{,}36x_3 + 1{,}42; \\ x_3 = 0{,}12x_1 + 0{,}08x_2 - 0{,}14x_3 - 0{.}24x_4 - 0{,}83; \\ x_4 = 0{,}15x_1 - 0{,}35x_2 - 0{,}18x_3 - 1{.}42. \end{cases}$$

№ 5.
$$\begin{cases} x_1 = 0{,}18x_1 - 0{,}34x_2 - 0{,}12x_3 + 0{.}15x_4 - 1{,}33; \\ x_2 = 0{,}11x_1 + 0{,}23x_2 - 0{,}15x_3 + 0{.}32x_4 + 0{,}84; \\ x_3 = 0{,}05x_1 - 0{,}12x_2 + 0{,}14x_3 - 0{.}18x_4 - 1{,}16; \\ x_4 = 0{,}12x_1 + 0{,}08x_2 + 0{,}06x_3 + 0{.}57. \end{cases}$$

№ 6.
$$\begin{cases} x_1 = 0{,}13x_1 + 0{,}23x_2 - 0{,}44x_3 - 0{.}05x_4 + 2{.}13; \\ x_2 = 0{,}24x_1 - 0{,}31x_3 + 0{,}15x_4 - 0{.}18; \\ x_3 = 0{,}06x_1 + 0{,}15x_2 - 0{,}23x_4 + 1{.}44; \\ x_4 = 0{,}72x_1 - 0{,}08x_2 - 0{,}05x_3 + 2{.}42. \end{cases}$$

№ 7.
$$\begin{cases} x_1 = \ \ 0{,}17x_1 + 0{,}31x_2 - 0{,}18x_3 + 0{,}22x_4 - 1{,}71; \\ x_2 = -0{,}21x_1 + 0{,}33x_3 + 0{,}22x_4 + 0{,}62; \\ x_3 = \ \ 0{,}32x_1 - 0{,}18x_2 + 0{,}05x_3 - 0{,}19x_4 - 0{,}89; \\ x_4 = \ \ 0{,}12x_1 + 0{,}28x_2 - 0{,}14x_3 + 0{,}94. \end{cases}$$

№ 8.
$$\begin{cases} x_1 = \ \ 0{,}13x_1 + 0{,}27x_2 - 0{,}22x_3 - 0{,}18x_4 + 1{,}21; \\ x_2 = -0{,}21x_1 - 0{,}45x_3 + 0{,}18x_4 - 0{,}33; \\ x_3 = \ \ 0{,}12x_1 + 0{,}13x_2 - 0{,}33x_3 + 0{,}18x_4 - 0{,}48; \\ x_4 = \ \ 0{,}33x_1 - 0{,}05x_2 + 0{,}06x_3 - 0{,}28x_4 - 0{,}17. \end{cases}$$

№ 9.
$$\begin{cases} x_1 = \ \ 0{,}19x_1 - 0{,}07x_2 + 0{,}38x_3 - 0{,}21x_4 - 0{,}81; \\ x_2 = -0{,}22x_1 + 0{,}08x_2 + 0{,}11x_3 + 0{,}33x_4 - 0{,}64; \\ x_3 = \ \ 0{,}51x_1 - 0{,}07x_2 + 0{,}09x_3 - 0{,}11x_4 + 1{,}71; \\ x_4 = \ \ 0{,}33x_1 - 0{,}41x_2 - 1{,}21. \end{cases}$$

№ 10.
$$\begin{cases} x_1 = 0{,}22x_2 - 0{,}11x_3 + 0{,}31x_4 + 2{,}7; \\ x_2 = 0{,}38x_1 - 0{,}12x_3 + 0{,}22x_4 - 1{,}5; \\ x_3 = 0{,}11x_1 + 0{,}23x_2 - 0{,}51x_4 + 1{,}2; \\ x_4 = 0{,}17x_1 - 0{,}21x_2 + 0{,}31x_3 - 0{,}17. \end{cases}$$

№ 11.
$$\begin{cases} x_1 = 0{,}07x_1 - 0{,}08x_2 + 0{,}11x_3 - 0{,}18x_4 - 0{,}51; \\ x_2 = 0{,}18x_1 + 0{,}52x_2 + 0{,}21x_4 + 1{,}17; \\ x_3 = 0{,}13x_1 + 0{,}31x_2 - 0{,}21x_4 - 1{,}02; \\ x_4 = 0{,}08x_1 - 0{,}33x_3 + 0{,}28x_4 - 0{,}28. \end{cases}$$

№ 12.
$$\begin{cases} x_1 = 0{,}05x_1 - 0{,}06x_2 - 0{,}12x_3 + 0{,}14x_4 - 2{,}17; \\ x_2 = 0{,}04x_1 - 0{,}12x_2 + 0{,}08x_3 + 0{,}11x_4 + 1{,}4; \\ x_3 = 0{,}34x_1 + 0{,}08x_2 - 0{,}06x_3 + 0{,}14x_4 - 2{,}1; \\ x_4 = 0{,}11x_1 + 0{,}12x_2 - 0{,}03x_4 - 0{,}8. \end{cases}$$

№ 13.
$$\begin{cases} x_1 = 0{,}08x_1 - 0{,}03x_2 - 0{,}04x_4 - 1{,}2; \\ x_2 = 0{,}31x_2 + 0{,}27x_3 - 0{,}08x_4 + 0{,}81; \\ x_3 = 0{,}33x_1 - 0{,}07x_3 + 0{,}21x_4 - 0{,}92; \\ x_4 = 0{,}11x_1 + 0{,}03x_3 + 0{,}58x_4 + 0{,}17. \end{cases}$$

№ 14.
$$\begin{cases} x_1 = 0{,}12x_1 - 0{,}23x_2 + 0{,}25x_3 - 0{,}16x_4 + 1{,}24; \\ x_2 = 0{,}14x_1 + 0{,}34x_2 - 0{,}18x_3 + 0{,}24x_4 - 0{,}89; \\ x_3 = 0{,}33x_1 + 0{,}03x_2 + 0{,}16x_3 - 0{,}32x_4 + 1{,}15; \\ x_4 = 0{,}12x_1 - 0{,}05x_2 + 0{,}15x_4 - 0{,}57. \end{cases}$$

№ 15.
$$\begin{cases} x_1 = 0{,}23x_1 - 0{,}14x_2 + 0{,}06x_3 - 0{,}12x_4 + 1{,}21; \\ x_2 = 0{,}12x_1 + 0{,}32x_3 - 0{,}18x_4 - 0{,}72; \\ x_3 = 0{,}08x_1 - 0{,}12x_2 + 0{,}23x_3 + 0{,}32x_4 - 0{,}58; \\ x_4 = 0{,}25x_1 + 0{,}22x_2 + 0{,}14x_3 + 1{,}56. \end{cases}$$

№ 16.
$$\begin{cases} x_1 = 0{,}14x_1 + 0{,}23x_2 + 0{,}18x_3 + 0{,}17x_4 - 1{,}42; \\ x_2 = 0{,}12x_1 - 0{,}14x_2 + 0{,}08x_3 + 0{,}09x_4 - 0{,}83; \\ x_3 = 0{,}16x_1 + 0{,}24x_2 - 0{,}35x_4 + 1{,}21; \\ x_4 = 0{,}23x_1 - 0{,}08x_2 + 0{,}05x_3 + 0{,}25x_4 + 0{,}65. \end{cases}$$

№ 17.
$$\begin{cases} x_1 = 0{,}24x_1 + 0{,}21x_2 + 0{,}06x_3 - 0{,}34x_4 + 1{,}42; \\ x_2 = 0{,}05x_1 + 0{,}32x_3 + 0{,}12x_4 - 0{,}57; \\ x_3 = 0{,}35x_1 - 0{,}27x_2 - 0{,}05x_4 + 0{,}68; \\ x_4 = 0{,}12x_1 - 0{,}43x_2 + 0{,}04x_3 - 0{,}21x_4 - 2{,}14. \end{cases}$$

№ 18.
$$\begin{cases} x_1 = 0{,}17x_1 + 0{,}27x_2 - 0{,}13x_3 - 0{,}11x_4 - 1{,}42; \\ x_2 = 0{,}13x_1 - 0{,}12x_2 + 0{,}09x_3 - 0{,}06x_4 + 0{,}48; \\ x_3 = 0{,}11x_1 + 0{,}05x_2 - 0{,}02x_3 + 0{,}12x_4 - 2{,}34; \\ x_4 = 0{,}13x_1 + 0{,}18x_2 + 0{,}24x_3 + 0{,}43x_4 + 0{,}72. \end{cases}$$

№ 19.
$$\begin{cases} x_1 = 0{,}15x_1 + 0{,}05x_2 - 0{,}08x_3 + 0{,}14x_4 - 0{,}48; \\ x_2 = 0{,}32x_1 - 0{,}13x_2 - 0{,}12x_3 + 0{,}11x_4 + 1{,}24; \\ x_3 = 0{,}17x_1 + 0{,}06x_2 - 0{,}08x_3 + 0{,}12x_4 + 1{,}15; \\ x_4 = 0{,}21x_1 - 0{,}16x_2 + 0{,}36x_3 - 0{,}88. \end{cases}$$

№ 20.
$$\begin{cases} x_1 = 0{,}28x_2 - 0{,}17x_3 + 0{,}06x_4 + 0{,}21; \\ x_2 = 0{,}52x_1 + 0{,}12x_3 + 0{,}17x_4 - 1{,}17; \\ x_3 = 0{,}17x_1 - 0{,}18x_2 + 0{,}21x_3 - 0{,}81; \\ x_4 = 0{,}11x_1 + 0{,}22x_2 + 0{,}03x_3 + 0{,}05x_4 + 0{,}72. \end{cases}$$

№ 21.
$$\begin{cases} x_1 = 0{,}52x_2 + 0{,}08x_3 + 0{,}13x_4 - 0{,}22; \\ x_2 = 0{,}07x_1 - 0{,}38x_2 - 0{,}05x_3 + 0{,}41x_4 + 1{,}8; \\ x_3 = 0{,}04x_1 + 0{,}42x_2 + 0{,}11x_3 - 0{,}07x_4 - 1{,}3; \\ x_4 = 0{,}17x_1 + 0{,}18x_2 - 0{,}13x_3 + 0{,}19x_4 + 0{,}33. \end{cases}$$

№ 22. $\begin{cases} x_1 = 0{,}01x_1 + 0{,}02x_2 - 0{,}62x_3 + 0{,}08x_4 - 1{,}3; \\ x_2 = 0{,}03x_1 + 0{,}28x_2 + 0{,}33x_3 - 0{,}07x_4 + 1{,}1; \\ x_3 = 0{,}09x_1 + 0{,}13x_2 + 0{,}42x_3 + 0{,}28x_4 - 1{,}7; \\ x_4 = 0{,}19x_1 - 0{,}23x_2 + 0{,}08x_3 + 0{,}37x_4 + 1{,}5. \end{cases}$

№ 23. $\begin{cases} x_1 = 0{,}17x_2 - 0{,}33x_3 + 0{,}18x_4 - 1{,}2; \\ x_2 = 0{,}18x_2 + 0{,}43x_3 - 0{,}08x_4 + 0{,}33; \\ x_3 = 0{,}22x_1 + 0{,}18x_2 + 0{,}21x_3 + 0{,}07x_4 + 0{,}48; \\ x_4 = 0{,}08x_1 + 0{,}07x_2 + 0{,}21x_3 + 0{,}04x_4 - 1{,}2. \end{cases}$

№ 24. $\begin{cases} x_1 = 0{,}03x_1 - 0{,}05x_2 + 0{,}22x_3 - 0{,}33x_4 + 0{,}43; \\ x_2 = 0{,}22x_1 + 0{,}55x_2 - 0{,}08x_3 + 0{,}07x_4 - 1{,}8; \\ x_3 = 0{,}33x_1 + 0{,}13x_2 - 0{,}08x_3 - 0{,}05x_4 - 0{,}8; \\ x_4 = 0{,}08x_1 + 0{,}17x_2 + 0{,}29x_3 + 0{,}33x_4 + 1{,}7. \end{cases}$

№ 25. $\begin{cases} x_1 = 0{,}13x_1 + 0{,}22x_2 - 0{,}33x_3 + 0{,}07x_4 + 0{,}11; \\ x_2 = 0{,}45x_2 - 0{,}23x_3 + 0{,}07x_4 - 0{,}33; \\ x_3 = 0{,}11x_1 - 0{,}08x_3 + 0{,}18x_4 + 0{,}85; \\ x_4 = 0{,}08x_1 + 0{,}09x_2 + 0{,}33x_3 + 0{,}21x_4 - 1{,}7. \end{cases}$

№ 26. $\begin{cases} x_1 = 0{,}32x_1 - 0{,}16x_2 - 0{,}08x_3 + 0{,}15x_4 + 2{,}42; \\ x_2 = 0{,}16x_1 - 0{,}23x_2 + 0{,}11x_3 - 0{,}21x_4 + 1{,}43; \\ x_3 = 0{,}05x_1 - 0{,}08x_2 + 0{,}34x_4 - 0{,}16; \\ x_4 = 0{,}12x_1 + 0{,}14x_2 - 0{,}18x_3 + 0{,}06x_4 + 1{,}62. \end{cases}$

Образец выполнения задания

$$\begin{cases} x_1 = 0{,}32x_1 - 0{,}05x_2 + 0{,}11x_3 - 0{,}08x_4 + 2{,}15; \\ x_2 = 0{,}11x_1 + 0{,}16x_2 - 0{,}28x_3 - 0{,}06x_4 - 0{,}83; \\ x_3 = 0{,}08x_1 - 0{,}15x_2 + 0{,}12x_4 + 1{,}16; \\ x_4 = -0{,}21x_1 + 0{,}13x_2 - 0{,}27x_3 + 0{,}44. \end{cases}$$

Число шагов, дающих наверняка ответ с точностью до 0,001, определим с помощью соотношения

$$\| X^* - X^k \| \leqslant \frac{\| A \|^{k+1}}{1 - \| A \|} \cdot \| F \| \leqslant 0{,}001.$$

Здесь $\| A \|_1 = \max \{0{,}56;\ 0{,}61;\ 0{,}35;\ 0{,}61\} < 1$; значит, итерационный процесс сходится; $\| F \|_1 = 2{,}15$. Имеем

$$\frac{0{,}61^{k+1}}{0{,}39} \cdot 2{,}15 < 0{,}001; \quad 0{,}61^{k+1} < \frac{0{,}001 \cdot 0{,}39}{2{,}15};$$

$$(k+1) \cdot \lg 0{,}61 < -3 + \lg 0{,}39 - \lg 2{,}15;$$

$$k+1 > \frac{-3 + 1{,}5911 - 0{,}3324}{1{,}7853} = \frac{3{,}7413}{0{,}2147} = 17{,}5; \quad k \geqslant 17.$$

Вычисления располагаем в таблице:

k	x_1	x_2	x_3	x_4
0	2,15	−0,83	1,16	0,44
1	2.9719	−1,0775	1,5093	−0,4326
2	2.3555	−1,0721	1,5075	−0,7317
3	3.5017	−1,0106	1,5015	−0,8111
4	3.5511	−0,9277	1,4944	−0,8321
5	3.5637	−0,9563	1,4834	−0,8298
6	3.5678	−0,9566	1,4890	−0,8332
7	3.5700	−0,9575	1,4889	−0,8356
8	3.5709	−0,9573	1,4890	−0,8362
9	3.5712	−0,9571	1,4889	−0,8364
10	3.5713	−0,9570	1,4890	−0,8364

Сходимость в тысячных долях имеет место уже на 10-м шаге.
Ответ: $x_1 \approx 3{,}571$; $x_2 \approx -0{,}957$; $x_3 \approx 1{,}489$; $x_4 \approx -0{,}836$.

Тема 4. Интерполирование и экстраполирование функций

Задание 1

Найти приближенное значение функции при данном значении аргумента с помощью интерполяционного многочлена Лагранжа, если функция задана

1) в неравноотстоящих узлах таблицы
2) в равноотстоящих узлах таблицы

Варианты к заданию 1

Варианты 1,7,13,19,25			Варианты 2,8,14,20,26		
Дано		Найти У в точке х	Дано		Найти У в точке х
х	у		х	у	
0,43	1,63597	0,702	0,02	1,02316	0,102
0,48	1,73234	0,512	0,08	1,09590	0,114
0,55	1,87686	0,645	0,12	1,14725	0,125
0,62	2,03345	0,736	0,17	1,21483	0,203
0,70	2,22846	0,608	0,23	1,30120	0,154
0,75	2,35973		0,30	1,40976	

Варианты 3,9,15,21,27			Варианты 4,10,16,22,28		
Дано		Найти У в точке х	Дано		Найти У в точке х
х	у		х	у	
0,35	2,73951	0,526	0,41	2,57418	0,616
0,41	2,30080	0,453	0,46	2,32513	0,478
0,47	1,96864	0,482	0,52	2,09336	0,665
0,51	1,78776	0,552	0,60	1,86203	0,537
0,56	1,59502	0,436	0,65	1,74926	0,673
0,64	1,34310		0,70	1,62098	

Варианты 5,11,17,23,29			Варианты 6,12,18,24,30		
Дано		Найти У в точке х	Дано		Найти У в точке х
х	у		х	у	
0,68	0,80866	0,896	0,11	9,05421	0,314
0,73	0,89492	0,812	0,15	6,61659	0,235
0,80	1,02964	0,774	0,21	4,69170	0,332
0,88	1,20966	0,955	0,29	3,35106	0,275
0,93	1,34087	0,715	0,35	2,73951	0,186
0,99	1,52368		0,40	2,36522	

Варианты к заданию 2

Варианты 1,7,13,19,25			Варианты 2,8,14,20,26		
Дано		Найти У в точке х	Дано		Найти У в точке х
х	у		х	у	
1,375	5,04192	1,3832	0,115	8,65729	0,1264
1,380	5,17744	1,3926	0,120	8,29329	0,1315
1,385	5,32016	1,3862	0,125	7,95829	0,1232
1,390	5,47069	1,3934	0,130	7,64493	0,1334
1,395	5,62968	1,3866	0,135	7,36235	0,1285
1,400	5,79788		0,140	7,09613	

Варианты 3,9,15,21,27			Варианты 4,10,16,22,28		
Дано		Найти У в точке х	Дано		Найти У в точке х
х	у		х	у	
0,150	6,61659	0,1521	0,180	5,61543	0,1838
0,155	6,39989	0,1611	0,185	5,46693	0,1875
0,160	6,19658	0,1662	0,190	5,32634	0,1944
0,165	6,00551	0,1542	0,195	5,19304	0,1976
0,170	5,82558	0,1625	0,200	5,06649	0,2038
0,175	5,65583		0,205	4,94619	

Варианты 5,11,17,23,29			Варианты 6,12,18,24,30		
Дано		Найти У в точке х	Дано		Найти У в точке х
х	у		х	у	
0,210	4,83170	0,2121	1,415	0,888551	1,4179
0,215	4,72261	0,2165	1,420	0,889599	1,4258
0,220	4,61855	0,2232	1,425	0,890637	1,4396
0,225	4,51919	0,2263	1,430	0,891667	1,4236
0,230	4,42422	0,2244	1,435	0,892687	1,4315
0,235	4,33337		1,440	0,893698	

Образец выполнения задания

1)

x	y
0,05	0,050042
0,10	0,100335
0,17	0,171657
0,25	0,255342
0,30	0,309336
0,36	0,376403

Вычислить значение функции $f(x)=y(x)$ при $x=0{,}263$.

2)

x	y
0,101	1,26183
0,106	1,27644
0,111	1,29122
0,116	1,30617
0,121	1,32130
0,126	1,32660

Определить значение функции $y(x)$ при $x=0{,}1157$.

1) Воспользуемся формулой

$$f(x)\approx\Pi_{n+1}\cdot\sum_{i=0}^{n}(y_i/D_i)$$

где

$$\Pi_{n+1}=(x-x_0)(x-x_1)\ldots(x-x_n),$$

$$D_i=(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x-x_i)(x_i-x_{i+1})\ldots(x_i-x_n).$$

Вычисления приведены в таблице.

i	Разности						D_i	y_i/D_i
0	0,213	−0,05	−0,12	−0,20	−0,25	−0,31	$-0{,}19809\cdot10^{-4}$	−2526,2
1	0,05	0,163	−0,07	−0,15	−0,20	−0,26	$0{,}44499\cdot10^{-5}$	25547,7
2	0,12	0,07	0,093	−0,08	−0,13	−0,19	$-0{,}154365\cdot10^{-5}$	−111202,0
3	0,20	0,15	0,08	0,013	−0,05	−0,11	$0{,}1716\cdot10^{-6}$	1488007,0
4	0,25	0,20	0,13	0,05	−0,037	−0,06	$0{,}7215\cdot10^{-6}$	428740,0
5	0,31	0,26	0,19	0,11	0,06	−0,097	$-0{,}980402\cdot10^{-6}$	−38392,7

Итак, $\Pi_{5+1}=0{,}1506492\cdot10^{-6}$, $\sum_{i=0}^{5}(y_i/D_i)=1790173{,}8$. Следовательно,

$$f(0{,}263)\approx\Pi_{5+1}\cdot\sum_{i=0}^{5}(y_i/D_i)=0{,}1506492\cdot10^{-6}\cdot1790173{,}8=0{,}269678.$$

2) Для вычислений используем формулу

$$f(x)=y(x)\approx\Pi_{n+1}(t)\sum_{i=0}^{n}\frac{y_i}{(t-i)C_i},$$

где

$$\Pi_{n+1}(t)=t(t-1)\ldots(t-n);\quad t=(x-x_0)/h;\quad h=x_{i+1}-x_i;$$

i	x_i	l_i'	$t-i$	C_i	$(t-i)\cdot C_i$	$\frac{y_i}{(t-i)C_i}$
0	0,101	1,26183	2,94	−120	−352,8	−0,0035766
1	0,106	1,27644	1,94	24	46,56	0,0274149
2	0,111	1,29122	0,94	−12	−11,28	−0,1144691
3	0,116	1,30617	−0,06	12	−0,72	−1,8141250
4	0,121	1,32130	−1,06	−24	25,44	0,0519379
5	0,126	1,33660	−2,06	120	−247,2	−0,0054069

Итак, $\Pi_{5+1}(t)=-0{,}7024271$; $\sum_{i=0}^{5}\frac{y_i}{(t-i)C_i}=-1{,}858225$. Следовательно,

$f(0{,}1157)\approx1{,}30527$.

Тема 6. Численное решение обыкновенных дифференциальных уравнений

Задание 1

Составить решение задачи Коши для обыкновенного дифференциального уравнения первого порядка методом Эйлера- Коши. Вычисления выполнять с четырьмя десятичными знаками.

№ 1 $y'=0.133*(x^2+\sin 2x)+0.872y$	№ 2 $y'=0.215*(x^2+\cos 1.5x)+1.283y$
№ 3 $y'=0.158*(x^2+\sin 0.8x)+1.164y$	№ 4 $y'=0.133*(x^2+\sin 2x)+0.754y$
№ 5 $y'=0.221*(x^2+\sin 1.2x)+0.452y$	№ 6 $y'=0.163*(x^2+\cos 0.4x)+0.635y$
№ 7 $y'=0.218*(x^2+\sin 1.6x)+0.718y$	№ 8 $y'=0.145*(x^2+\cos 0.5x)+0.843y$
№ 9 $y'=0.213*(x^2+\sin 1.8x)+0.368y$	№ 10 $y'=0.127*(x^2+\cos 0.6x)+0.573y$
№ 11 $y'=0.232*(x^2+\sin 1.4x)+1.453y$	№ 12 $y'=0.417*(x^2+\cos 0.8x)+0.972y$
№ 13 $y'=0.324*(x^2+\sin 1.5x)+1.612y$	№ 14 $y'=0.263*(x^2+\cos 1.2x)+0.453y$
№ 15 $y'=0.221*(x^2+\sin 1.2x)+0.452y$	№ 16 $y'=0.343*(x^2+\cos 0.4x)+1.315y$
№ 17 $y'=0.276*(x^2+\sin 1.6x)+0.988y$	№ 18 $y'=0.173*(x^2+\cos 0.6x)+1.534y$
№ 19 $y'=0.258*(x^2+\sin 0.4x)+0.724y$	№ 20 $y'=0.317*(x^2+\cos 1.4x)+1.344y$
№ 21 $y'=0.166*(x^2+\sin 1.1x)+0.883y$	№ 22 $y'=0.215*(x^2+\cos 0.9x)+1.213y$
№ 23 $y'=0.188*(x^2+\sin 1.5x)+1.213y$	№ 24 $y'=0.314*(x^2+\cos 0.6x)+0.772y$
№ 25 $y'=0.418*(x^2+\sin 1.2x)+1.344y$	№ 26 $y'=0.273*(x^2+\cos 1.3x)+0.687y$
№ 27 $y'=0.176*(x^2+\sin 0.8x)+0.452y$	№ 28 $y'=0.245*(x^2+\cos 0.4x)+1.452y$
№ 29 $y'=0.184*(x^2+\sin 0.6x)+0.747y$	№ 30 $y'=0.212*(x^2+\cos 1.2x)+1.544y$

Задание 2

Используя метод Эйлера с уточнением, составить таблицу приближенных значений интеграла дифференциального уравнения $y'=f(x,y)$, удовлетворяющего начальным условиям $y(x_0)=y_0$ на отрезке [a,b] шаг h=0.1. Все вычисления вести с четырьмя десятичными знаками.

№ 1 $y'=x+\cos\frac{y}{\sqrt{5}}$	$y_0(1.8)=2.6$	$x\in[1.8;2.8]$
№ 2 $y'=x+\cos\frac{y}{3}$	$y_0(1.6)=2.6$	$x\in[1.6;2.6]$
№ 3 $y'=x+\cos\frac{y}{\sqrt{10}}$	$y_0(0.6)=0{,}8$	$x\in[0.6;1.6]$
№ 4 $y'=x+\cos\frac{y}{\sqrt{7}}$	$y_0(0{,}5)=0.6$	$x\in[0{,}5;1{,}5]$
№ 5 $y'=x+\cos\frac{y}{\pi}$	$y_0(1.7)=5{,}3$	$x\in[1.7;2.7]$
№ 6 $y'=x+\cos\frac{y}{2{,}25}$	$y_0(1.4)=2.2$	$x\in[1.4;2.4]$
№ 7 $y'=x+\cos\frac{y}{\ell}$	$y_0(1.4)=2.5$	$x\in[1.4;2.4]$

№ 8 $y' = x + \cos\frac{y}{\sqrt{2}}$ $y_0(0,8) = 1,4$ $x \in [0,8;1,8]$

№ 9 $y' = x + \cos\frac{y}{\sqrt{3}}$ $y_0(1.2) = 2.1$ $x \in [1.2;2.2]$

№ 10 $y' = x + \cos\frac{y}{\sqrt{11}}$ $y_0(2,1) = 2.5$ $x \in [2,1;3,1]$

№ 11 $y' = x + \sin\frac{y}{\sqrt{5}}$ $y_0(1.8) = 2.6$ $x \in [1.8;2.8]$

№ 12 $y' = x + \sin\frac{y}{3}$ $y_0(1.6) = 4.6$ $x \in [1.6;2.6]$

№ 13 $y' = x + \sin\frac{y}{\sqrt{10}}$ $y_0(0.6) = 0.8$ $x \in [0.6;1.6]$

№ 14 $y' = x + \sin\frac{y}{\sqrt{7}}$ $y_0(0,5) = 0.6$ $x \in [0,5;1,5]$

№ 15 $y' = x + \sin\frac{y}{n}$ $y_0(1.7) = 5.3$ $x \in [1.7;2.7]$

№ 16 $y' = x + \sin\frac{y}{\sqrt{2.8}}$ $y_0(1.4) = 2.2$ $x \in [1.4;2.4]$

№ 17 $y' = x + \sin\frac{y}{e}$ $y_0(1.4) = 2.5$ $x \in [1.4;2.4]$

№ 18 $y' = x + \sin\frac{y}{\sqrt{2}}$ $y_0(0.8) = 1.3$ $x \in [0.8;1.8]$

№ 19 $y' = x + \sin\frac{y}{\sqrt{3}}$ $y_0(1.1) = 1.5$ $x \in [1.1;2.1]$

№ 20 $y' = x + \sin\frac{y}{\sqrt{11}}$ $y_0(0.6) = 1.2$ $x \in [0.6;1.6]$

№ 21 $y' = x + \sin\frac{y}{1.25}$ $y_0(0.5) = 1.8$ $x \in [0.5;1.5]$

№ 22 $y' = x + \sin\frac{y}{\sqrt{15}}$ $y_0(0.2) = 1.1$ $x \in [0.2;1.2]$

№ 23 $y' = x + \sin\frac{y}{\sqrt{1.3}}$ $y_0(0.1) = 0.8$ $x \in [0.1;1.1]$

№ 24 $y' = x + \sin\frac{y}{\sqrt{0.3}}$ $y_0(0.5) = 0.6$ $x \in [0.5;1.5]$

№ 25 $y' = x + \sin\frac{y}{\sqrt{0.7}}$ $y_0(1.2) = 1.4$ $x \in [1.2;2.2]$

№ 26 $y' = x + \cos\frac{y}{1.25}$ $y_0(0.4) = 0.8$ $x \in [0.4;1.4]$

№ 27 $y' = x + \cos\frac{y}{\sqrt{1.5}}$ $y_0(0.3) = 0.9$ $x \in [0.3;1.3]$

№ 28 $y' = x + \cos\frac{y}{\sqrt{1.3}}$ $y_0(1.2) = 1.8$ $x \in [1.2;2.2]$

№ 29 $y' = x + \cos\frac{y}{\sqrt{0.3}}$ $y_0(0.7) = 2.1$ $x \in [0.7;1.7]$

№ 30 $y' = x + \cos\frac{y}{\sqrt{0.7}}$ $y_0(0.9) = 1.7$ $x \in [0.9;1.9]$

Задание 3

Используя метод Рунге- Кутта, составить таблицу приближенных значений интеграла дифференциального уравнения y'=f(x,y), удовлетворяющего начальным условиям $y(x_0)=y_0$ на отрезке [0,1] шаг h=0.1. Все вычисления вести с четырьмя десятичными знаками..

№ 1 $y' = 1 + 0.2y\sin x - y^2, y(0) = 0$

№ 2 $y' = \cos(x + y) + 0.5(x - y), y(0) = 0$

№ 3 $y' = \frac{\cos x}{x+1} - 0.5y^2, y(0) = 0$

№ 4 $y' = (1 - y^2)\cos x + 0.6y, y(0) = 0$

№ 5 $y' = 1 + 0.4y\sin x - 1.5y^2, y(0) = 0$

№ 6 $y' = \frac{\cos y}{x+2} + 0.3y^2, y(0) = 0$

№ 7 $y' = \cos(1.5x + y) + (x - y), y(0) = 0$

№ 8 $y' = 1 - \sin(x + y) + \frac{0.5y}{x+2}, y(0) = 0$

№ 9 $y' = \frac{\cos y}{1.5 + x} + 0.1y^2, y(0) = 0$

№ 10 $y' = 0.6\sin x - 1.25y^2 + 1, y(0) = 0$

№ 11 $y' = \cos(2x + y) - 1.5(x - y), y(0) = 0$

№ 12 $y' = 1 - \frac{0.1y}{x+2} - \sin(2x + y), y(0) = 0$

№ 13 $y' = \frac{\cos y}{1.25 + x} - 0.1y^2, y(0) = 0$

№ 14 $y' = 1 + 0.8y\sin x - 2y^2, y(0) = 0$

№ 15 $y' = \cos(1.5x + y) + 1.5(x - y), y(0) = 0$

№ 16 $y' = 1 - \sin(2x + y) + \frac{0.3y}{x+2}, y(0) = 0$

№ 17 $y' = \frac{\cos y}{1.75 + x} - 0.5y^2, y(0) = 0$

№ 18 $y' = 1 + (1 - x)\sin y - (2 + x)y, y(0) = 0$

№ 19 $y' = (0{,}8 - y^2)\cos x + 0.3y, y(0) = 0$

№ 20 $y' = 1 + 2.2\sin x + 1.5y^2, y(0) = 0$

№ 21 $y' = \cos(x + y) + 0.75(x - y), y(0) = 0$

№ 22 $y' = 1 - \sin(1.25x + y) + \frac{0.5y}{x+2}, y(0) = 0$

№ 23 $y' = \frac{\cos y}{x+2} - 0.3y^2, y(0) = 0$

№ 24 $y' = 1 - \sin(1.75x + y) + \frac{0.1y}{x+2}, y(0) = 0$

№ 25 $y' = \frac{\cos y}{1.23 + x} - 0.5y^2, y(0) = 0$

№26 $y' = \cos(1.5x + y) - 2.25(x + y), y(0) = 0$

№ 27 $y'=\dfrac{\cos y}{1.5+x}-1{,}25y^2, y(0)=0$ № 28 $y'=1-(x-1)\sin y+2(x+y), y(0)=0$

№ 29 $y'=1-\sin(0.75x-y)+\dfrac{1.75y}{x+1}, y(0)=0$ № 30 $y'=\cos(x-y)-\dfrac{1.25y}{1.5+x}, y(0)=0$

Справочный материал

1 Решение задачи Коши для дифференциального уравнения n-го порядка

$$y^{(n)}=f\left(x,y,y',...,y^{(n-1)}\right)$$

заключается в отыскании функции *y(x),* удовлетворяющей этому уравнению и начальным условиям $y(x_0)=y_0$, $y'(x_0)=y_0',....,y^{(n-1)}(x_0)=y_0^{(n-1)}$, где x_0, y_0, y_0', …., $y_0^{(n-1)}$ - заданные числа.

Для решения поставленной задачи применяются следующие методы:

а) метод степенных рядов. Решение ищут в виде суммы ряда

$$y(x)=y(x_0)+\frac{y'(x_0)}{1!}(x-x_0)+\frac{y''(x_0)}{2!}(x-x_0)^2+....+\frac{y^{(n)}(x_0)}{n!}(x-x_0)^n+....$$

Приближенное решение задачи дает частичная сумма этого ряда.

б) метод Эйлера для уравнения $y'=f(x,y)$ с начальным условием $y(x0)=y0$.
Составляют таблицу значений $y_k=y(x_k)$, где $x_k=x_0+kh$ (k=0,1,2,…,n), h=(b-a)/n, $[a,b]$ - отрезок, на котором ищется решение. Значения y_{k+1} определяются по формуле
$y_{k+1}=y_k+hf(x_k, y_k)$ (k=0,1,2,…,n-1).
Погрешность вычислений на каждом шаге составляет $R_k=0{,}5h^2y''(\varepsilon)$, где
$x_k\le\varepsilon\le x_{k+1}$;

в) усовершенствованный метод ломаных.
Сначала вычисляют промежуточные значения

$$x_{k+1/2}=x_k+\frac{h}{2};\ y_{k+1/2}=y_k+\frac{h}{2}f(x_k,y_k),$$

а затем находят $y_{k+1}=y_k+hf(x_{k+1/2},y_{k+1/2})$. Метод имеет несколько большую точность, чем метод Эйлера;

г) усовершенствованный метод Эйлера – Коши.
Сначала вычисляют «грубое» значение

$$y_{k+1}^{(0)}=y_k+hf(x_k,y_k),$$

которое затем уточняют по формуле

$$y_{k+1}=y_k+\frac{1}{2}h[f(x_k,y_k)+f(x_{k+1},y_{k+1})].$$

Погрешность метода на каждом шаге имеет порядок h^3;

д) усовершенствованный метод Эйлера с уточнением.
Сначала вычисляют

$$y_{k+1}^{(0)}=y_k+hf(x_k,y_k),$$

а затем это значение уточняют по формуле

$$y^{(i)}{}_{k+1} = y_k + \frac{h}{2}\left[f(x_k, y_k) + f\left(x_{k+1}, y^{(i-1)}{}_{k+1}\right)\right] \text{ (i=1,2,….).}$$

Итерации продолжают до тех пор, пока в пределах требуемой точности два последовательных приближения не совпадут. Погрешность метода на каждом этапе имеет порядок h^3;

е) метод Рунге-Кутта.

На каждом шаге вычисления выполняют по формуле

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}\right),$$

где

$$k_1^{(i)} = hf(x_i, y_i),\ k_2^{(i)} = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1^{(i)}}{2}\right),$$

$$k_3^{(i)} = hf\left(x_i + \frac{h}{2} y_i + \frac{k_2^{(i)}}{2}\right),\ k_4^{(i)} = hf\left(x_i + h, y_i + k_3^{(i)}\right).$$

Причем $x_i = x_0 + ih$ $(i=0,1,2,\ldots,n)$. Погрешность метода на каждом шаге имеет порядок h^5.

ж) метод Адамса. Формула с первыми разностями:

$$y_{k+1} = y_k + q_k + \frac{1}{2}\Delta q_{k-1},\ q_k = hf(x_k, y_k) \text{ (k=1,2,….).}$$

Значение y_1 находят другим способом. Погрешность на каждом шаге имеет порядок h^2.

Формула со вторыми разностями:

$$y_{i+1} = y_i + q_k + \frac{1}{2}\Delta_{k-1} + \frac{5}{12}\Delta^2 q_{k-2} \text{ (k=2,3,…).}$$

Значения y_1 и y_2 находят другим способом. Погрешность на каждом шаге имеет порядок h^3;

Минобрнауки России

ФГБОУ ВО «Тульский государственный университет»

Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ

по междисциплинарному курсу

МДК 04.01 Внедрение и поддержка компьютерных систем

профессионального модуля

ПМ.04 СОПРОВОЖДЕНИЕ И ОБСЛУЖИВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

специальности СПО

09.02.07 Информационные системы и программирование

Тула 2021

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «14» октября 2021 г. № 3

Председатель цикловой комиссии И.В. Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

Практическая работа №1
«РАЗРАБОТКА СЦЕНАРИЯ ВНЕДРЕНИЯ ПРОГРАММНОГО ПРОДУКТА ДЛЯ РАБОЧЕГО МЕСТА»

Цель работы: получить теоретические знания и практические навыки по разработке сценария внедрения программного продукта для рабочего места.

Теоретические сведения

Бизнес-цель - это описание фактора, побуждающего к выполнению проекта. Ее формирование производится на стратегическом уровне, то есть бизнес-цель выступает в качестве связующего звена между глобальными задачами, стоящими перед организациями, и планируемым к реализации проектом. При отходе от стратегического видения происходит смещение бизнес-цели в сторону тактических и даже операционных задач, на уровне которых целью проекта видится "просто выдать продукт", а не достичь какой-либо тактической цели, поддерживающей стратегические цели организации. Этого нельзя допускать: бизнес-цель проекта должна всегда носить тактический или стратегический характер, но в то же время быть предельно точной и ясной (очень редко удается применить широко известный метод SMART к построению бизнес-цели проекта. Так, например, бизнес-целью проекта по приобретению и установке нового производственного оборудования является не покупка и установка оборудования, а устранение узкого места в производственном процессе и обеспечение надлежащих объемов выпуска, гарантирующих удовлетворение спроса и завоевание определенной доли рынка. Аналогично, проект внедрения информационной системы имеет своей бизнес-целью не разворачивание технических средств, а создание информационно-технологического фундамента для поддержки принятия руководством компании своевременных управленческих решений, направленных на обеспечение ее развития и роста.

Весь спектр работ согласно пожеланиям заказчика, начиная от инсталляции, адаптации и наладки программного обеспечения и до интеграции с устройствами и передачи в эксплуатацию, называется внедрением программного обеспечения в систему. Время и стоимость комплекса работ зависят от множества факторов и критериев выполнения, указанных заказчиком или необходимых для стабильности, таких как:

- готовность персонала компании к переходу на новое ПО или его освоению;
- наличие необходимых для выполнения аппаратных средств;
- особенностей выполнения работы;
- масштаба предполагаемых действий;
- состояния баз данных на текущий момент, наличия резервных копий на крайний случай;
- наличия и работоспособности каналов связи. Процесс поэтапного внедрения программного обеспечения.

Поскольку процедура внедрения программного обеспечения может вызвать перебои в работе компании, процесс разделяется на несколько этапов, каждый из которых имеет свои нюансы и осуществляется после строгого согласования с заказчиком.

Обследование, изучение компании (этап 1).

Перед созданием проекта выполняется исследование текущей работы компании профессионалами. По окончании предварительного обследования и аудита заказчик получает рекомендации, связанные с разработкой технического задания на производство работ. В нем уделяется внимание каждой мельчайшей детали, подробно описаны требования по:

- подготовке и требованиям к техническим средствам;
- формату хранения и передачи данных и резервных архивов;
- составу и выполнению подготовительных работ для объекта;

- конфигурированию системы передачи информации;
- работе общего и прикладного программного обеспечения.

Качественно составленное техническое задание (ТЗ) гарантирует точность выполнения работ.

Составление контракта на работы (этап 2)

Контракт на производство работ составляется по совместному заключению заказчика и компании после выполнения анализа ТЗ.

Этот период — оценочный. Поскольку план работ назначен и сроки определены, компания-исполнитель может оценить всю процедуру в комплексе и определиться с ценой. Чаще всего первичный этап производится бесплатно или становится таковым на основании последующего заказа. Цена на выполнение работ по интеграции программного обеспечения может зависеть от следующих факторов:

- состава и количества рабочих мест, подсистем и модулей;
- проведения дополнительных работ по интеграции с другими подсистемами и системами, а также сложности ее исполнения;
- объема хранимой в БД информации и ее состояния (работоспособности и наличие резервных копий).

Создание группы по внедрению программного обеспечения (этап 3).

Третий период также входит в подготовительные работы. Компанией-исполнителем формируется группа внедрения программного обеспечения и назначаются ответственные.

Инсталляция и наладка программного обеспечения (этап 4).

В этот период производится инсталляция программного обеспечения на серверах и клиентских машинах, подключение связи, а также проверка и наладка рабочего состояния системы и ее тестирование под нагрузкой. В стандартный перечень работ по четвертому этапу входит:

- установка и подготовка общесистемного ПО сервера;
- инсталляция и наладка компонентов и функций серверной платформы;
- создание таблиц баз данных, загрузка информации и интеграция;
- перенос БД (при необходимости), конвертация в нужный формат, наладка и создание рабочих копий ПО, подготовка программ;
- установка и подготовка клиентских машин (общеприкладное и прикладное ПО);
- интеграция и адаптация с уже имеющимися системами и платформами;
- проверка работоспособности всей системы, тестирование функционирования комплекса программного обеспечения;
- окончательная настройка по результатам тестирования с целью получения максимальной производительности и оптимизации работы.

На этом процесс внедрения программного обеспечения завершен, однако существуют дополнительные процедуры, которые множество компаний называет пост-установочными.

Завершение внедрения и проведение дополнительных работ (этап 5).

Завершение внедрения программного обеспечения включает выполнение следующих работ:

- обучение группы специалистов со стороны заказчика работе с новым ПО — может производится удаленно или на территории заказчика;
- внесение изменений согласно опыту эксплуатации заказчиком нового ПО;
- по окончании внесения условленных изменений и устранения замечаний подписывается акт сдачи работ и приемки проекта согласно ТЗ, после чего система передается заказчику и операция по внедрению считается завершенной.

После интеграции программного обеспечения со стороны заказчика могут возникнуть проблемы. Это может быть человеческий фактор или недостаточная оптимизация и интеграция с незаявленными в ТЗ системами, которые косвенно касаются

внедренного ПО. В связи с этим компании оказывают техническую поддержку как своих, так и интегрированных сторонними компаниями систем. Поддержка и сопровождение работы серверов не входит в оплату по основным работам, производимым по техническому заданию.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Сформулируйте цели и задачи внедрения вашего ПО (см. индивидуальные варианты в Приложении А).

Задание 3. Получите задание у преподавателя и выполните его.

Задание 4. Разделитесь на рабочие группы в соответствии с классификацией, обсудите и распределите обязанности конкретно для каждого участника.

Задание 5. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие этапы внедрения программного обеспечения Вам известны?
2. Охарактеризуйте этап «Обследование, изучение компании».
3. Охарактеризуйте этап «Составление контракта на работы».
4. Охарактеризуйте этап «Создание группы по внедрению программного обеспечения».
5. Охарактеризуйте этап «Инсталляция и наладка программного обеспечения».

Практическая работа №2
«РАЗРАБОТКА РУКОВОДСТВА ОПЕРАТОРА»

Цель работы: получить теоретические знания и практические навыки по разработке руководства оператора.

Теоретические сведения

Руководство оператора должно состоять из следующих частей:

- Титульной;
- Информационной;
- Основной.

Титульная часть оформляется согласно ГОСТ 19.104-78 ЕСПД. Основные надписи. Информационная часть должна состоять из аннотации и содержания. В аннотации приводят сведения о назначении документа и краткое изложение основной части. Содержание включает перечень записей о структурных элементах основной части документа.

Согласно «ГОСТ 19.505-79 ЕСПД. Руководствооператора. Требования к содержанию и оформлению»:

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Структура и оформление программного документа устанавливаются в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

1.2. Руководство оператора должно содержать следующие разделы:

- назначение программы;
- условия выполнения программы;
- выполнение программы;
- сообщения оператору.

В зависимости от особенностей документа допускается объединять отдельные разделы или вводить новые.

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе "Назначение программы" должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

2.2. В разделе "Условия выполнения программы" должны быть указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.).

2.3. В разделе "Выполнение программы" должны быть: указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, также ответы программы на эти команды.

2.5. В разделе "Сообщения оператору" должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

2.6. Допускается содержание разделов иллюстрировать поясняющими примерами, таблицами, схемами, графиками.

2.7. В приложения к руководству оператора допускается включать различные материалы, которые нецелесообразно включать в разделы руководства.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Составить руководство оператора в соответствии с «ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению».

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. На какой ГОСТ необходимо опираться при составлении данного документа?
2. Что представляет собой руководство оператора?
3. Какие основные разделы содержит «ГОСТ 19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению»?

Практическая работа №3
«РАЗРАБОТКА (ПОДГОТОВКА) ДОКУМЕНТАЦИИ И ОТЧЕТНЫХ ФОРМ ДЛЯ ВНЕДРЕНИЯ ПРОГРАММНЫХ СРЕДСТВ»

Цель работы: получить теоретические знания и практические навыки по разработке (подготовке) документации и отчетных форм для внедрения программных средств.

Теоретические сведения

Программная документация, включает:

1) техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
2) текст программы (запись программы с необходимыми комментариями);
3) описание программы (сведения о логической структуре и функционировании программы);
4) пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
5) эксплуатационные документы.

К эксплуатационным документам относят:

1) описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
2) руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
3) руководство программиста (сведения для эксплуатации программы);
4) руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
5) описание языка (описание синтаксиса и семантики языка);
6) руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов.

Эксплуатационный документ "Описание языка" включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.

Эксплуатационный документ "Руководство по техническому обслуживанию" включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (возможности, основные характеристики, ограничения области применения);
- условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);
- описание задачи (указываются определения задачи и методы её решения);
- входные и выходные данные.

Руководство программиста

Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:

- назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);
- обращение к программе (способы передачи управления и параметров данных);
- входные и выходные данные (формат и кодирование);
- сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);
- условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);
- выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);
- сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Составить сводную ведомость документов и отчетных форм для внедрения программных средств (программного средства по индивидуальному заданию из Приложения А).

Задание 3. Проанализировать наличие и качество документации, сделать необходимые отметки.

Задание 4. Составить перечень недостающих документов.

Задание 5. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что отражается в эксплуатационном документе "Описание языка"?
2. Что отражается в эксплуатационном документе "Руководство по техническому обслуживанию"?
3. Что отражается в документе "Описание применения"?
4. Что отражается в документе "Руководство программиста"?

5. Что отражается в документе "Руководство оператора"?
6. Какие документы сопровождают процедуру внедрения программных средств?

Практическая работа №4
«ИЗМЕРЕНИЕ И АНАЛИЗ ЭКСПЛУАТАЦИОННЫХ ХАРАКТЕРИСТИК КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Цель работы: получить теоретические знания и практические навыки по измерению и анализу эксплуатационных характеристик качества программного обеспечения.

Теоретические сведения

Качество программного обеспечения – это совокупность свойств, характеризующих способность программного обеспечения удовлетворять потребностям пользователя в соответствии с предназначением.

Процесс оценки качества программного обеспечения осуществляется для каждой фазы его жизненного цикла и включает:

- выбор совокупности (номенклатуры) показателей качества оцениваемого программного средства;
- определение значений этих показателей;
- сравнение полученных значений с базовыми значениями показателей.

Рисунок 1.

Управление качеством – это система организационных, экономических, технологических и правовых мероприятий, осуществляемых для удовлетворения требований к качеству программного обеспечения в течение жизненного цикла.

Свойства программы – это особенности, объективно присущие программе, которые проявляются в ее жизненном цикле (разработке, применении, сопровождении).

Характеристика программы – это понятие, отражающее проявление отдельного измеримого фактора присущего программе свойства. Иначе говоря, характеристика – это проявляемый и измеримый атрибут свойства.

Измерение (оценка) одной или нескольких характеристик программы дает представление о том, насколько программе присуще то или иное свойство.

Каждому свойству соответствует одна или несколько характеристик программного обеспечения.

Для решения задачи количественной оценки характеристик программного обеспечения необходимо наличие системы измерений и методов оценки.

Система измерений характеристик программного обеспечения – это совокупность измеряемых характеристик, единиц измерения, измерительных шкал и связей, установленных между ними. Если между измеряемыми характеристиками установлены иерархические связи, систему измерений называют иерархической, в противном случае – одноранговой.

Измерительная шкала устанавливает границы (диапазон) и точность измерений характеристик свойств в установленных единицах.

Результаты измерений в избранной измерительной шкале позволяют обнаружить сходство и различие в свойствах программного обеспечения с целью последующей оценки и классификации. Применительно к ПО используют главным образом следующие известные виды измерительных шкал: номинальные (категорийные), порядковые, интервальные.

Номинальная (категорийная) шкала фиксирует наличие или отсутствие некоторой характеристики свойства без учета градаций и позволяет классифицировать программы по этому принципу.

Порядковая шкала фиксирует отношение порядка и позволяет ранжировать программы относительно некоторого опорного значения характеристик свойств.

Интервальная шкала фиксирует не только отношение порядка, но и величину, отличающую одно значение характеристики от другого (интервал между значениями).

Методы оценки характеристик программного обеспечения делят на следующие шесть групп:

1) Измерительные.
2) Регистрационные.
3) Органолептические.
4) Расчетные.
5) Экспертные.
6) Социологические.
7) Традиционные.

Измерительные методы основаны на получении информации о характеристиках программного обеспечения с использованием специальных инструментальных средств (технических или программных средств, обеспечивающих проведение измерений и их автоматизацию).

Регистрационные методы основаны на получении информации о характеристиках программного обеспечения во время испытаний или функционирования путем регистрации и подсчета определенных событий (например, моментов и количества ошибок, времени начала и окончания расчетов и т.д.), регистрируемых извне программы с помощью средств измерений общего назначения.

Органолептические методы основаны на получении информации о характеристиках программного обеспечения путем их восприятия органами чувств – в первую очередь зрения, слуха и осязания.

Расчетные методы основаны на получении информации о характеристиках программного обеспечения за счет использования теоретических или эмпирических зависимостей.

Экспертные методы используют опыт экспертов-специалистов, компетентных в оценке характеристик программного обеспечения.

Социологические методы используют обработку специальных анкет-опросников, содержащих качественные оценки характеристик программного обеспечения социальными группами, имеющими отношение к применению программного обеспечения.

Традиционные методы объединяют группу сформировавшихся и традиционно используемых в организациях, на предприятиях и иных учреждениях методов количественной оценки характеристик программного обеспечения.

Под *жизненным циклом программного обеспечения* понимается период времени с момента начала предпроектного обследования до момента полного выхода программы из употребления пользователями.

Весь период жизненного цикла программного обеспечения делится на следующие временные промежутки или фазы.

1. Анализ — этап определения требований к программному обеспечению, спецификация требований и формирования технического задания на проектирование программы.

2. Проектирование — этап разработки технического проекта.
3. Реализация — этап разработки программного обеспечения, средств тестирования и документации.
4. Тестирование — этап испытания программного обеспечения и устранение недостатков.
5. Изготовление — этап преобразования программного обеспечения в форму, готовую для поставки; завершение формирования документации.
6. Внедрение — этап подтверждения стабильной работы программного обеспечения; ввод в стадию активного применения.
7. Эксплуатация — этап применения программного обеспечения по назначению.
8. Сопровождение — этап устранения дефектов в процессе эксплуатации; усовершенствование, оптимизация и модификация используемого программного обеспечения при условии сохранности целостности программного продукта.

Оценка качества программного обеспечения на всех фазах жизненного цикла осуществляется на основе чётырехуровневой системы показателей.

Показатели первого уровня (факторы качества) характеризуют потребительски-ориентированные свойства программных средств, которые соответствуют потребностям пользователей. Факторы качества, собственно, определяют наиболее значимые (с точки зрения использования) свойства программ. Для оценки качества программного обеспечения используют следующие факторы:

• надежность;
• сопровождаемость;
• удобство применения;
• эффективность;
• универсальность;
• корректность.

Каждый фактор представляет собой интегральную оценку, которой соответствует несколько критериев качества (комплексных показателей второго уровня).

Оценочные элементы фактора «Надежность ПС»:

- Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных.
- Возможность обработки ошибочных ситуаций.
- Полнота обработки ошибочных данных.
- Наличие тестов для проверки допустимых значений входных данных.
- Наличие системы контроля полноты входных данных.
- Наличие средств контроля корректности входных данных.
- Наличие требований к программе по восстановлению процесса выполнения в случае сбоя ОС, внешних устройств, процессора.
- Наличие требований к программе по восстановлению результатов при отказах ОС, внешних устройств, процессора.
- Наличие средств восстановления при сбоях оборудования.
- Наличие возможности повторного старта с точки прерывания.
- Наличие обработки неопределенностей.
- Наличие централизованного управления процессами, конкурирующими из-за ресурсов.
- Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления.

Оценочные элементы фактора «Сопровождаемость»:

- Наличие комментариев в точках входа и выхода в программу осуществляется ли передача результатов работы модуля через вызывающий его модуль.

- Оценка программы по числу циклов.
- Используется ли язык высокого уровня.
- Наличие проверки корректности передаваемых данных.
- Использование при построении программ метода структурного программирования Соблюдение принципа процесса разработки программы сверху вниз.
- Наличие ограничений на размеры модуля.
- Наличие модульной схемы программы.

Оценочные элементы фактора «Корректность»:

- Наличие всех необходимых документов для понимания и использования ПС.
- Наличие описание схемы иерархии модулей программы.
- Наличие описаний основных функций.
- Наличие описаний частных функций.
- Наличие описания данных.
- Наличие описания алгоритмов.
- Наличие описания интерфейсов между модулями.
- Наличие описания всех параметров.
- Наличие описание методов настройки системы.
- Наличие описание способов проверки работоспособности программы.
- Реанимация всех модулей системы.
- Реанимация всех основных функций.
- Реанимация всех алгоритмов.
- Наличие определений всех данных: переменные, индексы, массивы и пр.
- Наличие интерфейсов с пользователем.
- Отсутствие противоречий в выполнении основных функций.
- Отсутствие противоречий в выполнении частных функций.
- Отсутствие противоречий в выполнении алгоритмов.
- Правильность взаимосвязей.
- Правильность реализаций интерфейса с пользователем.
- Отсутствие противоречий в настройке системы.
- Комплектность документации в соответствии со стандартами.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. В соответствие с приведенными выше критериями оцените характеристики качества программного продукта (созданного Вами ранее).

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как оценивается эффективность программного продукта?
2. Какие существуют критерии для оценки качества программного продукта?
3. Для чего предназначены программные продукты?
4. Какие варианты легального распространения программных продуктов существуют?
5. Чем определяется надежность программного продукта?

6. Что обозначает модифицируемость программного продукта?

Практическая работа №5
«ВЫЯВЛЕНИЕ И ДОКУМЕНТИРОВАНИЕ ПРОБЛЕМ УСТАНОВКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Цель работы: получить теоретические знания и практические навыки по выявлению возможных проблемам установки программного обеспечения, решению проблем установки ПО и написания Руководства по инсталляции программного средства.

Теоретические сведения

Отсутствие NET Framework необходимой версии

Самая частая причина, почему не устанавливаются программы, это отсутствие системной библиотеки NET Framework необходимой версии. В этой библиотеке содержатся ресурсы, которые нужны для нормальной работы той или иной программы. Поэтому если её нет, то и программа установиться не сможет.

Рекомендуется установить на свой компьютер все версии NET Framework, начиная с 2.0 и заканчивая самой старшей, которую поддерживает ваша Windows. Это обеспечит нормальную работу всех совместимых с вашей ОС программ.

Чтобы узнать, какая версия NET Framework установлена на вашем компьютере, нажмите на кнопку «Пуск», выберите «Панель управления», затем откройте раздел «Программы», а потом – «Программы и компоненты» (пример для Windows 7).

Как правило, если программы не устанавливаются по причине отсутствия нужной NET Framework, появляется сообщение с указанием версии, которую нужно докачать.

Отсутствие Visual C++ и Direct X необходимой версии

Следующей частой причиной, почему не устанавливаются программы, является отсутствие компонентов Visual C++ и Direct X. Visual C++ необходима для работы программ, которые написаны на популярном языке C++, а Direct X требуется для большинства игр. Как и в случае с NET Framework, есть разные версии этих компонентов.

Узнать версию Visual C++ можно так же, как и NET Framework – в разделе «Программы и компоненты».

А для того, чтобы узнать версию Direct X, нажмите «Пуск» и откройте «Выполнить» (или ctrl+R). Запишите команду dxdiag и нажмите «Ок».

Как правило, если программа не устанавливается из-за отсутствия этих компонентов или их правильной версии, появляется соответствующее сообщение.

Неправильная разрядность Windows

Windows могут быть 32-х и 64-х разрядными. На 64-х разрядные операционные системы можно установить 32-х и 64-х разрядные программы. А на 32-х разрядных системах будет работать только 32-х разрядная программа. Поэтому вы не сможете установить на неё 64-х разрядную программу.

Чтобы определить разрядность системы, нажмите правой кнопкой на «Компьютер» и выберите «Свойства».

Разрядность программы можно узнать в её описании. Если она не подходит, то в процессе установки появится соответствующая ошибка.

Повреждённый файл инсталляции

Если при установке программы появляется ошибка о том, что файл установки повреждён, то следует получить этот файл заново. Такое может случиться из-за того, что файл программы не до конца докачался, или был выложен на сайте уже повреждённым. Чтобы решить проблему, нужно попытаться скачать файл установки программы из другого места.

Отсутствие необходимой DLL-библиотеки

Редкой причиной, почему не устанавливаются программы, является отсутствие каких-то библиотек в системе, которые должны были бы быть в ней по умолчанию. Такое случается, если используется неофициальная сборка Windows или эти библиотеки были удалены преднамеренно.

Проблема проявляется в сообщении, которое ругается на какой-либо файл DLL.

Чтобы решить её, необходимо докачать нужный DLL файл и поместить его в нужный каталог (в system32 или SysWOW64).

После этого следует зарегистрировать библиотеку. Для этого нужно нажать «Пуск» и выбрать «Выполнить» (или клавиши ctrl+R). Затем записать cmd и нажать «Ок». В командной строке библиотеки регистрируются командой regsvr32 file.dll , где file.dll – это наименование файла библиотеки.

Кривые сборки программ

Нередкой причиной, почему не устанавливаются программы, являются кривые руки у пиратов, которые их взламывают и отдают вам в бесплатное пользование, либо как-то видоизменяют, то есть делают собственную сборку (репак). Чтобы избежать этого, ознакомьтесь с комментариями тех, кто уже попробовал то, что вы устанавливаете.

Сложно сказать, какие могут быть ошибки при установке таких программ. Может быть всё, что угодно.

Без прав администратора

Если вы работаете в офисе, то вы можете столкнуться с тем, что у вашего пользователя нет прав администратора. Ваш системный администратор может преднамеренно запретить установку программ. Чтобы решить это, обратитесь к нему за помощью.

Это может случиться не только в офисе, но и дома, если ваш пользователь на ПК не имеет прав администратора.

Блокировка программами безопасности

На всех компьютерах есть антивирусы и некоторый софт может блокироваться ими. Это тоже частая причина того, почему программы не устанавливаются. Решить её просто – отключите ваш антивирус или другую программу безопасности и установите программу. При необходимости внесите этот софт в доверительный список в антивирусе или в программе безопасности.

Будьте бдительны. Если вы не уверены в благонадёжности источника программы, то лучше не устанавливайте её, если антивирус ругается.

Конфликты с не полностью удалёнными старыми версиями программ

Если вы обновляете версию программы, и сначала удалили старую, а потом пытаетесь установить новую, то может случиться конфликт, если компоненты старой версии не были удалены полностью. Это часто бывает при использовании стандартного удаляльщика Windows.

Чтобы исправить эту проблему, необходимо удалить все компоненты предыдущей версии вручную. Также можно воспользоваться программами-чистильщикам, например, CCLeaner.

Недостатки оборудования

Сегодня появились такие программы и игры, которые могут установиться уже не на все компьютеры. Так они могут предъявлять особые требования к видеокартам, оперативной памяти, процессору.

Решить такую проблему докачиванием каких-то файлов не получится. Здесь нужно апгрейдить свой компьютер и докупать железки.

Программная несовместимость

Главной проблемой в использовании программ является их несовместимость с операционной системой. Дело в том, что разработчики ориентируют приложение на работу в одной или нескольких операционных системах, использующихся на момент его написания. Никто не может гарантировать, что эта программа будет нормально работать в новой операционной системе. В принципе, ничто не мешает скачать более свежую версию и использовать ее при работе в Windows Vista, однако это не всегда возможно. Во-первых, более новой версии может и не существовать. А во-вторых, часто бывает так, что использоваться должна именно данная версия, а не какая-нибудь другая. Элементарный пример – программа учета, написанная с применением устаревших технологий и систем доступа к базам данных. В этом случае очень часто работоспособность программы гарантируется только в определенных операционных системах, например в MS-DOS или Windows 95/98.

Для настройки совместимости программ с операционной системой предназначен специальный режим. От того, насколько качественно разработана эта функция, зависит корректность работы программы.

Используя возможности операционной системы Windows Vista, можно в любой момент настроить режим совместимости определенной программы, если вы не уверены в том, что она будет работать корректно, или уверены в этом наверняка.

Для этого выполните следующие действия. Предположим, ярлык программы выведен на Рабочий стол. В таком случае щелкните на нем правой кнопкой мыши и в появившемся контекстном меню выберите пункт Свойства (рис. 1).

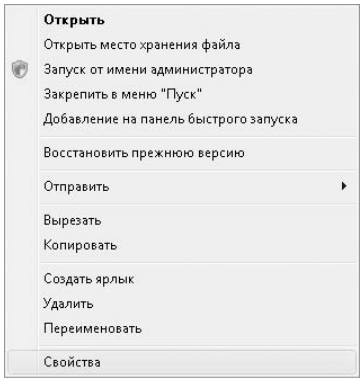


Рисунок 1. Выбираем пункт Свойства

В результате на экране появится окно свойств ярлыка программы (рис. 2), содержащее несколько вкладок с параметрами.

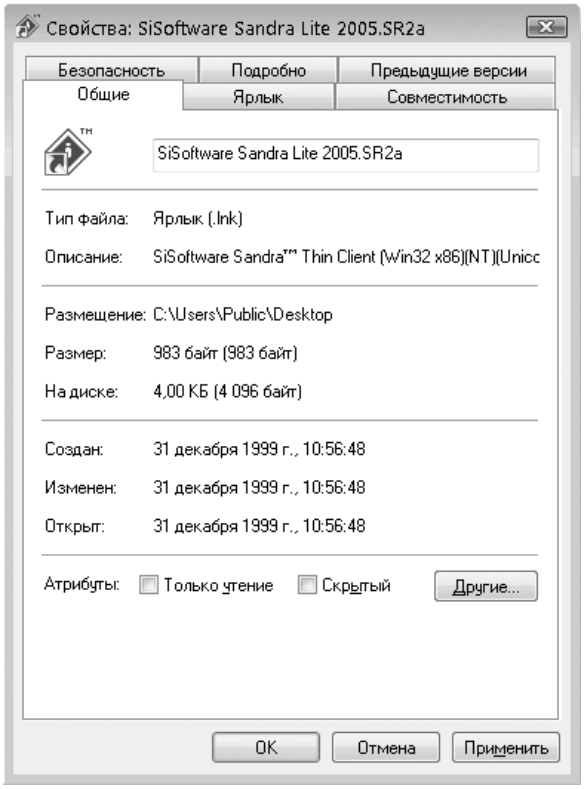


Рисунок 2. Окно свойств ярлыка программы

Перейдите в данном окне на вкладку Совместимость (рис. 3). В области Режим совместимости установите флажок Запустить программу в режиме совместимости с. Из раскрывающегося списка выберите операционную систему, в которой данная программа работает без сбоев. Нажав кнопку ОК, закройте окно и запустите приложение.

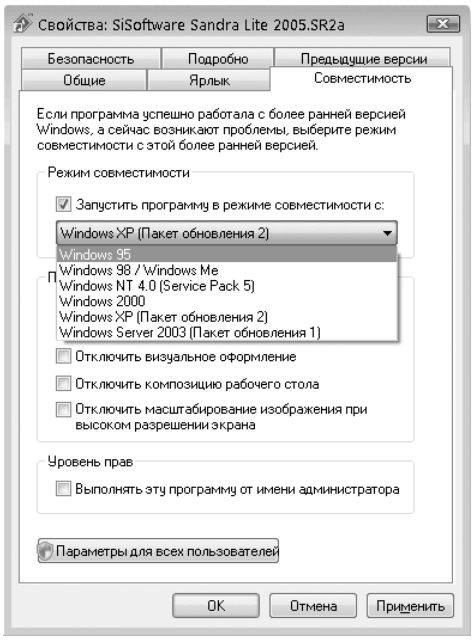


Рисунок 3. Устанавливаем флажок Запустить программу в режиме совместимости

Если программа все равно отказывается запускаться, можно попробовать установить флажок Выполнять эту программу от имени администратора, поскольку может потребоваться доступ к некоторым ресурсам, недоступным обычным пользователям. Если и это не дало результата, придется либо отказаться от использования данного приложения, либо найти его более новую версию.

Отказ программы устанавливаться на компьютер

Операционная система Windows обладает очень мощными механизмами защиты, которые позволяют обезопасить ее работу от вмешательства сторонних программ. Как следствие, система может заблокировать установку приложения. В этом случае, когда пользователь пытается установить «критичную» программу, система выдает соответствующее сообщение (рис. 4).

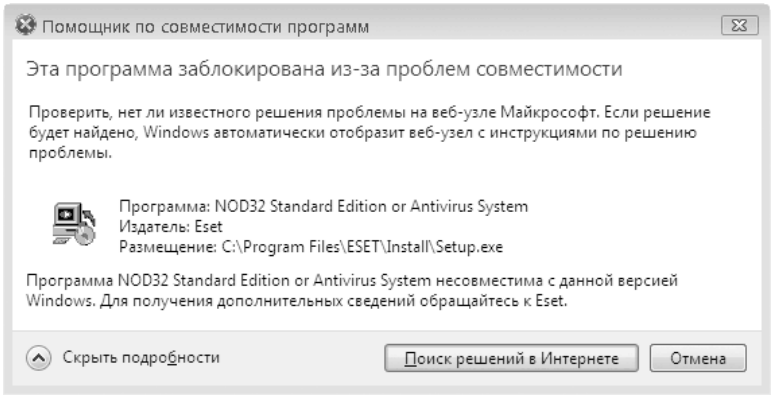


Рисунок 4. Установка программы заблокирована

Единственное, что остается сделать, – отказаться от установки данной версии программы и поискать ее более новый вариант.

Проблемы с Internet Explorer

Как ни стараются программисты Microsoft избавить «родной» браузер от множества ошибок и программных «дыр», сделать это не получается. Возможно, виноват «движок» браузера, возможно, сами разработчики, однако факт налицо: даже последняя, седьмая, версия браузера имеет недоработки, которые не позволяют использовать эту программу в полной мере.

Перечислю основные недостатки Internet Explorer.

• **Использование значительного количества памяти.** Практика показывает, что Internet Explorer не стесняет себя в использовании оперативной памяти. Мало того, чем дольше вы работаете в окне браузера, тем больше памяти он отнимает. Параллельно с этим увеличивается и файл подкачки операционной системы, что однозначно сказывается на скорости ее работы.

• **Зависание.** Зависание браузера при загрузке очередного веб-ресурса – история, которая тянется с давних времен. Разработчики либо не обращают на это внимания, либо не знают, как избавиться от подобного деффекта. Самое неприятное в этой ситуации то, что при принудительном завершении работы программы закрываются все окна и вкладки, которые были открыты из ее окна.

• **Неустранимые ошибки.** При работе браузера может появиться сообщение о неустранимой ошибке (ошибку выдает отладчик, используемый по умолчанию). При закрытии такого сообщения все окна, открытые из браузера, также закрываются.

• **Некорректная работа с всплывающими окнами.** Механизм отслеживания и блокирования всплывающих окон явно не доработан до того уровня, как это сделано в других браузерах, например Mozilla Firefox. Часто случается так, что механизм не срабатывает и пропускает подобного рода окна.

Этот список можно продолжать, однако остальные ошибки некритичны и, в принципе, не мешают нормальной работе. Примером такой некритичной ошибки может быть потеря фокуса у браузера, то есть если сначала вы можете открыть окно браузера и сразу же набирать адрес страницы, то по прошествии некоторого времени вы почему-то не можете этого сделать.

Зависание и некорректная работа программ

Как уже упоминалось, работа программы, будь то серьезный графический пакет или простенькое приложение, зависит от множества факторов. В первую очередь – от наличия достаточного количества ресурсов, а также возможности доступа к нужной информации и аппаратной части компьютера. Если хотя бы одно из этих условий не выполняется, то работа программы может сопровождаться разными неожиданностями (в частности, зависанием, внезапным завершением и т. д.).

Для примера рассмотрим популярную программу Adobe Acrobat Reader.

Предположим, ссылка на просматриваемой веб-странице относится к файлу с расширением PDF, за работу с которым отвечает программа Acrobat Reader (если, конечно, никакая другая подобная программа на вашем компьютере не установлена). Щелкнув на этой ссылке, вы просматриваете документ в окне браузера. Если браузер вам больше не нужен, вы закрываете его и занимаетесь чем-то другим. Как оказывается, модуль программы Acrobat Reader, который использовался для просмотра содержимого документа в браузере, остается «висеть» в оперативной памяти, хотя надобности в нем уже нет, и занимает определенный объем оперативной памяти, а кроме этого, еще и использует другие системные ресурсы (дескриптор работы с процессом, файл подкачки и т. д.).

Убедиться в правдивости подобной ситуации достаточно просто, открыв Диспетчер задач. Для этого щелкните правой кнопкой мыши на свободном участке Панели задач и в появившемся контекстном меню выберите пункт Диспетчер задач (для этого можно также нажать сочетание клавиш Ctrl+Alt+Delete). В открывшемся окне перейдите на вкладку Приложения и убедитесь в том, что программы Acrobat Reader нет в списке работающих приложений (рис. 5).

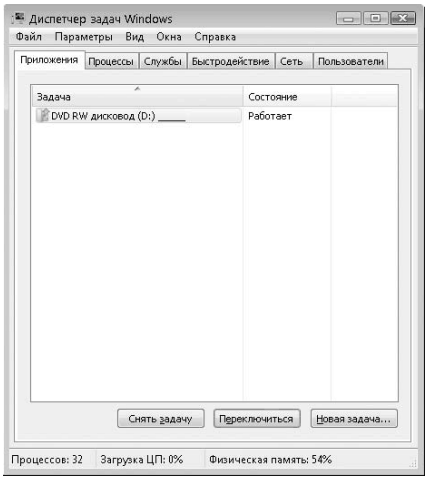


Рисунок 5. Список работающих приложений

На вкладке Процессы данного окна (рис. 6) вы увидите файл программы AcroRd32.exe, что говорит о том, что приложение в данный момент находится в оперативной памяти.

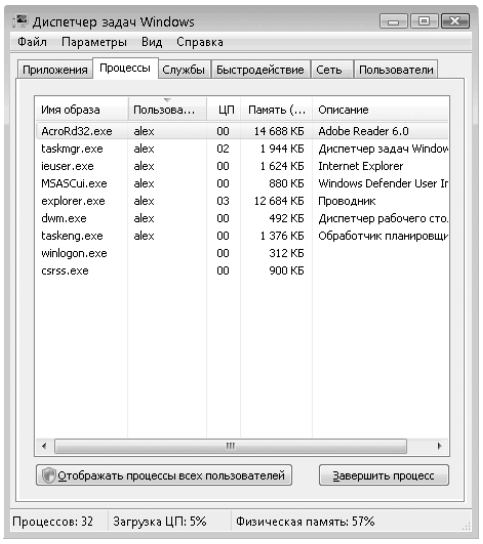


Рисунок 6. Список выполняющихся процессов

Единственное, что можно сделать при возникновении этой проблемы, – вручную остановить выполнение процесса. Для этого щелкните правой кнопкой мыши на названии процесса и в появившемся меню выберите пункт Завершить процесс или Завершить дерево процессов, что более предпочтительно в данной ситуации (рис. 7).

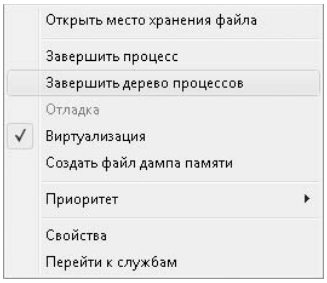


Рисунок 7. Завершаем выполнение процесса.

В результате вы освободите не только некоторый объем оперативной памяти и файла подкачки, но и занятые программой файлы и устройства, которые до этого не могли быть использованы другими процессами.

Кстати, аналогичным образом можно поступить и с другими подозрительными процессами, отнимающими ресурсы у системы. Главное при этом – не переусердствовать.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Из методических указаний выписать в отчет все основные причины проблем с установкой программного обеспечения и возможные способы их решения.

Задание 3. С помощью ресурсов Интернет найдите основные пункты, которые должны содержаться в Руководстве по инсталляции программного средства, выпишите их в отчет.

Задание 4. Скачайте пробную версию CorelDraw (или какое-либо другое ПО). Установите программу. После установки программы напишите в отчет Руководство по инсталляции программного средства. Задание 5. С помощью ресурсов Интернет найдите и перечислите в отчетет программы, не совместимые с ОС Windows 10.

Задание 6. Дооформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. С какими проблемами с установкой программы из вышеперечисленных вы ранее сталкивались?
2. Были ли у вас какие-то проблемы с установкой программы, которые не указаны в методических указаниях? Какие именно?
3. Что должно содержаться в описании программы, для того, чтобы предостеречь пользователя от проблем с совместимостью программ?

Практическая работа №6
«УСТРАНЕНИЕ ПРОБЛЕМ СОВМЕСТИМОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Цель работы: получить теоретические знания и практические навыки по устранению проблем совместимости программного обеспечения.

Теоретические сведения

Windows 10 уже больше двух лет, и все основные программы уже давно оптимизированы для работы в ней. Тем не менее существует ПО, вроде специализированных утилит или старых игр, которое не обновляется уже годами и не способно корректно работать в новых версиях Windows. К счастью, в Windows 10 существует возможность запуска любой программы в специальном режиме совместимости с предыдущими версиями системы, что помогает справиться с проблемами устаревшего ПО.

Способ 1: Запуск средства исправления неполадок совместимости

Как включить режим совместимости через свойства программы

Самый простой способ активации режима совместимости - его настройка в свойствах программы (а именно исполняемого файла, вроде .**exe**) или её ярлыка.

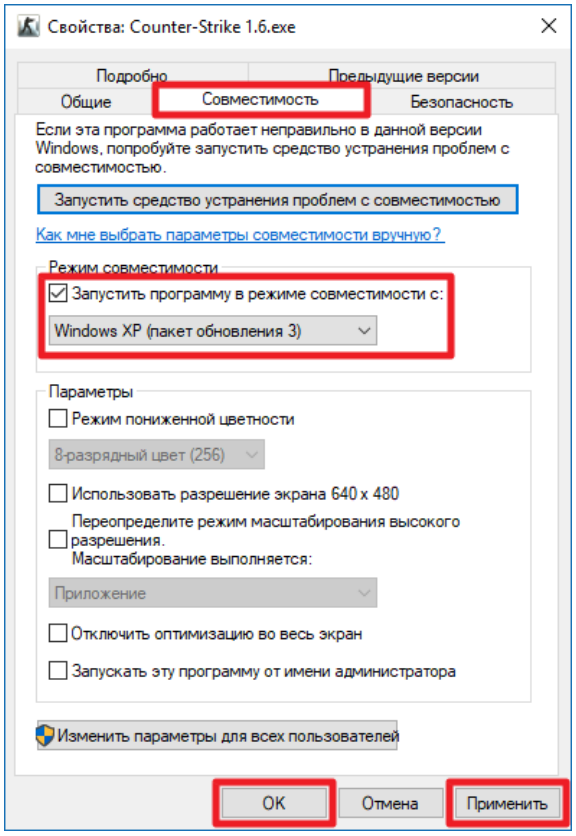
Где найти ярлык программы в Windows

Все ярлыки, которые различное ПО помещает в меню Пуск, можно найти в папке **C:\ProgramData\Microsoft\Windows\Start Menu\Programs**. Исполняемые файлы программ, а также ярлыки, которых нет в меню Пуск, чаще всего находятся в папке приложения. Её можно отыскать в следующих директориях:

- Самый распространённый вариант: **C:\Program Files** или **C:\Program Files (x86)**.
- Более редкий вариант: **C:\Users\имя_пользователя\AppData\Roaming**.

Как включить режим совместимости

1. Кликните по исполняемому файлу или ярлыку правой клавишей мыши и выберите пункт **Свойства**.
2. Перейдите во вкладку **Совместимость**. Поставьте галочку **Запустить программу в режиме совместимости с**.
3. В выпадающем списке выберите операционную систему, с которой данное приложение точно заработает.
4. Нажмите **Применить**, затем **ОК**.



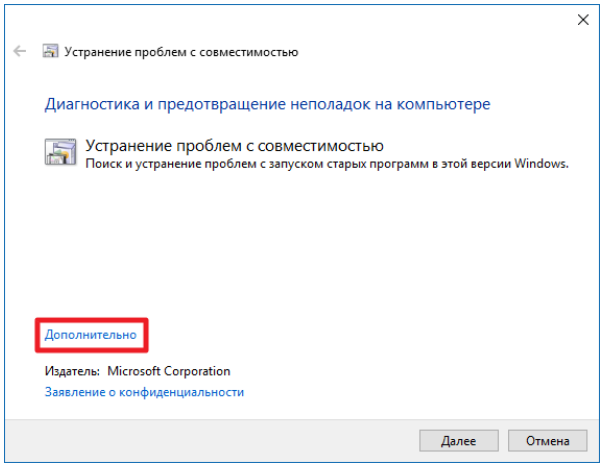
5. Запустите программу, чтобы проверить, будет ли она теперь работать как надо. Если проделанные операции не помогли, попробуйте указать более старую версию Windows в настройках режима совместимости.

Включение режима совместимости через встроенную утилиту диагностики

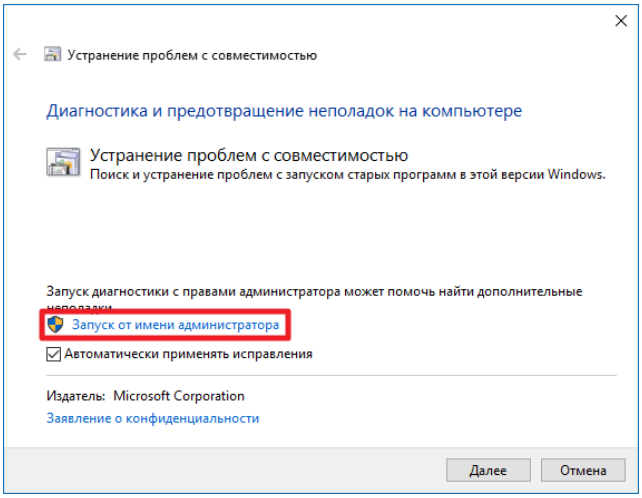
Немного более удобные настройки режима совместимости предоставляет встроенная в Windows утилита **Устранение проблем с совместимостью**.

1. Нажмите **Win + S**. Введите **Выполнение программ, созданных для предыдущих версий Windows**.
2. Запустите найденную утилиту.

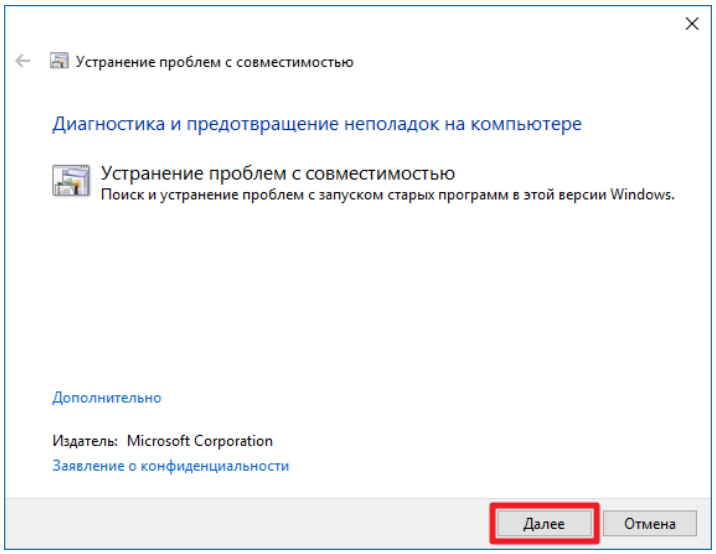
Нажмите на надпись **Дополнительно**.



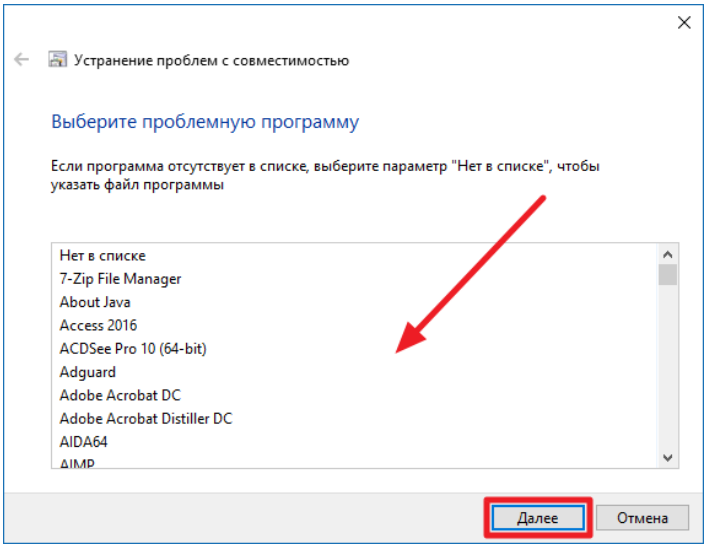
3. Выберите пункт **Запуск от имени администратора**.



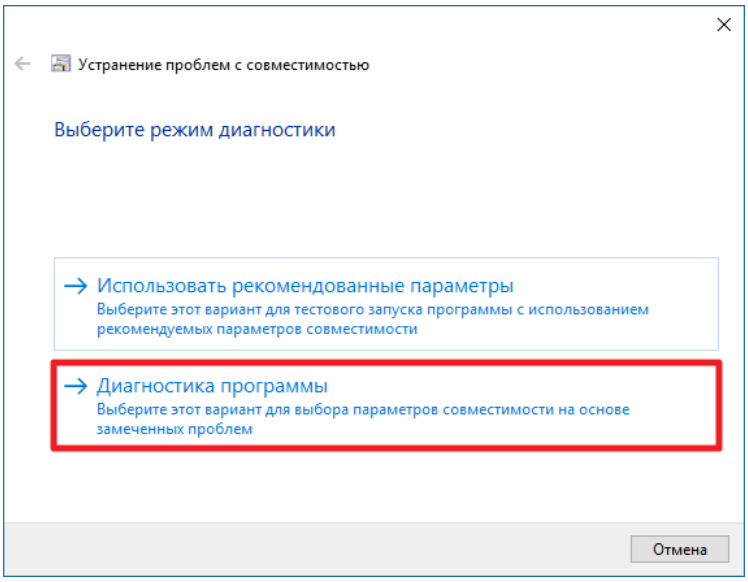
4. Кликните на кнопку **Далее**.



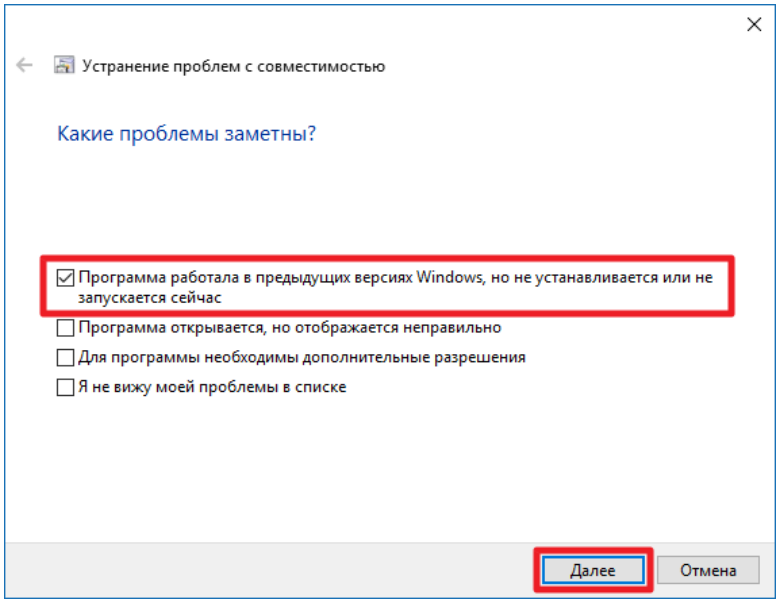
5. Найдите необходимую программу. Если её нет в списке (например, это портативное приложение), то выберите пункт **Нет в списке** и укажите путь к её исполняемому файлу.
6. Нажмите **Далее**.



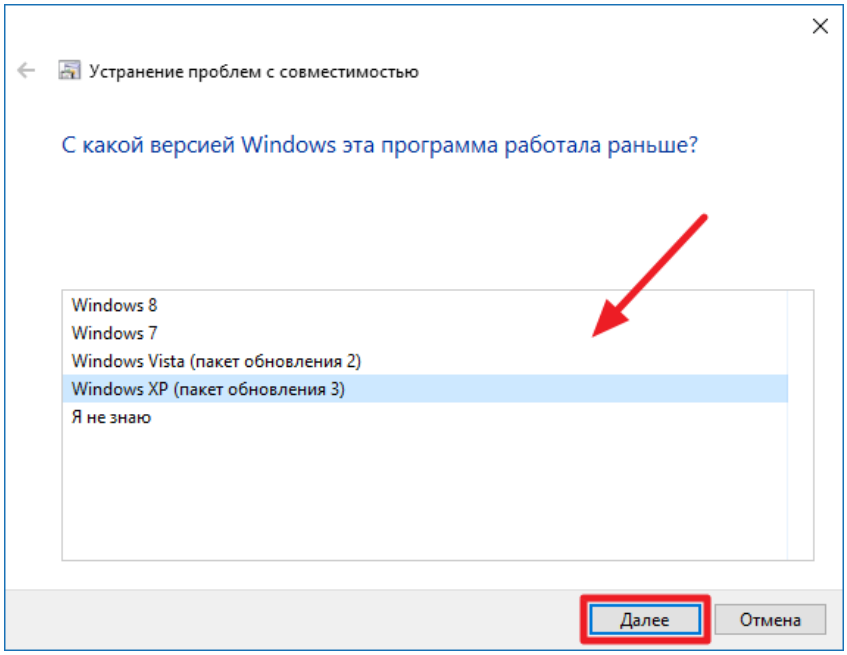
7. Выберите ручной или автоматический способ настройки параметров совместимости. Довольно часто автоматический режим не помогает исправить неполадки, так что мы рекомендуем использовать ручной. Для его запуска следует кликнуть **Диагностика программы**.



8. Отметьте галочкой пункт **Программа работала в предыдущих версиях Windows, но не устанавливается или не запускается сейчас** и нажмите **Далее**.

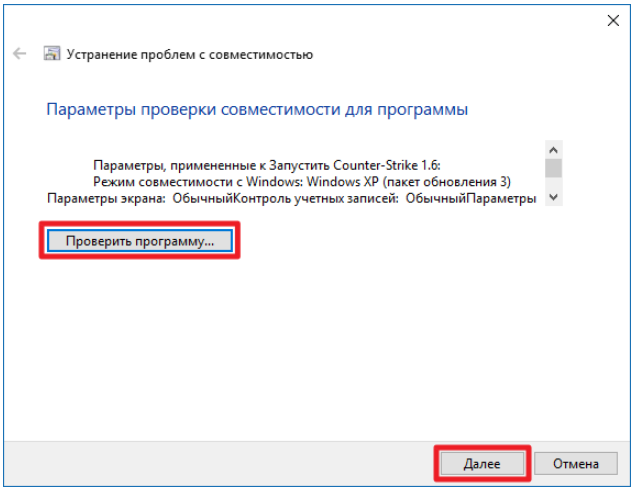


9. Укажите версию системы, в которой программа ранее работала корректно, и нажмите **Далее**.

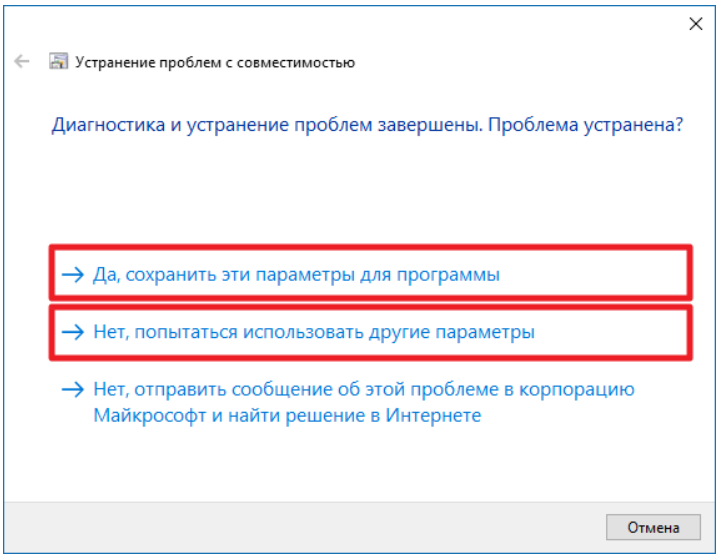


10. Кликните на кнопку **Проверить программу**, чтобы выполнить тестовый запуск.

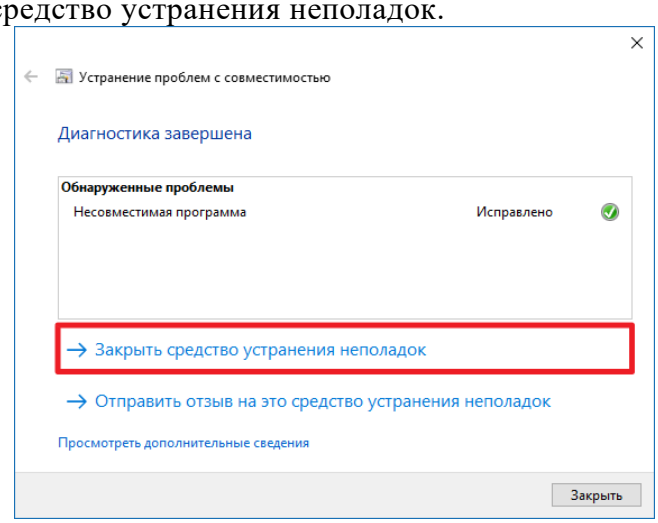
11. Нажмите **Далее**.



12. Если всё в порядке, тогда выберите пункт **Да, сохранить эти параметры для программы**. В ином случае воспользуйтесь кнопкой **Нет, попытаться использовать другие параметры** и настройте режим совместимости по-другому.



13. Закройте средство устранения неполадок.



В большинстве случаев эта инструкция поможет вам работать со старыми программами в актуальной операционной системе от Microsoft.

Способ 2: Ручные настройки совместимости

Аналогично предыдущему пункту настройки совместимости несложно выставить и самостоятельно, отредактировав свойства ярлыка/установщика. Однако по своей функциональности этот метод практически ничем не отличается от предыдущего за исключением некоторых второстепенных параметров. В связи с этим при безрезультатности средства исправления неполадок совместимости этот вариант также, вероятно, не принесет должного результата.

Способ 3: Отключение помощника по совместимости

Обратной предыдущим двум рекомендациям выступает эта. Дело в том, что иногда этот компонент намерено блокирует запуск программы, считая ее несовместимой с операционной системой, хотя по факту она вполне может работать на ней. Мы покажем, как производить отключение через «Редактор локальной групповой политики», а тем пользователям, у которых он отсутствует, подойдет альтернатива в виде «Редактора реестра».

Способ 4: Отключение UAC

UAC представляет собой встроенное в Windows приложение, обеспечивающее контроль учетных записей. По сути это некое обеспечение безопасности операционной системы, но иногда оно вызывает сбои в установке или запуске различных приложений. Поэтому имеет смысл на время отключить эту функцию.

Способ 5: Проверка пути установки

Некоторые старые программы (и не только старые) могут быть установлены в директории, в пути которых имеются русские символы. Из-за этого возникают неустранимые ошибки, и приложение не удается запустить.

Способ 6: Переустановка/обновление драйверов

Установленные программы, которые не удается запустить из-за рассматриваемой ошибки, иногда отказываются работать из-за драйверов. Если те слишком старые (реже, наоборот, очень новые для приложения), появляется та самая несовместимость. Нельзя однозначно ответить, какие именно драйверы необходимо переустановить, поскольку все зависит от типа программы, которую необходимо запустить. Здесь вы уже сами должны понять, на чем стоит сделать акцент. Например, если это игра или какое-то приложение, дающее нагрузку на видеокарту, значит стоит переустановить драйвер для нее. Мы лишь можем предоставить материалы, которые помогут вам разобраться с тем, как переустанавливать или обновлять драйверы.

Способ 7: Установка виртуальной машины

Когда никакие методы не помогают установить запустить программу или не дают возможности ее установить, вероятно, она попросту не может работать в Windows 10. Когда возникает острая необходимость ее запустить и никакие другие аналоги не подходят, единственной рекомендацией остается установка виртуальной машины с системой, на которой она будет корректно работать.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Составьте таблицу, содержащую минимальные системные требования для программ, необходимые для тестирования на совместимость. Дополните таблицу другими программными продуктами.

Программа	Процессор	Видеокарта	Оперативная память	ОС + разрядность	Объем памяти на жестком диске	Дополнительно
PhotoShop cs5						
Windows 10						
Microsoft Office 2019						
Компас-3D V13						
Visual Studio 2019						
AutoCAD						

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятия «операционная система».
   2. Какую функцию должна обеспечивать операционная система применительно к другому «стороннему» программному обеспечению?
   3. Дайте определение понятия «системные требования» к установке операционной системы (минимальные, рекомендуемые).
3. В чем состоит отличие 32-, 64-разрядных операционных систем друг от друга? Можно ли установить 32-разрядное приложение в среде 64-разрядной операционной системы? Что для этого нужно? Каковы недостатки данного варианта работы прикладного приложения?
4. В чем состоит отличие однопользовательской операционной системы от многопользовательской системы?

Практическая работа №7
«КОНФИГУРИРОВАНИЕ ПРОГРАММНЫХ И АППАРАТНЫХ СРЕДСТВ»

Цель работы: получить теоретические знания и практические навыки по конфигурированию программных и аппаратных средств.

Теоретические сведения

Классификация компьютеров.

Аппаратная и программная конфигурация.

Существуют различные классификации компьютерной техники:

1. по этапам развития (по поколениям);
2. по архитектуре;
3. по производительности (микрокомпьютеры, в том числе персональные компьютеры; миникомпьютеры; мэйнфреймы (универсальные компьютеры);суперкомпьютеры.
4. по условиям эксплуатации (офисные (универсальные) и специальные)
5. по количеству процессоров;
6. по потребительским свойствам и т.д.

Состав вычислительной системы называется ее *конфигурацией*. Различают аппаратную и программную конфигурацию. Современные компьютеры имеют блочную конструкцию. Аппаратную конфигурацию, необходимую для выполнения конкретных видов работ, можно собрать из готовых блоков и гибко изменять по мере необходимости.

Согласование между отдельными блоками выполняется с помощью устройств, называемых *аппаратными интерфейсами*. Стандарты на аппаратные интерфейсы называются *протоколами*. Аппаратные интерфейсы разделятся на *последовательные* и *параллельные*.

1. *Последовательный интерфейс* обеспечивает передачу данных последовательно, бит за битом и поэтому обеспечивают малую скорость передачи данных и имеют простое устройство.

1. Параллельные интерфейсы обеспечивают передачу данных одновременно группами битов, что повышает скорость передачи данных. Количество битов в группе называется разрядностью интерфейса. Существуют 8, 16, 32 и 64-разрядные интерфейсы.

В настоящее время базовая аппаратная конфигурация персонального компьютера включает следующие устройства:

1. системный блок;
2. монитор;
3. клавиатуру;
4. мышь.

Программная конфигурация.

Конечная цель выполнения любой программы – управление аппаратными средствами. Программное и аппаратное обеспечение работают в непрерывном взаимодействии, и их разделение является довольно условным. Между программами, также как между аппаратными средствами, существует взаимосвязь, поэтому можно говорить о программном интерфейсе. Программный интерфейс основан на протоколах – соглашениях о взаимодействии программ. Всё программное обеспечение вычислительной системы разбивается на несколько взаимодействующих между собой уровней (рис. 18). Каждый следующий уровень опирается на программное обеспечение предшествующих уровней.

Такое разделение программного обеспечения упрощает разработку и эксплуатацию программ. Каждый следующий уровень повышает функциональные возможности всей системы.

Базовый уровень. Это самый низкий уровень программного обеспечения. Базовое программное обеспечение отвечает за взаимодействие с базовыми аппаратными средствами.

Системный уровень. Этот уровень обеспечивает взаимодействие прочих программ вычислительной системы с программами базового уровня и непосредственно с аппаратным обеспечением. От программ этого уровня во многом зависят эксплуатационные показатели всей вычислительной системы. При подключении к системе нового оборудования на системном уровне должна быть установлена программа, обеспечивающая взаимодействие других программ с этим оборудованием. Конкретные программы, отвечающие за взаимодействие с конкретными устройствами, называются драйверами устройств.

Служебный уровень. Программное обеспечение этого уровня взаимодействует как с программным обеспечением базового уровня, так и с программным обеспечением системного уровня. Служебные программы называются утилитами. Они предназначены для автоматизации работ по проверке, наладке и настройке вычислительной системы, а также для расширения и улучшения функций системных программ.

Прикладной уровень. Программное обеспечение этого уровня представляет собой комплекс прикладных программ, с помощью которых на данном рабочем месте выполняются конкретные работы. Диапазон возможных приложений вычислительной системы зависит от наличия прикладных программ для разных видов деятельности. Широта функциональных возможностей компьютера напрямую зависит от типа используемой операционной системы.

Программа MyBIOS имеет полное сходство с интерфейсом и меню оригинальной программы BIOS Setup Utility материнской платы ASUS P5K.

Программа предназначена в первую очередь для высших и средних специальных учебных заведений, в которых может быть использована для проведения практических занятий, цель которых — научить студентов настраивать различные опции BIOS. Без использования данной программы проведение подобных практических занятий осложняется тем, что неправильная настройка BIOS может привести к неработоспособности или к сбоям в работе компьютера.

Программа MyBIOS не выполняет реальной настройки аппаратных устройств компьютера, она только эмулирует меню и опции BIOS Setup Utility материнской платы, поэтому не может нарушить работоспособность компьютера.

Важной особенностью программы является наличие режима работы, при котором студентам даются задания по настройке BIOS, после выполнения которых выводится оценка в баллах (один балл за каждое правильно выполненное задание). Список доступных заданий сгруппирован по разделам меню BIOS, задания из списка выбираются в случайном порядке.

После выхода из эмулятора и его повторного запуска все значения опций BIOS Setup Utility возвращаются в исходное положение.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Определите с помощью CPU-Z конфигурацию учебного ПК и запишите основные показатели в отчет.

Задание 3. С помощью ресурсов Интернет подобрать такую конфигурацию аппаратных средств компьютера, которая позволит работать со следующими программными средствами (на выбор):

- CorelDraw (демо-версия).
- 3D MAX (демо-версия).
- AutoCad (демо-версия).

Задание 4. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое конфигурация ПО?
2. Что такое конфигурационное управление?
3. Какое оно бывает?
4. Для чего нужен процесс управления конфигурацией?
5. Из каких этапов он состоит?
6. Назовите средства управление конфигурацией

Практическая работа №8
«НАСТРОЙКИ СИСТЕМЫ И ОБНОВЛЕНИЙ»

Цель работы: получить теоретические знания и практические навыки по настройке системы и обновлений.

Теоретические сведения

В Windows 10 подход к обновлению системы кардинально изменился, в связи с чем возможности по настройке автоматического обновления сильно урезаны. Кроме того, доступные настройки зависят от используемой редакции Windows.

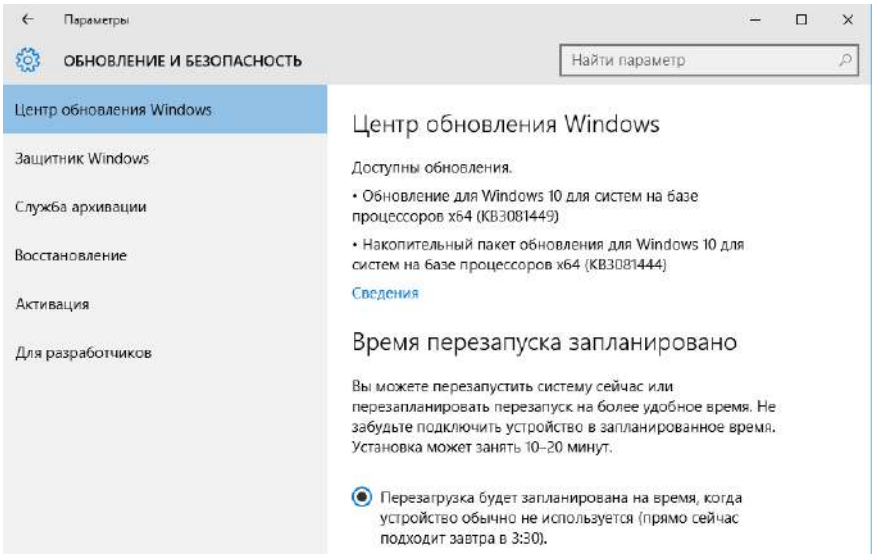
Так в Windows 10 Home все новые функции, апдейты и обновления безопасности устанавливаются с Windows Update, при этом у пользователя нет никакой возможности управления процессом обновления. Единственное, что может сделать пользователь — это ненадолго отложить перезагрузку. Редакция Pro более гуманна и позволяет управлять некоторыми параметрами обновления — выбирать источник обновления, задавать расписание и откладывать установку обновлений (кроме обновлений безопасности) на длительный срок. Возможность полного отключения автоматического обновления не предусмотрена ни в одной из редакций в принципе.

Конечно, автоматическое обновление является важным компонентом операционной системы, а регулярная загрузка и установка обновлений, особенно критических исправлений и обновлений безопасности, необходима для стабильного и безопасного функционирования системы. Однако у пользователя должна быть возможность выбрать когда, как и какие обновления загружать и устанавливать. И даже возможность полностью выключить автоматическую проверку наличия обновлений тоже должна быть. О том, как реализовать эту возможность и пойдет речь далее.

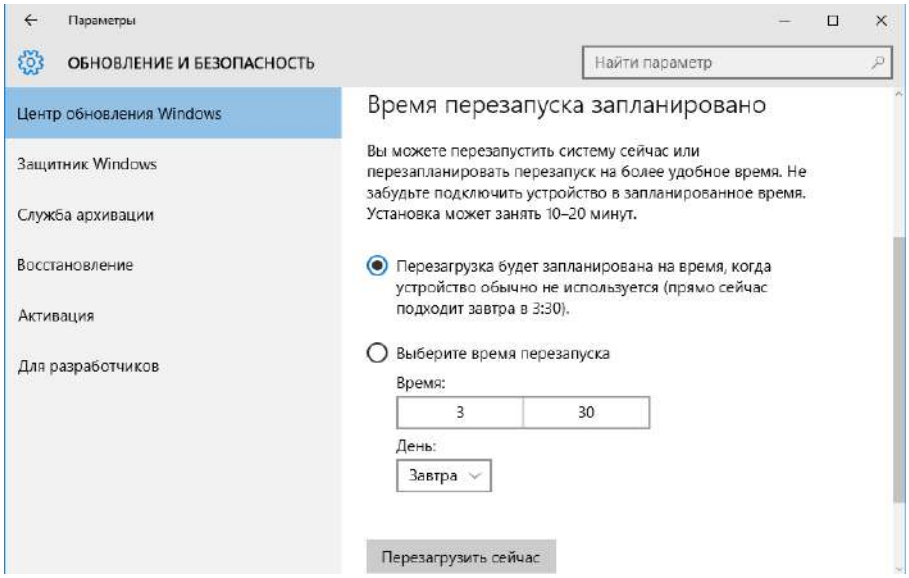
Центр обновления Windows

В Windows 10 привычный классический «Центр обновления Windows» окончательно убран из панели управления и более недоступен. Новый центр обновления находится в разделе параметров системы и для его открытия надо в меню «Пуск» перейти в раздел «Параметры» — «Обновление и безопасность».

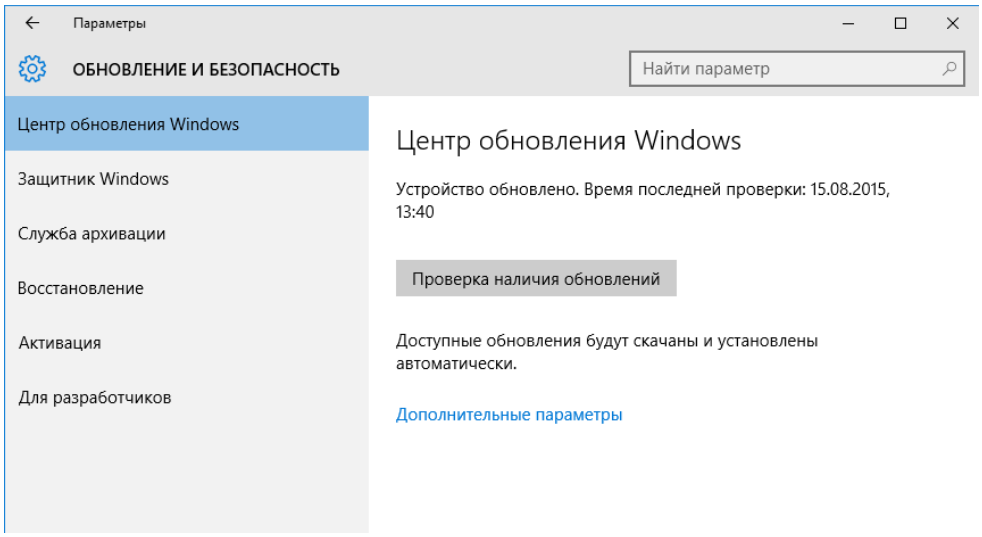
В основном окне центра обновления выбора практически нет, можно только просмотреть список доступных обновлений и сведения о них.



Единственная доступная настройка позволяет выбрать время для обязательной перезагрузки. Здесь можно указать желаемые день и время перезагрузки либо перезагрузиться сразу, чтоб долго не мучаться).

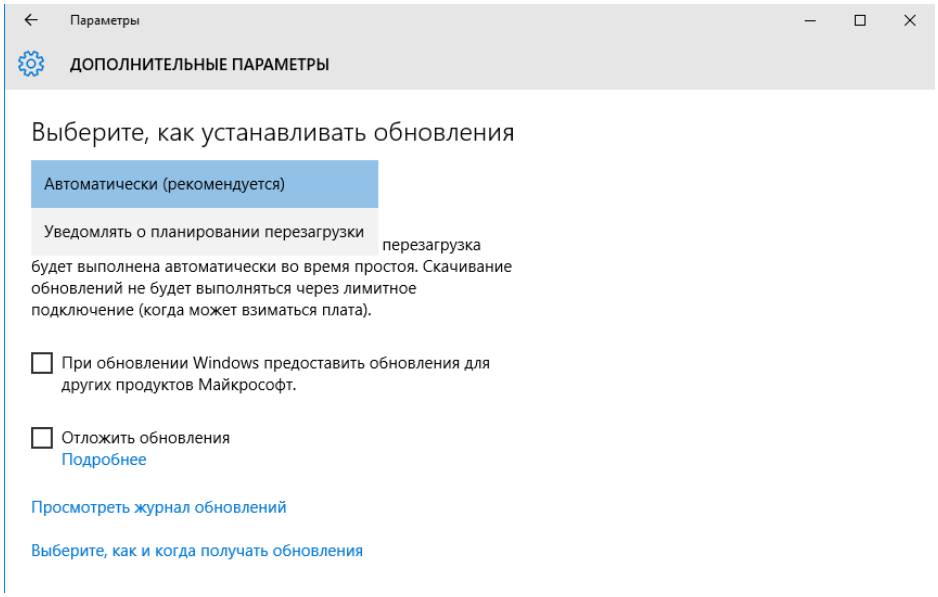


В редакции Home это все доступные настройки, а в старших редакциях Windows 10 (Pro, Enterprise и Education) пользователь может перейти по ссылке «Дополнительные параметры».



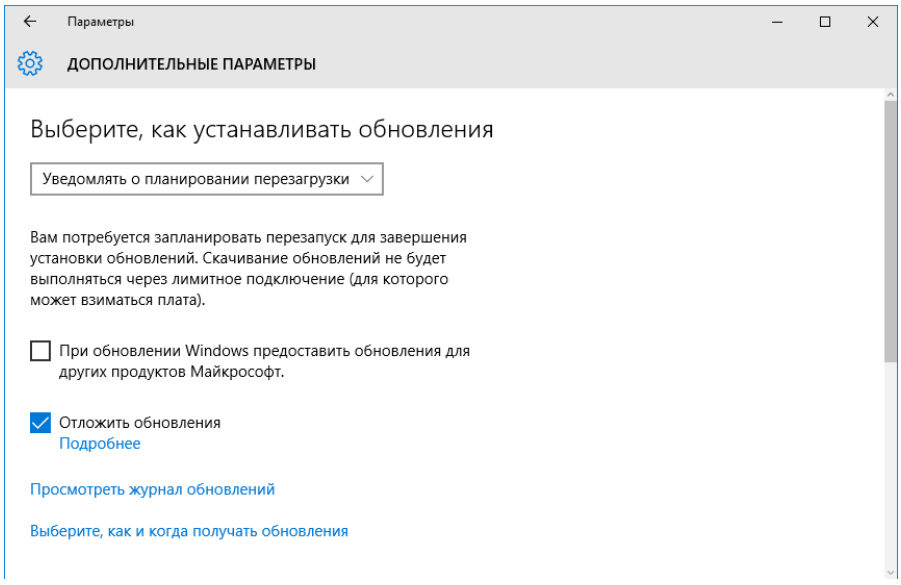
В дополнительных параметрах есть возможность выбрать способ установки обновлений. Правда выбор небогатый и состоит всего из двух вариантов:

• Автоматически — обновления автоматически загружаются и устанавливаются, после чего следует автоматическая перезагрузка;
• Уведомлять о планировании перезагрузки — загрузка и установка обновлений также происходит автоматически, но перезагрузку можно запланировать на удобное для себя время.



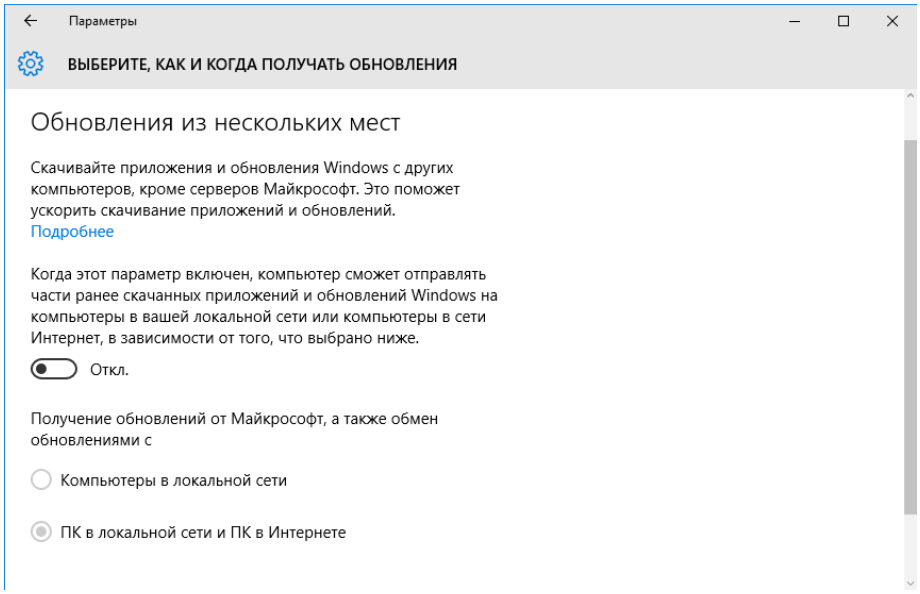
Также можно включить опцию «Отложить обновления», которая позволяет на некоторое время отключить загрузку и установку обновлений. Эта опция рассчитана в основном на корпоративных пользователей и предназначена для того, чтобы дать им возможность протестировать новый функционал (который теперь может включаться в обновления) перед его массовым внедрением в организации. У пользователей редакции Home такой возможности нет, поэтому они невольно выступают тестерами ☺ для корпоративного сектора.

В отложенные обновления не входят критические исправления и обновления безопасности, которые будут автоматически устанавливаться в любом случае. Точный срок, на который откладываются обновления, неизвестен, в разных источниках встречаются цифры от нескольких дней до нескольких месяцев.

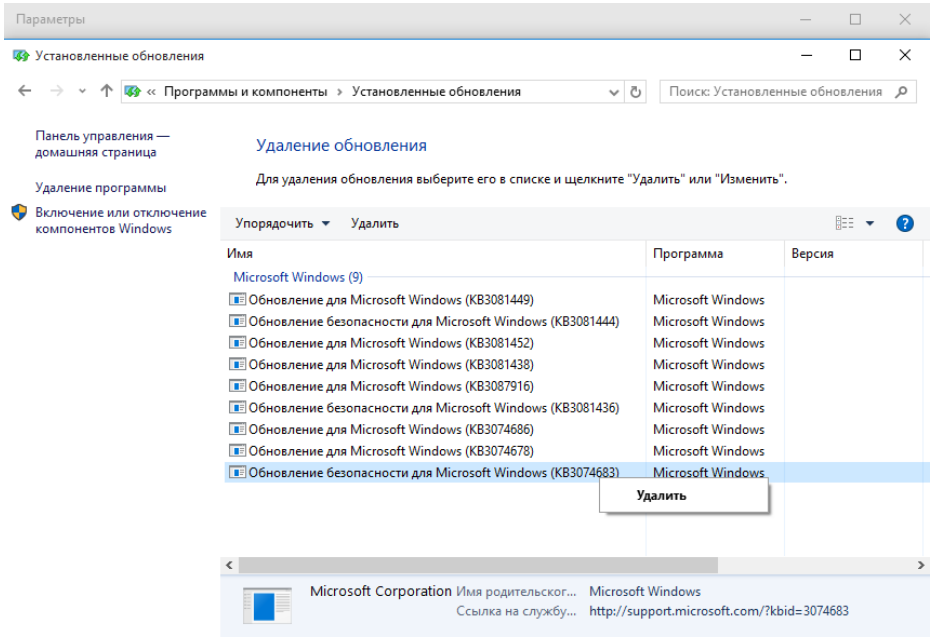


Еще один интересный момент. В Windows 10 встроен новый механизм доставки обновлений (Windows Update Delivery Optimisation), основанный на P2P технологии. Проще говоря, после загрузки обновлений с серверов Windows Update ваш компьютер начинает раздавать их на другие устройства в сети, по принципу торрент-клиента.

Эта функция включена по умолчанию и для ее отключения надо перейти по ссылке «Выберите, как и когда получать обновления» и установить ползунок в положение Откл.



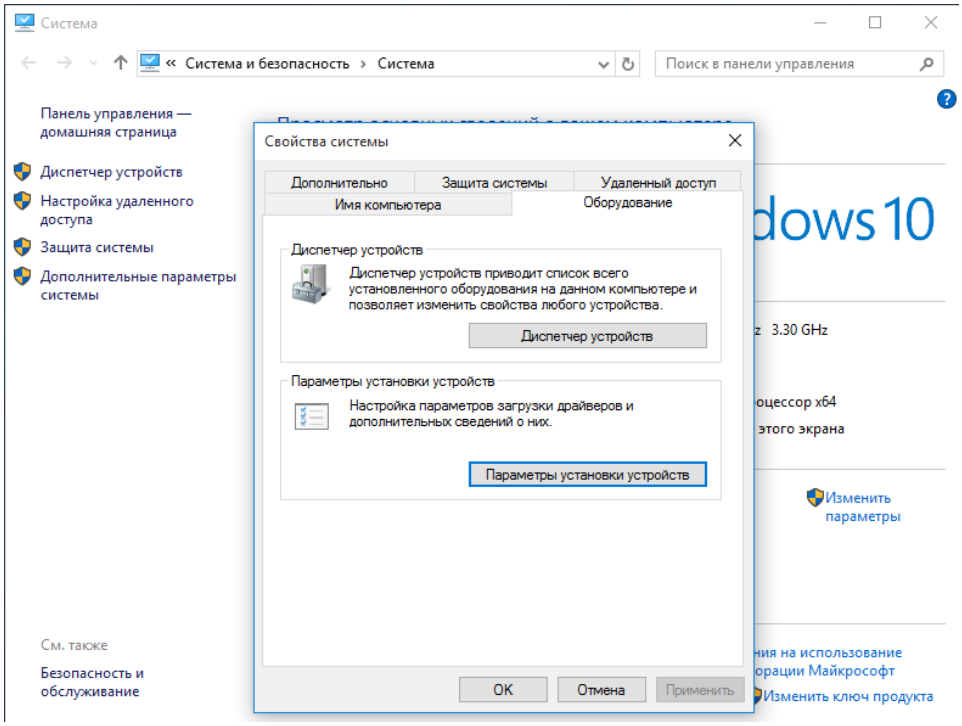
Просмотреть установленные обновления можно в Панели управления, перейдя в раздел «Программы и компоненты» — «Установленные обновления». Если после обновления появились проблемы, то здесь же любое из установленных обновлений можно удалить.



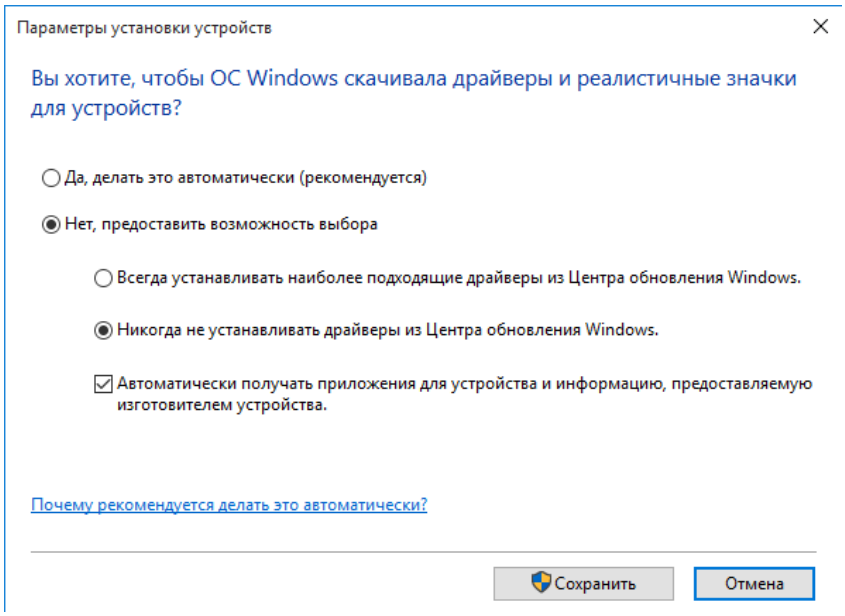
Отключение автоматического обновления драйверов

Операционная система Windows содержит в себе хранилище драйверов для наиболее распространенных устройств. Загрузка новых версий драйверов по умолчанию производится автоматически, с помощью Windows Update. В некоторых случаях наличие в системе нужного драйвера довольно удобно, однако Windows Update далеко не всегда содержит самый свежий\лучший драйвер, да и обновлять драйвера лучше вручную во избежании неожиданных проблем с оборудованием. Поэтому автоматическое обновление драйверов лучше все-таки отключить, благо эта возможность в Windows 10 пока еще есть.

Для того, чтобы добраться до настроек обновления драйверов, нужно открыть в панели управления раздел «Система» и выбрать пункт «Дополнительные параметры системы», затем перейти на вкладку «Оборудование» и нажать кнопку «Параметры установки устройств». Также открыть нужное окно можно, нажав **Win+R** и выполнив команду **rundll32 newdev.dll,DeviceInternetSettingUi.**

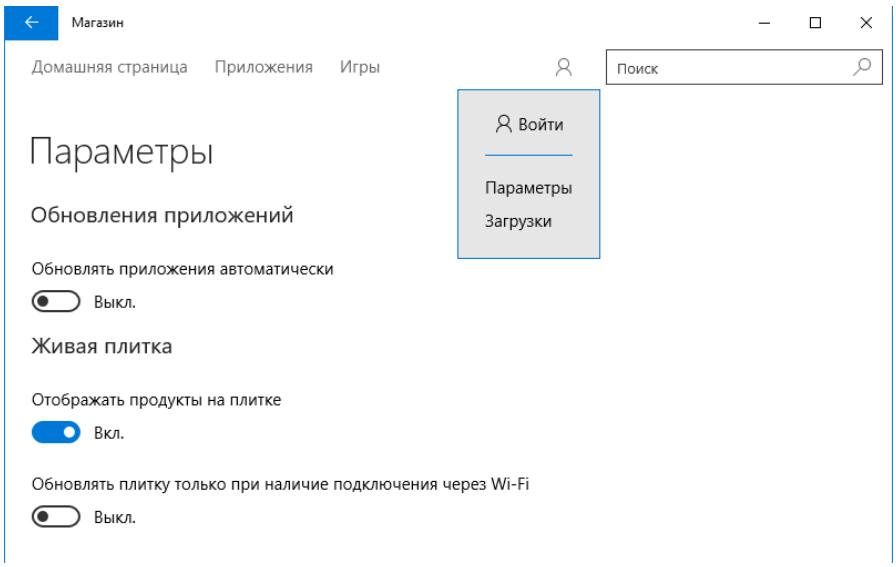


Для отключения автоматической загрузки драйверов выбираем пункт «Никогда не устанавливать драйверы из центра обновления Windows».

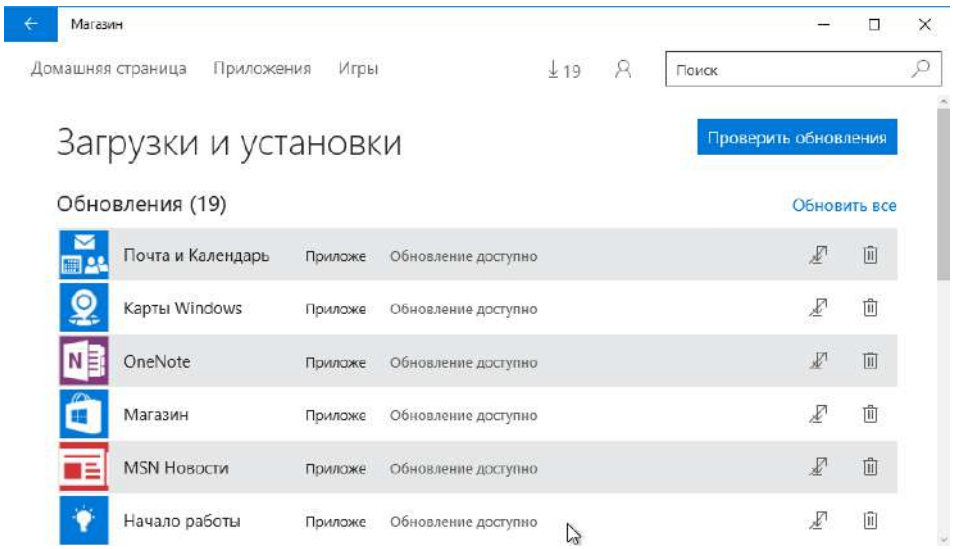


Отключение автоматического обновления современных приложений

Современные приложения (приложения из магазина Windows) также обновляются автоматически. Чтобы это исправить, надо в меню «Пуск» открыть пункт «Магазин», кликнуть по значку человечка, в открывшемся меню выбрать пункт «Параметры», а затем в пункте «Обновлять приложения автоматически» поставить переключатель в положение Выкл.



После этого приложения из магазина не будут обновляться автоматически, и при необходимости их придется обновлять вручную.

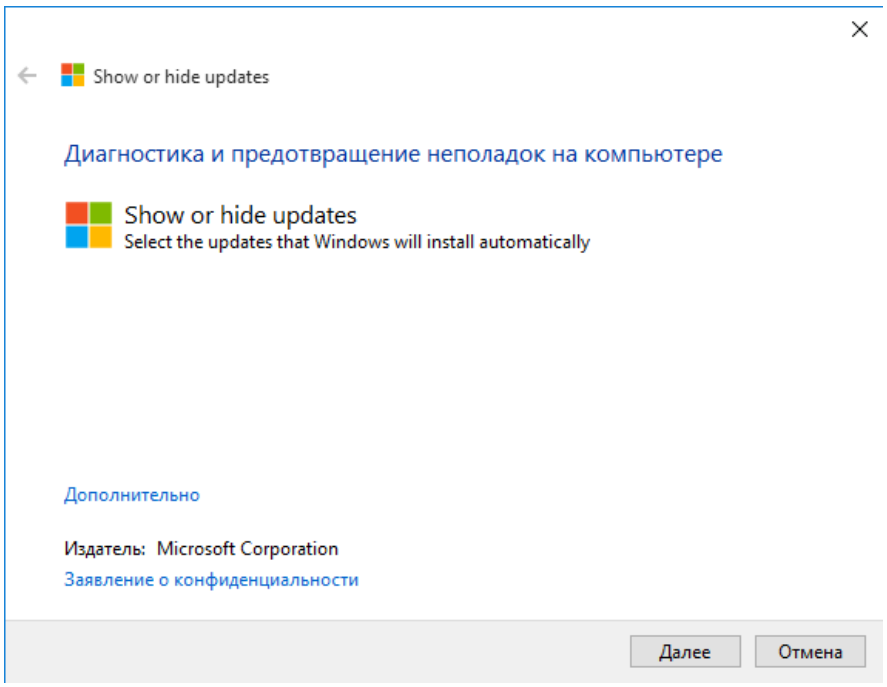


Примечание. Изначально пользователи Windows 10 Home не могли отключить автоматическое обновление приложений из магазина Windows. Эта возможность была добавлена позднее, с обновлением KB3081448.

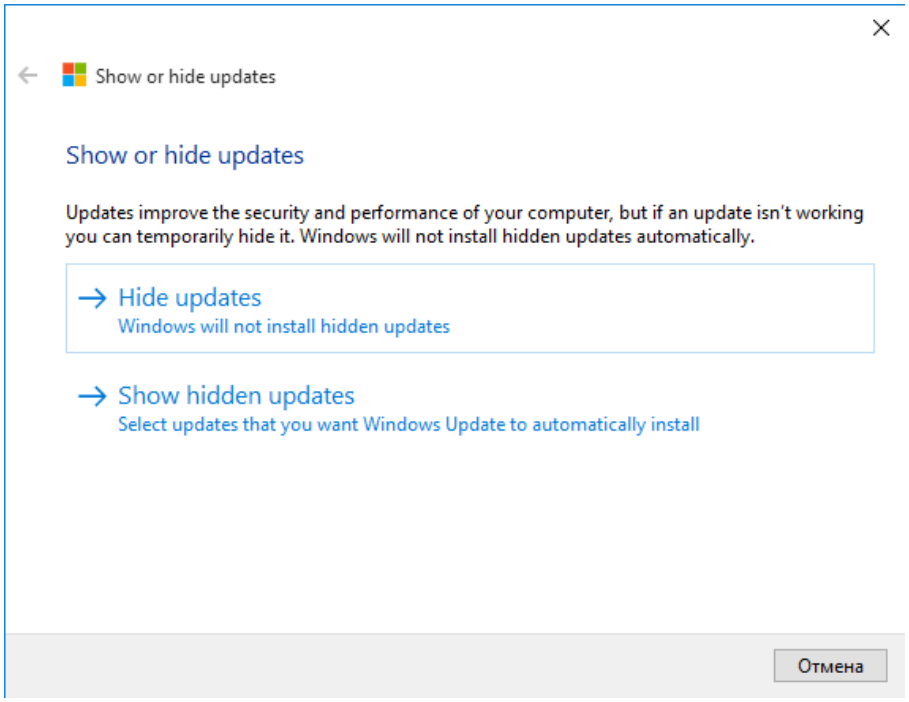
Утилита Show or hide update

Старый Центр обновления помимо прочего давал возможность выбирать и откладывать на неопределенный срок (скрывать) обновления. В новом этой возможности изначально нет, но зато есть специальная утилита от Microsoft с названием Show or hide update, которая умеет делать примерно то-же. Утилита не входит в состав системы а загружается отдельно.

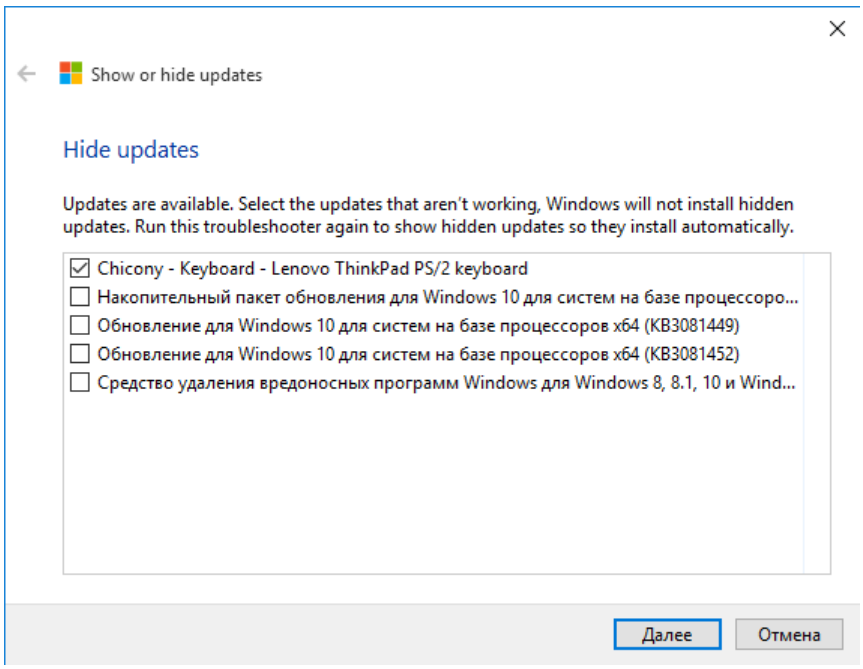
Установка не требуется, достаточно скопировать и запустить файл wushowhide.diagcab, после чего утилита приступает к сбору данных о доступных обновлениях.



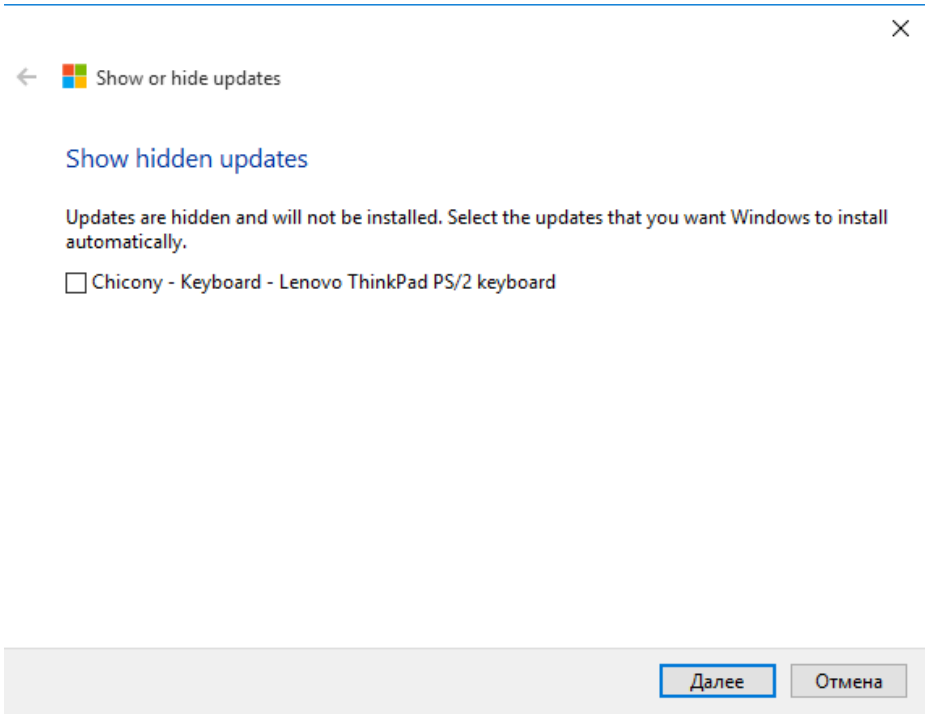
После сбора данных предлагается выбрать действие. Для скрытия обновлений выбираем «Hide updates».



Отмечаем те обновления, которые необходимо скрыть, и жмем «Далее».



Пока обновление скрыто, операционная система не будет его устанавливать. А при необходимости скрытые обновления можно восстановить, для чего надо снова запустить утилиту, выбрать «Show hidden updates» и поставить галочку напротив нужного обновления. После этого обновление будет установлено в общем порядке.



Использование «Show or hide update» дает возможность отложить установку любых проблемных обновлений (в том числе и обновлений безопасности), однако полностью избежать обновления таким образом не получится. Все мелкие обновления входят в состав крупных и рано или поздно будут установлены.

Это были простые способы, потихоньку переходим к более продвинутым.

Настройка автоматического обновления с помощью PowerShell

Для управления обновлениями можно воспользоваться модулем PSWindowsUpdate из центра сценариев Microsoft. Для использования скачиваем архив, распаковываем его и кладем в папку %WINDIR%\System32\WindowsPowerShell\v1.0\Modules. Затем запускаем консоль PowerShell и разрешаем выполнение неподписанных скриптов командой:

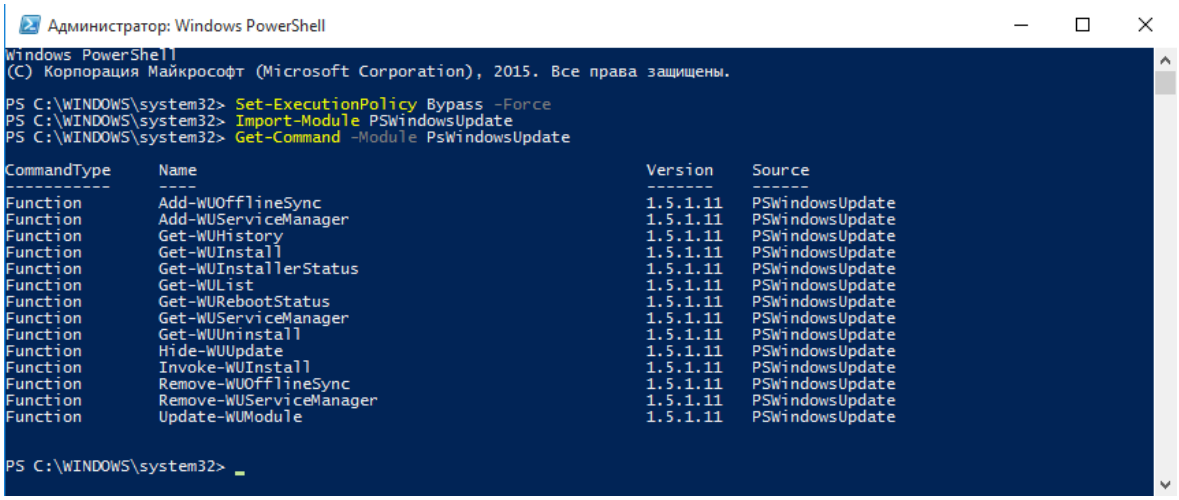
```
Set-ExecutionPolicy Bypass -Force
```

Импортируем модуль в текущий сеанс:

```
Import-Module PSWindowsUpdate
```

И выводим список команд для модуля:

```
Get-Command -Module PSWindowsUpdate
```



Модуль содержит 14 командлетов:

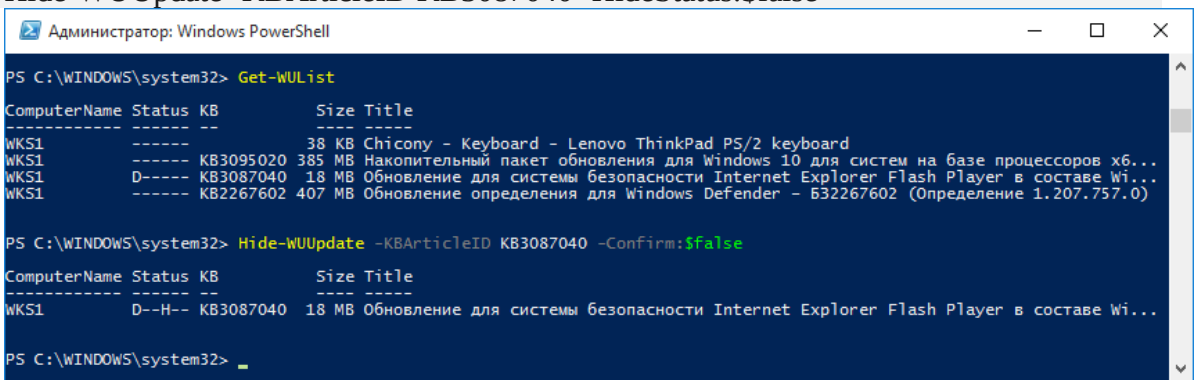
Get-WUList — выдает список доступных обновлений;
Get-WUInstall — запускает загрузку и установку обновлений;
Get-WUUninstall — удаляет выбранные обновления;
Invoke-WUInstall — служит для управления обновлением на удаленных компьютерах;
Hide-WUUpdate — скрывает выбранные обновления;
Get-WUHistory — выводит информацию об установленных обновлениях;
Add-WUOfflineSync — регистрирует службу которая позволяет производить установку обновлений из локального кеша (Offline sync service);
Remove-WUOfflineSync — удаляет зарегистрированную службу;
Get-WUServiceManager — выводит список доступных служб обновления (Windows Update, WSUS и т.п.);
Add-WUServiceManager — регистрирует выбранную службу обновления;
Remove-WUServiceManager — удаляет выбранную службу обновления;
Get-WUInstallerStatus — показывает статус службы Windows Update Installer;
Get-WURebootStatus — позволяет уточнить необходимость перезагрузки;
Update-WUModule — служит для централизованного обновления модуля PSWindowsUpdate на удаленных компьютерах.

Для примера выведем список доступных обновлений и скроем одно из них:

```
Get-WUList
Hide-WUUpdate -KBArticleID KB3087040 -Confirm:$false
```

После этого обновление KB3087040 не будет установлено. При необходимости его можно разблокировать такой командой:

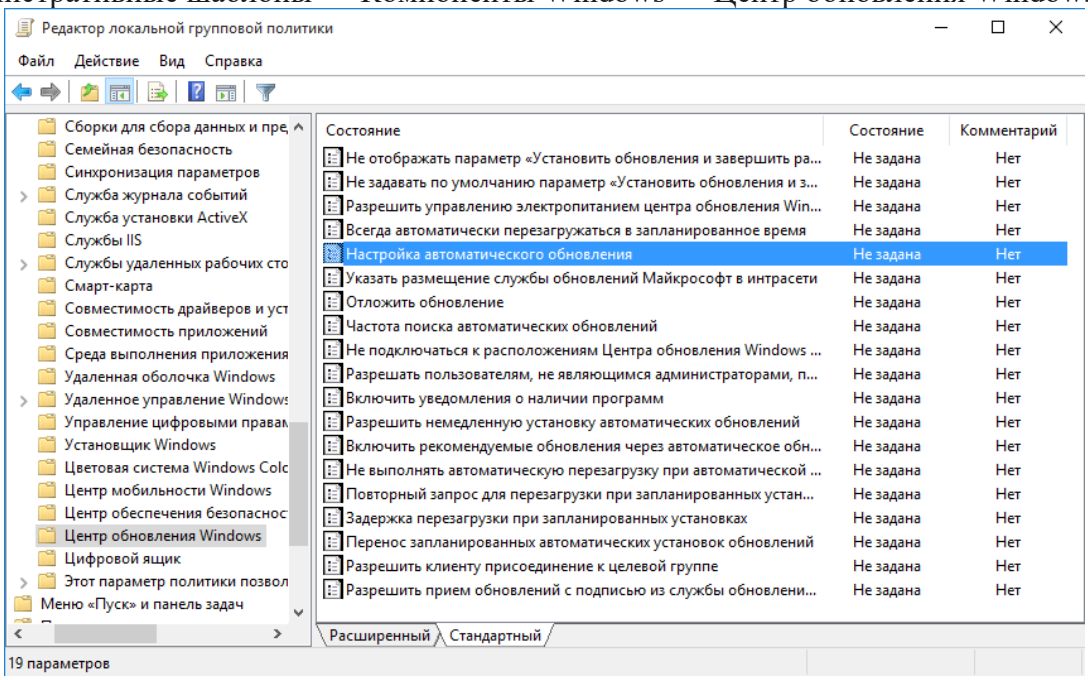
```
Hide-WUUpdate -KBArticleID KB3087040 -HideStatus:$false
```



Вообще модуль PSWindowsUpdate имеет довольно много возможностей, с которыми, по хорошему, надо подробно разбираться. Поддержка Windows 10 автором пока не заявлена, однако команды вполне корректно выполняются.

Настройка автоматического обновления с помощью групповых политик

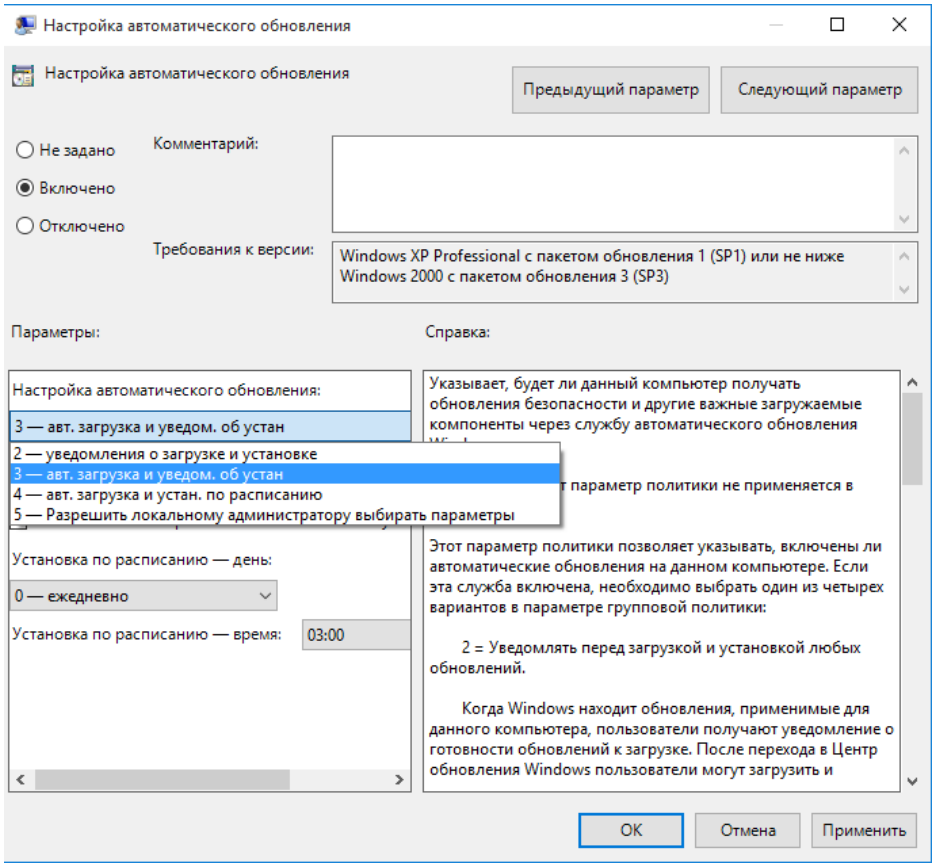
Этот способ доступен только для старших редакций Windows 10, т.к. в редакции Home нет редактора групповых политик. Для открытия оснастки редактора локальных групповых политик нажимаем **Win+R** и выполняем команду **gpedit.msc**. Настройки автоматического обновления находятся в разделе Конфигурация компьютера — Административные шаблоны — Компоненты Windows — Центр обновления Windows.



Все основные настройке задаются в политике «Настройка автоматического обновления». Сначала ее надо включить, после чего можно выбрать один из 4-х вариантов обновления:

2 — уведомление о загрузке и установке обновлений;
3 — автоматическая загрузка и уведомление об установке;
4 — автоматическая загрузка и установка по расписанию;
5 — разрешить локальному администратору выбирать параметры автоматического обновления.

Если выбран вариант 4, то дополнительно можно задать день и час для установки обновлений, а также указать устанавливать обновления только в период простоя, во время автоматического обслуживания системы.



Примечание. При активации данной политики настройки в Центре обновления Windows становятся недоступны. Исключение составляет вариант под номером 5, позволяющий локальным администраторам изменять режим обновления в центре обновления.

Если выбран вариант автоматической загрузки и установки по расписанию, то дополнительно можно использовать следующие настройки.

Всегда автоматически перезагружаться в запланированное время

Если эта политика включена, то после установки обновлений компьютер будет перезагружен независимо не от чего. Чтобы перезагрузка не была внезапной и пользователи успели сохранить результаты своей работы, можно задать таймер перезагрузки от 15 минут до 3 часов.

Не выполнять автоматическую перезагрузку при автоматической установке обновлений, если в системе работают пользователи

Тут все ясно из названия политики. Если эта политика включена, то после установки обновлений компьютер не перезагружается автоматически, а выводит уведомление о завершении установки и ждет перезагрузки пользователем. Отменяет действие предыдущей политики.

Повторный запрос для перезагрузки при запланированных установках

Эта политика определяет время, через которое система выдаст повторный запрос при отмене запланированной перезагрузки. Если эта политика не активна, то запросы будут выдаваться каждые 10 минут.

Задержка перезагрузки при запланированных установках

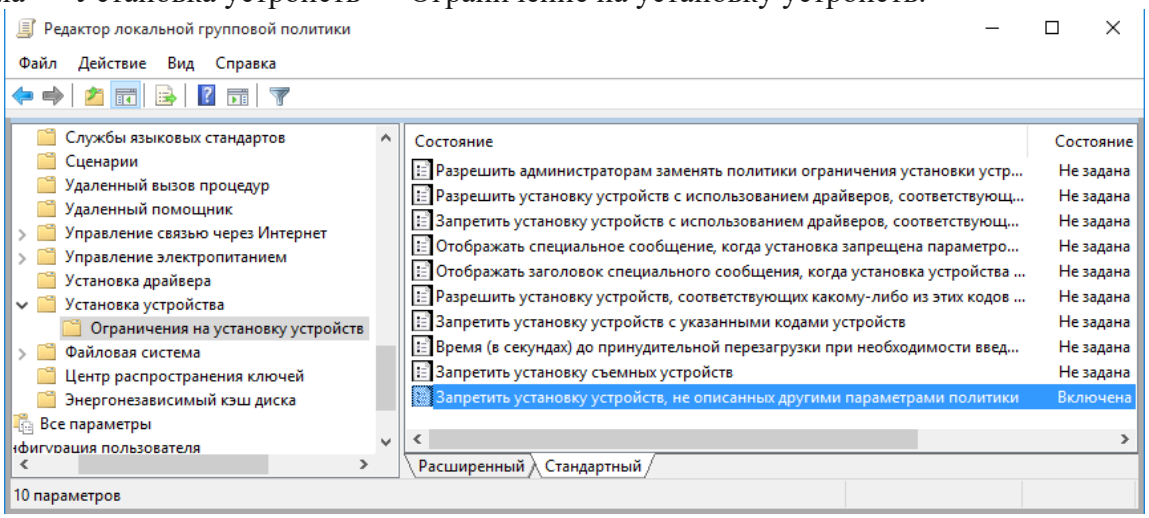
В этой политике задается время, которое должно пройти с момента окончания установки обновлений до перезагрузки.

Перенос запланированных автоматических установок обновлений

В том случае, если компьютер был выключен и установка обновлений не была произведена в запланированное время, она будет запущена сразу после следующего запуска компьютера. В этой политике можно указать время, которое должно пройти с момента загрузки системы до начала установки.

Запретить установку устройств, не описанных другими параметрами политики

Эта политика служит для отключения автоматического обновления драйверов и находится в разделе Конфигурация компьютера — Административные шаблоны — Система — Установка устройств — Ограничение на установку устройств.



Примечание. Я описал настройку локальных групповых политик, но есть и точно-такие же доменные. И если компьютер находится в сети предприятия и является членом домена Active Directory, то (как правило) настройки автоматического обновления определяются доменными политиками. Доменные политики имеют высший приоритет и переопределяют любые локальные настройки.

Настройка автоматического обновления с помощью реестра

Наиболее мощное средство управления системой — редактирование реестра. В реестре можно задать все те-же настройки, что и с помощью групповых политик, кроме того он дает возможность полностью выключить автоматическое обновление.

Для настройки открываем редактор реестра (Win+R -> Regedit) и заходим в раздел HKLM\SOFTWARE\Policies\Microsoft\Windows. Создаем новый раздел **WindowsUpdate**, в нем подраздел с именем **AU**. В разделе AU создаем параметры типа DWORD, которые отвечают за автоматическое обновление.

Вот наиболее важные из них.

AUOptions — основной параметр, отвечающий за способ получения и установки обновлений. Может иметь следующие значения:

• 2 — уведомлять перед загрузкой и установкой любых обновлений;
• 3 — автоматически загружать обновления и уведомлять о готовности установки;
• 4 — автоматически загружать обновления и устанавливать согласно расписанию;
• 5 — Разрешить локальным администраторам управлять параметрами обновления.

NoAutoUpdate — параметр, позволяющий полностью отключить автоматический поиск и установку обновлений. Значение **1** — автоматическое обновление отключено, **0** — обновления будут загружаться и устанавливаться согласно установкам, заданным в параметре **AUOptions.**

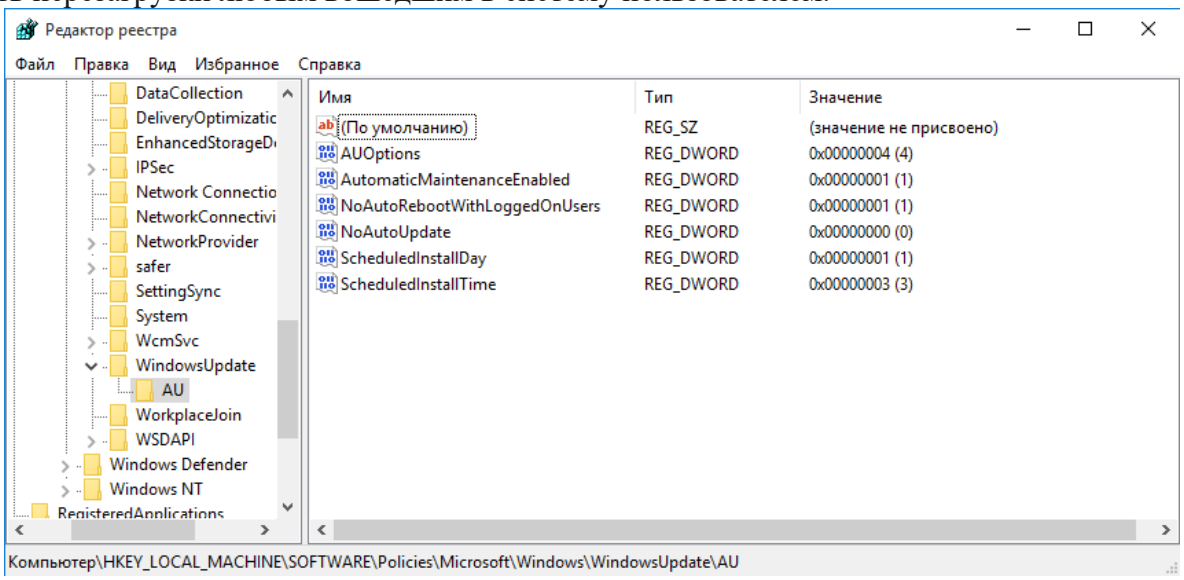
В случае, если значение включено автоматическая загрузка и установка по расписанию (AUOptions = 4), то дополнительно можно указать дополнительные параметры.

ScheduledInstallDay — день недели, на который запланирована установка обновлений. Значение **0** означает ежедневную установку, значения от **1** до **7** указывают на конкретный день недели (1 — понедельник).

ScheduledInstallTime -время, на которое запланирована установка обновлений. Для этого параметра доступны значения от 0 до 23 часов, что соответствует часам в сутках.

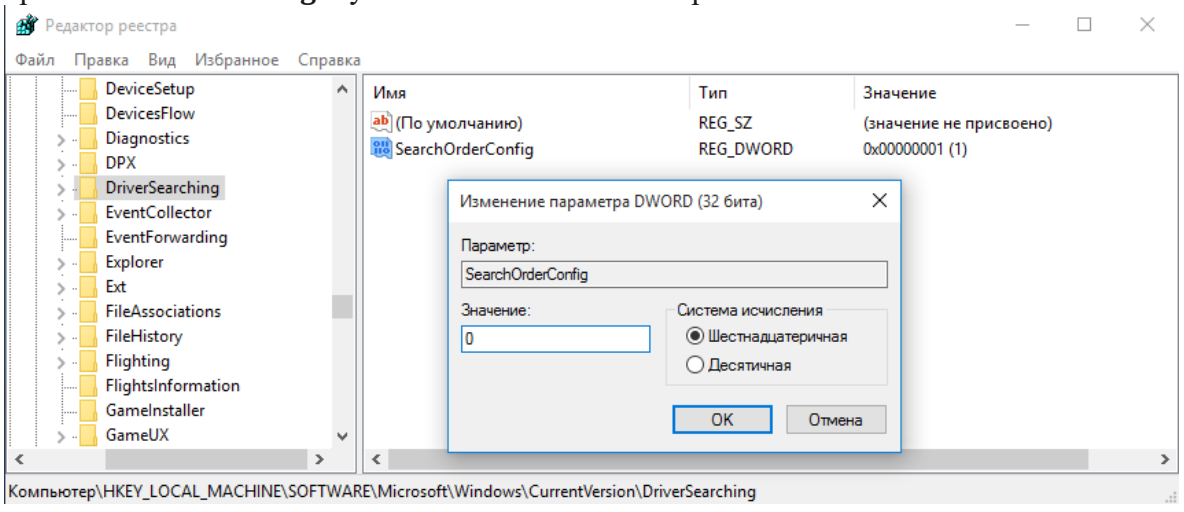
AutomaticMaintenanceEnabled — значение 1 означает, что установку обновлений надо производить во время простоя, в рамках автоматического обслуживания системы.

NoAutoRebootWithLoggedOnUsers — значение 1 запрещает автоматическую перезагрузку после установки обновлений. Служба автоматического обновления будет ожидать перезагрузки любым вошедшим в систему пользователем.



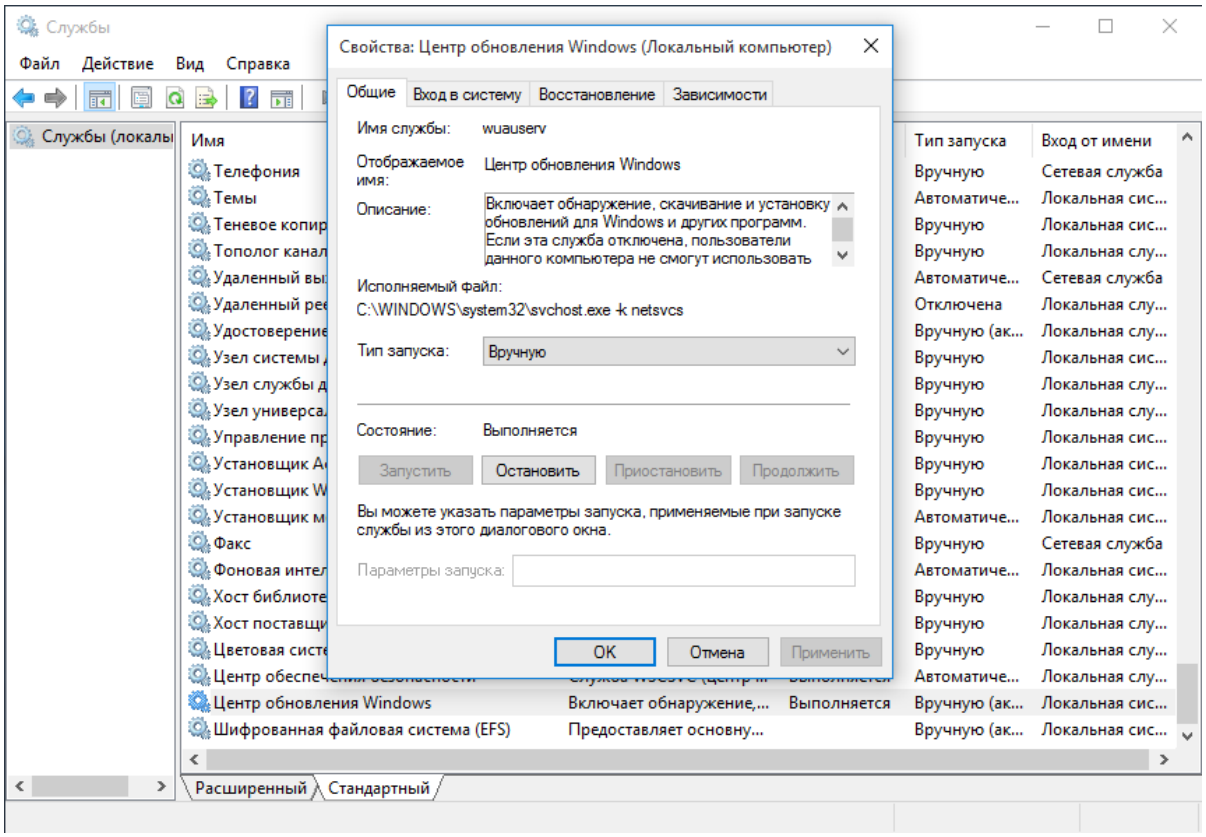
Примечание. Чисто теоретически управление обновлениями с помощью редактирования реестра может работать и в Windows 10 Home. На практике это не проверялось по причине отсутствия у меня нужной редакции.

Для отключения автоматического обновления драйверов надо в разделе HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\DriverSearching найти параметр **SearchOrderConfig** и установить его значение равным **0**.



Отключение службы автоматического обновления

И на самый экстренный случай отключить автоматическое обновление можно, остановив соответствующую службу. Для этого надо открыть оснастку «Службы» (Win+R -> services.msc), найти службу с названием «Центр обновления Windows» и остановить ее. А чтобы она не запустилась самостоятельно, тип запуска надо установить в положение Отключено.



То же самое можно сделать и с помощью PowerShell. Для остановки используем такую команду:

```
Stop-Service wuauserv -Force
```

Для отключения такую:

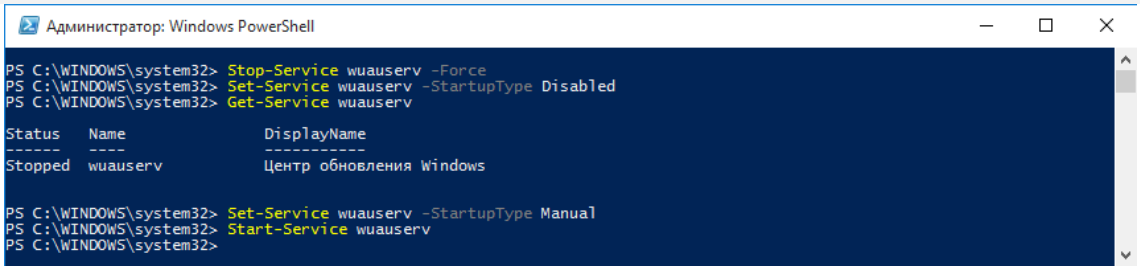
```
Set-Service wuauserv -StartupType Disabled
```

Посмотреть состояние службы можно так:

```
Get-Service wuauserv
```

Ну а так можно вернуть все обратно и запустить службу:

```
Set-Service wuauserv -StartupType Manual
Start-Service wuauserv
```



После отключения службы Центр обновления станет выдавать ошибку при попытке проверить наличие обновлений. Это крайне грубый способ, хотя он и работает на всех без

исключения редакциях десятки, но — использовать его стоит в экстренных случаях, например при необходимости срочно остановить процесс обновления.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Проверьте ОС на обновление данных. Произведите обновление. (Если обновление не касается на переход более новой ОС)

Задание 3. Составить подробную презентацию на тему «Обновление Windows» для версий 10.

Задание 4. Выписать различные способы обновления программ, привести примеры.

Задание 5. Составить подробную презентацию на тему «Обновление приложений в браузере».

Задание 6. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое обновление ПО?
2. Что такое пакет обновлений?
3. Как осуществляется автоматический поиск обновлений?
4. Какие программы автоматического поиска обновлений вы знаете?

Практическая работа №9
«СОЗДАНИЕ ОБРАЗА СИСТЕМЫ. ВОССТАНОВЛЕНИЕ СИСТЕМЫ»

Цель работы: получить теоретические знания и практические навыки по созданию образа системы и восстановление системы.

Теоретические сведения

Слишком много резервных копий никогда не бывает. Если вы можете создать резервную копию резервной копии, то обязательно сделайте это. К счастью, Microsoft значительно упрощает создание полной копии всего вашего компьютера с помощью встроенного инструмента “Создание образа системы”.

Что такое образ системы?

Образ системы в Windows 8.1 и Windows 10 включает полную копию всей системы. Данная резервная копия включает полную сборку операционной системы, настройки, установленные программы, приложения Windows и все персональные файлы.

Главное преимущество данной функции – это то, что она позволяет полностью восстановить систему в случае критической ошибки с ПО или аппаратного сбоя.

Недостаток данного метода заключается в том, что вы не сможете восстановить отдельные файлы, т.е. все файлы на жестком диске. Хотя вы будете иметь доступ к скопированным файлам в библиотеках Документы, Изображения, Музыка и др.

Кроме того, если вы не часто выполняете резервное копирование, то в случае системного сбоя, вы можете потерять ценные документы, настройки или программы, потому что вы сможете восстановить только данные на момент последней резервной копии.

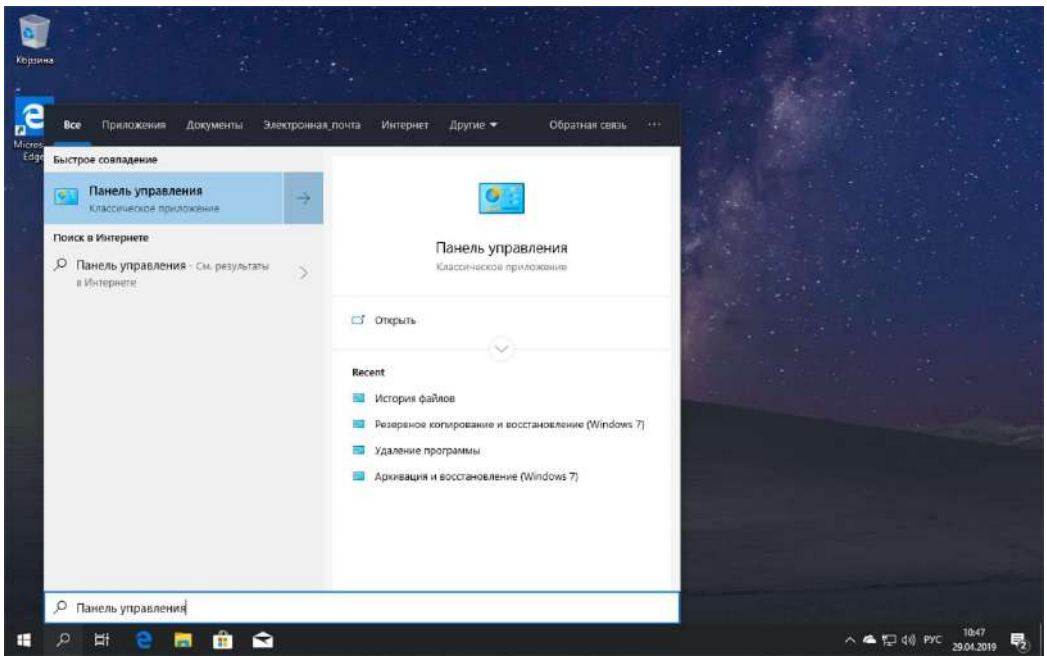
браз системы можно использовать для создания основной резервной копии компьютера, включающей саму установку Windows, все последние обновления, персонализированные конфигурации и программы. Для создания резервной копии файлов можно использовать **Историю файлов** или функцию **«Защита важных файлов»** сервиса OneDrive. Данный рецепт резервного копирование позволит избежать дублирование файлов, и у вас всегда будет актуальная копия всех личных документов.

Если компьютер работает корректно, то системная функция резервного копирования поможет во всех ситуациях, когда в систему были внесены изменения, которые могут нарушить работоспособность. При этом восстановление исходного состояния займет приемлемое время.

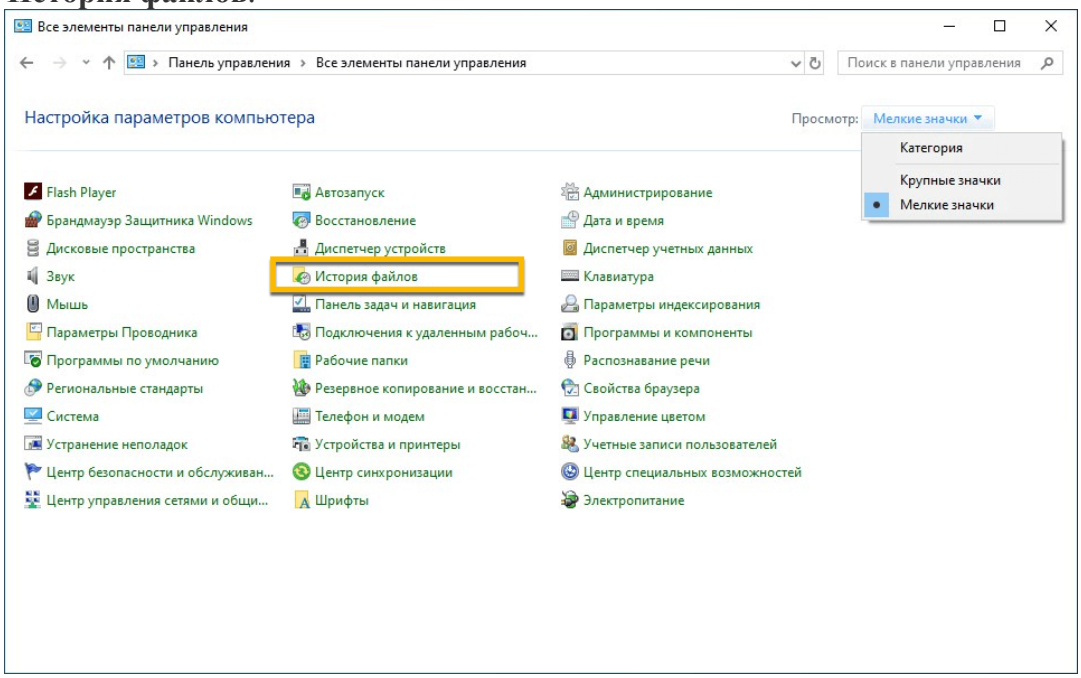
Представленные инструкции позволяют создать резервный образ системы в Windows 10, но они отлично работают и в Windows 8.1.

Как создать резервную копию образа системы в Windows 10 или 8.1

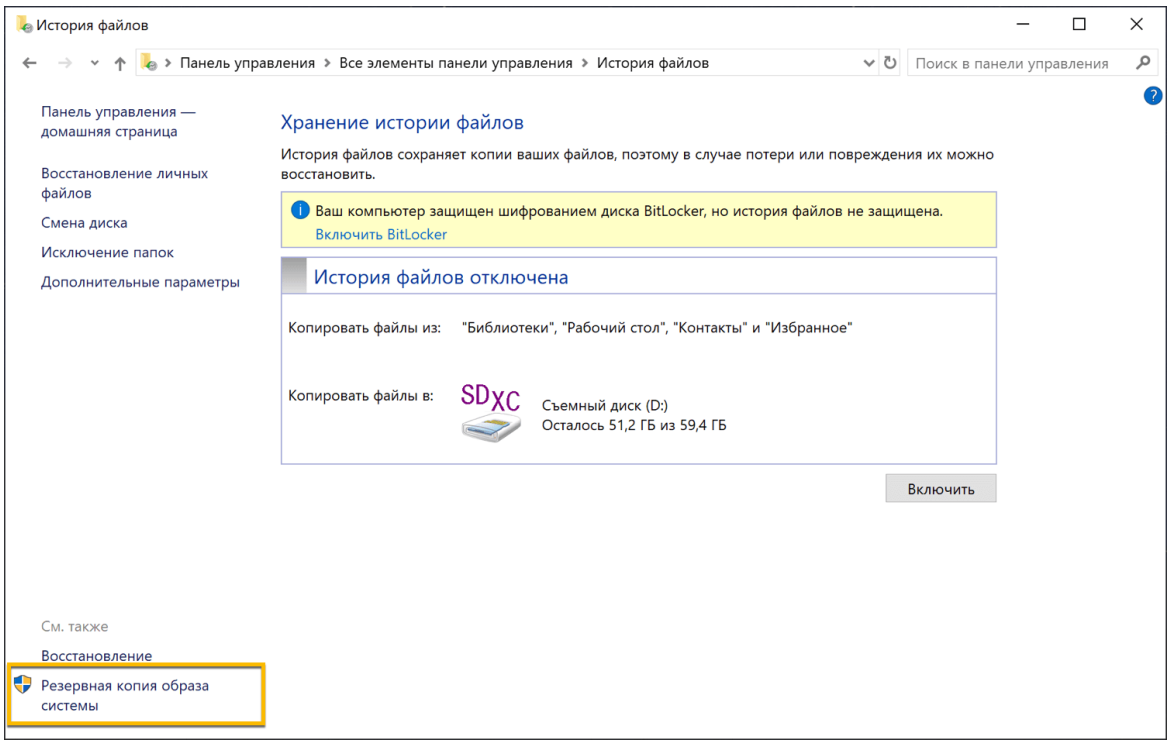
- Наберите в поиске меню Пуск **Панель управления** и перейдите по найденной ссылке.



- Включите режим просмотра *Крупные значки* или *Мелкие значки* и выберите раздел **История файлов**.



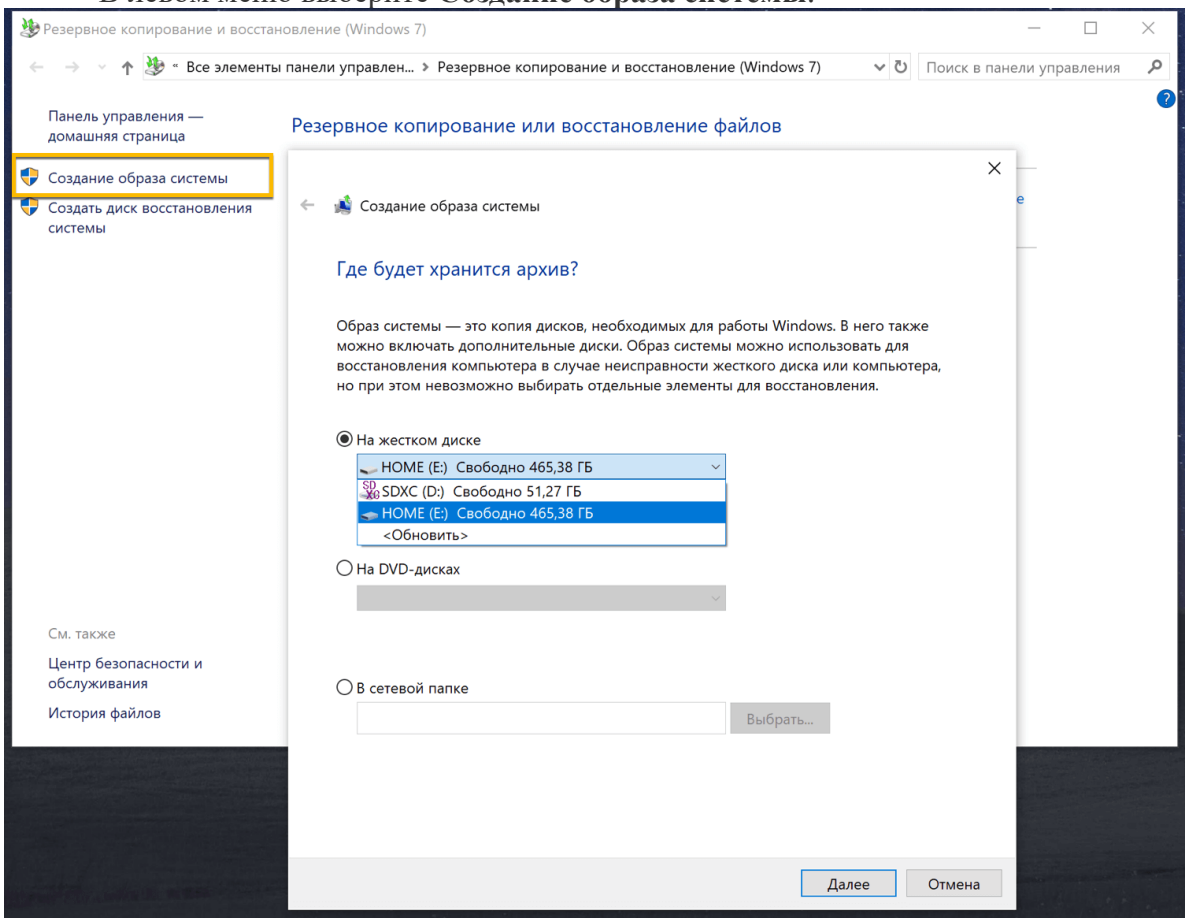
- Выберите ссылку **Резервная копия образа системы** в нижнем-левом углу окна.



- Подключите портативный жесткий диск (USB-накопитель), USB-флешку или SD-карту с достаточным объемом свободного пространства.

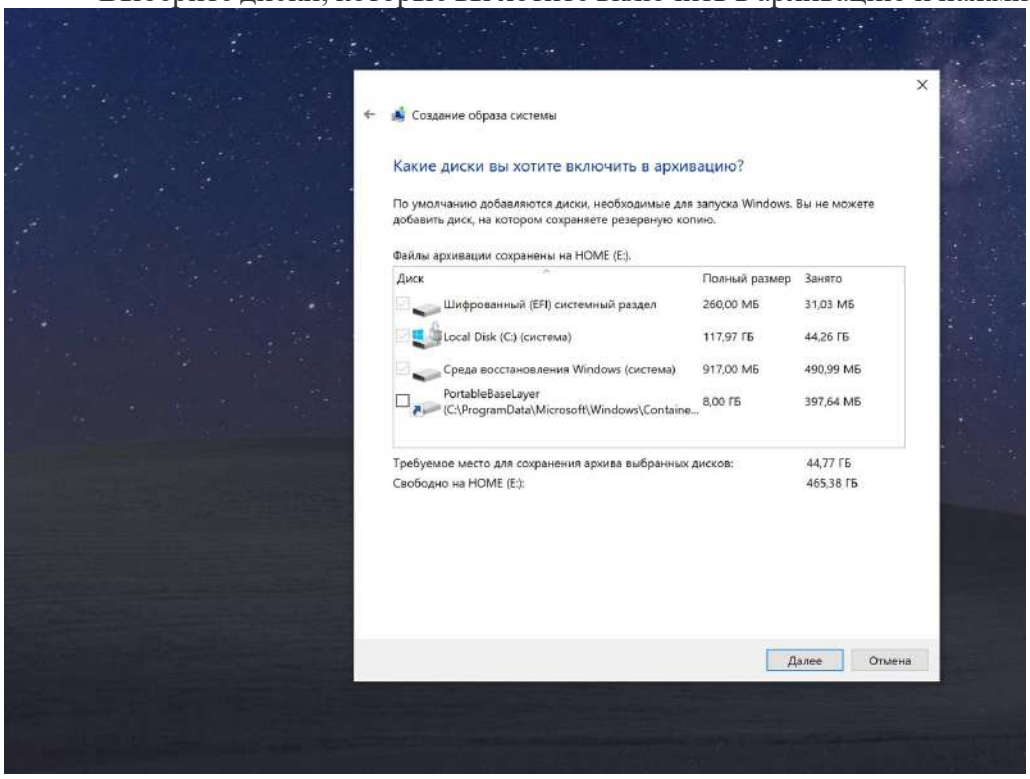
Примечание. Носитель должен быть отформатирован в файловой системе NTFS, чтобы его можно было использовать для хранения образа системы.

- В левом меню выберите **Создание образа системы**.

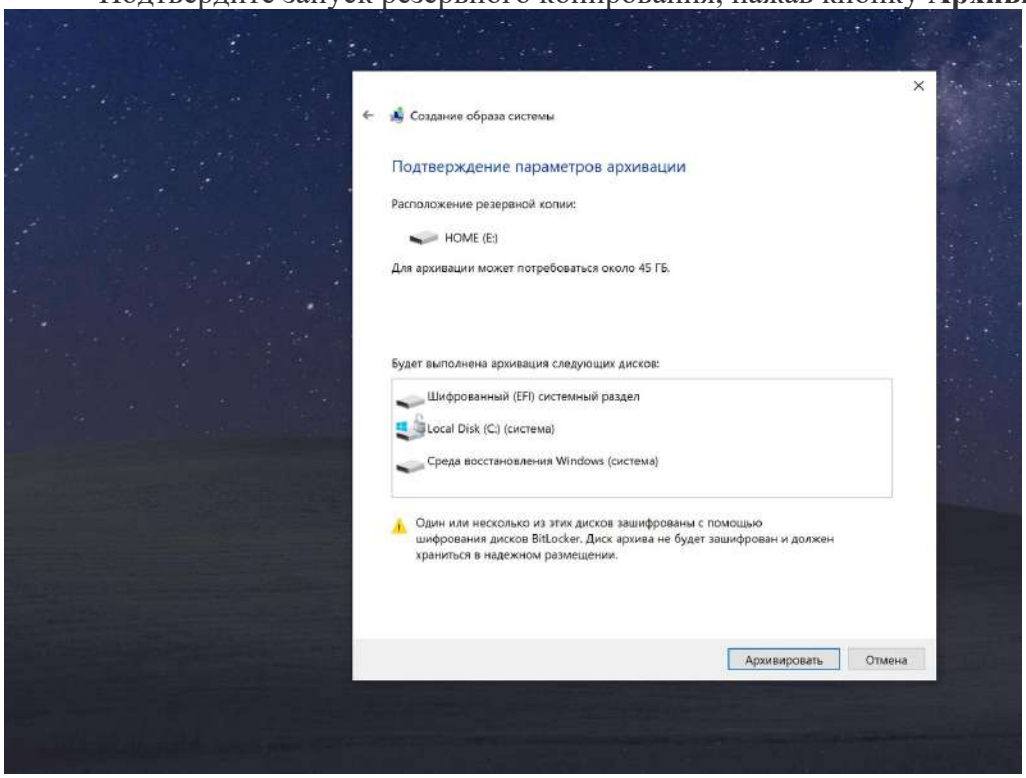


- Мастер создания образа системы попросит выбрать один из трех вариантов размещения резервной копии: на жестком диске, на DVD-дисках или в сетевой папке. Быстрее всего процедура пройдет при выборе жесткого диска. Нажмите **Далее**.

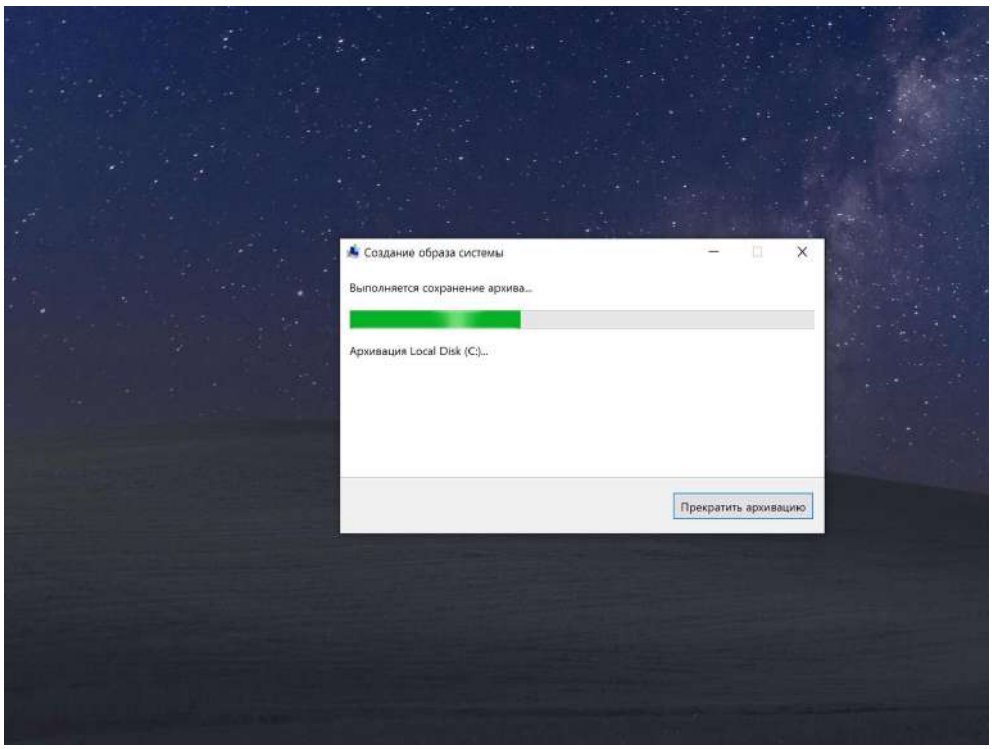
- Выберите диски, которые вы хотите включить в архивацию и нажмите **Далее**.



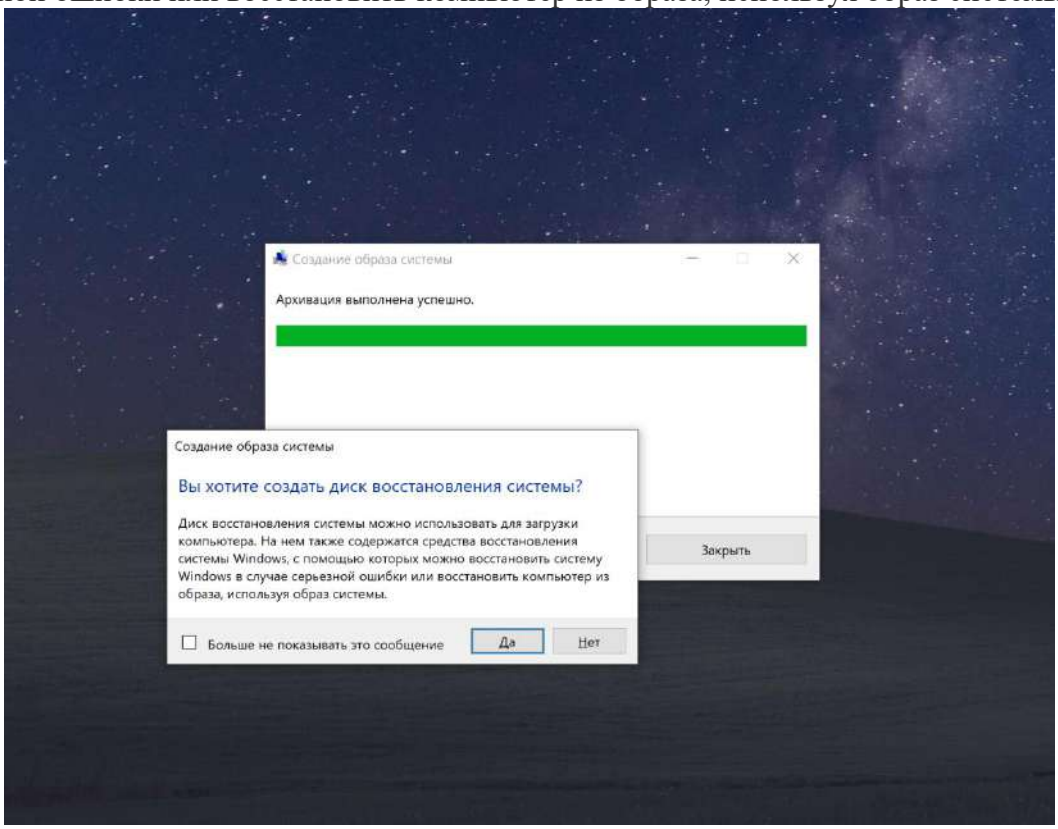
- Подтвердите запуск резервного копирования, нажав кнопку **Архивировать**.



- Процесс резервного копирования может занять от 10 минут до 2 часов – все зависит от количества архивируемых данных. Во время выполнения архивирования вы можете продолжать заниматься другими задачами, как при обычном использовании компьютера.



- После того, как утилита “Создание образа системы” завершит задание, вы можете создать **Диск восстановления системы**. Диск восстановления системы можно использовать для загрузки компьютера. На нем также содержатся средства восстановления системы Windows, с помощью которых можно восстановить систему Windows в случае серьезной ошибки или восстановить компьютер из образа, используя образ системы.



Положите портативный накопитель с образом системы и диск восстановления системы в безопасное место.

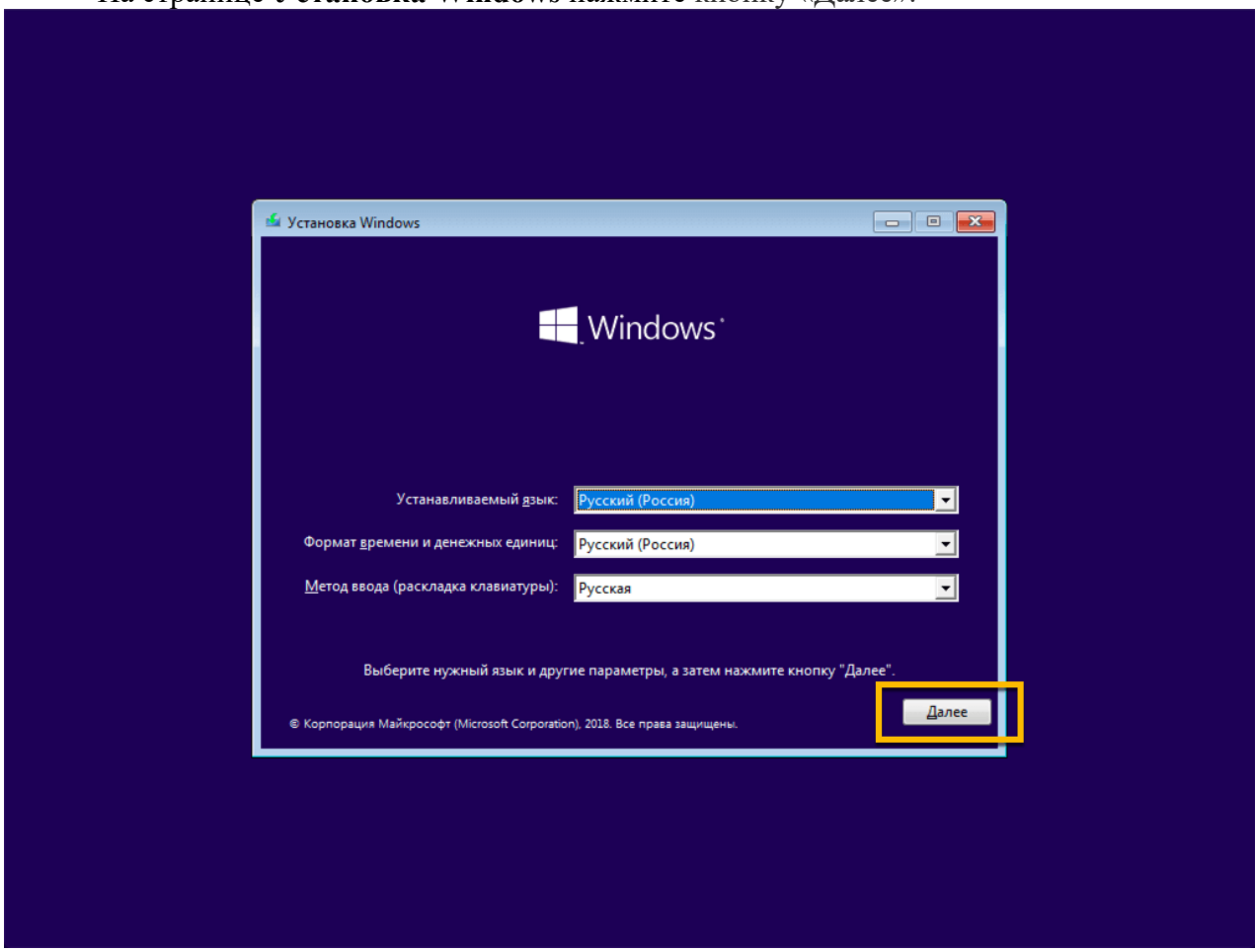
Восстановление компьютера из резервной копии

Если ваша система перестала запускаться или вы решили сменить основной жесткий диск на устройстве, то вы можете использовать следующие шаги, чтобы восстановить систему из резервной копии:

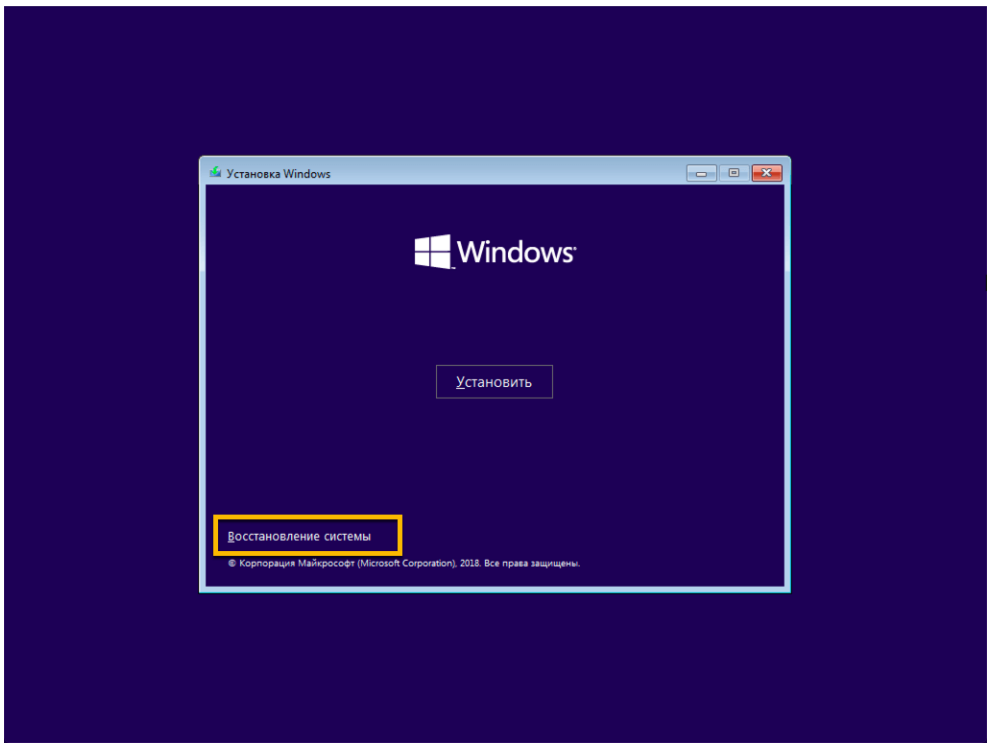
- Подключите к компьютеру диск восстановления системы или **установочный носитель Windows 10**.
- Подключите носитель, содержащий файлы восстановления системы (образ системы) к вашему устройству.
- Перезагрузите компьютер.

Совет: если мастер установки Windows не запускается автоматически, то вам следует изменить системные настройки BIOS. Обычно для перехода в BIOS нужно нажать одну из функциональных клавиш (F1, F2, F3, F10 или F12) или клавиши ESC и Delete. Горячие клавиши зависят от производителя устройства и даже от конкретной модели. Точную информацию можно узнать в разделе поддержки на официальном сайте производителя.

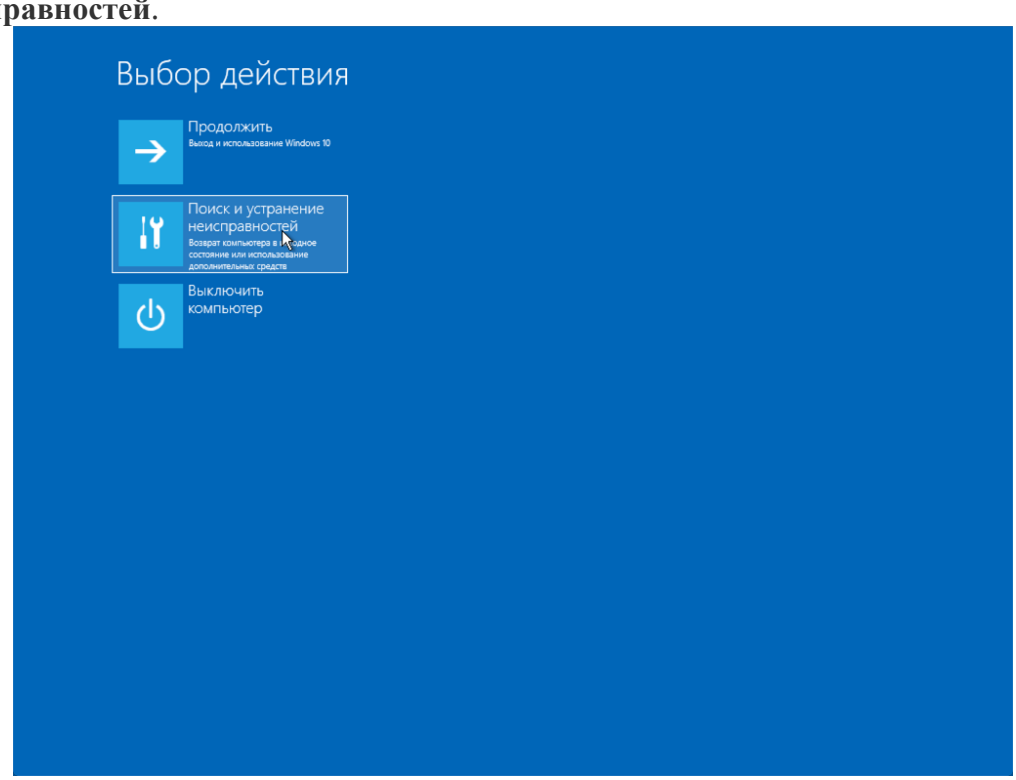
- На странице **Установка Windows** нажмите кнопку «Далее».



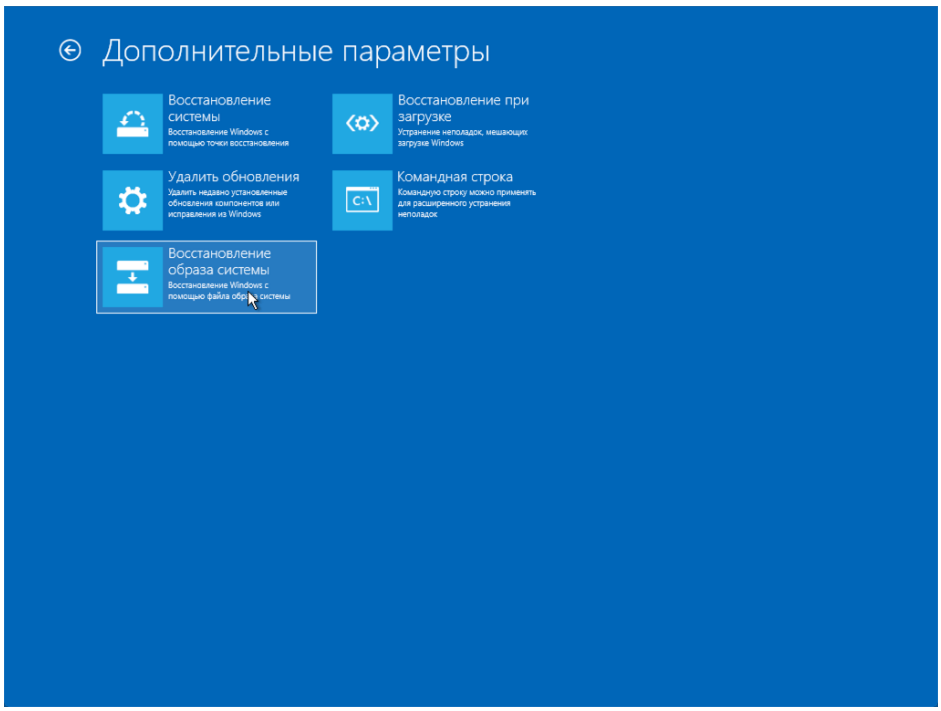
- Выберите ссылку **Восстановление системы** в нижнем-левом углу.



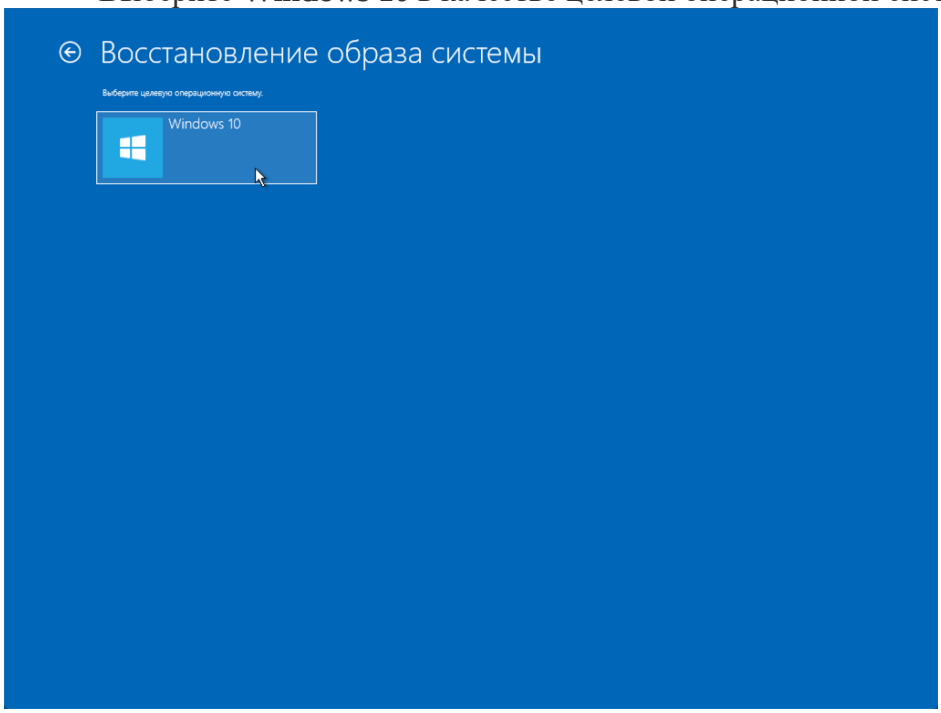
- На следующем экране выберите опцию **Поиск и устранение неисправностей**.



- На странице «Дополнительные параметры» выберите опцию **Восстановление образа системы**.



- Выберите **Windows 10** в качестве целевой операционной системы.



- В окне «Восстановление компьютера из образа» выберите опцию «Использовать последний доступный образ системы».

Совет: вы можете выбрать опцию «Выберите образ системы», если создали несколько системных резервных копий. В этом случае можно будет восстановить более старую копию.

- Нажмите кнопку «Далее».
- Если вы восстанавливаете полный системный образ на жесткий диск, вы можете также выбрать опцию «Форматировать и разбить на разделы диски» перед восстановлением резервной копии.
- Нажмите кнопку «Далее».
- Нажмите кнопку «Готово».
- В появившемся окне подтверждения выберите «Да».

После завершения данных шагов, начнется процесс восстановления системы. Время выполнения операции зависит от объема восстанавливаемых данных и характеристик вашего оборудования. Любые прерывания процесса могут нарушить работу компьютера и препятствовать его дальнейшей загрузке, поэтому убедитесь, что ваш компьютер подключен к ИБП. В случае с ноутбуком, перед запуском восстановления, убедитесь, что устройство подключено к сетевому источнику питания.

Так как с момента создания резервной копии могло пройти много времени, сразу после завершения восстановления рекомендуется перейти в **Параметры > Обновление и безопасность > Центр обновления Windows** и выбрать опцию «Проверить наличие обновлений» для скачивания и установки новейших патчей безопасности.

«Создание образа системы» – полезный инструмент во многих сценариях

Резервный образ системы может быть очень полезным во многих случаях. Например, если вы собираетесь обновиться до новой версии Windows 10, обязательно создайте копию системы, чтобы упростить процесс отката до предыдущей версии в случае серьезных ошибок и сбоев после обновления. Кроме того, рекомендуется регулярно создавать резервные копии в случае, если вам нужно быстро восстановить систему после аппаратного или системного отказа, вредоносного заражения или других проблем.

Тем не менее, нужно помнить об ограничениях и особенностях данного способа восстановления. Например, пользователь должен самостоятельно проявлять инициативу при создании резервных копий, поскольку восстановить состояние системы и файлы можно только на момент создания полной резервной копии. Любые данные, настройки и приложения, созданные, измененные или установленные после резервного копирования, не будут восстановлены в ходе этого процесса.

Данная функция, предназначенная для восстановления всей системы полностью, а не для восстановления отдельных файлов, настроек и приложений. Если вы хотите сохранить актуальную копию своих файлов, вам следует подумать о дополнительном использовании функций История файлов или «Защита важных файлов» сервиса OneDrive.

Создание образа системы является устаревшей функцией в Windows 10. Это означает, что, хотя вы все еще можете создавать резервные копии, функция может внезапно перестать работать или может быть удалена в будущих версиях.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Создайте образ системы.

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что представляет собой процедура резервного копирования?
2. Какие способы восстановления системы вам известны?

Практическая работа №10
«РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО СРЕДСТВА»

Цель работы: получить теоретические знания и практические навыки по разработке модулей программного средства и работы в команде над общим проектом.

Теоретические сведения

Каждый программный продукт состоит из модулей. Модуль может разрабатываться отдельно и, таким образом, модернизировать программное средство, улучшая его функциональность.

При разработке программного модуля целесообразно придерживаться следующего порядка:

- изучение и проверка спецификации модуля, выбор языка программирования;
- выбор алгоритма и структуры данных;
- программирование модуля;
- шлифовка текста модуля;
- проверка модуля;
- компиляция модуля.

Первый шаг разработки программного модуля в значительной степени представляет собой смежный контроль структуры программы снизу: изучая спецификацию модуля, разработчик должен убедиться, что она ему понятна и достаточна для разработки этого модуля. В завершении этого шага выбирается язык программирования: хотя язык программирования может быть уже предопределен для всего ПС, все же в ряде случаев (если система программирования это допускает) может быть выбран другой язык, более подходящий для реализации данного модуля (например, язык ассемблера).

На втором шаге разработки программного модуля необходимо выяснить, не известны ли уже какие-либо алгоритмы для решения поставленной и или близкой к ней задачи. И если найдется подходящий алгоритм, то целесообразно им воспользоваться. Выбор подходящих структур данных, которые будут использоваться при выполнении модулем своих функций, в значительной степени предопределяет логику и качественные показатели разрабатываемого модуля, поэтому его следует рассматривать как весьма ответственное решение.

На третьем шаге осуществляется построение текста модуля на выбранном языке программирования. Обилие всевозможных деталей, которые должны быть учтены при реализации функций, указанных в спецификации модуля, легко могут привести к созданию весьма запутанного текста, содержащего массу ошибок и неточностей. Искать ошибки в таком модуле и вносить в него требуемые изменения может оказаться весьма трудоемкой задачей. Поэтому весьма важно для построения текста модуля пользоваться технологически обоснованной и практически проверенной дисциплиной программирования. Впервые на это обратил внимание Дейкстра, сформулировав и обосновав основные принципы структурного программирования. На этих принципах базируются многие дисциплины программирования, широко применяемые на практике. Наиболее распространенной является дисциплина пошаговой детализации.

Следующий шаг разработки модуля связан с приведением текста модуля к завершенному виду в соответствии со спецификацией качества ПС. При программировании модуля разработчик основное внимание уделяет правильности реализации функций модуля, оставляя недоработанными комментарии и допуская некоторые нарушения требований к стилю программы. При шлифовке текста модуля он должен отредактировать имеющиеся в тексте комментарии и, возможно, включить в него дополнительные комментарии с целью обеспечить требуемые примитивы качества. С этой же целью

производится редактирование текста программы для выполнения стилистических требований.

Шаг проверки модуля представляет собой ручную проверку внутренней логики модуля до начала его отладки (использующей выполнение его на компьютере), реализует общий принцип, сформулированный для обсуждаемой технологии программирования, о необходимости контроля принимаемых решений на каждом этапе разработки ПС.

И, наконец, последний шаг разработки модуля означает завершение проверки модуля (с помощью компилятора) и переход к процессу отладки модуля.

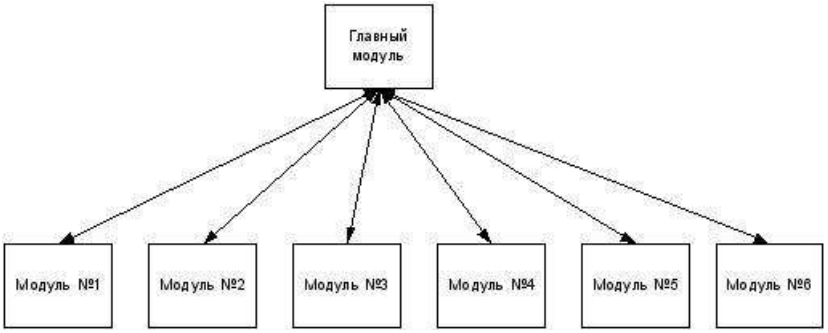


Рисунок 1. Схема подключения модулей

С помощью модулей решаются различные профессиональные задачи обработки данных разного типа.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Разделитесь в группе на 5 подгрупп. Каждая подгруппа должна разработать модуль программы «Инженерный калькулятор». Модули программы в каждой подгруппе пишутся на одном и том же языке программирования. Язык программирования группа выбирает совместно. После написания каждого модуля одна из подгрупп объединяет все части программы и производит отладку и тестирование.

Программа должна содержать следующие модули:

1) Интерфейс программы «Инженерный калькулятор».
2) Обычный пользовательский калькулятор со сложением, вычитанием, умножением, делением и подсчитыванием процентов.
3) Калькулятор, который выполняет тригонометрические вычисления.
4) Калькулятор для вычисления степени числа, извлечения корня, вычисления факториала.
5) Калькулятор для перевода числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную и наоборот.

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что является «модулем» ПО?
2. Что такое структурное программирование?
3. Каков порядок разработки программного модуля?
4. Что такое пошаговая детализация программного модуля?
5. Что называют модулем в контексте модульного программирования?
6. Сколько входов и выходов имеет один модуль?
7. Существуют ли типы данных, которые невозможно обработать при помощи

8. модулей?
9. Зависит ли результат работы модуля от работы других модулей?
10. Чем должны быть ограничены размер и сложность модуля?

Практическая работа №11
«НАСТРОЙКА СЕТЕВОГО ДОСТУПА»

Цель работы: получить теоретические знания и практические навыки по настройке сетевого доступа.

Теоретические сведения

Настройка сетевого доступа к дискам

Вы можете открыть пользователям локальной сети доступ к дискам вашего компьютера, что позволит им просматривать, редактировать и сохранять файлы на этих дисках, создавать и удалять папки, прослушивать хранящиеся на вашем компьютере аудиозаписи, устанавливать с вашего винчестера различные программы. Совместное использование дисковых ресурсов может быть необходимо, например, в случае, если только ваш компьютер во всей сети оснащен приводом CD-ROM или DVD.

Чтобы открыть пользователям локальной сети доступ к дисковым ресурсам вашего компьютера, необходимо проделать следующее:

- откройте системное окно Мой компьютер;
- щелкните правой кнопкой мыши на изображении диска, к которому вы хотите открыть доступ по сети, и выберите в появившемся меню пункт Свойства;
- в открывшемся окне Свойства: локальный диск перейдите ко вкладке Доступ и выберите пункт Если вы хотите открыть доступ к корневой папке диска, щелкните здесь (для MS Windows XP), в другой операционной системе семейства Windows достаточно установить переключатель в положение Общий ресурс;
- в разделе Сетевой совместный доступ и безопасность установите флажок рядом с пунктом Открыть общий доступ к этой папке и введите в поле Общий ресурс сетевое имя своего диска — оно будет отображаться в папке Сетевое окружение других пользователей локальной сети (рис. 1);
- если вы хотите открыть пользователям сети полный доступ к своему диску, то есть разрешить им создавать, удалять, перемещать и переименовывать файловые объекты на вашем винчестере, установите флажок рядом с пунктом Разрешить изменение файлов по сети. Если флажок сброшен, пользователи смогут обращаться к диску в режиме «только чтение»;
- щелкните на кнопке ОК, чтобы сохранить внесенные вами изменения. Диск, к которому открыт доступ из локальной сети, будет показан в папке Мой компьютер с помощью специальной метки в виде изображения открытой ладони.

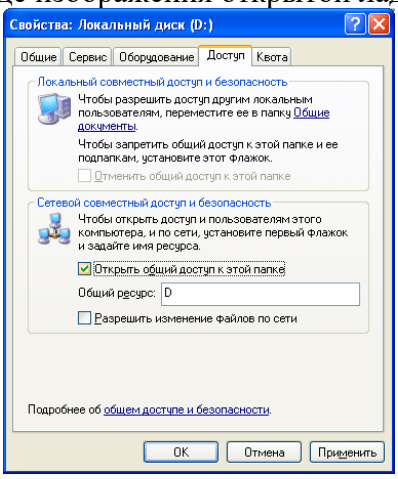


Рисунок 1. Настройка общего доступа к локальному ресурсу

В целях безопасности не рекомендуется открывать доступ к диску или логическому дисковому разделу, на котором установлена Microsoft Windows. Кто-либо из пользователей локальной сети может случайно или намеренно внести изменения в системные файлы, в результате чего Windows придет в неработоспособное состояние.

Управление сетевым доступом к папкам

Открытие сетевого доступа к дискам и дисковым разделам является потенциально опасным для хранящихся на винчестере данных, поскольку пользователь локальной сети может случайно или намерено уничтожить, переименовать или изменить файлы, предназначенные только для вашего личного пользования. С точки зрения безопасности лучше открыть доступ не к диску в целом, а к одной дисковой директории, предназначенной для совместного использования в локальной сети. Вы можете назначить такой папке произвольное сетевое имя, например, аналогичное системному имени дискового раздела, благодаря чему пользователям будет казаться, что они работают непосредственно с диском вашего компьютера, в то время как доступ к каким-либо ресурсам за пределами данной директории будет для них закрыт. Чтобы настроить сетевой доступ к какой-либо папке на жестком диске компьютера, необходимо проделать описанные ниже шаги.

- Перейдите на один из дисков своего компьютера и создайте папку с произвольным именем, которую вы хотите сделать доступной из локальной сети.
- Щелкните на значке папки правой кнопкой мыши и в появившемся меню выберите пункт Свойства.
- В открывшемся окне Свойства папки перейдите к вкладке Доступ.
- В разделе Сетевой совместный доступ и безопасность установите флажок рядом с пунктом Открыть общий доступ к этой папке и введите в поле Сетевой ресурс сетевое имя вашей папки. Оно может совпадать с именем вашего диска, например C, D, E или F, либо быть произвольным, например, Netfolder. Папка, сетевое имя которой совпадает с именем одного из дисковых разделов, фактически может находится на любом диске. Например, папка с сетевым именем C может храниться на диске D. Локальное и сетевое имя папки могут быть различными.
- Если вы хотите открыть пользователям сети полный доступ к данной папке, установите флажок рядом с пунктом Разрешить изменение файлов по сети. Если флажок сброшен, пользователи смогут обращаться к папке в режиме «только чтение».
- Щелкните на кнопке О К, чтобы сохранить внесенные вами изменения. Папка, к которой открыт сетевой доступ, будет отображаться в окне Проводника с помощью специальной метки в виде изображения открытой ладони.

Управление доступом к локальному принтеру

Вы можете открыть пользователям локальной сети доступ к принтеру, подключенному к вашему компьютеру, чтобы они могли печатать свои документы по сети. Для этого:

- перейдите в системную папку Принтеры и факсы, выполнив команды Пуск→
- Панель управления →Принтеры и другое оборудование→Принтеры и факсы; а щелкните на значке установленного в вашей системе принтера правой кнопкой мыши и выберите в появившемся меню пункт Свойства;
- перейдите к вкладке Доступ диалогового окна Свойства: Принтер, установите переключатель в положение Общий доступ к данному принтеру и введите в поле Сетевое имя произвольное сетевое имя принтера;
- щелкните на кнопке ОК, чтобы сохранить внесенные изменения. Принтер, к которому открыт сетевой доступ, будет отображаться в окне Принтеры и факсы с помощью специальной метки в виде изображения открытой ладони.

Подключение сетевого принтера

Если принтер подключен не к вашему, а к другому компьютеру локальной сети, вы можете использовать его для распечатки своих документов. Для этого:

- перейдите в системную папку Принтеры и факсы, выполнив команды Пуск→Панель управления→Принтеры и другое оборудование→Принтеры и факсы;
- щелкните на пункте Установка принтера в командном меню Задачи печати;
- в появившемся окне Мастера установки принтеров нажмите на кнопку Далее;
- в следующем окне Мастера установки принтеров выберите пункт Сетевой принтер, подключенный к другому компьютеру и снова нажмите Далее;
- в следующем окне установите переключатель в положение Обзор принтеров и щелкните на кнопке Далее;
- в предложенном списке принтеров, доступных в локальной сети, выберите нужный и снова нажмите Далее (рис. 2);
- если вы хотите сделать этот принтер используемым в вашей системе по умолчанию, установите в следующем окне переключатель в положение Да и щелкните на кнопке Далее;
- настройка сетевого принтера завершена. Нажмите на кнопку Готово, чтобы покинуть окно Мастера установки принтеров. Теперь все документы, распечатываемые вами из приложений Windows, будут направляться на этот принтер.
-

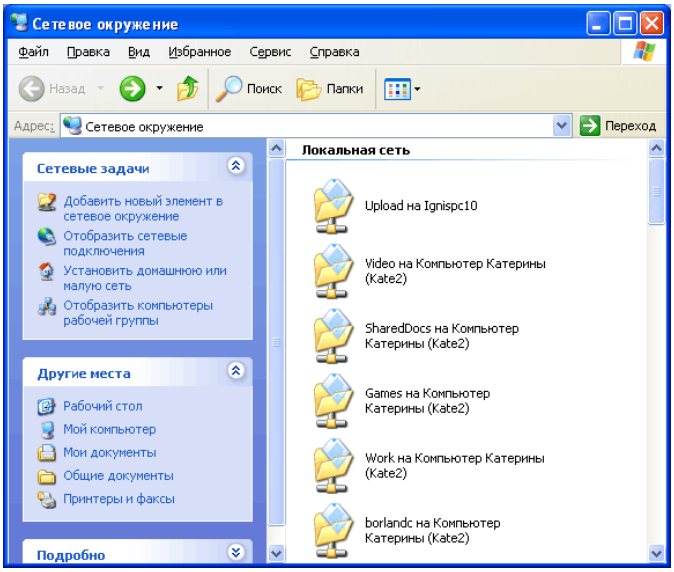


Рисунок 2. Выбор сетевого принтера из списка

Подключение сетевого диска

Некоторые программы MS Windows, работающие с файловыми ресурсами других сетевых компьютеров (например, сетевая версия бухгалтерского пакета «1С») требуют, чтобы физический диск или дисковый раздел удаленного компьютера был подключен к вашей системе как сетевой диск. Сетевые диски отображаются в системном окне Мой компьютер наравне с вашими локальными дисками, вы можете обращаться к ним и работать с их содержимым так же, как с содержимым собственного винчестера. Для того чтобы подключить к системе сетевой диск, необходимо выполнить следующие операции:

- щелкните правой кнопкой мыши на расположенном на Рабочем столе Windows значке Мой компьютер и выберите в появившемся меню пункт Подключить сетевой диск. На экране появится окно одноименного Мастера подключения сетевого диска;
- выберите в меню Диск символ, которым будет обозначаться подключаемый к вашей системе сетевой диск, затем щелкните на расположенной рядом кнопке Обзор;

- в открывшемся окне Обзор папки выберите из списка доступный для совместного использования диск удаленного компьютера и нажмите кнопку ОК.
- если вы хотите, чтобы соединение с данным сетевым диском автоматически восстанавливалось всякий раз при включении вашего компьютера, в окне Мастера подключения сетевого диска установите флажок рядом с функцией Восстанавливать при входе в систему. Щелкните на кнопке Готово.

Созданный вами сетевой диск будет обозначен в окне Мой компьютер выбранным вами символом и сетевым именем компьютера, которому фактически принадлежит.

Чтобы отключить сетевой диск, щелкните на его изображении в окне Мой компьютер правой кнопкой мыши и в появившемся контекстном меню выберите пункт Отключить.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Создать папку с общем доступом на своем ПК. Проверить работоспособность. (Перекинуть документы через сеть). Поставить на документы Доступ. Проверить работоспособность.

Задание 3. Подключить принтер через общий доступ. Вывести тестовую страницу принтера.

Задание 4. Отключить общий доступ к своей папке и общий доступ к принтеру преподавателя.

Задание 5. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каким образом внешний компьютер идентифицируется на вашем компьютере?
2. В каких случаях лучше использовать МАСТЕР НАСТРОЙКИ СЕТИ, а в каких лучше самостоятельно настроить?
3. Как осуществить доступ к Вашим каталогам с другого ПК?
4. Как настроить общий доступ к файлам и папкам?

Минобрнауки России
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ

по междисциплинарному курсу

МДК 04.02 Обеспечение качества функционирования компьютерных систем

профессионального модуля

ПМ.04 СОПРОВОЖДЕНИЕ И ОБСЛУЖИВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

специальности СПО

09.02.07 Информационные системы и программирование

Тула 2021

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «14» октября 2021 г. № 3

Председатель цикловой комиссии И.В. Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

Практическая работа №1
«ТЕСТИРОВАНИЕ ПРОГРАММНЫХ ПРОДУКТОВ»

Цель работы: получить навыки тестирования программного продукта и использования различных техник.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

По мнению опытных разработчиков, тестирование программного продукта должно проводиться прям с самого начала его создания. Но при этом, сами опытные разработчики в тестировании не должны принимать участия, так как не царское это дело. Тестировать программный продукт должны специально обученные сотрудники, называемые тестировщиками, ибо даже самый опытный разработчик не сможет увидеть свою ошибку, даже с использованием самых новейших оптических приборов.

Тем не менее, все разработчики сходятся во мнении, что тестирование программного продукта с точки зрения классификации по целям должно делиться на два класса:

- Функциональное тестирование
- Нефункциональное тестирование

Функциональное тестирование

Под функциональным тестированием понимается проверка соответствия программного продукта функциональным требованиям, указанным в техническом задании на создание это продукта. Если говорить проще, то при функциональном тестировании проверяется выполняет ли программный продукт все функции, которые должен.

Для проведения функционального тестирования персоналом отдела технического контроля разрабатывается документ программа и методика испытаний функционала приложения (ПМИ). Документ ПМИ содержит перечень сценариев тестирования программного продукта (test cases) с подробным описанием шагов. Каждый шаг сценария тестирования характеризуется действиями пользователя (специалиста по тестированию) и ожидаемыми результатами – ответной реакции программы на эти действия. Программа и методика испытаний обязана имитировать эксплуатацию программного продукта в реальном режиме. Это означает, что сценарий тестирования должен быть построен на основе анализа операций, которые будут выполнять будущие пользователи системы, а не быть искусственно составленной последовательностью понятных только разработчику манипуляций

Обычно, функциональное тестирование проводится на двух уровнях:

- Компонентное (модульное) тестирование. Тестирование отдельных компонентов программного продукта, сфокусированное на их специфике, назначении и функциональных особенностях.
- Интеграционное тестирование. Данный вид тестирования проводится после компонентного тестирования и направлен на выявление дефектов взаимодействия различных подсистем на уровне потоков управления и обмена данными.

Нефункциональное тестирование

Нефункциональное тестирование оценивает такие качества программного продукта, как, например, эргономику или производительность.

Думаю, важность данного вида тестирования понятна и не требует обоснования. Ведь всем понятно, что если, к примеру, производительность системы не достаточна, то пользователям придется по пол дня ждать отклика на свои действия, что может привести к их массовой спячке.

Как следует из названия, при нефункциональном тестировании проверяется соответствие программного продукта нефункциональным требованиям из технического задания на его создание. И, как в случае с функциональным тестированием, для нефункционально разрабатывается программа и методика испытаний.

Тестирование встроенного ПО и соблюдение стандартов в эру Agile

Соблюдение отраслевых стандартов – это не то, чем вы можете пренебречь или заняться позже; это неотъемлемая часть процесса разработки встроенного программного обеспечения (ПО). Традиционно, тестирование играет важную роль в разработке встраиваемых систем для регулируемых стандартами отраслей. Однако за последние годы устоявшиеся практики и процессы тестирования, их место и роль в подобных проектах значительно преобразились. Это резко изменило все правила игры, а когда правила игры меняются, необходимо меняться вместе с ними, чтобы выиграть.

В условиях постоянного развития новых, ультрасовременных технологий компаниям необходимо быстро предлагать рынку надежные, безопасные, простые в использовании и совместимые с другими системами продукты – просто чтобы не потеряться в быстро меняющемся технологическом мире. В такой ситуации традиционная каскадная модель, где процесс разработки ПО строго последователен и тестирование выполняется в самом его конце, уходит в прошлое. Большую популярность приобретают методы DevOps и Agile, поскольку они позволяют инженерам выполнять задачи, которые раньше следовали друг за другом, одновременно.

Тестирование производительности

В ходе этапа тестирования производительности в первую очередь проводят нагрузочное тестирование, целью которого является проверка, будет ли система адекватно реагировать на внешние воздействия в режиме, близком к режиму реальной эксплуатации.

Кроме нагрузочного тестирования проводят испытания в условиях минимальных аппаратных средств и максимальной нагрузки – стрессовое тестирование, а также, испытания в условиях предельных объемов обрабатываемой информации – объемное тестирование.

Выделяют еще один вид тестирования: тестирование стабильности и надежности, которое включает в себя не только длительное испытание программного продукта в нормальных условиях, но и способность его возвращаться в нормальный режим функционирования после непродолжительных периодов стрессовых нагрузок.

Документация для тестирования

Как уже было указано выше, тестирование проводится в соответствии с программой и методикой испытаний, которая разрабатывается в соответствии с ГОСТ 34.603-92.

Для проведения тестирования разрабатывается контрольный пример, который должен содержать достаточно данных для проверки всех режимов работы программного продукта. Обычно, контрольный пример создается совместно заказчиком и исполнителем на основе реальных данных.

Для проведения всех видов тестирования производительности чаще всего создается так называемый генератор данных, который позволяет в автоматическом режиме создать достаточное количество данных, для достижения объективного результата при оценке производительности.

В ходе проведения тестирования составляется протокол тестирования, куда заносится информация о прохождении всех этапов и шагов тестирования и замечаниях полученных на испытаниях.

Если результат тестирования отрицательный, проводится устранение недостатков и повторное тестирование.

Исследовательское тестирование

Исследовательское тестирование (ad hoc тестирование - подвид функционального тестирования. Оно применяется в быстрорастущих проектах с гибкими методиками разработки, где нет четкой документации и требований. Исследовательское тестирование - высший пилотаж в тестирование программного обеспечения. Качественное тестирование доступно специалистам с высшей квалификацией и практически полностью зависит от исполнителя, его опыта, знаний (как в предметной области, так и в методиках тестирования), способности быстро проникать в суть.

Нагрузочное тестирование

Нагрузочное тестирование - процесс анализа производительности тестируемой системы под воздействием нагрузок. Цель нагрузочного тестирования- определить способность приложения к внешним нагрузкам. Обычно испытания проводятся в несколько этапов.

1. Генерация тестовых сценариев

Для эффективного анализа сценарии должны быть наиболее близки к реальным сценариям использования. Важно понимать, что всегда возможны исключения, и даже самый подробный план тестирования может не покрыть отдельно взятого случая.

2. Разработка тестовой конфигурации

Имея сценарии тестирования, важно распределить порядок возрастания нагрузки. Для успешного анализа необходимо выделить критерии оценки производительности (скорость отклика, время обработки запроса и т.д.).

3. Проведение тестового испытания

При проведении тестов важно своевременно следить за исполнением сценариев и откликом тестируемой системы. Для эмуляции высоких нагрузок требуется серьезная аппаратная и программная инфраструктура. В некоторых случаях для удешевления работ применяются методы математического моделирования. За основу берутся данные, полученные при низких нагрузках, и аппроксимируются. Чем выше уровень моделируемой нагрузки, тем ниже точность оценки. Однако подобный способ существенно сокращает расходы.

Автоматизация тестирования

Основная особенность автоматизированного тестирования - возможность быстрого проведения регрессионных тестов. Главными плюсами автоматизации (по даннымотчета компании Worksoft) является увеличение эффективности персонала, более раннее обнаружение дефектов и более высокое качество бизнес-процессов.

Тестирование юзабилити

Любое приложение создается для того, чтобы им воспользовались. Удобство использования - важный качественный показатель программы. IT индустрия знает множество примеров, когда проекты взлетали после удачного исправления удобства использования. Чем шире аудитория, тем важнее фактор юзабилити. Тестирование юзабилити включает в себя детальный анализ поведения пользователей. Для оценки эргономики важно иметь данные не только о скорости выполнения бизнес-задачи, но и об эмоциях пользователя, мимике лица, тембра голоса.

Конфигурационное тестирование

Конфигурационное тестирование дает уверенность, что приложение заработает на разных платформах, а значит у максимального числа пользователей. Для ВЕБ-приложений обычно выбирают тестирование на кросс-браузерность. Для Windows приложений - тестирование на различных операционных системах и битностях (x86, x64). Важной составляющей конфигурационного тестирования является тестовая инфраструктура: для проведения испытаний нужно постоянно поддерживать парк тестовых машин. Их число варьируется от 5 до нескольких десятков.

Интеграционное тестирование

Если в вашем проекте более одной компоненты, он нуждается в интеграционном тестировании. При сложной архитектуре приложения необходимым условием обеспечения качества является проверка на взаимодействие частей программы. Тестирование достигается путем разработки и проведения "сквозных" кейсов. Интеграционное тестирование проводится после компонентного. Поэтому очень важно учитывать опыт компонентного тестирования, при этом соблюдая бизнес-ориентацию тест-кейсов.

Стресс тестирование

У любой системы есть предел нормального функционирования. При превышении предела система попадает в состояние стресса и значительно меняет свое поведение. Стресс тестирование проверяет работу приложения в условиях превышения предлов нормального функционирования. Особенно это важно для "критичных" программ: банковского ПО, программ авиационной отрасли, медицины. Стресс тестирование проводят не только на стадии разработки программного обеспечения, но и на протяжении всего цикла функционирования с целью получения и обработки данных поведения системы за долгий период времени.

Требования к этапу тестирования

Провести тестирование на всех трех уровнях тестирования (модульном, интеграционном, системном) в соответствии с целями тестирования:

1. Приемочное тестирование.
2. Установочное тестирование.
3. Альфа- и бета-тестирование.
4. Функциональные тесты/тесты соответствия.
5. Тестирование производительности.
6. Нагрузочное тестирование.
7. Конфигурационное тестирование.
8. Тестирование удобства и простоты использования.

Использовать 5 видов техник тестирования из представленных ниже:

- Специализированное тестирование.
- Таблицы принятия решений или тесты на основе конечного автомата.
- Тесты на основе потоков данных.
- Ссылочные модели для тестирования, ориентированного на код.
- Предположение ошибок.
- Операционный профиль.
- Объектно-ориентированное тестирование.
- Компонентно-ориентированное тестирование.
- Тестирование на соответствие протоколам.
- Тестирование систем реального времени.
- Функциональное и структурное.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.

2 Проведите тестирование программного обеспечения (разработанного ранее) в соответствии с предъявляемыми требованиями к этапу тестирования.

2. Оформите в виде отчета протокол проведенного тестирования.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие уровни тестирования вам известны?
2. Какое существует тестирование ПО?
3. Перечислите известные вам техники тестирования.

Практическая работа №2
«СРАВНЕНИЕ РЕЗУЛЬТАТОВ ТЕСТИРВОАНИЯ С ТРЕБОВАНИЯМИ ТЕХНИЧЕСКОГО ЗАДАНИЯ И/ИЛИ СПЕЦИФИКАЦИЕЙ»

Цель работы: получить практические навыки выполнения анализа технического задания и его соотношения с результатами тестирования.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Во-первых, техническое задание – это, как правило, основной документ в рамках проектной документации. Именно в ТЗ описываются все основные требования на разработку программного обеспечения, будь то создание либо простенькой программы или сайта, либо же разработка крупномасштабной информационной системы или программно-аппаратного комплекса. Причем, говоря языком ГОСТов, техническое задание может разрабатываться как в рамках эскизного проекта (это когда только описание функций и структуры системы без рассмотрения технологий реализации решения), так и в дальнейшем «перекочевать» в технический проект (более детальное описание с учетом выбранных технологий).

Во-вторых, техническое задание может быть как поверхностным (например, общеконцептуальное ТЗ, предназначенное для инвесторов проекта), так и более детальным (например, подробное ТЗ для программиста).

В-третьих, в некоторых случаях можно обойтись только подготовкой одного технического задания для описания разрабатываемой системы.

Тестирование – процесс, направленный на оценку корректности, полноты и качества разработанного программного обеспечения.

Техническое задание должно содержать следующие разделы:

- название программы и область применения;
- основание для разработки;
- назначение разработки;
- технические требования к программе или программному изделию;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- приложения.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Выполните анализ имеющегося технического задания на программный продукт. Определить корректность, полноту документа.
3. В соответствии с материалами практической работы №1 произвести сравнение результатов тестирования с требованиями технического задания.
4. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что представляет техническое задание?
2. Какие разделы ТЗ отражают процедуру тестирования?
3. Какой ГОСТ описывает содержание и структуру ТЗ на ПО?

Практическая работа №3
«АНАЛИЗ РИСКОВ»

Цель работы: получение практических навыков управления внедрением программных продуктов, управлением рисками при их внедрении.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Современные информационные системы представляют собой достаточно сложные и комплексные решения, и их внедрение, как правило, требует значительных инвестиций со стороны предприятия. ИТ системы - это комплекс программных и технических средств, которые используются для анализа, хранения и обработки данных. Средства анализа, входящие в ИТ системы, позволяют с высокой скоростью обрабатывать гигантские массивы данных и на основе полученных результатов предлагать корректирующие действия в соответствии со сложившейся ситуацией.

Перечислим несколько типичных причин возникновения рисков при реализации ИТ-проектов:

1. Неготовность топ-менеджмента к изменениям в бизнес-процессах предприятия и организационной структуры.
2. Незаинтересованность руководителей основных подразделений и их прямых подчиненных.
3. Смена в ходе реализации проекта менеджера проекта.
4. Недостаточная квалификация менеджера проекта и ответственных исполнителей.
5. Отсутствие ясных и четких методологических основ этого процесса.

Основываясь на перечисленных факторах, управление рисками проектов по внедрению информационных технологий (ИТ-проектов) заключается в том, чтобы заранее выявить все возможные риски и провести комплекс предупреждающих мероприятий для избежания серьезных проблем во время реализации проекта.

Проекты в специфических предметных областях, таких как ИТ или проекты, осуществляемые с применением узкоспециальных технологий, проекты для вертикальных рынков (таких как здравоохранение, высшее образование, правительственная деятельность, некоммерческие услуги и пр.), а также проекты со специфическим конечным продуктом могут содержать риски, уникальные для своей области. Например, в области информационной безопасности существуют риски, связанные с кражей, потерей или искажением информации в результате злоумышленного действия или случайного события. При работе над проектами в таких областях полезно изучить классификации этих специальных рисков или же расширить имеющиеся классификации рисков общего назначения.

Выбор классификации рисков будет зависеть от специфики ИТ-проекта и профессиональных предпочтений менеджера проекта. Оговоримся, что одни и те же риски могут немного отличаться по содержанию для разных видов деятельности и разных типов проектов. Но в основном риски ИТ-проекта можно классифицировать следующим образом:

1. **Технические риски**. Практически в любом ИТ-проекте существуют риски, связанные с техникой (отказ и сбои в работе оборудования, ошибки в монтаже и т.п.).
2. **Риски оценки сроков.** Для большинства ИТ-проектов (особенно в проектах по разработке и внедрению программного обеспечения) характерны ошибки в оценках сроков работ проекта.
3. **Интеграционные риски**. Интеграционные риски в ИТ-проектах, особенно в крупных компаниях, всегда высоки, поскольку любое ИТ решение должно быть интегрировано в существующую инфраструктуру. Наиболее характерны риски перехода на новую систему, которые включают в себя расходы на остановку предприятия во время внедрения ИТ решений, обучение персонала и т.д.
4. **Риски не принятия продукта проекта пользователями**. Любой проект, в т.ч. в ИТ сфере - это в первую очередь изменение технологии работы. Техническая составляющая любого проекта, безусловно важна, но не менее важна организационная часть.

5. **Коммерческие риски.** Это риски, связанные с выбором технологии и поставщика. Необходимо оценить успешность технологии на рынке, ее актуальность на протяжении жизненного цикла ИТ-проекта, доступность необходимого аппаратного и программного обеспечения, его качество, частоту модернизации.

6. **Риски не соблюдения технологии.** Эти риски возникают в случае, если менеджер проекта имеет единоличное решение по рискам (идентификация, анализ, выбор метода реагирования). Чем больше и сложнее проект, тем выше данный риск.

Говоря о проектах внедрения ИТ, нужно отметить, что любые новые технологии реализуются в условиях большой неопределенности и негативного воздействия окружающей среды. Это вызвано тем, что осуществление большинства ИТ-проектов, особенно крупных, происходит в условиях, когда трудно применить стандартные методы управления. Уникальность целей проекта и отсутствие подобных практик в компании порождает неопределенность относительно выбора новых технологий, определения методов и средств достижения поставленной цели, принятия той или иной методологии.

Управление рисками в современных преуспевающих организациях является тщательно планируемым процессом. Процесс управления рисками должен рассматриваться не как отдельно стоящая задача, требующая решения, а как часть изменения общей корпоративной системы управления. Целью управления рисками, в конечном счете, является повышение эффективности бизнеса за счет контроля деятельности компании и максимальная отдача от используемой методики.

Управление рисками ИТ-проектов - это определение, оценка и контроль эффекта, внутренних и внешних факторов, которые могут негативно повлиять на стоимость и процесс внедрения новых информационных технологий в компании. Таким образом, задачей управления рисками проектов ИТ является своевременное определение факторов, связанных с внедрением информационной системы или системы автоматизации, которые могут негативно повлиять на реализацию проекта внедрения, а также оптимальное планирование действий по минимизации этих факторов.

Управление рисками тесно связано с общим жизненным циклом проекта. На ранних этапах преобладают риски, связанные с бизнесом, рамками проекта, требованиями к конечному продукту и проектированием этого продукта. На стадии реализации доминируют технологические риски, далее возрастает роль рисков, связанных с поддержкой и сопровождением системы. На протяжении всего жизненного цикла проекта возникают новые риски, что требует проведения дополнительных операций анализа и планирования.

Тем не менее, анализ исследований в области управления рисками ИТ-проектов с учетом требований современной экономики позволяет нам выделить основные принципы управления:

- **Разбивать крупные проекты на более мелкие** (принцип «Дельфины вместо китов»). При этом обязательно должен быть единый человек (как правило директор программы), который одновременно управляет всеми проектами и добивается не локальных успехов, а реализацию решения в целом.
- **Привлекать для управления проектами профессионалов в управлении, а не узко технических специалистов**. Эти специалисты видят проект в первую очередь со своей технической точки зрения и забывают об единой управленческой составляющей.
- **Привлекать независимых (не включенных в проектную команду) экспертов для оценки рисков**. Если все решения по рискам проекта принимают только люди, которые изначально мотивированы на успех проекта, многие технические и технологические трудности они невольно могут рассматриваться как несущественные.
- **Учитывать риски связанные с организационной составляющей проекта**. Для успешной реализации проекта необходимо большое количество согласований и соблюдения формальностей. Следует детально продумать систему организации и протоколирования проектных встреч, согласования документов, принятия результатов, обучение пользователей и т.п.

Целью управления рисками проекта является повышение вероятности реализации и значимости позитивных событий и снижение вероятности реализации событий, негативных для целей проекта.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Составьте подробное описание информационной системы (получите задание у преподавателя).
3. На основании описания системы провести анализ осуществимости. В ходе анализа ответить на вопросы:

- Что произойдет с организацией, если система не будет введена в эксплуатацию?
- Какие текущие проблемы существуют в организации и как новая система поможет их решить?
- Каким образом система будет способствовать целям бизнеса?
- Требует ли разработка системы технологии, которая до этого использовалась в организации?

4. Разделиться на подгруппы и распределить роли (руководитель проекта-разработчик, системный аналитик-разработчик, тестер-разработчик).
5. Заполнить разделы плана:

- Введение
- Организация выполнения проекта
- Анализ рисков

6. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие риски возможны при внедрении программного продукта?
2. Дайте классификацию рисков ИТ-проекта.

Практическая работа №4
«ВЫЯВЛЕНИЕ ПЕРВИЧНЫХ И ВТОРИЧНЫХ ОШИБОК»

Цель работы: обнаружение первичных и вторичных ошибок программного средства.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Важной особенностью процесса выявления ошибок в программах является отсутствие полностью определенной программы-эталона, которой должны соответствовать текст и результаты функционирования разрабатываемой программы. Поэтому установить наличие и локализовать дефект непосредственным сравнением с программой без ошибок в большинстве случаев невозможно. При отладке и тестировании обычно сначала обнаруживаются вторичные ошибки и риски, т. е. последствия и результаты проявления некоторых внутренних дефектов или некорректностей программ. Эти внутренние дефекты следует квалифицировать как первичные ошибки или причины обнаруженных аномалий результатов. Последующая локализация и корректировка таких первичных ошибок должна приводить к устранению ошибок, первоначально обнаруживаемых в результатах функционирования программ.

Первичные ошибки в программах проектов можно анализировать с разной степенью детализации и в зависимости от различных факторов. Практический опыт показал, что наиболее существенными факторами, влияющими на характеристики обнаруживаемых ошибок являются:

- методология, технология и уровень автоматизации системного и структурного проектирования ПС, а также непосредственного программирования компонентов;
- длительность с начала процесса тестирования и текущий этап разработки или сопровождения, и модификации комплекса программ;
- класс ПС, масштаб (размер) и типы компонентов, в которых обнаруживаются ошибки;
- методы, виды и уровень автоматизации верификации и тестирования, их адекватность характеристикам компонентов и потенциально возможным в программах ошибкам;
- виды и достоверность эталонов-тестов, которые используются для обнаружения ошибок.

Первичные ошибки в ПС в порядке уменьшения их влияния на сложность обнаружения и масштабы корректировок можно разделить на следующие группы:

- ошибки, обусловленные сложностью компонентов и ПС в целом и наиболее сильно влияющие на размеры модификаций;
- ошибки вследствие большого масштаба – размера комплекса программ, а также высоких требований к его качеству;
- ошибки планирования и корректности требований модификаций часто могут быть наиболее критичным для общего успеха ЖЦ ПС и системы;
- ошибки проектирования, разработки структуры и функций ПС в более полные и точные технические описания сценариев того, как комплекс программ и система будут функционировать;
- системные ошибки, обусловленные отклонением функционирования ПС в реальной системе, и характеристик внешних объектов от предполагавшихся при проектировании;
- алгоритмические ошибки, связанные с неполным формированием необходимых условий решения и некорректной постановкой целей функциональных задач;
- ошибки реализации спецификаций изменений – программные дефекты, возможно, ошибки нарушения требований или структуры компонентов ПС;
- программные ошибки, вследствие неправильной записи текстов программ на языке программирования и ошибок трансляции текстов изменений программ в объектный код;
- ошибки в документации, которые наиболее легко обнаруживаются и в наименьшей степени влияют на функционирование и применение версий ПС;
- технологические ошибки подготовки физических носителей и документации, а также ввода программ в память ЭВМ и вывода результатов на средства отображения.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.

2. Выявите первичные и вторичные ошибки программного продукта (задание получите у преподавателя) в соответствии с параметрами, указанными в теоретических сведениях к работе.

3. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что представляют собой первичные ошибки?
2. Что представляют собой вторичные ошибки?

Практическая работа №5
«ОБНАРУЖЕНИЕ ВИРУСА И УСТРАНЕНИЕ ПОСЛЕДСТВИЙ ЕГО ВЛИЯНИЯ»

Цель работы: получение практических навыков проведения сравнительного анализа вирусных программ.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Термин «вредоносное ПО» используется для описания любой вредоносной программы на компьютере или мобильном устройстве. Эти программы устанавливаются без согласия пользователей и могут вызывать ряд неприятных последствий, таких как снижение производительности компьютера, извлечение из системы персональных данных пользователя, удаление данных или даже воздействие на работу аппаратных средств компьютера. Поскольку киберпреступники придумывают все более сложные способы проникновения в системы пользователей, рынок вредоносных программ существенно расширился. Давайте рассмотрим некоторые из наиболее распространенных типов вредоносных программ, которые можно встретить в интернете.

1. Вирусы

Компьютерные вирусы получили свое название за способность «заражать» множество файлов на компьютере. Они распространяются и на другие машины, когда зараженные файлы отправляются по электронной почте или переносятся пользователями на физических носителях, например, на USB-накопителях.

2. Черви

В отличие от вирусов, червям для распространения не требуются вмешательства человека: они заражают один компьютер, а затем через компьютерные сети распространяются на другие машины без участия их владельцев. Используя уязвимости сети, например, недостатки в почтовых программах, черви могут отправлять тысячи своих копий и заражать все новые системы, и затем процесс начинается снова. Помимо того, что многие черви просто «съедают» системные ресурсы, снижая тем самым производительность компьютера, большинство из них теперь содержит вредоносные «составляющие», предназначенные для кражи или удаления файлов.

3. Рекламное ПО

Одним из наиболее распространенных типов вредоносных программ является рекламное ПО. Программы автоматически доставляют рекламные объявления на хост-компьютеры. Среди разновидностей Adware - всплывающие рекламные объявления на веб-страницах и реклама, входящая в состав «бесплатного» ПО. Некоторые рекламные программы относительно безвредны, в других используются инструменты отслеживания для сбора информации о вашем местонахождении или истории посещения сайтов и вывода целевых объявлений на экран вашего компьютера.

4. Шпионское ПО

Шпионское ПО делает то, что предполагает его название - следит за вашими действиями на компьютере. Оно собирает информацию (например, регистрирует нажатия клавиш на клавиатуре вашего компьютера, отслеживает, какие сайты вы посещаете и даже перехватывает ваши регистрационные данные), которая затем отправляется третьим лицам, как правило, киберпреступникам. Оно также может изменять определенные параметры защиты на вашем компьютере или препятствовать сетевым соединениям.

5. Программы-вымогатели

Программы-вымогатели заражают ваш компьютер, затем шифруют конфиденциальные данные, например, личные документы или фотографии, и требуют выкуп за их расшифровку. Если вы отказываетесь платить, данные удаляются. Некоторые типы программ-вымогателей могут полностью заблокировать доступ к вашему компьютеру. Они могут выдавать свои действия за работу правоохранительных органов и обвинить вас в каких-либо противоправных поступках.

6. Боты

Боты - это программы, предназначенные для автоматического выполнения определенных операций. Они могут использоваться для легитимных целей, но злоумышленники приспособили их для своих вредоносных целей. Проникнув в компьютер, боты могут заставить его выполнять определенные команды без одобрения или вообще без ведома пользователя.

7. Руткиты

Руткиты позволяют третьей стороне получать удаленный доступ к компьютеру и управлять им. Эти программы используются IT-специалистами для дистанционного устранения сетевых проблем. Но в руках злоумышленников они превращаются в инструмент мошенничества: проникнув в ваш компьютер, руткиты обеспечивают киберпреступникам возможность получить контроль над ним и похитить ваши данные или установить другие вредоносные программы. Руткиты умеют качественно маскировать свое присутствие в системе, чтобы оставаться незамеченными как можно дольше. Обнаружение такого вредоносного кода требует ручного мониторинга необычного поведения, а также регулярного внесения корректировок в программное обеспечение и операционную систему для исключения потенциальных маршрутов заражения.

8. Троянские программы

Более известные как троянцы, эти программы маскируются под легитимные файлы или ПО. После скачивания и установки они вносят изменения в систему и осуществляют вредоносную деятельность без ведома или согласия жертвы.

9. Баги

Баги - ошибки в фрагментах программного кода - это не тип вредоносного ПО, а именно ошибки, допущенные программистом. Они могут иметь пагубные последствия для вашего компьютера, такие как остановка, сбой или снижение производительности. В то же время баги в системе безопасности - это легкий способ для злоумышленников обойти защиту и заразить вашу машину. Обеспечение более эффективного контроля безопасности на стороне разработчика помогает устранить ошибки, но важно также регулярного проводить программные корректировки, направленные на устранение конкретных багов.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Изучите существующие вирусные программы, их воздействия и последствия.
3. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое вирус?
2. Какие разновидности вирусов вы знаете?
3. Какие воздействия они оказывают?
4. Какие последствия могут быть?

Практическая работа №6
«УСТАНОВКА И НАСТРОЙКА АНТИВИРУСА. НАСТРОЙКА ОБНОВЛЕНИЙ С ПОМОЩЬЮ ЗЕРКАЛА»

Цель работы: получение практических навыков установки и настройки антивируса, настройки обновлений с помощью зеркала.

Краткие теоретические сведения

Основа антивирусной защиты компьютера - это использование надежной антивирусной программы. Антивирусные программы бывают разные - от простейших приложений для мобильных телефонов до корпоративных продуктов, обеспечивающие безопасность больших гетерогенных сетей. К каждому виду антивирусов, кроме общих надежности и незаметности для пользователя, предъявляются свои требования: в одном случае на первое место выдвигается необходимость работать с очень ограниченными системными ресурсами (мобильный телефон), в другом - оперировать с огромными базами данных, разрешать удаленное централизованное управление и предоставлять подробную статистику о вирусной ситуации в большой сети.

При создании любого приложения программисты дают гарантию, что их продукт будет работать на технике с определенными характеристиками: например, для работы браузера **Internet Explorer** необходимо наличие на компьютере установленной операционной системы семейства **Microsoft Windows**, на **Linux** и любой другой ***nix** -операционной системе он работать не будет. Это требования к программному обеспечению. Бывают также требования к аппаратному обеспечению - в этом случае постулируется необходимость наличия на компьютере некоторого минимального объема оперативной памяти (если ее меньше, то *программа* будет очень медленно работать или же не запустится вовсе), свободного пространства на диске (для размещения всех необходимых в работе приложения файлов), тактовой частоты процессора, от которой зависит *производительность* компьютера и другое.

В случае антивирусных программ часто выдвигается дополнительное требование отсутствия на компьютере другого антивирусного средства, *совместная работа* с которым может вызвать конфликты.

Системные требования обычно приводятся в сопровождающем *дистрибутив* текстовом файле и/или в документации к продукту. Также всегда с ними можно ознакомиться на сайте компании-производителя.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Изучите современные инструменты для устранения последствий воздействия вирусов.
3. Установите любой на выбор антивирус (не самой последней версии), выполните его обновление с помощью зеркала.
4. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие требования необходимо учитывать при установке антивируса?
2. Какие существуют современные антивирусные средства для ПК?
3. Назовите основные этапы установки антивируса.

Практическая работа №7
«НАСТРОЙКА ПОЛИТИКИ БЕЗОПАСНОСТИ»

Цель практической работы: приобретение практических навыков в области политики безопасности.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Политика безопасности системы является одной из важнейших составляющих в обеспечении надежной и защищенной работы Windows. Настройка политики безопасности осуществляется в программе Local Security Settings: Пуск\Панель управления\Администрирование\Локальная политика безопасности\Назначение прав пользователя.

Windows защищает ценные данные с помощью уровней безопасности приложений.

В следующей таблице обобщены Windows и возможности безопасности для приложений:

Меры безопасности	Возможности & возможности
Управление приложениями в Защитнике Windows	Управление приложениями является одним из наиболее эффективных элементов управления безопасностью для предотвращения запуска нежелательного или вредоносного кода. Он перемещается от модели доверия приложения, где предполагается, что все коды являются надежными, к модели, в которой приложениям необходимо заработать доверие для запуска. Дополнительные. Управление приложениями для Windows
Application Guard в Microsoft Defender	Application Guard использует аппаратную изоляцию на основе чипов, чтобы изолировать ненастоячивые веб-сайты и ненастоячивые Office файлы, беспрепятственно запускать ненастоячивые веб-сайты и файлы в изолированном контейнере на Hyper-V, отдельно от операционной системы настольных компьютеров, и убедиться, что все, что происходит в контейнере, остается изолированным от рабочего стола. Дополнительные Application Guard в Microsoft Defender обзор.
Песочница Windows	Windows "Песочница" обеспечивает легкую среду рабочего стола для безопасного запуска приложений в изоляции. Программное обеспечение, установленное Windows среде песочницы, остается "песочницой" и выполняется отдельно от хост-машины. Песочница временная. Когда он закрыт, все программное обеспечение и файлы и состояние удаляются. Каждый раз при открываемом приложении вы получаете совершенно новый экземпляр песочницы. Дополнительные дополнительные Windows песочницы
Безопасность электронной почты	Благодаря Windows безопасности электронной почты S/MIME пользователи могут шифровать исходяющие сообщения и вложения, поэтому их могут читать только предполагаемые получатели с помощью цифровой идентификации (ID),также называемой сертификатом. Пользователи могут в цифровом формате подписать сообщение, которое проверяет удостоверение отправитель и гарантирует, что сообщение не было подделано. Настройка S/MIME для Windows 10
Фильтр SmartScreen в Microsoft Defender	Фильтр SmartScreen в Microsoft Defender защищает от фишинговых и вредоносных веб-сайтов и приложений, а также от скачивания потенциально опасных файлов. Дополнительные сведения: фильтр SmartScreen в Microsoft Defender обзор

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Произведите настройку Политики безопасности на своем ПК.
3. Произведите настройку Параметров безопасности на своем ПК.
4. Произведите настройку Политики обновления на своем ПК.
5. Составьте подробный отчет по выполненной работе.

Контрольные вопросы

1. Определите назначение политики безопасности системы.
2. Где производится настройка политики безопасности системы?
3. Как запретить доступ сетевых пользователей к компьютеру?

Практическая работа №8
«НАСТРОЙКА БРАУЗЕРА»

Цель работы: получить практические навыки настройки различных браузеров.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Бра́узер, или **веб-обозрева́тель** (от англ. *web browser*, МФА: [wɛb ˈbraʊ.zə(ɹ), -zɚ]; устар. **бро́узер**) — прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач. В глобальной сети браузеры используют для запроса, обработки, манипулирования и отображения содержания веб-сайтов. Многие современные браузеры также могут использоваться для обмена файлами с серверами FTP, а также для непосредственного просмотра содержания файлов многих графических форматов (gif, jpeg, png, svg), аудио- и видеоформатов (mp3, mpeg), текстовых форматов (pdf, djvu) и других файлов.

Функциональные возможности браузеров постоянно расширяются и улучшаются благодаря конкуренции между их разработчиками и высоким темпам развития и внедрения информационных технологий. Несмотря на то, что браузеры разных изготовителей базируются на разных технологических решениях, большинство современных браузеров придерживается международных стандартов и рекомендаций W3C в области обработки и отображения данных. Стандартизация позволяет добиться предсказуемости в визуальном представлении информации конечному пользователю независимо от технологии, которая использована для её отображения в браузере. Со времени начала применения браузеров во Всемирной паутине в начале 1990-х годов из простого средства просмотра текстовой информации браузер превратился в комплексное прикладное программное обеспечение для обработки данных и обеспечения интерфейса между информационными ресурсами и человеком. В последние годы многие разработчики браузеров сосредоточили свои усилия на повышении удобства пользовательского интерфейса браузеров для их использования в аппаратных устройствах, в которых применяются сенсорные экраны.

Браузеры распространяются, как правило, бесплатно. Потребителям браузер может быть поставлен в форме самостоятельного (автономного) приложения или в составе комплектного программного обеспечения. К примеру, браузеры Internet Explorer и Microsoft Edge поставляются в составе операционной системы Microsoft Windows; Mozilla Firefox — отдельно или в составе дистрибутивов Linux (например, Ubuntu); Safari — в составе операционной системы Mac OS X; Google Chrome, Opera и другие браузеры — как самостоятельные приложения во множестве вариантов для различных операционных систем.

Все браузеры позволяют выполнить некоторые настройки для оптимизации работы пользователей в Интернете.

В браузере Internet Explorer основная часть настроек содержится в меню Сервис – Свойства обозревателя. Вкладка «Общие» позволяет задать адрес домашней страницы, которая будет автоматически загружаться в окно браузера при его запуске, цвета гиперссылок по умолчанию, название шрифта по умолчанию. Здесь же определяется, сколько дней будет храниться ссылка посещенных страниц в журнале. Кроме того, для ускорения просмотра. Все посещенные страницы помещаются в специальную папку, и с помощью кнопки «Параметры» можно задать разные способы обновления таких страниц. С помощью вкладки «Безопасность» можно создать списки надежных узлов и узлов с ограниченными функциями. Зона Интернет будет при этом включать все остальные узлы, не вошедшие в эти две папки. Для каждой из них с помощью кнопки Другой можно изменить параметры безопасности, установленные для них по умолчанию. Здесь можно запретить выполнение сценариев, отображение всплывающих окон, загрузку файлов и т.д. Вкладка «Конфиденциальность» дает возможность настроить работу с файлами cookie, с помощью которых информация о пользователе автоматически передается на сервер. Вкладка «Содержание» позволяет ограничить доступ к некоторой информации (насилие, ненормативная лексика и т.д.). Вкладка «Подключения» позволяет установить подключение к Интернету. На вкладке «Дополнительно» можно задать

некоторые дополнительные параметры работы (отключить загрузку графических изображений, отменить подчеркивание ссылок, запретить отладку сценариев и т.д.). Вкладка «Программы» позволяет определить программы, которые будут по умолчанию использоваться службами Интернета (почтовые программы, html-редакторы и т.п.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Для различных браузеров (Mozilla Firefox , Internet Explorer, Chrome и др.): установите начальную веб-страницу, настройте кэш-память браузера, установите правильную кодировку для отображения веб-страниц.
3. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Порядок настройки браузеров.
2. Настройка домашней страницы.
3. Настройка подключения к сети Интернет.
4. Настройка безопасности работы в Интернете.
5. Настройка дополнительных параметров браузера.

Практическая работа №9
«РАБОТА С РЕЕСТРОМ»

Цель работы: ознакомиться с системным с реестром ОС Windows и получить практические навыки по работе с ним.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Реестр является одной из важнейших составляющих операционной системы Windows, и неправильные действия с реестром могут причинить серьёзный ущерб работе операционной системы. Всегда перед началом работы создавайте резервную копию реестра (ветви, с которой работаете).

Реестр Windows (системный реестр) - это иерархическая (древовидная) база данных, содержащая записи, определяющие параметры и настройки операционных систем Microsoft Windows. Реестр в том виде, как он выглядит при просмотре редактором реестра, формируется из данных, источниками которых являются файлы реестра и информация об оборудовании, собираемая в процессе загрузки. В описании файлов реестра на английском языке используется термин "Hive". В некоторых работах его переводят на русский язык как "Улей". В документации от Microsoft этот термин переводится как "Куст".

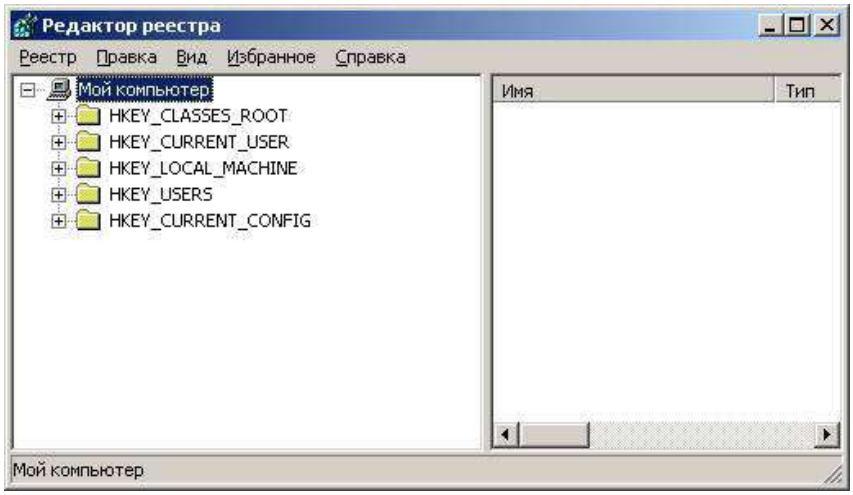
Файлы реестра создаются в процессе установки операционной системы и хранятся в папке **%SystemRoot%\system32\config** (обычно C:\windows\system32\config). Для операционных систем Windows это файлы с именами:

- **default**
- **sam**
- **ecurity**
- **software**
- **system**

В операционных системах Windows Vista/Windows 7/8/10, файлы реестра располагаются также в каталоге \Windows\system32\config и имеют такие же имена, однако в этих ОС добавился новый раздел реестра для хранения данных конфигурации загрузки (Boot Configuration Data) с именем **BCD00000000** . Файл с данными этого раздела имеет имя **bcd** и находится в скрытой папке **Boot** активного раздела (раздела, с которого выполняется загрузка системы). Обычно, при стандартной установке Windows 7, создается активный раздел небольшого размера (около 100 мегабайт), который скрыт от пользователя и содержит только служебные данные для загрузки системы – загрузочные записи, менеджер загрузки **bootmgr**, хранилище конфигурации загрузки BCD, файлы локализации и программы тестирования памяти . Расположение куста **bcd** зависит от того, как сконфигурирован загрузчик системы при ее установке, и может находиться на том же разделе, где и каталог Windows.

Место расположения файлов реестра в любой версии Windows можно просмотреть с помощью редактора реестра. В разделе ***HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\hivelist*** *хранится информация о всех кустах, включая пользовательские профили, со ссылками на их расположение в файловой системе Windows.*

В процессе загрузки система получает монопольный доступ к файлам реестра и, поэтому, их невозможно открыть для просмотра, скопировать, удалить или переименовать обычным образом. Для работы с содержимым системного реестра используется специальное программное обеспечение - редакторы реестра (REGEDIT.EXE, REGEDT32.EXE), являющиеся стандартными компонентами операционной системы. Для запуска редактора реестра можно использовать меню кнопки "Пуск"- "Выполнить" - regedit.exe



После старта редактора, в левой части основного окна вы видите список ***корневых разделов (root keys)*** реестра. Каждый корневой раздел может включать в себя ***вложенные разделы (subkeys)*** и ***параметры (value entries) или ключи реестра.*** Основное назначение корневых разделов: **HKEY_CLASSES_ROOT (** Общепринятое сокращенное обозначение **HKCR)** - Ассоциации между приложениями и расширениями файлов и информацию о зарегистрированных объектах COM и ActiveX. **HKEY_CURRENT_USER (HKCU)**- Настройки для текущего пользователя (рабочий стол, личные папки, настройки приложений). Этот раздел представляет собой ссылку на раздел HKEY_USERS\Идентификатор пользователя (SID) в виде S-1-5-21-854245398 -1035525444-...

SID - это уникальный номер, идентифицирующий учетную запись пользователя, группы или компьютера. Он присваивается учетной записи при создании каждого нового пользователя системы. Внутренние процессы Windows обращаются к учетным записям по их кодам SID, а не по именам пользователей или групп. Если удалить, а затем снова создать учетную запись с тем же самым именем пользователя, то предоставленные прежней учетной записи права и разрешения не сохранятся для новой учетной записи, так как их коды безопасности будут разными. Аббревиатура SID образована от Security ID.

Идентификатор SID представляет собой числовое значение переменной длины, формируемое из номера версии структуры SID, 48-битного кода агента идентификатора и переменного количества 32-битных кодов субагентов и/или относительных идентификаторов (Relative IDentifiers, RID). Код агента идентификатора определяет агент, выдавший SID, и обычно таким агентом является локальная операционная система или домен под управлением Windows. Коды субагентов идентифицируют попечителей, уполномоченных агентом, который выдал SID, а RID - дополнительный код для создания уникальных SID на основе общего базового SID.
Для идентификатора S-1-5-21-854245398 -1035525444: 1000, номер версии равен 1, код агента идентификатора - 5, а далее следуют коды четырех субагентов. В Windows NT и старше, при установке системы, создается один фиксированный (код 21) и три генерируемых случайным образом (числа после "S-1-5-21") кода субагентов. Также в процессе установки создаются некоторые (одинаковые для всех систем) учетные записи, как например, учетная запись администратора, которая всегда имеет RID равный 500

В процессе загрузки и функционирования операционной системы выполняется постоянное обращение к данным реестра, как для чтения, так и для записи. Файлы реестра постоянно изменяются, поскольку не только система, но и отдельные приложения могут использовать реестр для хранения собственных данных, параметров и настроек. Другими словами, обращение к реестру - это одна из наиболее распространенных операций. Даже если пользователь не работает за компьютером, обращения к реестру все равно выполняются системными службами, драйверами и приложениями.

Нарушение целостности файлов реестра (нарушение структуры данных) или неверное значение отдельных критических параметров может привести к краху системы . Поэтому,

прежде чем экспериментировать с реестром, позаботьтесь о возможности его сохранения и восстановления.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.

I. Отключить службу индексирования

1. Откройте окно Мой компьютер
2. Вызовите окно свойств жесткого (логического) диска
3. Снимите флажок Разрешить индексирование диска для быстрого поиска
4. Нажмите кнопку Применить и в новом окне установите переключатель в положение Применить ко всем вложенным файлам и папкам
5. Дождитесь завершение процесса применения новых атрибутов ко всем вложенным файлам и папкам. Будьте готовы, что он может занять некоторое время.
6. Повторите эти же действия для всех остальных дисков.
7. Подготовьте отчет о проделанной работе.

II. Отключить визуальных эффектов

1. Вызовите окно Свойства системы и перейти в нем на вкладку Дополнительно. Здесь нажмите в области быстродействие на кнопку параметры. Откроется окно Параметры быстродействия.
2. Установите положение Обеспечить наилучшее быстродействие сделает картинку намного скромнее, его производительность системы при этом резко возрастет.
3. С помощью меню особые эффекты в индивидуальном порядке поработайте с различными типами визуального эффекта.
4. Верните состояние системы в исходное положение, установив переключатель в положение Восстановить значения по умолчанию.
5. Подготовьте отчет о проделанной работе.

III. *Обслуживание дисков*

1. Вызовите диалоговое окно свойств диска и прейдите на вкладку сервис.
2. Проверьте диск на наличие ошибок.
3. Запустите программу дефрагментации.
4. Заархивируйте содержимое диска.

V. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое реестр?
2. Как запустить редактор реестра?
3. Как сохранить реестр перед редактированием?
4. Как восстановить реестр?

Практическая работа №10
«РАБОТА С ПРОГРАММОЙ ВОССТАНОВЛЕНИЯ ФАЙЛОВ И ОЧИСТКИ ДИСКОВ»

Цель работы: получить практические навыки восстановления жесткого диска после сбоев.

Краткие теоретические сведения

На сегодняшний день жёсткие диски занимают доминирующее место на рынке накопителей информации. К плюсам жёстких дисков можно отнести низкую стоимость за Гбайт памяти и практичность в использовании. Поэтому возникает необходимость в своевременном **обслуживании, тестировании** и выявлении критического состояния жесткого диска.

В состав утилит современной операционной системы, в том числе Windows входят программы, позволяющие осуществлять дефрагментацию и очистку жесткого диска. Для этого необходимо выполнить команду **Пуск/Стандартные/Служебные** и из появившегося списка программ выбрать нужную.

Кроме того, современные накопители имеют систему оперативного наблюдения за своим состоянием - S.M.A.R.T. (Self-Monitoring, Analysis And Reporting Technology) – технология самодиагностики, анализа и отчета. Это набор программ, вшитых в ПЗУ диска. Эта технология позволяет в любое время оценить такие важные параметры накопителя, как количество отработанных часов, число возникших в процессе чтения/записи ошибок, температуру накопителя среднюю производительность, количество циклов запуска/останова шпинделя, время раскрутки шпинделя, количество переназначенных секторов, количество ошибок позиционирования головок и многое другое. Технология позволяет предсказать возможный выход из строя накопителя.

Исходя из огромной важности корректной работы жесткого диска, существует большое количество программ, позволяющих восстанавливать удаленные файлы с диска, файловую систему, критически важные структуры жесткого диска, такие как главная загрузочная запись, таблица разделов и т.д.

1. Partition Magic

Power Quest @ Partition Magic - это утилита, которая позволяет быстро и легко создавать, удалять, объединять или преобразовывать файловые системы и разделы на жестком диске, не уничтожая существующие данные. Новый инструмент кластерного анализа исследует FAT-дисководы и рекомендует подходящий размер кластера. Кроме того, есть возможность создавать, перемещать и изменять размер разделов типа FAT, FAT 32, файловой системы Windows NT (Windows NT File System, NTFS), HPFS (High-Performance File System - высокопроизводительная файловая система).

Partition Magic помогает надежно устанавливать и использовать несколько операционных систем на одном жестком диске. Partition Magic включает в себя Boot Magic - мощный администратор загрузки, который помогает безопасно устанавливать новые операционные системы и позволяет выбирать через меню систему при загрузке компьютера.

Программа имеет наглядный доброжелательный интерфейс.

В процессе установки программы можно сделать две загрузочные дискеты - на одной будет DOS от Caldera, а на другой - Partition Magic for DOS. С помощью этих дискет можно подготовить новый диск к работе с нуля, т.к. программа наряду с организацией разделов выполняет и их форматирование, причем эти процедуры выполняются намного быстрее, чем при использовании традиционных программ.

Прежде чем начать работу с программой Partition Magic обязательно нужно выполнить следующие рекомендации:

- Установить самые последние обновления для операционных систем Windows. Удостовериться, что самые последние исправления для операционных систем Windows.
- Сделайте копию вашего жесткого диска. Данные на диске - самая ценная часть компьютера. Хоть это и маловероятно, чтобы Partition Magic повредил бы данные, но влияние других ошибок типа системных отказов аппаратных средств, программного обеспечения, или питания, могут привести к повреждению данных в момент выполнения программы PartitionMagic. Используя программу Power Quest's Drive Image, можно создать резервную копию раздела, который будет изменяться. Можно также использовать эту программу и для полного восстановления раздела к первоначальному состоянию.
- Создать загрузочный диск Windows. Загрузочный диск позволит загрузить Windows при возникновении проблемы.
- Запустить опцию проверки ошибок на диске. Для раздела, который будет проверяться, нажать Partition > Check for Errors. Небольшие ошибки могут быть исправлены Partition Magic, однако более серьезные ошибки прекратят выполнение программы. Проверить и исправить обычные ошибки на диске. Проверка загрузочного раздела операционной системы Windows невозможно, так как есть всегда открытые файлы. Для этого раздела, можно воспользоваться Partition > MS ScanDisk.
- Закрыть все запущенные приложения. Нельзя запускать Partition Magic вместе с другими приложениями, включая вирусные сканеры. Если осуществляется работа в сети под управлением Windows NT, перед выполнением Partition Magic, необходимо удостовериться что другие пользователи, не подключены к вашему компьютеру.
- Использовать UPS (Источник бесперебойного питания). Partition Magic не способна восстановить данные, если в процессе разделения диска происходит сбой питания. Используя источник бесперебойного питания (UPS) можно избежать проблем, вызванных сбоем питания.

Совет. Из-за несовместимости аппаратной и системной конфигурации одного компьютера с другим, не рекомендуется переносить с одного на другой компьютер, жесткий диск, разделенный с помощью программы Partition Magic, во избежание потери данных.

Проверка целостности жесткого диска.

Программа Partition Magic проверяет целостность диска сложной системой анализа и проверки достоверности, которая скрыто начинает свою работу, каждый раз, когда запускается программа или завершается операция. Первоначальная проверка на целостность диска, сообщает о любых проблемах связанных с разделами, которые могут препятствовать нормальной работе программы Partition Magic. Проверка целостности действует как ранняя система предупреждения, которая сообщит о том, что структура диска полностью проверена и проанализирована еще до изменения.

Если физический диск проходит первоначальную проверку целостности диска, то появляется таблица разделов, и вы можете начинать работу с программой. В случае появления сообщения об ошибке вместо таблицы разделов, указывается проблема с жестким диском, а не с программой Partition Magic (так как никакие изменения с диском еще не проводились). Необходимо исправить проблему с жестким диском и перезапустить Partition Magic. Для получения **дополнительной информации** можно воспользоваться кнопкой помощи на панели инструментов.

В дополнение проверки целостности при запуске программы, Partition Magic выполняет еще две проверки в течение любой операции. До операции разделения диска проверяется файловая система (наподобие CHKDSK или MS ScanDisk) , после проверяется целостность данных. Partition Magic анализирует диск и немедленно сообщает о найденных ошибках.

Ниже представлены особенности других программ по обслуживанию жестких дисков в процессе их эксплуатации.

2. PARAGON PARTITION MANAGER

Функции программы во многом совпадают с возможностями предыдущей программы - любые разделы можно создавать, удалять, форматировать, перемещать, конвертировать между файловыми системами, объединять и изменять их атрибуты, уменьшать или увеличивать размер разделов - и все это без потери данных. Кроме того, программа от отечественных производителей. Программа способна работать практически с любыми накопителями - жесткими дисками (PATA/SATA/SCSI) неограниченным объемом, внешними жесткими дисками (USB/FireWire), Zip, Jazz и Flash-устройствами.

3. ACRONIS DISK DIRECTOR SUITE

Еще одна отечественная разработка. Возможности утилиты по редактированию разделов дублируют функциональность предыдущих. Дополнительно в комплект входит утилита Acronis Disk Editor, благодаря которой можно вручную редактировать огромное количество параметров жесткого диска и содержащихся на нем разделов. В частности, можно править таблицу разделов, загрузочные секторы FAT и NTFS, настройки FAT и даже все данные, хранящиеся на накопителе (в шестнадцатеричном виде).

4. ACRONIS RECOVERY EXPERT

Нередко проблемы потери данных выходят за рамки гибели пары файлов, порой случается и так, что бесследно исчезают и целые разделы. Список причин, в результате которых может случиться подобная неприятность, довольно обширен - простая невнимательность или неосторожность пользователя, сбой в работе жесткого диска, проказы вируса, ошибка в исполняемой программе, скачок напряжения в сети и многое другое. Помочь может эта программа. Сначала она сканирует неразмеченную область диска на предмет нахождения пропавших разделов, затем удостоверяется у пользователя, что конкретно надо восстановить, после чего приступает к окончательной процедуре восстановления. Программа понимает большинство распространенных файловых систем. Утилита распространяется в составе предыдущей программы.

5. PARTITION TABLE DOCTOR

Одна из самых распространенных неприятностей - это частичное повреждение главной загрузочной записи (Master Boot Record), таблицы разделов (Partition Table) или загрузочных секторов (Boot Sectors), в результате чего система может вообще отказаться запускаться. Справиться с этими проблемами, и поможет данная программа. Помимо непосредственного лечения с помощью утилиты можно сделать резервную копию таблицы разделов и загрузочных секторов. Программа может создать загрузочную дискету или CD со своим полнофункциональным модулем.

6. *PARAGON MOUNT EVERYTHING*

Возникают проблемы совместимости при появлении в системе нового накопителя с другой файловой системой. Данная программа позволяет решить эти проблемы: моментально подключает разделы NTFS, Ext2, Ext3 в любой версии Windows, после чего работа с ними никак не будет отличаться от использования стандартных разделов FAT. Подключенным разделам присваивается буква, на них можно копировать, открывать, редактировать любые файлы и даже запускать приложения. Утилита может управлять разделами - создавать, удалять и форматировать.

7. DISK DIRECTOR SUITE

Эта программа предназначена для профессиональной работы с жестким диском. Это комплексный программный пакет, который включает в себя менеджер разделов, позволяющий осуществлять копирование, перемещение и изменение любых разделов Windows и Linux без риска потери данных, инструмент для восстановления разделов на жестком диске, а также менеджер загрузки, позволяющий установить несколько ОС на один ПК и управлять их запуском. Уже при загрузке программа производит проверку имеющихся дисков. Есть возможность запустить программу с загрузочного CD или дискеты, что позволяет восстановить

разделы даже в ситуациях, когда загрузка компьютера невозможна. Программа оснащена паролем на вход и файлом помощи.

8. EASY RECOVERY PRO

Эта программа предназначена для восстановления утраченных или недоступных (в результате их повреждения) данных. Утилита позволяет без особого труда восстановить данные на жестком диске при утере их вследствие случайного удаления, атаки вирусов, повреждения из-за отключения или резких колебаний напряжения в электросети, ошибок в программе, проблем при создании разделов, неправильного включения ПК, повреждения структуры файловой системы. При помощи команды Drive Test можно проверить диск на наличие физических проблем.

9. FILE RECOVERY

Утилита предназначена для восстановления удаленных или стертых в результате форматирования жесткого диска, данных. Работает с файловыми системами FAT 12/16/32 и NTFS, а также умеет восстанавливать зашифрованные и сжатые файлы. Имеется возможность восстановления информации не только на жестком диске, но и на съемных носителях - дискетах, картах SmartMedia, CompactFlash, Memory Stick и т.д.

10. RESTORER Data RECOVERY

Это мощная программа, которая поможет быстро и просто восстановить нужные файлы, утерянные в результате случайного удаления, а также восстановить отформатированные или разрушенные диски. Утилита поддерживает возможность создания образа диска, это очень полезно для таких задач, как восстановление жесткого диска с большим количеством неработоспособных секторов. Можно установить размер сканируемой области, в зависимости от этого будет меняться время выполнения, которое программа автоматически подсчитывает.

11. HDD Temperature Pro

Это очень маленькая утилита, предназначена для отслеживания состояния жестких дисков. Используя технологию SMART, встроенную во все современные жесткие диски, она анализирует и показывает текущую температуру диска. Здесь возможна установка максимальной температуры накопителя, при превышении которой программа выдаст сообщение. Можно сделать так, чтобы эта утилита самостоятельно загружалась при входе в ОС, так что она будет незаметна, но в нужный момент предупредит о возможной опасности перегрева диска.

12. TREESIZE

Эта утилита предназначена для мониторинга пространства на жестком диске и его освобождении. Она умет искать старые и неиспользуемые, а также временные файлы и удаляет их. С помощью этой утилиты можно найти папки, которые занимают больше всего места на диске, сравнить их объем в процентном соотношении в виде графика. Примерно такие же возможности имеют программы: FCLEANER, FREESPACE.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал.
2. Изучите 4 различных (на выбор) программных средств по восстановлению носителей информации, удаленных файлов. Определите их особенности и возможности.
3. Составьте подробный отчет по выполненной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите известные Вам программы по обслуживанию жестких дисков в процессе их эксплуатации и определите их назначение.

2. Опишите последовательность восстановления удаленной информации, если файл удален в Корзину или файл удален в Корзину и затем очистили Корзину.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Технический колледж им. С.И. Мосина

Методические указания по выполнению самостоятельных работ по профессиональному модулю ПМ.04 СОПРОВОЖДЕНИЕ И ОБСЛУЖИВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

основной профессиональной образовательной программы среднего профессионального образования – программы подготовки специалистов среднего звена

по специальности
09.02.07 Информационные системы и программирование

квалификация
Программист

Тула 2021

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «14» октября 2021 г. № 3
Председатель цикловой комиссии _____ И.В. Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью и задачами изучения профессионального модуля (дисциплины) является формирование у студентов общих и профессиональных компетенций, знаний, умений и практического опыта, указанных в разделах 1, 4 рабочей программы профессионального модуля (дисциплины). В рамках достижения указанной цели учебным планом и рабочей программой предусмотрена самостоятельная работа студентов.

Самостоятельная работа студентов является одной из важнейших составляющих образовательного процесса. Независимо от полученной специальности и характера работы любой начинающий специалист должен обладать фундаментальными знаниями, профессиональными умениями и навыками деятельности своей квалификации, опытом творческой и исследовательской деятельности по решению новых проблем, опытом социально-оценочной деятельности.

Все эти составляющие образования формируются именно в процессе самостоятельной работы студентов, так как предполагает максимальную индивидуализацию деятельности каждого студента и может рассматриваться одновременно и как средство совершенствования творческой индивидуальности.

Основным принципом организации самостоятельной работы студентов является комплексный подход, направленный на формирование навыков репродуктивной и творческой деятельности студента в аудитории, при внеаудиторных контактах с преподавателем на консультациях и домашней подготовке.

Самостоятельная работа студента предполагает углубленное изучение тем, входящих в содержание профессионального модуля (дисциплины) (пункт 2.2 рабочей программы) и выполнение дополнительных заданий теоретического и практического характера.

Среди основных видов самостоятельной работы студентов традиционно выделяют: подготовка к лекциям, лабораторным работам и практическим занятиям, зачетам и экзаменам, презентациям и докладам; написание рефератов, выполнение лабораторных и контрольных работ.

1 Виды и формы организации самостоятельной работы студентов

Любой вид занятий, создающий условия для зарождения самостоятельной мысли, познавательной и творческой активности студента связан с самостоятельной работой. В широком смысле под самостоятельной работой понимают совокупность всей самостоятельной деятельности студентов как в учебной аудитории, так и вне ее, в контакте с преподавателем и в его отсутствие.

Самостоятельная работа может реализовываться:

- непосредственно в процессе аудиторных занятий – на лекциях, лабораторных работах, практических занятиях, при выполнении контрольных и лабораторных работ и др.;
- в контакте с преподавателем вне рамок аудиторных занятий – на консультациях по учебным вопросам, в ходе творческих контактов, при ликвидации задолженностей, при выполнении индивидуальных заданий и т.д.;
- в библиотеке, дома, в общежитии и других местах при выполнении студентом учебных и творческих заданий.

Цель самостоятельной работы студента – осмысленно и самостоятельно работать сначала с учебным материалом, заложить основы самоорганизации и самовоспитания с тем, чтобы привить умение в дальнейшем непрерывно повышать свою профессиональную квалификацию.

В учебном процессе выделяют два вида самостоятельной работы:

- аудиторная – самостоятельная работа выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию;
- внеаудиторная – самостоятельная работа выполняется студентом по заданию преподавателя, но без его непосредственного участия.

Содержание аудиторной и внеаудиторной самостоятельной работы студентов определяется в соответствии с рекомендуемыми видами учебных заданий, представленными в рабочей программе профессионального модуля (дисциплины).

Самостоятельная работа помогает студентам:

1) овладеть знаниями:

- чтение текста (учебника, первоисточника, дополнительной литературы и т.д.);
- составление плана текста, графическое изображение структуры текста, конспектирование текста, выписки из текста и т.д.;
- работа со справочниками и др. справочной литературой;
- ознакомление с нормативными и правовыми документами;
- учебно-методическая и научно-исследовательская работа;
- использование компьютерной техники и Интернета и др.;

2) закреплять и систематизировать знания:

- работа с конспектом лекции;
- обработка текста, повторная работа над учебным материалом учебника, первоисточника, дополнительной литературы, аудио и видеозаписей;

- подготовка плана;
- составление таблиц для систематизации учебного материала;
- подготовка ответов на контрольные вопросы;
- заполнение рабочей тетради;
- аналитическая обработка текста;
- подготовка мультимедиа презентации и докладов к выступлению на семинаре (конференции, круглом столе и т.п.);
- подготовка реферата;
- составление библиографии использованных литературных источников;
- разработка тематических кроссвордов и ребусов;
- тестирование и др.;

3) формировать умения:

- решение ситуационных задач и упражнений по образцу;
- выполнение расчетов (графические и расчетные работы);
- решение профессиональных кейсов и вариативных задач;
- подготовка к контрольным работам;
- подготовка к тестированию;
- подготовка к деловым играм;
- проектирование и моделирование разных видов и компонентов профессиональной деятельности;
- опытно-экспериментальная работа;
- анализ профессиональных умений с использованием аудио- и видеотехники и др.

Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности и уровня умений студентов.

Контроль результатов самостоятельной работы студентов должен осуществляться в пределах времени, отведенного на обязательные учебные занятия и внеаудиторную самостоятельную работу студентов по профессиональному модулю (дисциплине), может проходить в письменной, устной или смешанной форме.

Формы самостоятельной работы студента могут различаться в зависимости от цели, характера, профессионального модуля (дисциплины), объема часов, определенных учебным планом: подготовка к лекциям, семинарским, практическим и лабораторным занятиям; изучение учебных пособий; изучение и конспектирование хрестоматий и сборников документов; изучение в рамках программы курса тем и проблем, не выносимых на лекции и семинарские занятия; написание тематических докладов, рефератов и эссе на проблемные темы; аннотирование монографий или их отдельных глав, статей; выполнение исследовательских и творческих заданий; написание контрольных и лабораторных работ; составление библиографии и реферирование по заданной теме.

2 Требования к организации самостоятельной работы студентов при подготовке к аудиторным занятиям

2.1. Подготовка к лекциям

Главное в период подготовки к лекционным занятиям – научиться методам самостоятельного умственного труда, сознательно развивать свои творческие способности и овладевать навыками творческой работы. Для этого необходимо строго соблюдать дисциплину учебы и поведения. Четкое планирование своего рабочего времени и отдыха является необходимым условием для успешной самостоятельной работы.

В основу его нужно положить рабочие программы изучаемых в семестре дисциплин.

Каждому студенту следует составлять еженедельный и семестровый планы работы, а также план на каждый рабочий день. С вечера всегда надо распределять работу на завтрашний день. В конце каждого дня целесообразно подводить итог работы: тщательно проверить, все ли выполнено по намеченному плану, не было ли каких-либо отступлений, а если были, по какой причине это произошло. Нужно осуществлять самоконтроль, который является необходимым условием успешной учебы. Если что-то осталось невыполненным, необходимо изыскать время для завершения этой части работы, не уменьшая объема недельного плана.

Самостоятельная работа на лекции

Слушание и запись лекций – сложный вид аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность студента. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим студентом.

Не надо стремиться записать дословно всю лекцию. Такое «конспектирование» приносит больше вреда, чем пользы. Запись лекций рекомендуется вести по возможности собственными формулировками. Желательно запись осуществлять на одной странице, а следующую оставлять для проработки учебного материала самостоятельно в домашних условиях.

Конспект лекции лучше подразделять на пункты, параграфы, соблюдая красную строку. Этому в большой степени будут способствовать пункты плана лекции, предложенные преподавателям. Принципиальные места, определения, формулы и другое следует сопровождать замечаниями «важно», «особо важно», «хорошо запомнить» и т.п. Можно делать это и с помощью разноцветных маркеров или ручек. Лучше если они будут собственными, чтобы не приходилось просить их у однокурсников и тем самым не отвлекать их во время лекции.

Целесообразно разработать собственную «маркографию» (значки, символы), сокращения слов. Не лишним будет и изучение основ стенографии. Работая над конспектом лекций, всегда необходимо использовать не только учебник, но и ту литературу, которую дополнительно рекомендовал лектор.

Именно такая серьезная, кропотливая работа с лекционным материалом позволит глубоко овладеть знаниями.

2.2 Подготовка отчета по лабораторно-практической работе

Отчёт по лабораторно-практической работедолжен иметь структуру:

- тема;
- цель;
- ход работы;
- результаты работы;
- ответы на контрольные вопросы.

Критерии оценки работы:

- использование рабочего времени;
- уровень знания возможностей конкретной программы;
- последовательность выполнения задания;
- самостоятельность в работе;
- форма фиксации результатов работы;
- достаточная полнота и формулировка ответов на контрольные вопросы.

Работа, сделанная не по своему варианту или копирующая работу другого студента, засчитывается не будет вне зависимости от качества её выполнения.

2.3 Подготовка презентации и доклада

Презентация, согласно толковому словарю русского языка Д.Н. Ушакова: «… способ подачи информации, в котором присутствуют рисунки, фотографии, анимация и звук».

Для подготовки презентации рекомендуется использовать: Libre Office Impress, MS Power Point, текстовый процессор, Acrobat Reader.

Для подготовки презентации необходимо собрать и обработать начальную информацию. Последовательность подготовки презентации:

1. Четко сформулировать цель презентации: вы хотите свою аудиторию мотивировать, убедить, заразить какой-то идеей или просто формально отчитаться.

2. Определить каков будет формат презентации: живое выступление (тогда, сколько будет его продолжительность) или электронная рассылка (каков будет контекст презентации).

3. Отобрать всю содержательную часть для презентации и выстроить логическую цепочку представления.

4. Определить ключевые моменты в содержании текста и выделить их.

5. Определить виды визуализации (картинки) для отображения их на слайдах в соответствии с логикой, целью и спецификой материала.

6. Подобрать дизайн и форматировать слайды (количество картинок и текста, их расположение, цвет и размер).

7. Проверить визуальное восприятие презентации.

К видам визуализации относятся иллюстрации, образы, диаграммы, таблицы. Иллюстрация – представление реально существующего зрительного ряда. Образы – в отличие от иллюстраций – метафора. Их назначение – вызвать эмоцию и создать отношение к ней, воздействовать на аудиторию. С помощью хорошо продуманных и представляемых образов, информация может надолго остаться в памяти человека. Диаграмма – визуализация количественных и качественных связей. Их используют для убедительной демонстрации данных, для пространственного мышления в дополнение к логическому. Таблица – конкретный, наглядный и точный показ данных. Ее основное назначение – структурировать информацию, что порой облегчает восприятие данных аудиторией.

Практические советы по подготовке презентации

- готовьте отдельно: печатный текст + слайды + раздаточный материал;
- слайды – визуальная подача информации, которая должна содержать минимум текста, максимум изображений, несущих смысловую нагрузку, выглядеть наглядно и просто;
- текстовое содержание презентации – устная речь или чтение, которая должна включать аргументы, факты, доказательства и эмоции;
- рекомендуемое число слайдов 10-12;
- обязательная информация для презентации: тема, фамилия и инициалы выступающего; план сообщения; краткие выводы из всего сказанного; список использованных источников;
- раздаточный материал – должен обеспечивать ту же глубину и охват, что и живое выступление: люди больше доверяют тому, что они могут унести с собой, чем исчезающим изображениям, слова и слайды забываются, а раздаточный материал остается постоянным осязаемым напоминанием; раздаточный материал важно раздавать в конце презентации; раздаточный материалы должны отличаться от слайдов, должны быть более информативными.

Доклад, согласно толковому словарю русского языка Д.Н. Ушакова: «… сообщение по заданной теме, с целью внести знания из дополнительной литературы, систематизировать материл, проиллюстрировать примерами, развивать навыки самостоятельной работы с научной литературой, познавательный интерес к научному познанию».

Тема доклада должна быть согласованна с преподавателем и соответствовать теме учебного занятия. Материалы при его подготовке, должны соответствовать научно-методическим требованиям и быть указаны в докладе. Необходимо соблюдать регламент, оговоренный при получении задания. Иллюстрации должны быть достаточными, но не чрезмерными.

Работа студента над докладом-презентацией включает отработку умения самостоятельно обобщать материал и делать выводы в заключении, умения ориентироваться в материале и отвечать на дополнительные вопросы слушателей, отработку навыков ораторства, умения проводить диспут.

Докладчики должны знать и уметь: сообщать новую информацию; использовать технические средства; хорошо ориентироваться в теме всего

семинарского занятия; дискутировать и быстро отвечать на заданные вопросы; четко выполнять установленный регламент (не более 10 минут); иметь представление о композиционной структуре доклада и др.

Структура выступления

Вступление помогает обеспечить успех выступления по любой тематике. Вступление должно содержать: название, сообщение основной идеи, современную оценку предмета изложения, краткое перечисление рассматриваемых вопросов, живую интересную форму изложения, акцентирование внимания на важных моментах, оригинальность подхода.

Основная часть, в которой выступающий должен глубоко раскрыть

суть затронутой темы, обычно строится по принципу отчета. Задача основной части – представить достаточно данных для того, чтобы слушатели заинтересовались темой и захотели ознакомиться с материалами. При этом логическая структура теоретического блока не должны даваться без наглядных пособий, аудио-визуальных и визуальных материалов.

Заключение – ясное, четкое обобщение и краткие выводы, которых всегда ждут слушатели.

2.4. Подготовка к зачету и экзамену

Каждый учебный семестр заканчивается сессией. Подготовка к сессии, выполнение контрольной работы, сдача зачетов и экзаменов является также самостоятельной работой студента. Основное в подготовке к сессии – повторение всего учебного материала междисциплинарного курса, профессионального модуля (дисциплины).

Только тот студент успевает, кто хорошо усвоил учебный материал. Если студент плохо работал в семестре, пропускал лекции, слушал их невнимательно, не конспектировал, не изучал рекомендованную литературу, то в процессе подготовки к сессии ему придется не повторять уже знакомое, а заново в короткий срок изучать весь учебный материал. Все это зачастую невозможно сделать из-за нехватки времени.

Для такого студента подготовка к зачету или экзамену будет трудным, а иногда и непосильным делом, а конечный результат – возможное отчисление из учебного заведения.

3 Требования к студентам при подготовке письменных работ

3.1. Подготовка реферата

Реферат – письменный доклад по определенной теме, в котором собрана информация из одного или нескольких источников. Рефераты пишутся обычно стандартным языком, с использованием типологизированных речевых оборотов вроде: «важное значение имеет», «уделяется особое внимание», «поднимается вопрос», «делаем следующие выводы», «исследуемая проблема», «освещаемый вопрос» и т.п.

К языковым и стилистическим особенностям рефератов относятся слова и обороты речи, носящие обобщающий характер, словесные клише. У

рефератов особая логичность подачи материала и изъяснения мысли, определенная объективность изложения материала.

Признаки реферата

Реферат не копирует дословно содержание первоисточника, а представляет собой новый вторичный текст, создаваемый в результате систематизации и обобщения материала первоисточника, его аналитико-синтетической переработки.

Будучи вторичным текстом, реферат составляется в соответствии со всеми требованиями, предъявляемыми к связанному высказыванию: так ему присущи следующие категории: оптимальное соотношение и завершенность (смысловая и жанрово-композиционная). Для реферата отбирается информация, объективно-ценная для всех читающих, а не только для одного автора. Автор реферата не может пользоваться только ему понятными значками, пометами, сокращениями.

Работа, проводимая автором для подготовки реферата должна обязательно включать самостоятельное мини-исследование, осуществляемое студентом на материале или художественных текстов по литературе, или архивных первоисточников по истории и т.п.

Организация и описание исследования представляет собой очень сложный вид интеллектуальной деятельности, требующий культуры научного мышления, знания методики проведения исследования, навыков оформления научного труда и т.д. Мини-исследование раскрывается в реферате после глубокого, полного обзора научной литературы по проблеме исследования.

В зависимости от количества реферируемых источников выделяют следующие виды рефератов:

- монографические – рефераты, написанные на основе одного источника;
- обзорные – рефераты, созданные на основе нескольких исходных текстов, объединенных общей темой и сходными проблемами исследования.

Структура реферата

1. Титульный лист
2. Содержание
3.Введение
4. Основная часть
5. Заключение
6. Список использованных источников
7. Приложения

Титульный лист является первой страницей и заполняется по строго определенным правилам (Приложение А).

После титульного листа помещают содержание, в котором приводятся все заголовки работы и указываются страницы, с которых они начинаются. Заголовки оглавления должны точно повторять заголовки в тексте. Сокращать их или давать в другой формулировке и последовательности нельзя. Все заголовки начинаются с прописной буквы без точки на конце. Последнее слово каждого заголовка соединяют отточием с соответствующим ему номером

страницы в правом столбце оглавления. Заголовки одинаковых ступеней рубрикации необходимо располагать друг под другом.

Введение к реферату – важнейшая его часть. Здесь обычно обосновывается актуальность выбранной темы, цель и задачи, краткое содержание, указывается объект рассмотрения, приводится характеристика источников для написания работы и краткий обзор имеющейся по данной теме литературы. Актуальность предполагает оценку своевременности и социальной значимости выбранной темы, обзор литературы по теме отражает знакомство автора с имеющимися источниками, умение их систематизировать, критически рассматривать, выделять существенное, определять главное.

Основная часть. Основная часть реферата структурируется по главам и параграфам (пунктам и подпунктам), количество и название которых определяются автором. Содержание глав основной части должно точно соответствовать теме работы и полностью ее раскрывать. Данные главы должны показать умение студента сжато, логично и аргументировано излагать материал, обобщать, анализировать и делать логические выводы. Основная часть реферата, помимо почерпнутого из разных источников содержания, должна включать в себя собственное мнение студента и сформулированные выводы, опирающиеся на приведенные факты.

В основной части реферата обязательными являются ссылки на авторов, чьи позиции, мнения, информация использованы в реферате. Ссылки на источники могут быть выполнены по тексту работы постранично в нижней части страницы (фамилия автора, его инициалы, полное название работы, год издания и страницы, откуда взята ссылка) или в конце цитирования - тогда достаточно указать в квадратных скобках номер литературного источника из списка использованной литературы с указанием конкретных страниц, откуда взята ссылка, (например, [7, с. 67–89]). Номер литературного источника должен указываться после каждого нового отрывка текста из другого литературного источника.

Цитирование и ссылки не должны подменять позиции автора реферата. Излишняя высокопарность, злоупотребления терминологией, объемные отступления от темы, несоразмерная растянутость отдельных глав, разделов, параграфов рассматриваются в качестве недостатков основной части реферата.

Заключительная часть предполагает последовательное, логически стройное изложение обобщенных выводов по рассматриваемой теме. Заключение не должно превышать объем 2 страниц и не должно слово в слово повторять уже имеющийся текст, но должно отражать собственные выводы о проделанной работе, а может быть, и о перспективах дальнейшего исследования темы. В заключении целесообразно сформулировать итоги выполненной работы, краткого и четкого изложить выводы, представить анализ степени выполнения поставленных во введении задач и указать то новое, что лично для себя студент вынес из работы над рефератом.

Список использованных источников составляет одну из частей работы, отражающую самостоятельную творческую работу автора, и позволяет судить о степени фундаментальности данного реферата. В список использованной литературы необходимо внести все источники, которые были изучены студентами в процессе написания реферата.

В работах используются следующие способы построения библиографических списков: по алфавиту фамилий авторов или заглавий; по тематике; по видам изданий; по характеру содержания; списки смешанного построения. Литература в списке указывается в алфавитном порядке (более распространенный вариант – фамилии авторов в алфавитном порядке), после указания фамилии и инициалов автора указывается название литературного источника без кавычек, место издания и название издательства – при города Москва и Санкт-Петербург как место издания обозначаются сокращенно – М.; СПб., название других городов пишется полностью. (М.: Академия), год издания, страницы – общее количество или конкретные.

Список использованных источников, приводится в следующей последовательности: 1) законодательные акты (в хронологическом порядке);

2) статистические материалы и нормативные документы (в хронологическом порядке); 3) литературные источники (в алфавитном порядке) – книги, монографии, учебники и учебные пособия, периодические издания, зарубежные источники, Интернет-источники. Например:

1. Указ Президента РФ “О защите потребителей от недобросовестной рекламы” от 10.06.94 г. № 1183// Российская газета. 1994. 16 июня. № 112.

2. Блинова М.С. Социология миграции: история становления и перспективы развития: учебное пособие/ М.С. Блинова. – М.: КДУ, 2009. – 192 с.

Для работ из журналов и газетных статей необходимо указать фамилию и инициалы автора, название статьи, а затем наименование источника со всеми элементами титульного листа, после чего указать номер страницы начала и конца статьи. Например:

1. Петренко К.В. Демографические характеристики трудового потенциала нефтегазодобывающих регионов Севера России// Научное обозрение. Серия 2. Гуманитарные науки. – М., 2012. – № 5. – С. 85 – 89.

2. Артемьев З. Мигрантам дадут на работу три года// Вечерняя Москва. 2013. № 184

21

(26509).

Для Интернет-источников необходимо указать название работы, источник работы и сайт. Например:

1. О мерах по созданию и развитию малых предприятий [Электронный ресурс]: постановление Совета министров СССР от 8 авг. 1990 г. № 790. – Режим доступа:

[14.05.2012]// http://www.consultant.ru. – Загл. с экрана.

2. Информационные ресурсы справочно-поисковой системы Рамблер -//http:// www.rambler.ru

После списка использованных источников могут быть помещены различные приложения (таблицы, графики, диаграммы, иллюстрации и пр.). В приложение рекомендуется выносить информацию, которая загромождает текст реферата и мешает его логическому восприятию. В содержательной части работы эта часть материала должна быть обобщена и представлена в сжатом виде. На все приложения в тексте реферата должны быть ссылки. Каждое приложение нумеруется и оформляется с новой страницы.

Требования к оформлению реферата

Работа выполняется на компьютере (гарнитура Times New Roman, шрифт 14) через 1,5 интервала с полями: верхнее, нижнее – 2; левое – 3; правое – 1,5. Отступ первой строки абзаца – 1,25. Сноски – постраничные (шрифт 12), их нумерация должна быть сквозной по всему тексту реферата.

Нумерация страниц должна быть сквозной (номер не ставится на титульном листе, но в общем количестве страниц учитывается).

Таблицы и рисунки встраиваются в текст работы, их нумерация должна быть сквозной по всему реферату. Они все должны иметь название и в самом тексте реферата на них должна быть ссылка. (Например: Как следует из таблицы 1 общая численность безработных в первое десятилетие XXI века в разрезе ряда европейских стран резко увеличивалась). После названия таблицы и рисунка точка не ставится.

Общее количество страниц в реферате, без учета приложений, не должно превышать 15 страниц. Значительное превышение установленного объема является недостатком работы и указывает на то, что студент не сумел отобрать и переработать необходимый материал.

В приложении помещают вспомогательные или дополнительные материалы, которые загромождают текст основной части работы (таблицы, рисунки, карты, графики, неопубликованные документы, переписка и т.д.).

Каждое приложение должно начинаться с новой страницы с указанием в центре верхней строки слова «ПРИЛОЖЕНИЕ», иметь номер и тематический заголовок. При наличии в работе более одного приложения они нумеруются русскими буквами (без знака «№»). Нумерация страниц, на которых даются приложения, должна быть сквозной и продолжать общую нумерацию страниц основного текста. Связь основного текста с приложениями осуществляется через ссылки, которые помещаются в круглые скобки – например, (ПРИЛОЖЕНИЕ А).

ПРИЛОЖЕНИЕ А

Пример оформления титульного листа реферата

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

РЕФЕРАТ

по МДК «……»

на тему: « »

Автор работы,
студент гр._____ А.А.Петров

Руководитель,
преподаватель П.П.Иванов

Тула 20__

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

по междисциплинарному курсу
МДК 11.01 Технология разработки и защиты баз данных»

профессионального модуля
ПМ.11 Разработка, администрирование и защита баз данных

специальности СПО
09.02.07 Информационные системы и программирование

Тула 2021

УТВЕРЖДЕНЫ

на заседании цикловой комиссии информационных технологий
Протокол от «14» октября 2021 г. № 3

Председатель цикловой комиссии И.В. Миляева

Автор: Сафронова М.А., преподаватель, канд. техн. наук

Методические указания определяют задачи, цель и организацию курсовой работы по междисциплинарному курсу МДК 11.01 Технология разработки и защиты баз данных, требования к содержанию и объему курсовой работы, оформлению текстовой части и графической документации, порядок представления курсовой работы к защите.

1. ЦЕЛЬ И ЗАДАЧИ КУРСОВОЙ РАБОТЫ

Курсовая работа по междисциплинарному курсу МДК 11.01 Технология разработки и защиты баз данных является составной часть реализации профессионального модуля ПМ.11 Разработка, администрирование и защита баз данных ОПОП по специальности СПО 09.02.07 Информационные системы и программирование.

Цель курсовой работы – систематизация, закрепление, углубление и обобщение знаний, полученных при изучении МДК 11.01 Технология разработки и защиты баз данных.

2. ОРГАНИЗАЦИЯ РАБОТЫ НАД КУРСОВОЙ РАБОТОЙ

На выполнение и защиту курсовой работы отводится двадцать часов. Выполнение курсовой работы начинается с изучения настоящих методических указаний и анализа задания.

В процессе выполнения курсовой работы обучающийся может пользоваться консультациями и должен не реже одного раза в неделю являться к руководителю курсовой работы для отчета о выполненной работе. Курсовая работа выполняется в соответствии с графиком, который доводится до обучающихся одновременно с выдачей задания.

В течение последней недели перед защитой курсовой работы обучающийся должен получить отзыв руководителя о его готовности к защите.

Если руководитель сделает вывод о невозможности допуска к защите курсовой работы, то обучающийся обязан переработать материал в соответствии с замечаниями и вновь представить его руководителю для заключения о готовности к защите.

3. ТЕМАТИКА КУРСОВОЙ РАБОТЫ

Тематика курсовой работы - разработка базы данных конкретного объекта:

1. Кафедра.
2. Поликлиника.
3. Склад.
4. Типография.
5. Универсам.
6. Организация дорожного движения.
7. Кадры.
8. Клиника.
9. Аптека.
10. Сборочный цех.

11.Группа.
12.Предприятие.
13.Сессия.
14.Посещаемость студентов
15.Кинотеатр.
16.Гостиница.
17.Ремонт автомобилей.
18.Абитуриент.
19.Туристическое агентство.
20.Железнодорожное расписание.
21.Доставка почты.
22.Ремонт квартир.
23.Стационар.
24.Меню.
25.Квартплата.
26.Расписание занятий.

При выполнении курсовой работы могут разрабатываться комплексные темы, связанные с решением одной и той же задачи несколькими обучающимися.

Степень сложности поставленных и решенных в курсовой работе вопросов учитывается при оценке выполненной работы.

4. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ КУРСОВОЙ РАБОТЫ

Задание на курсовую работу, содержащее наименование темы, постановку задачи и исходные данные, оформляются на типовом бланке, подписываются руководителем и обучающимся.

Законченная работа должна состоять из пояснительной записки. Объем пояснительной записки 25-35 страниц, без учета приложений, машинописного текста, распечатанного на бумаге формата А4.

Все страницы текста, а также все рисунки, таблицы и формулы пояснительной записки должны быть пронумерованы. Рисунки и таблицы выполняются либо непосредственно на листах записки, либо аккуратно вклеиваются в записку.

При выполнении вычислений расчетные формулы сначала приводятся в общем виде, затем после подстановки в них числовых значений и, наконец, результат с указанием размерности.

Расшифровка символов, входящих в формулу должна быть приведена непосредственно под формулой (если они не были введены ранее).

При использовании литературных источников обязательно делается ссылка на источник. При этом в квадратных скобках указывается его порядковый номер по перечню использованной литературы, например, [3].

Пояснительная записка курсовой работы должна быть оформлена в соответствии с межгосударственным стандартом ГОСТ 7.32–2017 «Система

стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Введен в действие с 01.07.2018 приказом Росстандарта от 24.10.2017 № 1494-ст.

Список использованных источников оформляется в соответствии с требованиями ГОСТ 7.1, ГОСТ 7.80, ГОСТ 7.82.

5. СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Пояснительная записка должна брошюроваться в следующем порядке:

Титульный лист.
Задание на курсовую работу (на бланке).
Аннотация
Содержание
Введение
Основная часть
Заключение
Список использованных источников

Аннотация должна содержать краткое изложение основного содержания курсовой работы с указанием специфики расчетных методов и приемов, используемых в работе, оригинальности разработки. Объем аннотации не более одной страницы.

Во введении приводятся сведения, касающиеся состояния вопроса, поставленного в задании, с обзором соответствующей технической литературы. Окончанием введения должна быть постановка задачи проектирования. Объем введения не более 5 страниц.

Содержание основной части пояснительной записки зависит от темы работы, метода решения поставленной задачи и определяется руководителем.

В заключении приводятся результаты разработанной базы данных, показывается в какой степени, полученные результаты отвечают требованиям технического задания.

6. ЗАЩИТА КУРСОВОЙ РАБОТЫ

При подготовке к защите курсовой работы обучающийся анализирует замечания руководителя, вносит необходимые исправления и составляет доклад, рассчитанный примерно на 5-7 минут. В докладе должна быть сформулирована поставленная задача, изложены пути и методы ее решения, полученные результаты. Особенно следует подчеркнуть практическую целесообразность принятия тех или иных решений и возможность их технической реализации.

После доклада студент отвечает на вопросы преподавателя по содержанию работы, общетеоретическому материалу.

Оценка за курсовую работу проставляется на бланке задания, в ведомость и зачетную книжку.

Обучающийся, не сумевший защитить курсовую работу, получает неудовлетворительную оценку. На бланке задания руководитель излагает мо-

тивы своего решения и предложения, касающиеся доработки курсовой работы или выдачи нового задания. К повторной защите обучающийся допускается лишь после выполнения рекомендаций руководителя и при наличии направления учебной части.

СПИСОК РЕКОМЕНДУЕМЫХ ИСТОЧНИКОВ

Основные источники

1 Кумскова, И.А. Базы данных : учебник для среднего профессионального образования / Кумскова И.А. — Москва : КноРус, 2020. — 400 с. — ISBN 978-5-406-07467-1. — Текст : электронный // ЭБС Book.ru [сайт]. — URL: https://book.ru/book/932493

2 Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник / В. К. Волк. — Санкт-Петербург : Лань, 2020. — 244 с. — ISBN 978-5-8114-4189-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/126933

3 Советов, Б. Я. Базы данных : учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2020. — 420 с. — (Профессиональное образование). — ISBN 978-5-534-09324-7. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/453635

Дополнительные источники

1 Стасышин, В. М. Базы данных: технологии доступа : учебное пособие для среднего профессионального образования / В. М. Стасышин, Т. Л. Стасышина. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 164 с. — (Профессиональное образование). — ISBN 978-5-534-09888-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/455863

2 Стружкин, Н. П. Базы данных: проектирование : учебник для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2020. — 477 с. — (Профессиональное образование). — ISBN 978-5-534-11635-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/457135

3 Стружкин, Н. П. Базы данных: проектирование. Практикум : учебное пособие для среднего профессионального образования / Н. П. Стружкин, В. В. Годин. — Москва : Издательство Юрайт, 2020. — 291 с. — (Профессиональное образование). — ISBN 978-5-534-08140-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/455865

4 Нестеров, С. А. Базы данных : учебник и практикум для среднего профессионального образования / С. А. Нестеров. — Москва : Издательство Юрайт, 2020. — 230 с. — (Профессиональное образование). — ISBN 978-5-534-11629-8. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/457142

5 Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для среднего профессионального образования / В. М. Илюшечкин. — испр. и доп. — Москва : Издательство Юрайт, 2020. — 213 с. — (Профессиональное образование). — ISBN 978-5-534-01283-5. —

Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/452874

6 Разработка и защита баз данных в Microsoft SQL Server 2005 : учебное пособие для СПО / . — Саратов : Профобразование, 2019. — 148 c. — ISBN 978-5-4488-0366-6. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/86207.html

7 Лаврищева, Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. — Москва : Издательство Юрайт, 2020. — 280 с. — (Высшее образование). — ISBN 978-5-534-01056-5. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/452156 .

8 Программно-аппаратные средства обеспечения информационной безопасности : учебное пособие / А. В. Душкин, О. М. Барсуков, Е. В. Кравцов, К. В. Славнов ; под редакцией А. В. Душкина. — Москва : Горячая линия-Телеком, 2018. — 248 с. — ISBN 978-5-9912-0470-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/111053

Периодические издания:

1 Системный администратор : [журнал]. - Москва, 2020

2 Программирование : научный журнал / учредители : ФГБОУ ВО МГУ им. М.В.Ломоносова, РАН, Отделение информатики, вычислительной техники и автоматизации РАН. - Москва : Наука, 2020 - . - ISSN 0132-3474. - Текст : электронный // НЭБ eLibrary [сайт]. — URL: https://www.elibrary.ru/title_about_new.asp?id=7966

3 Информационно-управляющие системы : научный журнал / учредитель : ООО «Информационно[управляющие системы». - Санкт-Петербург : Изд-во Санкт-Петербургского государственного университета аэрокосмического приборостроения, 2020 - . - ISSN 1684-8853. - Текст : электронный // НЭБ eLibrary [сайт]. — URL: https://www.elibrary.ru/title_about.asp?id=25785

Интернет-ресурсы

1 ЭБС Юрайт. - Интернет- ссылка https://urait.ru/

2 ЭБС BOOK.ru. - Интернет- ссылка https://www.book.ru/

3 ЭБС Лань. - Интернет-ссылка https://e.lanbook.com/

4 ЭБС IPRBooks. - Интернет- ссылка http://www.iprbookshop.ru/

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНО-ПРАКТИЧЕСКИХ РАБОТ

по междисциплинарному курсу
МДК 11.01 Технология разработки и защиты баз данных»

профессионального модуля
ПМ.11 Разработка, администрирование и защита баз данных

специальности СПО
09.02.07 Информационные системы и программирование

Тула 2021

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «14» января 2021 г. № 3
Председатель цикловой комиссии И.В. Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

Практическая работа №1
«СБОР И АНАЛИЗ ИНФОРМАЦИИ»

Цель работы: получить теоретические знания и практические навыки по сбору и анализу информации, о банках и базах данных, основных понятиях теории баз данных, моделях данных.

Теоретические сведения

1. *Банк данных и базы данных*

Стержневые идеи современных информационных технологий базируются на концепции баз данных. Согласно этой концепции, основой информационных технологий являются данные, которые должны быть организованы в базы данных в целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Одним из важнейших понятий в теории баз данных является понятие информации.

Под **информацией** понимаются любые сведения о каком-либо событии, процессе, объекте.

Данные — это информация, но представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные — это информация в дискретном, фиксированном виде, удобная для хранения, обработки на ЭВМ, а также для передачи по каналам связи.

В начальный период применения вычислительной техники организацией данных занимались прикладные программисты. Каждая прикладная программа снабжалась своими собственными механизмами управления данными. Пока на ЭВМ решались отдельные, не связанные друг с другом задачи, такое положение было естественным. Кроме того, в качестве хранилища данных на первых порах достаточно было использовать файлы операционной системы.

Однако позднее применение вычислительной техники приобрело **комплексный характер.** Это означает, что задачи, решаемые на ЭВМ, оказались взаимосвязаны. Комплекс программ призван управлять тем или иным сложным объектом, обработкой регулярных потоков данных. В связи с этим, возникла потребность в разных задачах обращаться к одним и тем же данным, формировать единую информационную модель объекта. Например, в системе управления предприятием данные о персонале используются не только в системе кадрового учета, но и при решении задач планирования, начисления зарплаты и т. д. Поэтому естественно организовать хранение и управление данными отдельно от конкретных прикладных задач.

Тогда любые прикладные программы получат возможность черпать нужные им данные из общего хранилища данных совершенно единообразным способом. Отсюда следует первое требование, которое предъявляется к системе управления базами данных (СУБД): **обеспечить независимый от прикладной задачи интерфейс по управлению данными.** Итак, смена парадигм управления данными произошла. И необходимой дополнительной предпосылкой и стимулом для нее явилось возникновение внешнего запоминающего устройства большого объема и высокой скорости доступа – жесткого диска. Все существовавшие до того внешние запоминающие устройства накладывали слишком большие ограничения либо по скорости (магнитные ленты), либо по объему информации (магнитные барабаны), либо по тому и другому (перфокарты, перфоленты).

Сегодня в основе решения многих задач лежит обработка информации. Для облегчения обработки информации создаются информационные системы (ИС).

Информационная система — это взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Ядром информационной системы являются хранимые в ней данные. На любом предприятии данные различных отделов, как правило, пересекаются, то есть используются в нескольких подразделениях или вообще являются общими. Например, для целей управления часто нужна информация по всему предприятию. Заказ комплектующих невозможен без наличия информации о запасах. Хранящиеся в информационной системе данные должны быть легко доступны в том виде, в каком они нужны для конкретной производственной деятельности предприятия. При этом не имеет существенного значения способ хранения данных.

Сегодня на предприятии мы можем встретить систему обработки данных традиционного типа, в которой служащий вручную помещает данные в скоросшиватель, и рядом с ней - современную систему с применением самой быстродействующей ЭВМ, сложнейшего оборудования и программного обеспечения. Несмотря на поразительную несхожесть, обе эти системы обязаны предоставлять достоверную информацию в определенное время, определенному лицу, в определенном месте и с ограниченными затратами.

Автоматизированными называют ИС, в которых применяют технические средства, в частности ЭВМ. Большинство существующих ИС являются автоматизированными, поэтому для краткости просто будем называть их ИС.

В *широком понимании* под определение ИС подпадает любая система обработки информации. По **области применения** ИС можно разделить на системы, используемые в производстве, образовании, здравоохранении, науке, военном деле, социальной сфере, торговле и других отраслях.

По **целевой функции** ИС можно условно разделить на следующие основные категории: управляющие, информационно-справочные, поддержки принятия решений.

Заметим, что иногда используется более *узкая трактовка понятия ИС* как совокупности аппаратно-программных средств, задействованных для решения некоторой прикладной задачи. В организации, например, могут существовать информационные системы, на которых соответственно возложены следующие задачи: учет кадров и материально-технических средств, расчет с поставщиками и заказчиками, бухгалтерский учет и т. п.

Чтобы понять процесс построения информационной системы, необходимо знать ряд терминов, которые применяются при описании и представлении данных.

Данные — формы представления информации, с которыми имеют дело информационные системы и их пользователи (ISO/IEC 10746-2:1996)[ISO/IEC 10746-2:1996, Information technology — Open Distributed Processing — Reference Model: Foundations].

Банк данных является разновидностью ИС, в которой реализованы функции централизованного хранения и накопления обрабатываемой информации, организованной в одну или несколько баз данных.

Банк данных (БнД) в общем случае состоит из следующих компонентов: базы (нескольких баз) данных, системы управления базами данных, словаря данных, администратора, вычислительной системы и обслуживающего персонала.

Рис. 1.1. Классификация банков данных

Обрабатываемая информация. Так называемые OLTP-системы (On-Line Transaction Processing) предназначены для реализации сравнительно простых запросов к хранимым данным. Напротив, в OLAP-системах (On-Line Analytical Processing) предусмотрены возможности проведения сложных аналитических вычислений.

Словарь данных - это централизованное хранилище сведений об объектах, составляющих их элементах данных, взаимосвязях между объектами, их источниках, значениях, использовании и форматах представления.

Словарь данных (СД) представляет собой подсистему БнД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т. п.

Администратор базы данных (АБД) есть лицо или группа лиц, отвечающих за выработку требований к БД, ее проектирование, создание, эффективное использование и сопровождение. В процессе эксплуатации АБД обычно следит за функционированием информационной системы, обеспечивает защиту от несанкционированного доступа, контролирует избыточность, непротиворечивость, сохранность и достоверность хранимой в БД информации.

Для однопользовательских информационных систем функции АБД обычно возлагаются на лиц, непосредственно работающих с приложением БД.

В вычислительной сети АБД, как правило, взаимодействует с администратором сети. В обязанности последнего входят контроль за функционированием аппаратно-программных средств сети, реконфигурация сети, восстановление программного обеспечения после сбоев и отказов оборудования, профилактические мероприятия и обеспечение разграничения доступа.

Любой банк данных имеет определенные **стадии своего существования**:

проектирование, реализация, эксплуатация, модернизация и развитие, реорганизация. На каждом из этих этапов с БНД работает определенная категория пользователей. Основными категориями пользователей являются следующие.

1. **Конечные пользователи.** Это основная категория пользователей, именно для нее в конечном счете и создается БНД. Эти пользователи могут быть случайными (например, клиенты, просматривающие электронный каталог фирмы) и регулярными (например, сотрудники этой фирмы, в чьи служебные обязанности входит работа с данными, содержащимися в БНД, на уровне выполнения запросов).

2. **Администраторы БНД**. Эта группа пользователей работает с БНД на всех этапах его существования. Администраторы, в частности, обеспечивают оптимальную организацию БНД в многопользовательском режиме, отвечают за поддержку целостности БНД, и т. д.

3. **Разработчики и администраторы приложений**. Эта категория пользователей также работает с БНД на всех этапах его существования и отвечает за разработку специализированных программ-приложений для обработки данных.

База данных (БД) представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Базу данных (БД) можно определить как унифицированную совокупность данных, совместно используемую различными задачами в рамках некоторой единой автоматизированной информационной системы (ИС).

БД бывают **централизованными** (хранятся на одном компьютере) и распределенными (хранятся на нескольких компьютерах некоторой сети).

База данных является ядром любого банка данных. Имеется множество признаков БД, в соответствии с которыми может быть проведена их **классификация**.

1. **По форме представления информации** БД делятся на символьные, визуальные БД, аудиосистемы и средства мультимедиа. Информация, хранимая в БД, может быть представлена в виде разном виде – в виде изображений (рисунки, чертежи и схемы, фотографии, движущиеся изображения, анимация), звука и т. д.

2. **По характеру организации данных БД** могут быть неструктурированными, частично-структурированными и структурированными.

Эта классификация относится к символьным БД. Неструктурированными называются БД, информация в которых представлена в виде так называемых семантических сетей. Частично-структурированные БД содержат информацию в виде текста. В структурированных БД перед заполнением их данными должна быть предварительно описана модель их структуры. В зависимости от типа используемой модели структурированные БД делятся на иерархические, сетевые, реляционные, постреляционные, многомерные и объектно-ориентированные.

3. **По типу хранимой информации** БД делятся на документальные, фактографические и лексикографические.

Документальные БД являются частично-структурированными и ориентированы, главным образом, на хранение текстовых данных в различных форматах. Информационной единицей в документальных БД является документ-текст. Среди этих моделей выделяют библиографические, реферативные и полнотекстовые модели.

Лексикографические модели организованы на принципах организации словарей и содержат в себе определенные языковые конструкции. Основное назначение этих моделей – использование в системах-переводчиках.

Фактографические модели являются структурированными и в зависимости от способа структуризации делятся на теоретико-графовые (иерархическая и сетевая модели), теоретико-множественные (реляционная, постреляционная и многомерная) и объектно-ориентированные.

Логическую структуру хранимых в базе данных называют **моделью представления данных.** К основным моделям представления данных (моделям данных) относятся следующие: иерархическая, сетевая, реляционная, постреляционная, многомерная и объектно-ориентированная.

4. **По характеру организации хранения данных** БД бывают персональными и распределенными. Персональные БД предназначены для одного конкретного пользователя. Распределенные БД предполагают возможность одновременного обращения к данным со стороны множества пользователей.

Система управления базами данных (СУБД) — это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

В зависимости от физического расположения различают локальные и сетевые СУБД.

Локальная СУБД целиком размещается на компьютере пользователя. Если таких пользователей несколько, то каждый из них должен иметь свою локальную копию СУБД.

Сетевые СУБД делятся на файл-серверные, клиент-серверные и распределенные. В **файл-серверной** модели как СУБД, так и БД, как правило, размещаются на одном компьютере, который называется файл-сервером. Пользователи получают доступ к информации со своих персональных компьютеров (клиентские места) посредством развертывания локальной сети. Таким образом, между локальными и файл-серверными вариантами принципиальных различий нет.

Клиент-серверные СУБД являются фактически двухзвенными, поскольку в этом случае часть СУБД размещается на сервере БД. Эта часть СУБД отвечает за получение запроса от клиента, отыскание в данных нужной информации и передачу ее клиенту.

Распределенные СУБД могут размещаться на десятках и сотнях серверов БД.

С точки зрения пользователя, СУБД реализует **функции** хранения, изменения (пополнения, редактирования и удаления) и обработки информации, а также разработки и получения различных выходных документов.

2. Основные понятия теории баз данных

Предметной областью называется часть реальной системы, представляющая интерес для данного исследования или совокупность объектов реального мира, рассматриваемого в рамках определенного контекста.

При проектировании автоматизированных информационных систем **предметная область отображается моделями данных нескольких уровней.** Число используемых уровней зависит от сложности системы, но в любом случае включает логический и физический уровни. Предметная область может относиться к любому типу организации (например, банк, университет, больница или завод).

Необходимо различать **полную предметную область** (крупное производственное предприятие, склад, универмаг и т.д.) и **организационную единицу этой предметной области**. Организационная единица в свою очередь может представлять свою предметную область (например, цех по производству кузовов автомобильного завода или отдел обработки данных предприятия по производству ЭВМ). В данном случае цехи и отделы сами могут соответствовать определенным предметным областям.

Информация, необходимая для описания предметной области, зависит от реальной модели и может включать сведения о персонале, заработной плате, товарах, накладных, счетах, отчетах по сбыту, лабораторных тестах, финансовых сделках, историях болезней, то есть сведения о людях, местах, предметах, событиях и понятиях.

Данные и связи между ними могут быть описаны и представлены различными способами. В настоящее время существует **стандарт ANSI** (American National Standards Institute), в соответствии с которым имеется три уровня представления данных – физический (внутренний) уровень, концептуальный уровень и уровень внешних моделей.

Под **физическим уровнем** понимаются собственно данные, специальным образом организованные и размещенные в файловых либо странично-сегментных структурах на внешних носителях.

Концептуальный уровень является наиболее общим. На этом уровне отражается обобщенная модель предметной области. Уровень **внешних моделей** отражает особенности видения данных отдельными приложениями. При этом каждое приложение имеет доступ только к тем данным, которые ему необходимы.

Объектом называется элемент информационной системы, информацию о котором мы сохраняем. В реляционной теории баз данных объект называется **сущностью**.

Объект может быть реальным (например, человек, какой-либо предмет или населенный пункт) и абстрактным (например, событие, счет покупателя или изучаемый студентами курс). Так, в области продажи автомобилей примерами объектов могут служить МОДЕЛЬ АВТОМОБИЛЯ, КЛИЕНТ и СЧЕТ. На товарном складе - это ПОСТАВЩИК, ТОВАР, ОТПРАВЛЕНИЕ и т. д. Каждый объект обладает определенным набором свойств, которые запоминаются в информационной системе. При обработке данных часто приходится иметь

дело с совокупностью однородных объектов, например таких, как служащие, и записывать информацию об одних и тех же свойствах для каждого из них.

Классом объектов называют совокупность объектов, обладающих одинаковым набором свойств. Таким образом, для объектов одного класса набор свойств будет одинаков, хотя значения этих свойств для каждого объекта, конечно, могут быть разными. Например, класс объектов МОДЕЛЬ АВТОМОБИЛЯ будет иметь одинаковый набор свойств, описывающих характеристики автомобилей, и каждая модель будет иметь различные значения этих характеристик.

Объекты и их свойства являются понятиями реального мира. В мире информации, существующем в представлении программиста, говорят об атрибутах объектов.

Атрибут - это информационное отображение свойств объекта. Каждый объект характеризуется рядом основных атрибутов. Например, модель автомобиля характеризуется типом кузова, рабочим объемом двигателя, количеством цилиндров, мощностью, габаритами, названием и т. д. Клиент магазина, продающего автомобили, имеет такие атрибуты, как фамилию, имя, отчество, адрес и, возможно, идентификационный номер. Каждый атрибут в модели должен иметь уникальное имя - идентификатор. Атрибут при реализации информационной модели на каком-либо носителей информации часто называют элементом данных, полем данных или просто полем.

Информацию о некоторой предметной области можно представить с помощью нескольких объектов, каждый из которых описывается несколькими элементами данных. Принимаемые элементами данных значения называются данными. Единичный набор принимаемых элементами данных значений называется экземпляром объекта. Объекты связываются между собой определенным образом.

Данные – это представление фактов и идей в формализованном виде, пригодном для передачи и переработки в некотором процессе.

Тип данных характеризует вид хранящихся данных. Понятие типа данных в информационной модели полностью адекватно понятию типа данных в языках программирования. Обычно в современных СУБД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (например, суммы в денежных единицах), а также данных специального формата (дата, время, временной интервал и пр.). В любом случае при выборе типа данных следует учитывать возможности той СУБД, с помощью которой будет реализовываться физическая модель информационной системы.

Источник данных – среда, из которой поступает информация. Примеры – больница (история болезни, коечный фонд), авиалинии (самолеты, рейсы, места), торговля (товар, покупатель, поставщик).

Первичные документы – носители информации, используемые в источнике данных.

Доменом называется набор значений элементов данных одного типа, отвечающий поставленным условиям.

Понятие домена более специфично для баз данных, хотя и имеет определенные аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных, который "забраковывает" недопустимые значения. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена. Например, домен "НАИМЕНОВАНИЕ" в нашем примере определен набазовом типе строк символов, но в число его значений могут входить только те строки, которые могут изображать имя (в частности, очевидно, что такие строки не должны начинаться с какого-нибудь специфичного символа типа знака подчеркивания или тире). В упрощенном виде понятие домена может характеризоваться как потенциальное множество допустимых значений одного типа. Например, значением атрибута "ПОЛ" может быть только либо "мужской", либо "женский".

Транзакцией называется некоторая неделимая последовательность операций над данными БД, которая отслеживается СУБД от начала и до завершения. Механизм транзакций используется в СУБД для поддержания целостности данных в базе. Если по каким-либо причинам (сбои и отказы оборудования, ошибки в программном обеспечении, включая приложение) транзакция остается незавершенной, то она отменяется.

В зависимости от времени, требуемого для выполнения, выделяют обычные и продолжительные транзакции. Продолжительные транзакции могут охватывать часы, дни и даже месяцы. Такие транзакции могут возникать в процессе проектирования и разработки сложных систем крупным коллективом людей. Кроме того, помимо обычных плоских транзакций, используется модель вложенных транзакций. В последнем случае транзакция рассматривается как набор взаимосвязанных подзадач (субтранзакций), каждая из которых также может состоять из произвольного количества субтранзакций.

Контроль транзакций важен в однопользовательских и в многопользовательских СУБД, где транзакции могут быть запущены параллельно.

3. Модели данных

Хранимые в базе данные имеют определенную логическую структуру — иными словами, описываются некоторой **моделью представления данных** (моделью данных), поддерживаемой СУБД.

К числу классических относятся следующие модели данных:

1. иерархическая,
2. сетевая,
3. реляционная.

Кроме того, в последние годы появились и стали более активно внедряться на практике следующие модели данных:

4. объектно-реляционная (постреляционная),
5. многомерная,
6. объектно-ориентированная.

- плоские файлы
- «сущность-связь»
- полуструктурированные модели

Разрабатываются также всевозможные системы, основанные на других моделях данных, расширяющих известные модели. В их числе можно назвать объектно-реляционные, дедуктивно-объектно-ориентированные, семантические, концептуальные и ориентированные модели. Некоторые из этих моделей служат для интеграции баз данных, баз знаний и языков программирования.

В некоторых СУБД поддерживаются одновременно несколько моделей данных.

3.1 Иерархическая модель

Иерархическая модель БД является одной из первых моделей БД. Это обусловлено прежде всего тем, что именно такая модель наиболее естественным образом отражает множественные связи между объектами реального мира, когда один объект выступает в качестве главного (родительского), с которым связано большое количество подчиненных (дочерних) объектов.

Принцип иерархической модели БД заключается в том, что все связи между данными описываются с помощью построения упорядоченного графа или дерева (рис. 2).

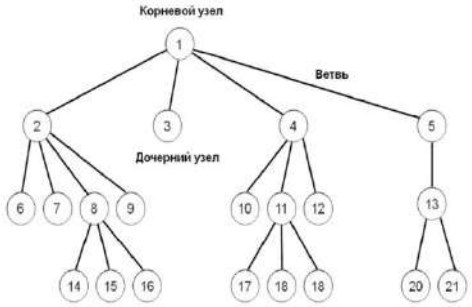


Рис. 2. Упорядоченный граф – древовидная структура

Для описания структуры (схемы) иерархической БД на некотором языке программирования используется тип данных «дерево». Тип «дерево» схож с типами данных «структура» языков программирования С и «запись» языка Паскаль. В них допускается вложенность типов, каждый из которых находится на некотором уровне.

Тип «дерево» является составным. Он включает в себя подтипы («поддеревья»), каждый из которых, в свою очередь, является типом «дерево». Каждый из типов «дерево» состоит из одного «корневого» типа и упорядоченного набора (возможно, пустого) подчиненных типов. Каждый из элементарных типов, включенных в тип «дерево», является простым или составным типом «запись». Простая «запись» состоит из одного типа, например числового, а составная «запись» объединяет некоторую совокупность типов, например, целое, строку символов и указатель (ссылку). Пример типа «дерево» как совокупности типов показан на рис. 3.

Корневой тип – это такой тип, который имеет подчиненные типы и не имеет родительских. Дочерние типы, имеющие один и тот же родительский тип, называются близнецами. Каждый из подчиненных типов для данного корневого типа может являться как простым, так и составным типом “запись”.

Основным **достоинством** иерархической модели БД является относительно высокая скорость обработки информации при обращении к данным. К основным **недостаткам** иерархических моделей следует отнести: неэффективность, медленный доступ к сегментам данных нижних уровней иерархии, четкая ориентация на определенные типы запросов и др. Также недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

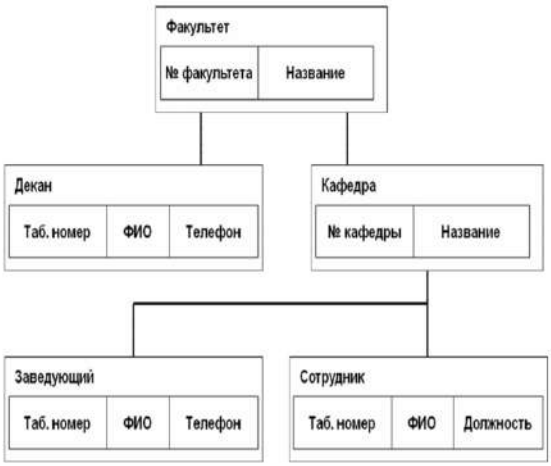


Рис. 3. Пример структуры иерархической модели

Различают три вида деревьев – сбалансированные, несбалансированные и двоичные.

В сбалансированном дереве (рис. 3) каждый узел имеет одно и то же количество ветвей. Такая организация данных физически является наиболее простой, однако часто логическая структура данных требует переменного количества ветвей в каждом узле, что соответствует

несбалансированному дереву (рис. 4). Двоичные деревья допускают наличие не более двух ветвей для одного узла.

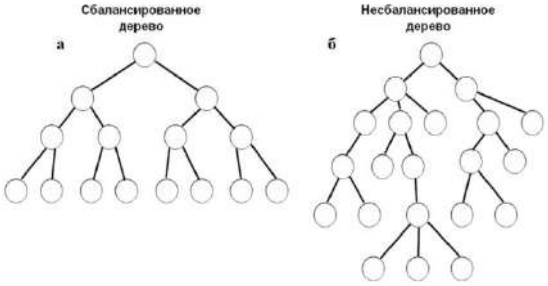


Рис. 4. Примеры сбалансированного и несбалансированного графов

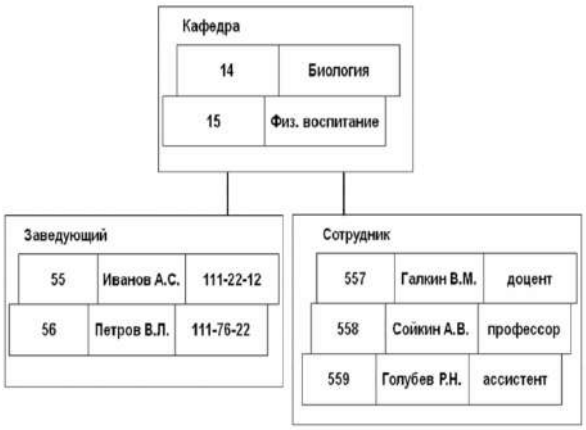


Рис. 5. Примеры представления данных в иерархической модели

На иерархической модели данных основано сравнительно ограниченное количество СУБД, в числе которых можно назвать зарубежные системы Information Management System (IMS) фирмы IBM (1966-1968 г.)., Time-Shared Date Management System (TDMS) компании System Development Corporation, Team-Up и Data Edge, Google App Engine Datastore API, а также отечественные системы Ока, ИНЭС и МИРИС.

По принципу иерархической БД построены иерархические файловые системы и Реестр Windows, серверы каталогов, такие, как LDAP и Active Directory (допускают чёткое представление в виде дерева).

Иерархические СУБД быстро прошли пик популярности, которая обусловливалась их ранним появлением на рынке. Затем их недостатки сделали их неконкурентоспособными, и в настоящее время иерархическая модель представляет исключительно исторический интерес.

3.2 Сетевая модель

Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель данных (рис. 6).

Для описания схемы сетевой БД используется две группы типов: «узел» и «связь». Тип «связь» определяется для двух типов «узлов»: предка и потомка. Сетевая БД состоит из набора **узлов** и **набора соответствующих связей.** На формирование связи особых ограничений не накладывается. Если в иерархических структурах запись-потомок могла иметь только одну запись-предка, то в сетевой модели данных запись-потомок может иметь произвольное число записей-предков (сводных родителей).

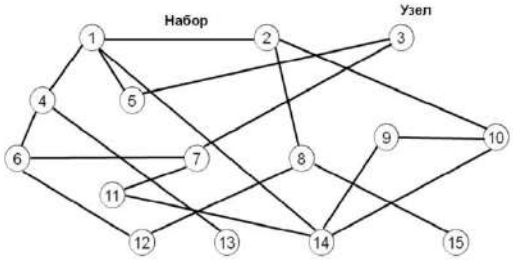


Рис. 6. Пример сетевой модели

Рис. 7. Примеры структур данных в сетевой модели

Сетевая СУБД — СУБД, построенная на основе сетевой модели данных.

К основным понятиям **сетевой модели базы данных** относятся: уровень, элемент (узел), связь.

Узел — это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. В сетевой структуре каждый элемент может быть связан с любым другим элементом.

Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию.

Несмотря на то, что эта модель решает некоторые проблемы, связанные с иерархической моделью, выполнение простых запросов остается достаточно сложным процессом.

Также, поскольку логика процедуры выборки данных зависит от физической организации этих данных, то эта модель не является полностью независимой от приложения. Другими словами, если необходимо изменить структуру данных, то нужно изменить и приложение.

Разница между иерархической моделью данных и сетевой состоит в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных у потомка может иметься любое число предков.

Сетевая БД состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного набора экземпляров типа записи потомка. Для данного типа связи L с типом записи предка P и типом записи потомка C должны выполняться следующие два условия:

- каждый экземпляр типа записи P является предком только в одном экземпляре типа связи L;
- каждый экземпляр типа записи C является потомком не более чем в одном экземпляре типа связи L.

Примерный набор операций манипулирования данными:

- найти конкретную запись в наборе однотипных записей;
- перейти от предка к первому потомку по некоторой связи;
- перейти к следующему потомку в некоторой связи;

- перейти от потомка к предку по некоторой связи;
- создать новую запись;
- уничтожить запись;
- модифицировать запись;
- включить в связь;
- исключить из связи;
- переставить в другую связь и т. д.

Несомненным **достоинством** сетевой модели данных является возможность более гибкого отображения множественных связей между объектами. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей. К основным преимуществам сетевых СУБД относятся следующие: обработка больших объемов информации (возможность построения на основе таких СУБД «хранилищ данных»); поддержка аналитической обработки данных; эффективная реализация обработки данных по показателям затрат памяти и оперативности.

Один из наиболее существенных **недостатков** заключается в высокой сложности схемы построения БД, что усугубляется ослаблением контроля за целостностью связей ввиду их многочисленности. Пользователи сетевых СУБД ограничены связями, определенными для них разработчиками БД-приложений. Подобно иерархическим, сетевые СУБД предполагают разработку БД приложений опытными программистами и системными аналитиками.

3.3 Реляционная модель

Реляционная модель данных предложена сотрудником фирмы IBM англичанином Эдгаром Коддом и основывается на понятии отношение (relation). В 60-х — 70-х годах он работал над своими теориями хранения данных. В 1970 издал работу «A Relational Model of Data for Large Shared Data Banks», которая считается первой работой по реляционной модели данных.

Отношение представляет собой множество элементов, называемых кортежами. Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица.

№	ФИО	Должность	Оклад
1	Соколов А.А.	инженер	1000
2	Брехунов Г.А.	администратор	35 000
3	Самойлов С.В.	менеджер	18 000

Рис. 8. Пример реляционного отношения

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Таблица имеет **строки** (записи) и **столбцы** (колонки). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют **кортежи**, а столбцам — **атрибуты отношения**.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- Каждый элемент таблицы является одним элементом данных
- Каждый столбец обладает своим уникальным именем
- Одинаковые строки в таблице отсутствуют
- Все столбцы в таблице однородные, то есть все элементы в столбце имеют одинаковый тип
- Порядок следования строк и столбцов может быть произвольным.

В каждом отношении всегда должен присутствовать атрибут или набор атрибутов, однозначно определяющий единственный кортеж этого отношения – **первичный ключ.**

Первичный ключ - это атрибут (или группа атрибутов), которые единственным образом идентифицируют каждую строку в таблице.

Альтернативный ключ - это атрибут (или группа атрибутов), несовпадающий с первичным ключом и уникально идентифицирующий экземпляр объекта.

Например, для объекта "служащий", который имеет атрибуты "ИДЕНТИФИКАТОР СЛУЖАЩЕГО", "ФАМИЛИЯ", "ИМЯ" и "ОТЧЕСТВО", группа атрибутов "ФАМИЛИЯ", "ИМЯ", "ОТЧЕСТВО" может являться альтернативным ключом по отношению к атрибуту "ИДЕНТИФИКАТОР СЛУЖАЩЕГО" (в предположении, что на предприятии не работают полные тезки).

Для отражения связи между объектами используется связывание таблиц по определенным правилам с использованием так называемых **внешних ключей**. Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование. Поскольку в рамках одной таблицы не удается описать более сложные логические структуры данных из предметной области, применяют **связывание таблиц.** Связь - это функциональная зависимость между сущностями.

Если между некоторыми сущностями существует связь, то факты из одной сущности ссылаются или некоторым образом связаны с фактами из другой сущности.

Поддержание непротиворечивости функциональных зависимостей между сущностями называется ссылочной целостностью. Поскольку связи содержатся "внутри" реляционной модели, реализация ссылочной целостности может выполняться как приложением, так и самой СУБД (с помощью механизмов декларативной ссылочной целостности и триггеров).

Связи могу быть представлены пятью основными характеристиками: тип связи (идентифицирующая, не идентифицирующая, полная/неполная категория, неспецифическая связь); родительская сущность; дочерняя (зависимая) сущность; мощность связи (cordiality); допустимость пустых (null) значений.

Связь называется **идентифицирующей**, если экземпляр дочерней сущности идентифицируется (однозначно определяется) через ее связь с родительской сущностью.

Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней сущности. Дочерняя сущность при идентифицирующей связи всегда является зависимой. Связь называется не идентифицирующей, если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав не ключевых атрибутов дочерней сущности.

Мощность связи представляет собой отношение количества экземпляров родительской сущности к соответствующему количеству экземпляров дочерней сущности. Для любой связи, кроме неспецифической, эта связь записывается как 1:n.

Ключевым элементом данных называется такой элемент, по которому можно определить значения других элементов данных. Однозначно идентифицировать объект могут два и более элемента данных. В этом случае их называют "кандидатами" в ключевые элементы данных. Вопрос о том, какой из кандидатов использовать для доступа к объекту, решается пользователем или разработчиком системы. Выбирать ключевые элементы данных следует тщательно, поскольку правильный выбор способствует созданию достоверной концептуальной модели данных.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми.

Основными **недостатками** реляционной модели являются следующие: отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

Примерами зарубежных реляционных СУБД для ПЭВМ являются следующие: dBase II Plus и dBase IY (фирма Ashton-Tate), FoxPro (Fox Software), Paradox и dBASE for Windows

(Borland), Visual FoxPro и Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) и Oracle 7.x (Oracle).

К отечественным СУБД реляционного типа относятся системы: ПАЛЬМА (ИК АН УССР), а также система HyTech (МИФИ).

Реляционные СУБД все еще доминируют в системах обработки финансовых транзакций, но сегодня компании все шире применяют СУБД новой архитектуры NoSQL — горизонтально масштабируемые, распределенные и разрабатываемые в открытых кодах. Примеры таких систем — Hadoop, MapReduce и VoltDB. По оценкам аналитиков Forrester, около 75% данных на предприятиях это либо полуструктурированная информация (XML, электронная почта и EDI), либо неструктурированная (текст, изображения, аудио и видео), и лишь 5% от этих данных хранится в реляционных СУБД, а остальное — в базах других типов или в виде файлов, и неподвластно обработке реляционными системами.

3.4 Постреляционная модель

Развитие реляционной модели привело к появлению так называемой постреляционной модели данных, основным отличием которой является допустимость многозначных полей (полей, значения которых состоят из множества подзначений).

Многозначные поля можно интерпретировать как самостоятельные таблицы, встроенные в исходную таблицу. Кроме того, в постреляционной модели поддерживаются множественные ассоциированные поля, в совокупности образующих ассоциацию: в каждой строке первое значение одного столбца ассоциации соответствует первым значениям всех остальных столбцов ассоциации.

Рис. 9 иллюстрирует сравнение представления данных в реляционной и постреляционной моделях.

Рис. 9. Сравнение способов представления данных в реляционной и постреляционной моделях

Основное **достоинство** постреляционной модели заключается в том, что она позволяет более эффективно хранить данные, а количество таблиц в этой модели заметно меньше по сравнению с реляционной. **Недостатком** является сложность обеспечения поддержания логической согласованности данных.

3.5 Объектно-ориентированная модель

Основным отличием объектно-ориентированной модели от рассмотренных выше является использование объектно-ориентированных методов манипулирования данными – **инкапсуляции**, **наследования** и **полиформизма**.

Инкапсуляция означает возможность разграничения доступа различных программ, приложений, методов и функций (в более широком смысле и доступа различных категорий пользователей) к различным свойствам объектов данных. В контексте термина “инкапсуляция” часто используется понятие видимости – степень доступности отдельных свойств объекта.

В отличие от инкапсуляции **наследование** предполагает полную передачу всех свойств родительского объекта дочерним объектам. При необходимости наследование свойств одного объекта можно распространить и на объекты, не являющиеся по отношению к нему дочерними.

Полиморфизм означает возможность одного и того же приложения манипулировать с данными разных типов – приложения (методы, процедуры и функции), обрабатывающие объекты различных типов, могут иметь одно и то же имя.

Основным **достоинством** объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

Заметим, что последние версии реляционных СУБД имеют некоторые свойства объектно-ориентированных систем. Такие СУБД часто называют объектно-реляционными. Примером такой системы можно считать продукты Oracle 8.x.

3.6 Объектно-реляционная модель

Объектно-реляционная СУБД (ОРСУБД) - реляционная система управления базами данных, использующая в своей работе заимствования и методы свойственные объектно-ориентированному подходу. Три ведущих компании в области разработки систему управления базами данных, а именно Oracle, Informix и IBM, расширили свои системы до объектно-реляционного уровня

3.7 Многомерная модель

Теория многомерных моделей данных активно развивается в последнее время. Понятие многомерной модели означает многомерность логического представления структуры информации. Основными понятиями многомерной модели являются **измерение** и **ячейка**.

Измерение (Dimension) — это множество однотипных данных, образующих одну из граней гиперкуба. Примерами наиболее часто используемых временных измерений являются Дни, Месяцы, Кварталы и Годы. В качестве географических измерений широко употребляются Города, Районы.

В многомерной модели данных измерения играют роль индексов, служащих для идентификации конкретных значений в ячейках гиперкуба.

Ячейка (Cell) или показатель — это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен как цифровой.

Для работы с многомерными моделями данных используются специальные многомерные СУБД, в основе которых лежат понятия агрегируемости, историчности и прогнозируемости. Под **агрегируемостью** данных подразумеваются различные уровни обобщения информации. **Историчность** данных означает высокий уровень статичности как самих данных, так и связей между ними, а также упорядочение данных во времени в процессе их обработки и представления пользователям. Обеспечение **прогнозируемости** задается использованием специальных функций прогнозирования.

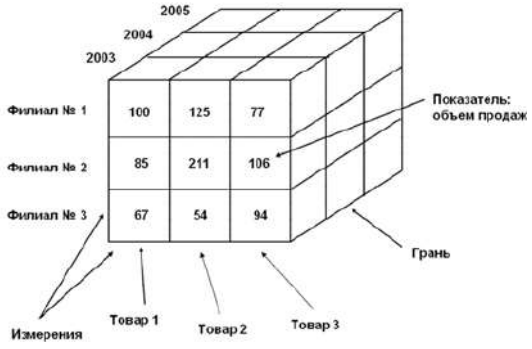


Рис. 10. Трехмерный куб в многомерной модели

Основным **достоинством** многомерной модели является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. **Недостатком** многомерной модели является ее громоздкость для простейших задач обычной оперативной обработки информации.

Примеры систем: Essbase, Media Multi-matrix (и реляционные тоже), Oracle Express Server.

3.8 Полуструктурированная модель данных

Модель полуструктурированных данных, допускающая необязательные атрибуты, может использоваться для интегрирования данных, полученных их разных источников, и для представления данных с нечетко определенной или меняющейся структурой.

Полуструктурированная модель данных (semi-structured data model) – определяется по-разному. Считается, что информация, представляемая в рамках этой модели, не разделяется на данные и схему, т.е. происходит нивелирование различия между данными и метаданными. Эта модель изначально была ориентирована на обмен данными между гетерогенными информационными системами, так как она обеспечивает гибкий формат обмена между различными типами баз данных.

Позднее стали говорить о базах полуструктурированных данных, основным свойством которых является отсутствие предписывающей схемы. Многие свойства этих баз вытекают из этого фундаментального свойства. Заметим, что традиционно системы баз данных, будь то реляционные или объектные, опираются на предписывающую схему данных.

Для обращения к базе полуструктурированных данных используются описывающие схемы, каждая из которых соответствует цели обращения.

Полуструктурированные модели данных делят на тяжеловесные (heavyweight) и легковесные (lightweight). К легковесным моделям данных относят простые и очень гибкие модели данных. Например, модель данных RDF можно отнести к легковесным моделям полуструктурированных данных.

Если из общей совокупности полуструктурированных данных удается выделить данные, которые имеют устойчивую структуру, то для них можно выделить постоянное место в записи и, например, использовать индексы для эффективного доступа, а в общем случае средства, которые предоставляют традиционные модели данных. В этом случае говорят о **тяжеловесной модели** полуструктурированных данных.

В соответствии с этой моделью база данных рассматривается как совокупность своеобразных таблиц, причем каждая таблица представляется парой файлов. По функциональному назначению различают файлы «словари» и файлы «области данных», хотя и те, и другие организованы абсолютно одинаково. С каждым словарем может быть связано несколько областей данных, в том числе и ни одной.

Файлы состоят из записей переменной длины. Каждая запись файла, будь то словарь или область данных, представляет собой строку переменной длины, которая получается при сохранении в долговременной памяти трехмерного динамического массива – основной структуры, с которой приходится оперировать при написании программ обработки (отсюда название – D3). Слово «динамический» означает, что границы измерений априорно не определены и могут быть сколь угодно большими.

Преимущества построения базы данных с использованием полуструктурированной модели следующие: в рамках этой модели можно представлять данные, не связанные некоторой предписывающей схемой; структурированные данные всегда можно рассматривать как полуструктурированные, при этом вместо предписывающей схемы, в соответствии с которой создавались данные, всегда может быть использовано несколько описывающих схем, которые создаются непосредственно в процессе эксплуатации.

С другой стороны, полуструктурированные данные всегда можно представить как структурированные, переходя на более высокий уровень абстракции при отображении предметной области.

Примерами работы с полуструктурированными данными являются СУБД D3, модели OEM (Object Exchange Model) для СУБД Lore.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Осуществите сбор и анализ информации для создания БД по предметной области по индивидуальному варианту (см. Приложение А).

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что называется «Банком данных»?
2. Что называется «Базой данных»?
3. Что называется «Моделью данных»?
4. Дайте характеристику иерархической модели базы данных?
5. Дайте характеристику сетевой модели
6. Дайте характеристику реляционной модели
7. Дайте характеристику постреляционной модели
8. Дайте характеристику объектно-ориентированной модели
9. Дайте характеристику объектно-реляционной модели
10. Дайте характеристику многомерноймодели
11. Дайте характеристику полуструктурированной модели.

Практическая работа №2

РАБОТА «ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ СХЕМЫ БАЗЫ ДАННЫХ В СРЕДЕ СУБД»

Цель работы: получить теоретические знания и практические навыки проектирования реляционной схемы базы данных в среде СУБД.

Теоретические сведения

1. Основные сведения о проектировании БД

Проектирование БД является одним из этапов жизненного цикла информационной системы.

Основной задачей, решаемой в процессе проектирования БД является проектирование структуры данных.

Проектирование программных систем:

- проектирование процессов;
- проектирование данных;
- проектирование событий.

В силу специфики курса мы рассматриваем лишь проектирование данных – процесс разработки структуры базы данных в соответствии с требованиями заказчиков.

Моделирование данных – это процесс создания логического представления структуры баз данных. Правильно сконструированная модель данных должна быть адекватная предметной области, то есть соответствовать всем пользовательским представлениям данных. Если база данных будет неверно отражать пользовательское представление, то пользователи найдет это приложение неудобным, неполным и непригодным для работы. Поэтому моделирование данных является основным этапом в разработке системы баз данных, основой всей последующей работы при разработке БД и приложений.

В ходе разработки модели или структуры баз данных нужно ответить на следующие вопросы:

- что представляют собой требования заказчиков, и в какой форме они выражены;
- как они преобразуются в структуру базы данных;
- как часто и каким образом структура базы данных должна перестраиваться.

В настоящее время рассматриваются три уровня абстракции для определения структуры данных: концептуальный (точка зрения заказчика), логический (точка зрения разработчика) и физический (точка зрения администратора БД). В соответствии с этим рассматриваются три уровня модели и три шага проектирования.

Под **информационной моделью данных** понимают способ описания информации, содержащейся в предметной области. В дальнейшем будут рассматриваться структурированные модели данных. Для этих моделей существует **4 основных уровня моделей**: концептуальный, даталогический или логический, физический и уровень внешних моделей (рис. 1). В соответствии с подходом Питера Чена, изложенным в основополагающей работе по диаграммам "сущность-связь", выделим **четыре уровня представления моделей данных,**

Рис. 1. Четырехуровневая модель представления данных

Концептуальный уровень – наиболее общее представление об информационном содержании предметной области. Представляется в виде **концептуальной модели** (КМ), которая часто называется концептуальной схемой или информационной структурой. КМ обладает высокой степенью стабильности, она проблемно-ориентирована и не зависит от конкретной СУБД, операционной системы и аппаратного обеспечения. Ее поведение должно быть полностью предсказуемо.

На первом уровне описание предметной области строится так, чтобы оно было как можно более общим, не зависело от особенностей выбираемой впоследствии СУБД, а информация была бы доступна широкой категории пользователей: от заказчиков до системных программистов, которые будут заниматься проектированием БД на основе этой модели. Для этого исходная информация о предметной области анализируется и представляется в некотором формализованном виде. Это формализованное описание предметной области должно отражать ее специфику и использоваться на следующих этапах проектирования структуры БД в контексте особенностей выбранной конкретной СУБД. Такое формализованное описание предметной области называется **концептуальной моделью.**

Концептуальное представление оперирует основными элементарными данными предметной области, называемыми **сущностями**. Сущности описываются атрибутами. Данные могут находиться в некотором отношении друг с другом: образовывать ассоциации. Эти ассоциации называются связями. КМ должна поддерживать согласованность связей в пределах уровня детализации.

Различаются две важных стадии концептуального проектирования: **анализ данных и организация их хранения.**

Содержание первой стадии – сбор полной и точной информации о данных предметной области. Заметим, что речь идет о первоначальном сборе информации, в процессе проектирования, как правило, выясняются дополнительные обстоятельства. Нередко для проведения данной работы прибегают к одному из двух методов (или к обоим): анкетированию и работе с экспертами.

Вторая стадия сводится к разработке графического представления полученной информации в виде схемы, которая включает, в частности, исходные данные с формирующими их процессами и результирующие со ссылкой на использующие процессы. На этом же этапе уточняется степень важности данных, выявляются и фиксируются связи между ними. К данной работе, наряду с проектировщиком, полезно привлекать администратора баз данных и представителей пользователя.

Обычно для концептуального представления используется модель «Сущность-Связь» (ER-модель), введенная Ченом, которая графически выражается ER-диаграммами. Существуют различные модификации представления (нотации) диаграмм. Добавим, что, согласно предложению Чена, не только сущности, но и связи могут иметь атрибуты, выражающие их свойства. Представление модели внешне напоминает структуру базы данных и служит для отображения на логическую модель.

Логический уровень представления оперирует такими понятиями, как запись, компоненты записи, связи между записями. Соответствующая ему модель называется логической (ЛМ), она представляет собой отображение концептуальной модели в среду

конкретной СУБД. Иногда рассматривают не конкретную СУБД, а только ее класс (модель) – иерархическую, сетевую или реляционную.

На этом этапе необходимо определить отношения и атрибуты, выделить ключи. На ряд атрибутов могут быть наложены ограничения, которые выражаются в функциональных зависимостях между ними.

В результате должна сформироваться логическая схема БД, находящаяся в 3 нормальной форме. Эта схема, разумеется, не окончательная, в процессе проектирования она может неоднократно корректироваться, в результате чего нормализованность может нарушиться. В этом случае добавляется специальный этап **нормализации** схемы.

Физический уровень демонстрирует физическое хранение данных. На этом уровне используются такие понятия, как физические блоки, файлы, хранимые записи, указатели.

Под физической моделью БД понимают способ размещения данных на устройствах внешней памяти и способ доступа к этим данным. Каждая СУБД по-разному организует методы хранения и доступа к данным, однако поскольку в подавляющем большинстве случаев СУБД работает под управлением конкретной операционной системы, эти методы в большой степени зависят от методов работы с данными самой операционной системы.

Разрабатывается так называемая схема хранения данных. Поскольку в разных СУБД имеются различные возможности и особенности физической организации данных, то физическое моделирование проводится только после разработки логической модели.

Уровень внешних модулей. Ряд современных СУБД обладают возможностями описания структуры БД с точки зрения конкретного пользователя. Такое описание называется внешней моделью. Для каждого типа пользователей внешнее моделирование позволяет разработать подсхему БД исходя из потребностей различных категорий пользователей.

2. Этапы проектирования баз данных

Процесс проектирования БД является итерационным – зачастую происходит возврат к предыдущим этапам для пересмотра принятых ранее решений.

Рис.2. Этапы проектирования базы данных

Онтологическое моделирование – формализация предметной области и ее описание с помощью концептульных схем. Исторически, понятие онтологии появилось в одной из ветвей философии, называемой метафизикой, которая изучает устройство реального мира. Основной характерной чертой онтологического анализа является, в частности, разделение реального мира на составляющие (классы объектов) и выявление совокупности

Онтологический анализ обычно начинается с составления словаря терминов, который используется при обсуждении и исследовании характеристик объектов и процессов, составляющих рассматриваемую систему, а также создания системы точных определений этих терминов. Кроме того, документируются основные логические взаимосвязи между соответствующими введенным терминам понятиями.

В дальнейшем мы не будем делать различия между понятиями и терминами. Результатом этого анализа является онтология системы, или же совокупность словаря терминов, точных их определений взаимосвязей между ними.

Для создания онтологического описания предметной области существует **стандарт онтологического моделирования IDEF5.**

Рис.3 Классификация сущностей предметной области

На рисунке 3 показана схема классификации объектов (сущностей) предметной области.

Объекты взаимодействуют между собой через свои свойства, что порождает ситуации (прецеденты). Все объекты предметной области находятся между собой в определенных отношениях.

Связи между объектами могут выть очень разнообразными, но для проектирования баз данных наиболее важны следующие типы связей: **классификация и композиция.** Классификация означает разбиение всех объектов по типам. Композиция означает утверждение: данный объект состоит из …__

При проектировании базы важны количественные отношения между экземплярами различных сущностей, т. е. важно знать, сколько экземпляров сущности X соответствует одному экземпляру сущности Y (и наоборот).

Ситуации – это взаимосвязи, выражающие взаимоотношения между объектами.

Ситуации в предметной области описываются посредством высказываний о предметной области, например, объект, состоит из … (композиция), объект это … либо …. (классификация). Имеются более сложные связи: «Студент – лицо, зачисленное в установленном порядке в учебное заведение». Здесь связь между объектом студент и объектом «учебное заведение», носит более сложный характер. Она формирует признаки и ограничения объекта «студент». Роль прецедентов для описания понятийной модели предметной области заключается в том, что они содержат в себе ограничения, типы и структуры данных. Выявление ограничений, заложенных в прецедентах, является одним из этапов формализации предметной области и разработки концептуальной модели предметной области. Для прецедентов часто используется термин бизнес-правила.

Рис.4 Классификация ситуация предметной области

После того, как онтологическая модель создана, т. е. вывялены все объекты предметной области, их атрибуты и связи между ними, можно переходить к следующему этапу – созданию инфологической (информационно-логической) модели предметной области.

Инфологическая (информационно-логическая) **модель** – это информационная модель предметной области базы данных, ориентированная на человека и не зависимая от типа СУБД.

Инфологическая модель определяет совокупности информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области, а также характер информационных потребностей пользователей.

Цель инфологического моделирования обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.

Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Состав инфологической модели

−определения сущностей;
−уникальные идентификаторы сущностей;
−отношения между сущностями;
−диаграммы «сущность-связь» (Entity – Relationship Diagrams);
−определения атрибутов сущностей;
−супертипы и подтипы.

Сущность (объект) – описывается с помощью данных, именуемых свойствами или атрибутами (attributes) сущности. Сущность — это понятие, концепт, воображаемый объект, для которого у человека может быть создан образ. Сущность задает некоторый набор объектов описываемого мира, в том числе процессов или атрибутов.

Атрибуты – являются определениями в высказывании о сущности и обозначаются именами существительными естественного языка. Сущности вступают в связи друг с другом через свои атрибуты.

Экземпляр (instance) сущности. Каждая группа атрибутов, описывающих одно реальное проявление сущности, представляет собой экземпляр сущности.

Иными словами, экземпляры сущности – это конкретные проявления сущности, отличающиеся друг от друга и допускающие однозначную идентификацию.

Пример:
Сущность – студент.
Атрибуты: ФИО, дата рождения,
Экземпляр сущности: студент Петров А.Б, 1978 г.р.

Идентификатор сущности (Entity identifier) Одним из основных компьютерных способов распознавания сущностей в базе данных является присвоение сущностям идентификаторов. Задача выбора идентификатора сущности является субъективной задачей. Некоторые сущности имеют естественные идентификаторы. Например, естественным идентификатором счета-фактуры является его номер.

Ключ. Уникальный идентификатор экземпляра сущности называют ключом. Поскольку сущность определяется набором своих атрибутов, то для каждой сущности целесообразно выделить такое подмножество атрибутов, которое однозначно идентифицирует данную сущность.

Первичный ключ. Если экземпляры сущности имеют несколько уникальных идентификаторов (ключей), то проектировщик должен выбрать первичный ключ сущности.

Домен. Каждый атрибут сущности имеет домен. Домен – это область значений атрибута.

Проектировщик базы данных должен проконтролировать, чтобы в информационной модели предметной области для каждого атрибута сущностей был определен домен.

Сущности не существуют отдельно друг от друга. Между ними имеются реальные отношения, и они должны быть отражены в информационной модели предметной области.

Отношения и связи (Relationship). Отношение (связь) представляет собой взаимосвязь между двумя или более сущностями. Самый распространенный тип связей – информационные.

Зависимость между сущностями определяет связь между ними.

Связи делятся на три типа по степени связи сущностей: один-к-одному (1:1), один-ко-многим (или многие-к-одному) (1:M, M:1), многие-ко-многим (M:M). В общем случае между двумя сущностями может быть задано произвольное количество связей с различными смысловыми нагрузками.

Класс принадлежности сущности может быть **обязательным** (О), когда в связи должен участвовать каждый экземпляр сущности, и **необязательным** (Н), когда не все экземпляры сущности должны участвовать в связи. Связь может быть обязательной со стороны одной сущности и необязательной со стороны другой. На практике степень связи и класс ее принадлежности всегда определяются, исходя из анализа предметной области.

Рис.5. Примеры изображения связей между сущностями в различных нотациях

Преобразование инфологической модели в даталогическую

После того как инфологическая модель готова, необходимо преобразовать ее в даталогическую модель. Это преобразование можно выполнить в соответствии со следующим алгоритмом.

Алгоритм преобразования инфологической модели к реляционной (**даталогической**) модели данных:

1. Каждому набору сущностей поставить в соответствие отношение (таблицу) в базе данных.
2. Каждый атрибут набора сущностей соотнести с атрибутом отношения (определив его тип и дополнительные условия для обеспечения поддержания целостности данных).
3. Ключ набора сущности превращается в первичный ключ отношения.
4. В каждое отношение подчиненного набора сущностей добавляется набор атрибутов основного набора сущностей, являющихся первичным ключом. Этот набор атрибутов становится внешним ключом отношения (FOREIGN KEY).
5. При необязательности связи для атрибутов внешнего ключа устанавливается допустимость значения NULL (пустого значения), при обязательной связи – NOT NULL (недопустимость пустого значения атрибутов).
6. Для связи ―многие ко многим вводится дополнительное отношение, связанное с исходными отношениями связью ―один ко многим. Его атрибуты – первичные ключи исходных отношений.

Рис. 6. Алгоритм перехода к реляционной модели

Физическое проектирование – создание схемы базы данных для конкретной СУБД.

Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т.п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

3. Диаграмма "сущность-связь». Средства создания ER-модели

Для удобства проектирования концептуальной модели и повышения ее наглядности и легкости восприятия пользователями сущности, экземпляры сущностей и связи между ними представляются графически в виде диаграмм ER-экземпляров, диаграмм ER-типов (рис. 7) и **ER-диаграмм**. Представленная в виде таких диаграмм модель БД называется **ER-моделью.**

Таким образом, **ER-модель** является графическим описанием предметной области в терминах "сущность-свойство-связь" и представляет собой один из наиболее важных элементов концептуальной модели БД. Использование ER-моделирования прежде всего является удобным средством документирования проектируемой БД, не привязанным к какой-либо конкретной СУБД, что важно, поскольку, с одной стороны, выбор СУБД может быть произведен на более поздних этапах, а с другой стороны, при необходимости выбора другой СУБД не нужно заново проектировать модель БД.

При разработке ER-моделей мы должны получить следующую информацию о предметной области:

- Список сущностей предметной области.
- Список атрибутов сущностей.
- Описание взаимосвязей между сущностями.

ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы.

Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Рис.7. Основные понятия модели «сущность-связь»

Существует большое количество различных нотаций для построения ER-модели. В этих нотациях используются разные символы, линии и фигуры для указания на характеристики объектов БД и связей между ними. Так, в частности, для определения сущности используется, как правило, прямоугольный блок. Под блоком сущности указывается ее ключ, для чего часто используется подчеркивание ключевых атрибутов сущности. Обязательное участие сущности в связи может быть указано блоком с точкой внутри, смежным с блоком этой сущности. При необязательности связи дополнительный блок к блоку сущности не добавляется, а точка располагается на линии связи. Кроме того, на линии связи расставляются символы, указывающие на ее степень.

Построение ER-диаграммы нотации Баркера

Мы опишем работу с ER-диаграммами близко к нотации Баркера, как довольно легкой в понимании основных идей.

Сущность - это класс однотипных объектов, информация о которых должна быть учтена в модели. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе.

Примерами сущностей могут быть такие классы объектов как "Поставщик", "Сотрудник", "Накладная". Каждая сущность в модели изображается в виде прямоугольника с наименованием:

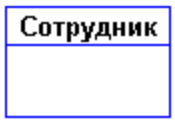


Рис. 8

Экземпляр сущности - это конкретный представитель данной сущности.

Например, представителем сущности "Сотрудник" может быть "Сотрудник Иванов". Экземпляры сущностей должны быть различимы, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

Пример класса сущностей СТУДЕНТ и конкретного экземпляра сущности показан на рис.

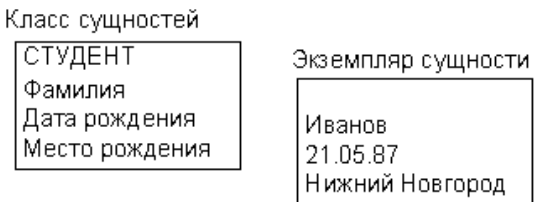


Рис.9. Класс сущностей и экземпляр сущности

Атрибут сущности - это именованная характеристика, являющаяся некоторым свойством сущности. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

Примерами атрибутов сущности "Сотрудник" могут быть такие атрибуты как "Табельный номер", "Фамилия", "Имя", "Отчество", "Должность", "Зарплата" и т.п.

Атрибуты изображаются в пределах прямоугольника, определяющего сущность:

Рис. 10

Ключ сущности - это неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Неизбыточность заключается в том, что при удаление любого ключа из сущности нарушается его уникальность. Сущность может иметь несколько различных ключей.

Ключевые атрибуты изображаются на диаграмме подчеркиванием:

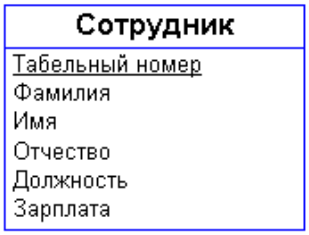


Рис. 11

Связь - это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с нею.

Например, связи между сущностями могут выражаться следующими фразами - "СОТРУДНИК может иметь несколько ДЕТЕЙ", "каждый СОТРУДНИК обязан числиться ровно в одном ОТДЕЛЕ".

Графически связь изображается линией, соединяющей две сущности:

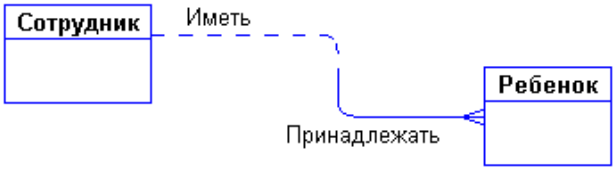


Рис. 12

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: "иметь", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих типов связи:

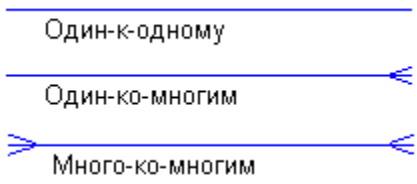


Рис. 13

Связь типа один-к-одному означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего

свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа один-ко-многим означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется родительской, правая (со стороны "много") - дочерней. Характерный пример такой связи приведен на Рис. 4.

Связь типа много-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи много-ко-многим является временным типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Каждая связь может иметь одну из двух модальностей связи:

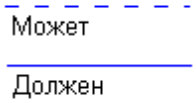


Рис. 14

Модальность "может" означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть и не связан ни с одним экземпляром. Модальность "должен" означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности.

Связь может иметь разную **модальность** с разных концов (как на Рис. 4).

Описанный графический синтаксис позволяет однозначно читать диаграммы, пользуясь следующей схемой построения фраз:

<Каждый экземпляр СУЩНОСТИ 1> <МОДАЛЬНОСТЬ СВЯЗИ> <НАИМЕНОВАНИЕ СВЯЗИ> <ТИП СВЯЗИ> <экземпляр СУЩНОСТИ 2>.

Каждая связь может быть прочитана как слева направо, так и справа налево. Связь на Рис. 12 читается так:

Слева направо: "каждый сотрудник может иметь несколько детей".

Справа налево: "Каждый ребенок обязан принадлежать ровно одному сотруднику".

Некоторые примеры, относящиеся к различным нотациям, приведены на рис.13-15.

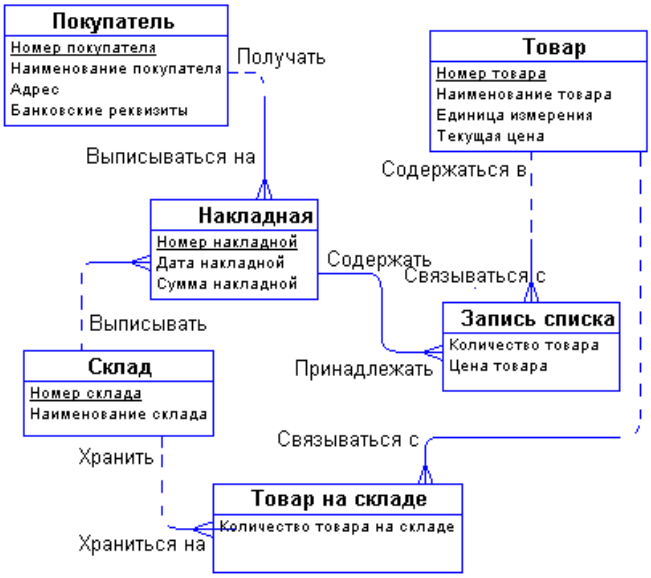


Рис. 13.

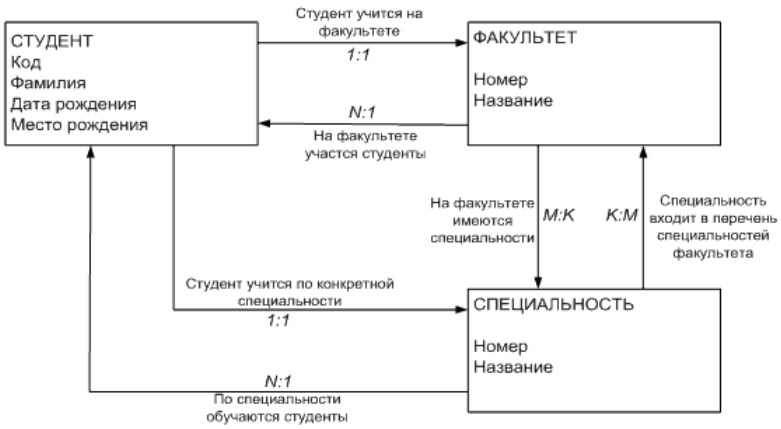


Рис. 14.

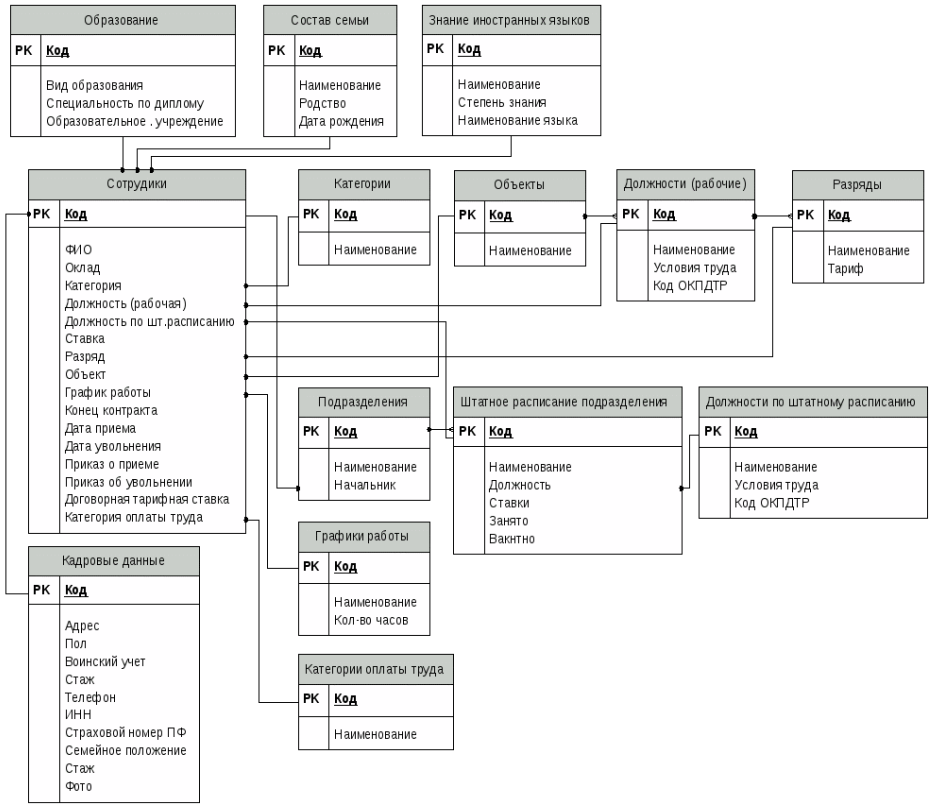


Рис. 15

В настоящее время существует целый ряд автоматизированных средств проектирования ER-моделей, называемых CASE-средствами. Среди них следует отметить Design/IDEF, Power Designer (S-Designor), Oracle, ERWin и т.п. Использование этих средств дает ряд преимуществ, а именно: снижаются требования к знанию языков описания данных и конкретных СУБД; автоматически контролируются целостность и согласованность схемы описания БД; сокращается в целом время проектирования; появляется возможность автоматического тестирования проекта на разных стадиях его проектирования.

Пример проектирования БД MySQL.

1 Инфологическое проектирование

1.1 Анализ предметной области и информационных задач пользователей

Основная задача любой библиотеки — обработка книжного фонда. Нетрудно выделить три основные группы пользователей системы: читатель, библиотекарь, администратор. Деятельность каждого из них показана на диаграмме вариантов использования (рисунок 1).

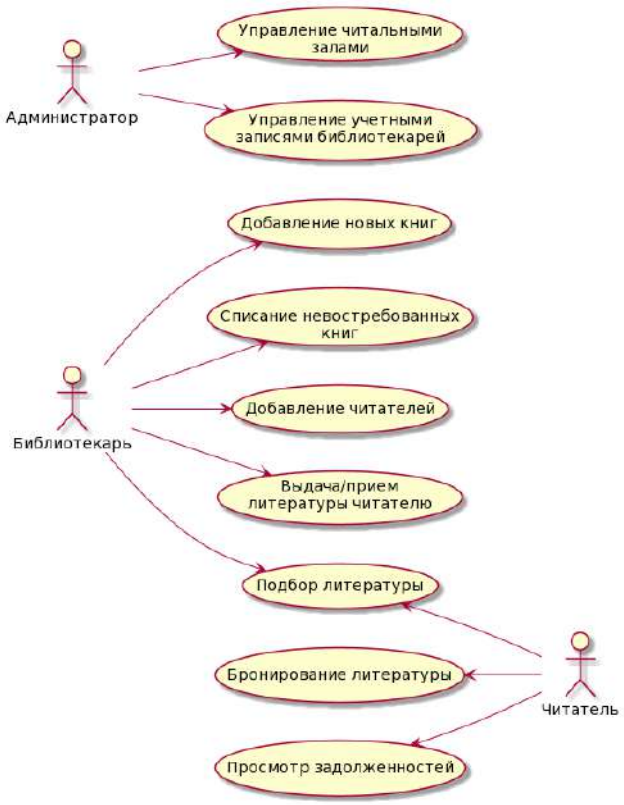


Рисунок 1.

Уже сейчас можно выделить некоторые сущности и отношения будущей базы данных (рисунок 2).

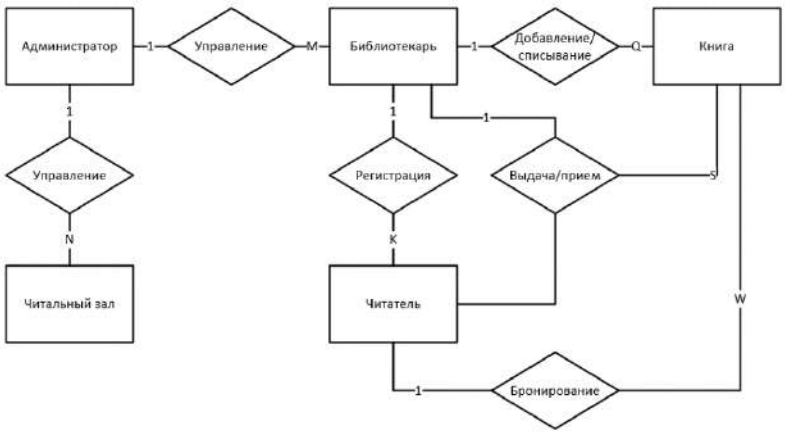


Рисунок 2.

При таком подходе не понятно как именно связать читателя с книгой (у читателя не проставлена арность в отношении «выдача/прием». Если книга имеет несколько экземпляров — то она может быть выдана нескольким читателям. Даже если же под книгой понимать один экземпляр — то при сохранении в таблице книг текущего читателя приведет к невозможности получения информации о том, кто (и сколько раз) брал эту книгу ранее.

Решением может быть введение дополнительной сущности — карточки о выдаче книги. При выдаче книги читателю заводится карточка, а при сдаче книги — в нее ставится соответствующая пометка. С помощью этих карточек определяются задолженности каждого пользователя и вычисляется статистика использования книг.

При бронировании литературы читателем — также заводится карточка, если забронированная литература не взята читателем в определенный срок — карточка уничтожается. Существует ограничение на количество книг, которые может забронировать читатель.

При подборе литературы пользователь просматривает каталог литературы с возможностью фильтрации результатов поиска по автору, названию, году издания.

Есть возможность расчета статистики по всем книгам библиотеки, при этом количество выданных экземпляров книги за заданный период времени. Также можно задать минимальное число экземпляров книг, для которых выполняется расчет. На основании этой статистики производится списание неиспользуемых книг из библиотеки.

Можно выделить следующие основные сущности предметной области:

- пользователь (библиотекари и администраторы);
- читатель;
- читальный зал;
- книга;
- карточка выдачи книги;
- карточка бронирования книги.

Доработанная ER- диаграмма базы данных приведена на рисунке 3.

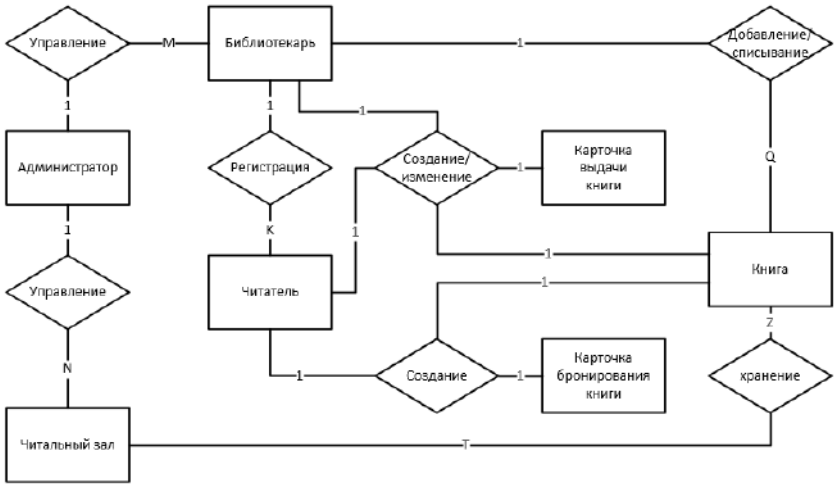


Рисунок 3 — ER диаграмма база данных (вариант 2)

В соответствии с прецедентами, показанными на рисунке 1, база данных должна реализовывать, следующие запросы (не полный перечень):

- отобразить книги, соответствующие заданным условиям;
- отобразить пользователей, имеющих незакрытые вовремя карточки выдачи книг (библиотекарь ищет должников);
- отобразить все книги, соответствующие незакрытым вовремя карточкам выдачи книг заданного пользователя (пользователь пришел в библиотеку за новыми книгами — надо посмотреть является ли он должником и сообщить ему об этом);
- удалить все карточки бронирования, созданные более чем N секунд назад;
- отобразить все книги, соответствующие незакрытым карточкам бронирования книг заданного пользователя (читатель заказал книги и пришел в библиотеку за ними — библиотекарю надо получить этот список чтобы выдать).

1.2 Формирование схемы данных

Для формирования схемы данных необходимо сначала дополнить ER-диаграмму реквизитами сущностей (уточнить ее). Иногда, при этом удается найти ошибки построения

ER-диаграммы — в этой задаче было обнаружено, что книгу необходимо «как-то» связать с залом библиотеки. Сделать это можно поместив в книгу реквизит «номер зала», однако при таком подходе одну и ту же книгу придется описывать в базе несколько раз (если она встречается в разных залах). Более правильный подход заключается во введении дополнительной сущности «размещение книги». На рисунке 4 показана ER-диаграмма с добавленной сущностью и реквизитами.

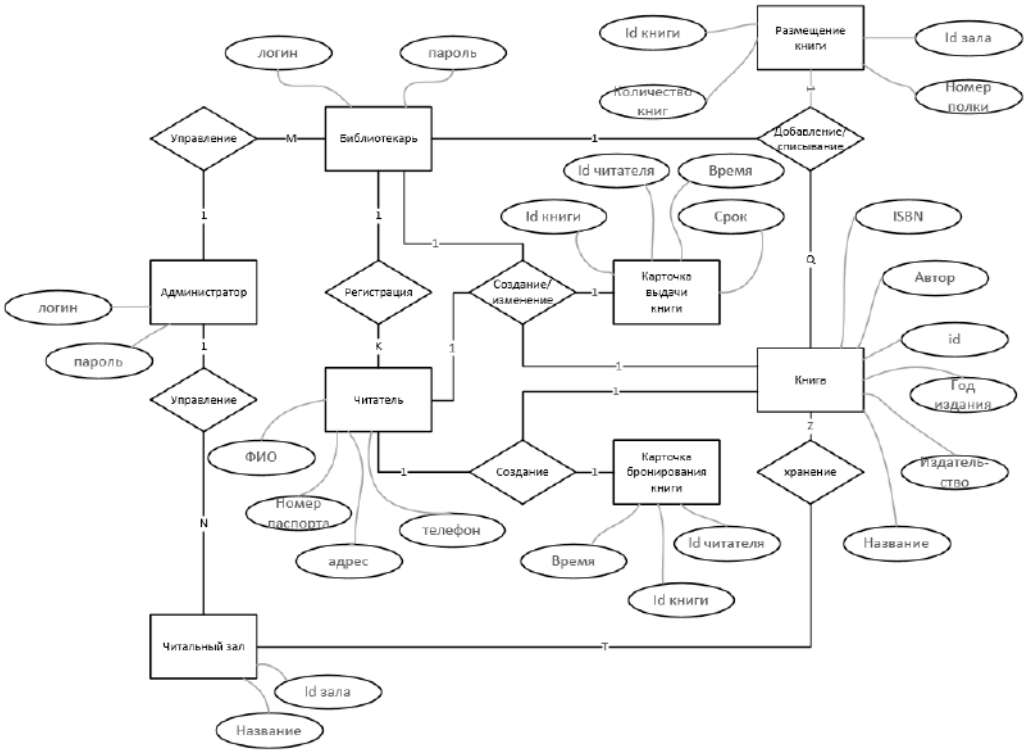


Рисунок 4.

Приведенная ER-диаграмма отражает основные таблицы, связи и атрибуты, на ее основе можно построить модель БД. На ER-диаграммы нет стандарта, но есть ряд нотаций (Чена, IDEFIX, Мартина и т.п.), но на модель предметной области не удалось найти ни стандарта, ни нотаций. Однако, в ходе построения такой диаграммы обязательно выделяются ключевые поля (внешние и внутренние), иногда — индексы и типы данных. Схема базы данных, приведенная на рисунке 5, выполнена с использованием открытого инструмента при этом:

- для связей используется нотация Мартина («вороньи лапки»);
- таблицы изображены прямоугольниками, разделенными на 3 секции:
  - имя таблицы;
  - внутренние ключи (помечаются маркером);
  - остальные поля, при этом обязательные помечаются маркером.

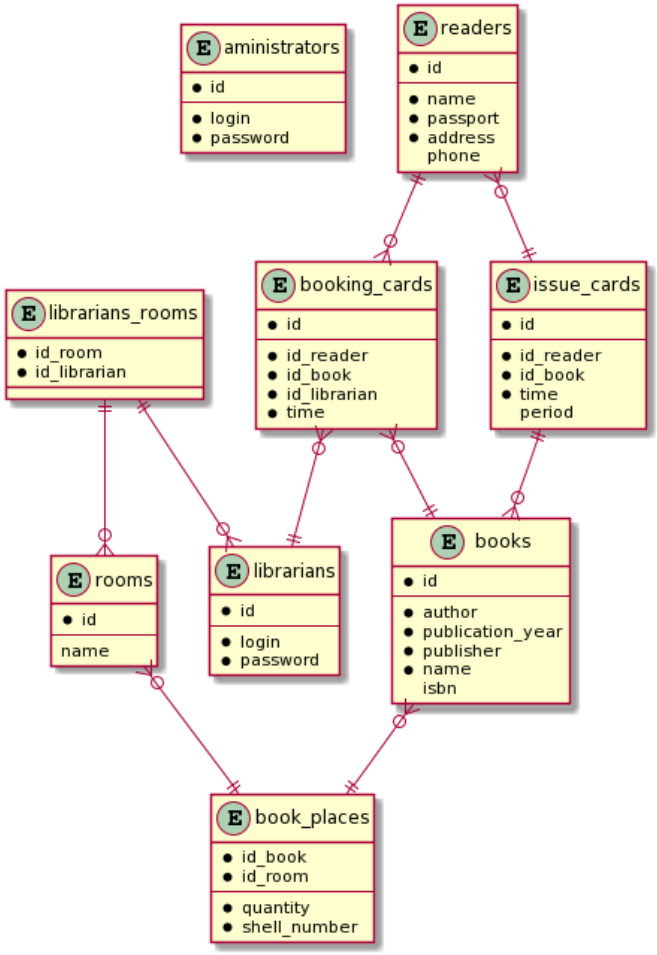


Рисунок 5.

При разработке этой модели возникало желание объединить таблицу администраторов с таблицей библиотекарей — добавить таблицу users, однако:

- администратор не связан с конкретным залом (пришлось бы заполнять соответствующее поле **null** -значениями);
- вероятно, это осложнило бы распределение прав доступа — сейчас доступ к таблице administrators имеет только администратор базы данных (работающий через специальную панель СУБД и не имеющий учетной записи в разрабатываемой системе). Однако при соединении таблиц пользовательские запросы требовали бы доступа к новой таблице.

При построении этой диаграммы был найден и исправлен недочет ER-диаграммы — добавлена таблица librarians_rooms , объединяющая библиотекарей и залы. Это нужно, так как один библиотекарь может работать в нескольких залах, но несколько библиотекарей могут работать в одном и том же зале.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Осуществите проектирования реляционной схемы базы данных по индивидуальному варианту (см. Приложение А).

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что входит в Проектирование БД?
2. Что входит в Моделирование данных?
3. Что такое «информационная модель данных»
4. Дайте описание Концептуальному уровню БД.
5. Дайте описание Логическому уровню БД.
6. Дайте описание Физическому уровню БД.
7. Что понимается под физической моделью БД?
8. Что понимается под Онтологическим моделированием?
9. Какая модель Бд является Инфологической (информационно-логической)?
10. Дайте определение понятию «Сущность (объект)».
11. Дайте определение понятию Атрибуты».
12. Дайте определение понятию Экземпляр (instance) сущности».
13. Дайте определение понятию Идентификатор сущности».
14. Дайте определение понятию Первичный ключ».
15. Дайте определение понятию Ключ».
16. Дайте определение понятию Домен».

Лабораторная работа №1
«ПРИВЕДЕНИЕ БД К НОРМАЛЬНОЙ ФОРМЕ 3НФ»

Цель работы: получить теоретические знания и практические навыки по вопросам нормализации отношений в базы данных, приведении базы данных к нормальной 3НФ.

Теоретические сведения

К проектированию БД можно подойти и более формально, превратив процесс проектирования в математическую процедуру. Цель этой процедуры одна – спроектировать оптимальную структуру данных. Рассмотрим предпосылки такого подхода.

При проектировании реляционной БД одним из основных этапов является **логическое проектирование**, которое заключается прежде всего в определении количества отношений со своими схемами, установлении количества и типов связей между отношениями. Другими словами, на этом этапе определяется **структуризация данных**.

В основе классического процесса проектирования лежит метод нормализации, который опирается на декомпозицию (разложение) отношения, находящегося в предыдущей нормальной форме, в два или более отношений, удовлетворяющих требованиям следующей нормальной формы.

Нормализация – разбиение таблицы на две или более, обладающие лучшими свойствами при добавлении, изменении и удалении данных. Нормализация осуществляется с целью оптимизации объема БД, быстродействия запросов и т.п. Вообще-то нормальных форм 5, но реально, на практике, используются 3 нормальные формы, точнее база данных в третьей нормальной форме (3НФ).

Процесс нормализации отношений осуществляется пошагово и заключается в последовательном переводе отношения от первой нормальной формы к нормальным формам более высокого порядка. При этом каждая следующая нормальная форма сохраняет все свойства предыдущих.

Важно еще раз подчеркнуть, что при нормализации отношений происходит их декомпозиция, т. е. после завершения этого процесса окончательный и исходный наборы отношений будут различными. Это означает, что исходная и окончательная схемы БД будут разными. Однако в основе теории нормализации лежит принцип обратимости, в соответствии с которым на любом этапе процесса нормализации исходная схема БД всегда может быть однозначно восстановлена.

Рис. 1. Проектирование логической схемы БД

Основные преимущества нормализации:

1. Лучшая общая организация базы данных
2. Сокращение избыточности информации
3. Непротиворечивость информации внутри базы данных

4. Более гибкий проект базы данных
5. Большая безопасность данных

Неформально цель нормализации можно описать так: один факт из предметной области должен храниться в одном месте (в одной записи одного отношения базы данных).

Метод нормальных форм основывается на фундаментальном понятии **зависимости** (функциональной зависимости) между атрибутами отношений. Функциональные зависимости — основной вид ограничения целостности данных в БД, которые не могут быть установлены никаким формальным алгоритмом — только проектировщиком БД.

Говорят, что один **атрибут функционально зависит от другого** (A→B, т.е. атрибут B зависит от атрибута A), если каждому значению независимого атрибута A однозначно соответствует определенное значение зависимого атрибута B.

Частичной функциональной зависимостью называется зависимость неключевого атрибута от части составного первичного ключа. Например, атрибут *Должн* находится в функциональной зависимости от атрибута *ФИО*, являющегося частью ключа.

Если же не ключевой атрибут зависит от всего составного первичного ключа, то такая зависимость называется **полной**.

Транзитивной зависимостью атрибута C от атрибута A называется такая зависимость, когда атрибут B функционально зависит от атрибута A, атрибут C зависит от B и при этом атрибут B не зависит от атрибута C. Если для атрибутов A,B,C выполняютсяусловия A—> B и

B—>C, но обратная зависимость отсутствует. Например, транзи-тивной зависимостью связаны атрибуты: ФИО-»Должн-»Оклад

Между атрибутами может иметь место многозначная зависимость. **Многозначной зависимостью** атрибута B от атрибута A называется такая зависимость, когда данному значению атрибута A соответствует множество значений атрибута B, не зависящих от других атрибутов в данном отношении. Многозначные зависимости могут быть: «один ко многим»(1:M),«многие к одному»(M:1) или «многие ко многим»(M:M), обозначаемые соответственно: $A \Rightarrow B$, $A \Leftarrow B$ и $A \Leftrightarrow B$.

Взаимно независимыми атрибутами отношения называются такие атрибуты, когда ни один из них не является функционально зависимым от других атрибутов.

Процесс проектирования БД с использованием метода нормальных форм является итерационным и заключается в последовательном переводе отношений из первой нормальной формы в нормальные формы более высокого порядка по определенным правилам. Каждая следующая нормальная форма ограничивает определенный тип функциональных зависимостей, устраняет соответствующие аномалии при выполнении операций над отношениями БД и сохраняет свойства предшествующих нормальных форм.

Выделяют следующую последовательность нормальных форм:

1. первая нормальная форма (1НФ);
2. вторая нормальная форма (2НФ);
3. третья нормальная форма (3НФ);
4. усиленная третья нормальная форма, или нормальная форма Бойса-Кодда (БКНФ);
5. четвертая нормальная форма (4НФ);
6. пятая нормальная форма (5НФ).

Первая нормальная форма. Отношение находится в 1НФ, если все его атрибуты являются простыми (имеют единственное значение). Исходное отношение строится таким образом, чтобы оно было в 1НФ.

Исходное отношение

Преподаватель	Предмет	Группа
Петренко А.Л.	Линейная алгебра	12-01
	Аналитическая геометрия	11-02
Грушин К.К.	Общая физика	07-12
	Молекулярная динамика	12-03

Отношение в 1НФ

Преподаватель	Предмет	Группа
Петренко А.Л.	Линейная алгебра	12-01
Петренко А.Л.	Аналитическая геометрия	11-02
Грушин К.К.	Общая физика	07-12
Грушин К.К.	Молекулярная динамика	12-03

Рис. 2. Приведение реляционного отношения к первой нормальной форме

Перевод отношения в следующую нормальную форму осуществляется методом «декомпозиции без потерь». Такая декомпозиция должна обеспечить то, что запросы (выборка данных по условию) к исходному отношению и к отношениям, получаемым в результате декомпозиции, дадут одинаковый результат.

Отношение находится в 1НФ тогда и только тогда, когда ни одна из его записей не содержит в поле более одного значения и ни одно из ключевых полей не пусто.

Эта частичная зависимость от ключа приводит к следующему:

1. В отношении присутствует явное и неявное избыточное дублирование данных, например:

• повторение сведений о стаже, должности и окладе преподавателей, проводящих занятия в нескольких группах и/или по разным предметам;

• повторение сведений об окладах для одной и той же должности или о надбавках за одинаковый стаж.

2. Следствием избыточного дублирования данных является проблема их редактирования. Например, изменение должности у преподавателя Иванова И.М. потребует просмотра всех кортежей отношения и внесения изменений в те из них, которые содержат сведения о данном преподавателе.

Часть избыточности устраняется при переводе отношения в 2НФ.

Вторая нормальная форма. Отношение находится во второй нормальной форме (2NF), если оно находится в 1NF и все его атрибуты, не входящие в первичный ключ (неключевые), связаны полной функциональной зависимостью с атрибутами первичного ключа. Для устранения частичной зависимости и перевода отношения в 2НФ необходимо, используя операцию проекции, разложить его на несколько отношений следующим образом:

• построить проекцию без атрибутов, находящихся в частичной функциональной зависимости от первичного ключа;

• построить проекции на части составного первичного ключа и атрибуты, зависящие от этих частей.

В результате получим два отношения R1 и R2 в 2НФ (рис. 3).

Рис. 3. Приведение реляционного отношения
ко второй нормальной форме

Приведенное на рис.3 исходное отношение отражает процесс сдачи студентами экзаменационной сессии. С учетом того, что студенты сдают несколько предметов, ясно, что первичный ключ в этом отношении является составным: для идентификации единственного кортежа отношения необходимо задать значения номера зачетной книжки студента и названия предмета.

Исходное отношение не находится во второй нормальной форме, поскольку в нем имеются неполные функциональные зависимости неключевых атрибутов от атрибутов первичного ключа, а именно: неключевые атрибуты “ФИО”и “Группа”однозначно определяются значением только номера зачетной книжки. Самым простым способом приведения исходного отношения ко второй нормальной форме является декомпозиция его на 2 новых отношения, причем таким образом, чтобы в одном из них оказались как раз те неключевые атрибуты, которые в исходном отношении давали неполные функциональные зависимости. На рис. 2.3 представлены схемы этих новых отношений.

Исследование отношений R1 и R2 показывает, что переход к 2НФ позволил исключить явную избыточность данных в таблице.

Для дальнейшего совершенствования отношения необходимо преобразовать его в ЗНФ.

Третья нормальная форма. Отношение находится в третьей нормальной форме (3NF), если оно находится в 2NF и все его неключевые атрибуты взаимно независимы и полностью зависят от первичного ключа.

Другое определение: отношение находится в третьей нормальной форме, если и только если оно находится во второй нормальной форме и ни одно из его неключевых полей не зависит функционально от любого другого неключевого поля (т.е. отсутствуют транзитивные функциональные зависимости).

Доказать справедливость этого утверждения несложно. Действительно, то, что неключевые атрибуты полностью зависят от первичного ключа, означает, что данное отношение находится в форме 2НФ. Взаимная независимость атрибутов (определение приведено выше) означает отсутствие всякой зависимости между атрибутами отношения, в том числе и транзитивной зависимости между ними. Таким образом, второе определение ЗНФ сводится к первому определению.

Приведенное на рис. 4 отношение содержит информацию о том, в какой группе числится студент, к какой кафедре относится эта группа и на каком факультете расположена эта кафедра. Первичным ключом в этом отношении, очевидно, является атрибут “№ зачетной книжки”. Ключ является простым, и отношение удовлетворяет определению второй нормальной формы. Однако отношение не находится в третьей нормальной форме, поскольку в нем имеется следующая транзитивная зависимость: №_зачетной книжки → Группа → Кафедра → Факультет. Приведение к третьей нормальной форме так же, как и в предыдущем случае, может быть произведено путем декомпозиции. Однако теперь уже требуется разбиение на 3 новых отношения, в каждом из которых транзитивные зависимости отсутствуют.

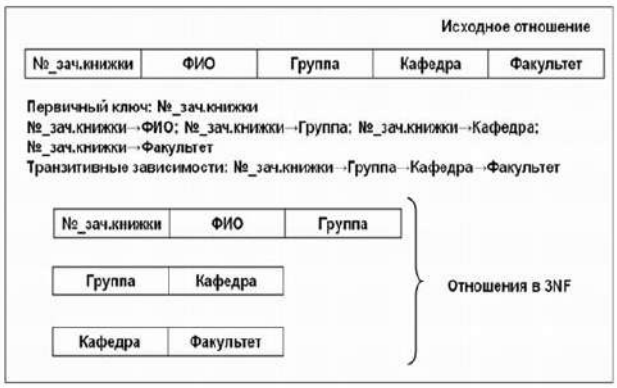


Рис. 4. Приведение реляционного отношения к третьей нормальной форме

На практике построение ЗНФ схем отношений в большинстве случаев является достаточным и приведением к ним процесс проектирования реляционной БД заканчивается. Действительно, приведение отношений к ЗНФ в нашем примере, привело к устранению избыточного дублирования.

Если в отношении имеется зависимость атрибутов составного ключа от неключевых атрибутов, то необходимо перейти к усиленной ЗНФ.

Усиленная ЗНФ или нормальная форма Бойса - Кодда (БКНФ). Отношение находится в БКНФ, если оно находится в 3НФ и в нем отсутствуют зависимости ключей (атрибутов составного ключа) от неключевых атрибутов.

У нас подобной зависимости нет, поэтому процесс проектирования на этом заканчивается. Результатом проектирования является БД, состоящая из следующих таблиц: Rl, R3, R4, R5. В полученной БД имеет место необходимое дублирование данных, но отсутствует избыточное.

Четвертая нормальная форма. Отношение находится в четвертой нормальной форме (4 NF) тогда и только тогда, когда в случае существования многозначной зависимости атрибута В от атрибута А все остальные атрибуты этого отношения функционально зависят от атрибута А.

На рис. 5 приведено исходное отношение, содержащее 3 атрибута. В этом отношении имеется 2 многозначные зависимости атрибутов “Предмет” и “№ зачетной книжки” от атрибута “Группа”. Для приведения этого отношения к четвертой нормальной форме необходимо декомпозировать его на два отношения таким образом, чтобы атрибуты “Предмет”и “№ зачетной книжки” оказались в разных отношениях, каждое из которых будет удовлетворять определению четвертой нормальной формы.

Рис. 5. Приведение реляционного отношения к четвертой нормальной форме

Пятая нормальная форма. Для определения пятой нормальной формы необходимо ввести понятие зависимости проекции-соединения (project-join зависимо-сти).

Отношение R с атрибутами X, Y, ..., Z удовлетворяет зависимости соединения (X, Y, ..., Z) тогда, когда оно может быть восстановлено без потерь путем соединения своих проекций на X, Y, ..., Z.

Отношение R находится в пятой нормальной форме (нормальной форме проекции-соединения PJ/NF) тогда и только тогда, когда любая зависимость соединения в этом отношении следует из существования в нем некоторого возможного ключа.

Исходное отношение

Преподаватель	Кафедра	Предмет
Глотов В.В.	Радиофизика	Общая физика
Козлов М.К.	Автоматика	Электротехника
Михеев С.В.	Уфология	Электротехника
Орлов М.У.	Автоматика	Колебания и волны
Соколов Б.Р.	Лазерная физика	Газовые лазеры
Соколов Б.Р.	Лазерная физика	Электротехника
Ткачук С.В.	Уфология	Маркетинг
Зотов Г.В.	Высшая математика	Мат. анализ
Белоус Ц.Д.	Высшая математика	Диф. уравнения
Шевченко П.П.	Химия	Мат. анализ

Рис. 5. Таблица базы данных

В качестве примера рассмотрим отношение, приведенное на рис. 6. Если предположить, что один и тот же преподаватель может совмещать работу сразу на нескольких кафедрах и, кроме того, вести занятия по нескольким предметам, то первичным ключом этого отношения является набор всех его атрибутов. Поэтому данное отношение удовлетворяет определению четвертой нормальной формы.

Произведем декомпозицию этого отношения в три новых отношения таким образом, как это показано на рис. 7.

R2={Преподаватель, Кафедра}

Преподаватель	Кафедра
Глотов В.В.	Радиофизика
Козлов М.К.	Автоматика
Михеев С.В.	Уфология
Орлов М.У.	Автоматика
Соколов Б.Р.	Лазерная физика
Соколов Б.Р.	Лазерная физика
Ткачук С.В.	Уфология
Зотов Г.В.	Высшая математика
Белоус Ц.Д.	Высшая математика
Шевченко П.П.	Химия

R3={Преподаватель, Предмет}

Преподаватель	Предмет
Глотов В.В.	Общая физика
Козлов М.К.	Электротехника
Михеев С.В.	Электротехника
Орлов М.У.	Колебания и волны
Соколов Б.Р.	Газовые лазеры
Соколов Б.Р.	Электротехника
Ткачук С.В.	Маркетинг
Зотов Г.В.	Мат. анализ
Белоус Ц.Д.	Диф. уравнения
Шевченко П.П.	Мат. анализ

R4={Кафедра, Предмет}

Кафедра	Предмет
Автоматика	Электротехника
Автоматика	Колебания и волны
Высшая математика	Мат. анализ
Высшая математика	Диф. уравнения
Лазерная физика	Газовые лазеры
Лазерная физика	Электротехника
Радиофизика	Общая физика
Уфология	Электротехника
Уфология	Маркетинг
Химия	Мат. анализ

Рис. 7. Декомпозиция исходного отношения

Если исходное отношение R1 удовлетворяет зависимости проекции соединения (Преподаватель, Кафедра, Предмет), то оно не находится в пятой нормальной форме, поскольку наличие зависимости проекции-соединения определяется наборами атрибутов, не являющихся возможными ключами.

Отношения R2, R3 и R4 находятся в пятой нормальной форме, что можно проверить, получив все попарные соединения (R2, R3), (R2, R4) и (R3, R4) так, как это показано на рис. 8.

Преподаватель	Кафедра	Предмет
Глотов В.В.	Радиофизика	Общая физика
Козлов М.К.	Автоматика	Электротехника
Козлов М.К.	Лаз. физ.	Электротехника
Козлов М.К.	Уфология	Электротехника
Михеев С.В.	Автоматика	Электротехника
Михеев С.В.	Лаз. физ.	Электротехника
Михеев С.В.	Уфология	Электротехника
Орлов М.У.	Автоматика	Кол. и волны
Соколов Б.Р.	Лаз. физ.	Газ. лазеры
Соколов Б.Р.	Автоматика	Электротехника
Соколов Б.Р.	Лаз. физ.	Электротехника
Соколов Б.Р.	Уфология	Электротехника
Ткачук С.В.	Уфология	Электротехника
Ткачук С.В.	Уфология	Маркетинг
Зотов Г.В.	Высш.матем.	Мат. анализ
Зотов Г.В.	Высш.матем.	Электротехника
Белоус Ц.Д.	Высш.матем.	Мат. анализ
Белоус Ц.Д.	Высш.матем.	Электротехника
Шевченко П.П.	Химия	Мат. анализ

Рис. 8. Соединение отношений R3 и R4

Недостатки нормализации

Хотя наиболее удачные базы данных всегда в той или иной степени нормализованы, у нормализованной базы данных есть один существенный недостаток, связанный со снижением ее производительности. Чтобы понять, почему это происходит, необходимо знать, что при выполнении базой данных запросов или транзакций существенную роль начинают играть такие факторы, как использование центрального процессора и памяти, а также система ввода/вывода.

Для обработки транзакций и запросов в случае нормализованной базы данных к центральному процессору, памяти и системе ввода/вывода предъявляются существенно большие требования, чем когда эта база данных ненормализована. Ведь для получения требуемой информации или обработки существующих данных нормализованная база данных должна сначала найти все необходимые таблицы, а затем объединить содержащуюся в них информацию.

Для проведения нормализации можно пользоваться и инфологической моделью базы данных. При этом нормализация такой модели повторяет нормализацию реляционной модели – можно проводить ее на инфологическом уровне.

Вот примерный **алгоритм нормализации ER-модели:**

1. Найти наборы сущностей, скрыто моделирующие несколько взаимосвязанных классов объектов. Если такие есть, то разделить наборы сущностей на несколько наборов и установить между ними связи (1НФ).

2. Проанализировать все наборы сущностей, имеющие составные ключи на наличие неполной функциональной зависимости неключевых атрибутов от атрибутов, входящих в ключ. Если такие есть, то разделить набор сущностей на два набора, для каждого определить ключ, установить между ними связи (2НФ).

3. Проанализировать неключевые атрибуты набора сущностей на наличие функциональных зависимостей друг от друга. Если такие есть, то разделить набор сущностей на несколько наборов, для каждого определить ключ, установить между наборами связи (3НФ).

Продолжение примера проектирования и создания БД MySQL (рассмотренного выше).

Нормализация полученных отношений

Разработанная схема БД находится в:

- первой нормальной форме, так как в качестве доменов выступают только скалярные значения и информация в таблицах не дублируется. Почти во всех таблицах есть идентификатор (id), а в остальных — librarian_rooms и book_places в качестве первичного ключа выступает пара полей, так как нет смысла добавлять одного и того же библиотекаря или книгу дважды в один зал. При повторном добавлении книги (если произошла приемка точно таких же книг) — надо выполнить поиск и изменить число экземпляров в существующей записи;
- второй и третьей нормальных формах, каждый не ключевой атрибут неприводимо и нетранзитивно зависит от первичного ключа. Для всех таблиц нашей БД это очевидно — Логин и Пароль зависят от Id и их нельзя вывести иным образом; количество книг и номер полки зависят от id книги и id комнаты и их тоже нельзя вывести никак иначе.

Таким образом, схема базы данных показанная на рисунках 5 и 6 находится в нормальной форме Бойса-Кодда.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Приведите БД (из предыдущей работы) к нормальной третьей форме (3НФ).

Задание 3. Оформите подробный отчет по работе.

.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что является структуризацией данных?
2. Что Вы понимаете под Нормализацией БД?
3. Дайте описание первой нормальной форме (1НФ), укажите ее особенности.
4. Дайте описание второй нормальной форме (2НФ), укажите ее особенности.
5. Дайте описание третьей нормальной форме (3НФ) , укажите ее особенности.
6. Дайте описание усиленной третьей нормальной форме, или нормальная форма Бойса-Кодда (БКНФ) , укажите ее особенности.
7. Дайте описание четвертой нормальной форме (4НФ), укажите ее особенности.
8. Дайте описание пятой нормальной форме (5НФ), укажите ее особенности.

Лабораторная работа №2
«СОЗДАНИЕ БАЗЫ ДАННЫХ В СРЕДЕ РАЗРАБОТКИ»

Цель работы: получить теоретические знания и практические навыки создания базы данных в среде разработки.

Теоретические сведения

1. Методика создания баз данных и работы с ней

1.1. Создание базы данных

Синтаксис команды:

```
CREATE DATABASE [IF NOT EXISTS] db_name;
```

Создается база данных с указанным именем *db_name*. Если попытаться создать уже существующую базу данных, возникнет ошибка. Для предотвращения ошибки оператор CREATE DATABASE необходимо снабдить конструкцией IF NOT EXISTS, при наличии которой база данных создаться, если она еще не существует, иначе – никаких действий не производится.

База данных в *MySQL* реализована в виде каталога, содержащего файлы, которые соответствуют таблицам в базе данных. CREATE DATABASE создает только каталог в каталоге данных *MySQL* и файл db.opt.

Имя может содержать основные латинские буквы (a-z, A-Z), цифры (0-9), символ доллара ‘$’, символ подчеркивания ‘_’. Имя может начинаться с цифры, но не должно состоять только из цифр, а также заканчиваться пробелами. Максимальная длина имени составляет 64 знака.

Пример 1. Создание базы данных с именем *GIS*.

При создании базы данных можно указать кодировку таблицам и столбцам по умолчанию. Для этого после имени базы данных следует указать ключевое слово DEFAULT CHARACTER SET *charset_name* – имя кодировки, например, cp1251, которая обозначает русскую *Windows*-кодировку.

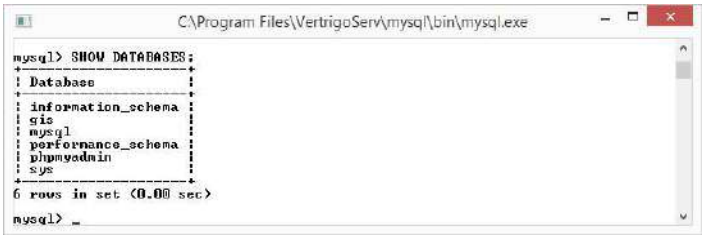
Пример 2. Создание базы данных с именем *GIS* с кодировкой.

1.2. Перечисление баз данных на сервере

Синтаксис команды:

```
SHOW DATABASES;
```

Оператор выводит список существующих баз данных.



1.3. Удаление базы данных

Синтаксис команды:

```
DROP DATABASE [IF EXISTS] db_name;
```

Удаляет все таблицы в базе данных db_name и удаляет саму базу данных. Ключевое слово IF EXISTS используется, для предотвращения возникновения ошибки, если база данных не существует.

Пример 3. Удаление базы данных с именем *GIS*.

1.4. Выбор базы данных

Перед созданием таблицы следует выбрать базу данных, с которой будет производиться работа. Выбирать базу данных необходимо в каждом сеансе работы с *MySQL*.

Синтаксис команды:

```
USE db_name;
```

Оператор сообщает *MySQL* имя базы данных *db_name* в качестве текущей базы данных по умолчанию для последующих операторов. База данных остается по умолчанию до конца.

Пример 4. Выборка базы данных с именем *GIS*.

1.5. Создание таблицы

Синтаксис команды:

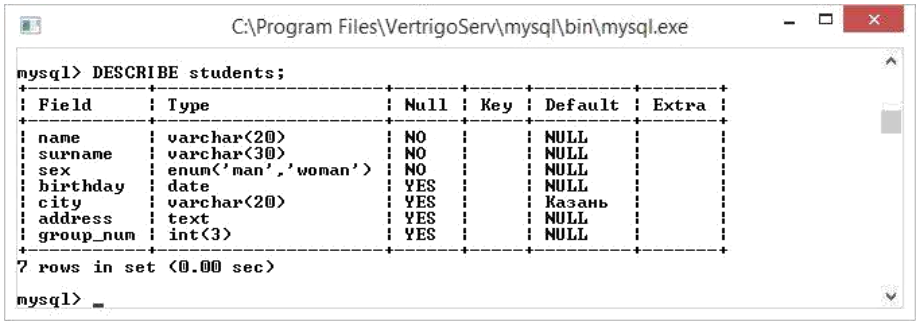
```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
        (create_definition, ...);
create_definition:
        col_name column_definition
```

Создается таблица с именем *tbl_name*. Имя может содержать основные латинские буквы (a-z, A-Z), цифры (0-9), символ доллара ‘$’, символ подчеркивания ‘_’. Имя может начинаться с цифры, но не должно состоять только из цифр, а также заканчиваться пробелами. Максимальная длина имени составляет 64 знака.

Ключевое слово IF NOT EXISTS предотвращает появление ошибки, если таблица существует. Тем не менее, нет никакой гарантии, что существующая таблица имеет структуру, идентичную указанной в инструкции CREATE TABLE.

Ключевое слово TEMPORARY используется для создания временной таблицы. Временная таблица видна только в текущей сессии, и при закрытии сессии удаляется автоматически.

В круглых скобках через запятую указываются имена столбцов



(*col_name*) и их тип (*data_type*).

column_definition:

data_type [NOT NULL | NULL] [DEFAULT *default_value*]

[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
[COMMENT '*string*']

[COLUMN_FORMAT
{FIXED|DYNAMIC|DEFAULT}] [STORAGE
{DISK|MEMORY|DEFAULT}] [*reference_definition*]

Пример 5. Создание таблицы *STUDENTS* в базе данных *GIS*.

Создается таблица STUDENTS с семью полями – name (имя), surname (фамилия), sex (пол), birthday (дата рождения), city (город), address (адрес проживания) и group_num (номер группы). Каждое имя поля снабжено описателем типа, характеризующее тип хранимых данных. В описание поля также можно вставлять дополнительные спецификаторы.

Для просмотра структуры таблиц служит оператор DESCRIBE.

Типы данных

Числовые типы

BOOL

0или1

TINYINT[(M)] [UNSIGNED] [ZEROFILL]

Очень малое целое число. Диапазон со знаком от -128 до 127. Диапазон без знака от 0 до 255.

SMALLINT[(M)] [UNSIGNED] [ZEROFILL]

Малое целое число. Диапазон со знаком от -32768 до 32767. Диапазон без знака от 0 до 65535.

MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]

Целое число среднего размера. Диапазон со знаком от -8388608 до 8388607. Диапазон без знака от 0 до 16777215.

INT[(M)] [UNSIGNED] [ZEROFILL]

Целое число нормального размера. Диапазон со знаком от -2147483648 до 2147483647. Диапазон без знака от 0 до 4294967295.

BIGINT[(M)] [UNSIGNED] [ZEROFILL]

Большое целое число. Диапазон со знаком от -9223372036854775808 до 9223372036854775807. Диапазон без знака от 0 до 18446744073709551615.

FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]

Малое число с плавающей точкой обычной точности. Диапазон со знаком от -3,402823466E+38 до -1,175494351E-38. Диапазон без знака от 1,175494351E-38 до 3,402823466E+38. Если указан атрибут UNSIGNED, отрицательные значения недопустимы. Атрибут *M* указывает количество выводимых пользователю знаков, а атрибут *D* – количество разрядов, следующих за десятичной точкой.

Обозначение FLOAT без указания аргументов или запись вида FLOAT(X), где X <=24 справедливы для числа с плавающей точкой обычной точности.

DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]

Число с плавающей точкой удвоенной точности нормального размера. Диапазон со знаком от -1,7976931348623157E+308 до -2,2250738585072014E-308.Диапазон без знака от 2,2250738585072014E-308 до 1,7976931348623157E+308. Если указан атрибут UNSIGNED, отрицательные значения недопустимы. Атрибут *M* указывает количество выводимых пользователю знаков, а атрибут *D* – количество разрядов, следующих за десятичной точкой. Обозначение DOUBLE без указания аргументов или запись вида DOUBLE (X), где 25 <= X <= 53 справедливы для числа с плавающей точкой двойной точности.

Календарные типы

DATE

Используется для величин с информацией только о дате. *MySQL* извлекает и выводит величины *DATE* в формате 'YYYY-MM-DD'. Поддерживается диапазон величин от '1000-01-01' до '9999-12-31'.

TIME

Используется для величин с информацией только о времени. *MySQL* извлекает и выводит величины *TIME* в формате 'HH:MM:SS'. Поддерживается диапазон величин от '-838:59:59' до '838:59:59'.

DATETIME

Комбинация даты и времени. Поддерживается интервал от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. *MySQL* выводит значения *DATETIME* в формате 'YYYY-MM-DD HH:MM:SS', но можно устанавливать значения в

столбце *DATETIME*, используя как строки, так и числа.

YEAR[(2|4)]

Год в двухзначном или четырехзначном форматах (по умолчанию формат четырехзначный). Допустимы следующие значения: с 1901 по 2155, 0000 для четырехзначного формата года и 1970 - 2069 при использовании двухзначного формата (70 - 69). *MySQL* выводит значения *YEAR* в формате *YYYY*, но можно задавать значения в столбце *YEAR*, используя как строки, так и числа.

Строковые типы

CHAR[(M)] [BINARY]

Строка фиксированной длины, которая справа дополняются пробелами до указанной длины, при хранении. Диапазон длины от 1 до 255 символов.
Завершающие пробелы удаляются, когда значение извлекается. Значения *CHAR* сортируются и сравниваются без учета регистра в зависимости от кодировки по умолчанию, если не установлен флаг *BINARY*.

VARCHAR(M)[BINARY]

Строка переменной длины. Примечание: конечные пробелы удаляются при сохранении. Диапазон длины от 1 до 255 символов. При хранении величин используется только то количество символов, которое необходимо, плюс один байт для записи длины. Значения *VARCHAR* сортируются и сравниваются без учета регистра, если не установлен флаг *BINARY*.

TINYTEXT [BINARY]

Строка текста с максимальной длиной 255 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

TEXT [BINARY]

Строка текста с максимальной длиной 65535 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

MEDIUMTEXT [BINARY]

Строка текста с максимальной длиной 16777215 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

LONGTEXT [BINARY]

Строка текста с максимальной длиной 4294967295 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

ENUM(value1, value2, value3, ...)

Перечисление – может принимать значение из списка допустимых значений, явно перечисленных в спецификации столбца в момент создания таблицы. Этим значением также может быть пустая строка ("") или *NULL*. Перечисление может иметь максимум 65535 элементов.

SET(value1, value2, value3, ...)

Строковый тип, который может принимать ноль или более значений, каждое из которых должно быть выбрано из списка допустимых значений, определенных при создании

таблицы. Элементы множества *SET* разделяются запятыми. Как следствие, сами элементы множества не могут содержать запятых.

TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB – представляет собой двоичный объект, который может содержать переменное количество данных.

По размеру аналогичны типу *TEXT*.

M – указывает максимальный размер вывода в символах. Максимально допустимый размер вывода составляет 255 символов.

D – употребляется для типов данных с плавающей точкой и указывает количество разрядов, следующих за десятичной точкой. Максимально возможная величина составляет 30 разрядов, но не может быть больше, чем *M*-2.

UNSIGNED – беззнаковое число, *ZEROFILL* – свободные позиции слева заполняются нулями.

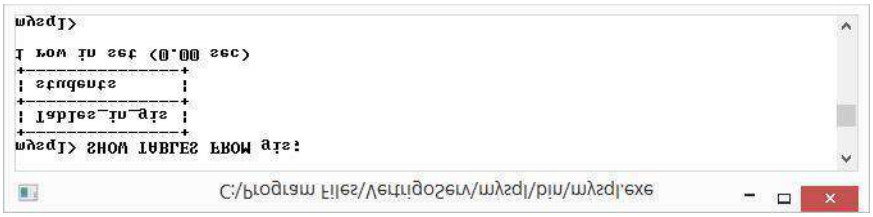
1.6. Просмотр таблиц

Синтаксис команды:

```
SHOW TABLES [FROM db_name]
```

Выводит список таблиц указанной базы данных *db_name*. Если имя базы данных не указано, выводится список текущей базы данных.

Пример 6. Получение списка таблиц базы данных *GIS*.



1.7. Удаление таблицы

Синтаксис команды:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

Оператор DROP TABLE удаляет одну или несколько таблиц. Все табличные данные и определения удаляются, так что будьте внимательны при работе с этой командой! Можно использовать ключевые слова IF EXISTS, чтобы предупредить ошибку, если указанные таблицы не существуют. Опции RESTRICT и CASCADE позволяют упростить перенос программы. В данный момент они не задействованы.

Опция TEMPORARY работает следующим образом: уничтожает только временные таблицы, не закрывает открытую транзакцию, права доступа не проверяются.

Использование слова TEMPORARY – это хороший способ удостовериться, что вы случайно не уничтожите настоящую таблицу.

1.8. Изменение структуры таблицы

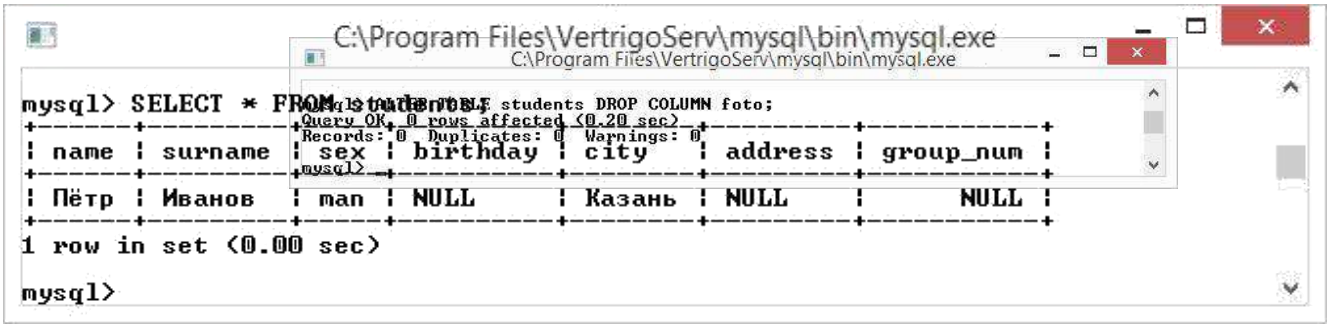
Синтаксис команды:

ALTER TABLE *tbl_name alter_spec* [, *alter_spec* ...]

Сразу после оператора следует имя таблицы, которая подвергается изменению, и производимое изменение которых может быть несколько.

Пример 7. Добавление нового столбца *foto* в таблицу *STUDENTS*.

Пример 8. Удаление столбца *foto* из таблицы *STUDENTS*.



Также существуют возможности добавлять столбцы в определенное место по отношению к другим столбцам, изменять название столбцов, их тип.

2. Методика внесения, изменение и удаления данных в таблице

2.1. Добавление данных

После успешного создания базы данных и таблиц перед разработчиком встает задача заполнения таблиц данными. Традиционно для осуществления этой операции применяют три подхода:

1) однострочный оператор *INSERT* – добавляет в таблицу данных новую запись;
2) многострочный оператор *INSERT* – добавляет в таблицу данных несколько новых записей;
3) пакетная загрузка данных – добавляет в таблицу данных из внешнего файла.

2.1.1. Однострочный оператор INSERT

Синтаксис команды:

INSERT [INTO] *tbl_name* [(*col_name*, ...)] VALUES (*expr*, ...)

Данный оператор вставляет строку в таблицу *tbl_name*, список столбцов, которые вносятся значения, указываются через запятую после имени таблицы, а значения столбцов указываются в скобках через запятую после ключевого слова VALUES. Если не указаны столбцы, то должны быть определены значения для всех столбцов в списке VALUES(). Любой столбец, для которого явно не указано значение, будет установлен в свое значение по умолчанию.

Выражение *expr* может относится к любому столбцу, который ранее был внесен в список значений. Например, можно указать следующее:

INSERT INTO *tbl_name* (col1, col2) VALUES(15, col1*2);

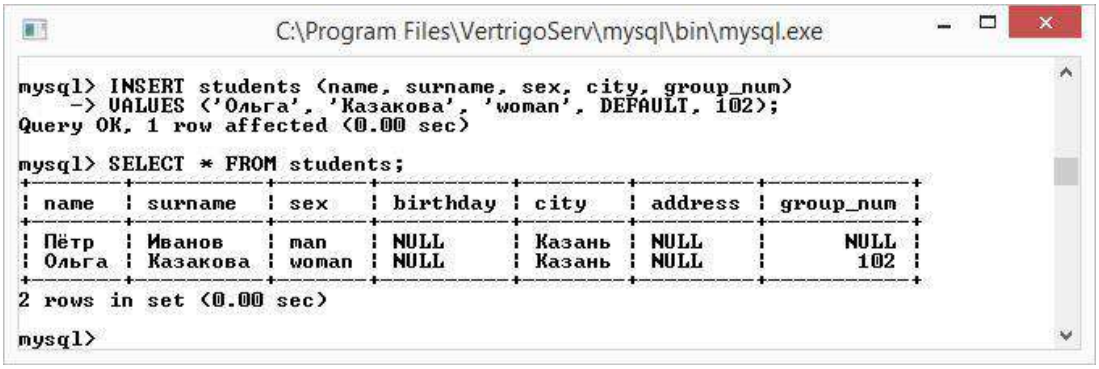
Но нельзя указать:

INSERT INTO *tbl_name* (col1, col2) VALUES(col2*2, 15);

Пример 9. Добавление одной строки в таблицу STUDENTS.

Выполним оператор выборки данных из таблицы *STUDENTS* для просмотра веденной строки:

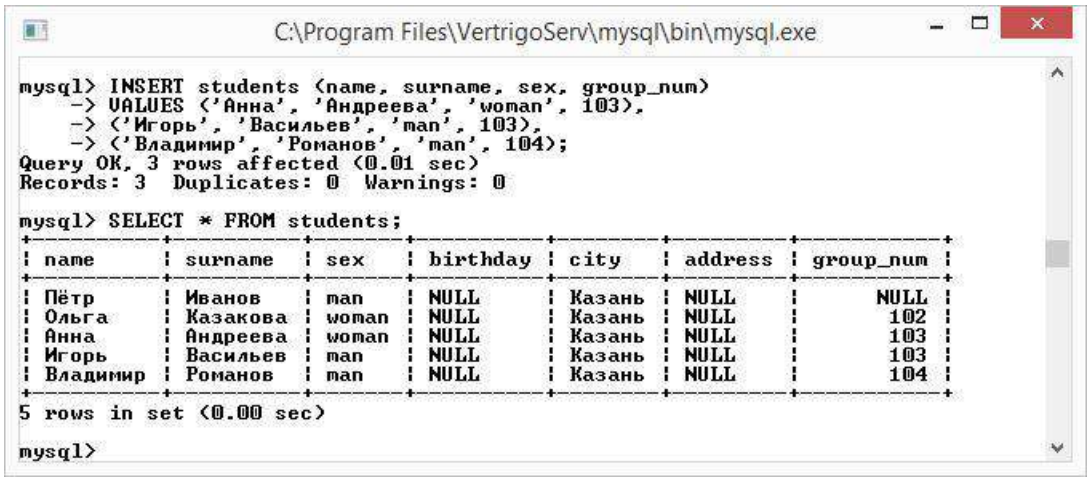
Пример 10. Добавление одной строки со значением *DEFAULT* в таблицу *STUDENTS*.



2.1.2. Многострочный оператор INSERT

Многострочный оператор INSERT совпадает с однострочным оператором. В нем после ключевого слова VALUES добавляется несколько expr.

Пример 11. Добавление несколько строк в таблицу *STUDENTS*.



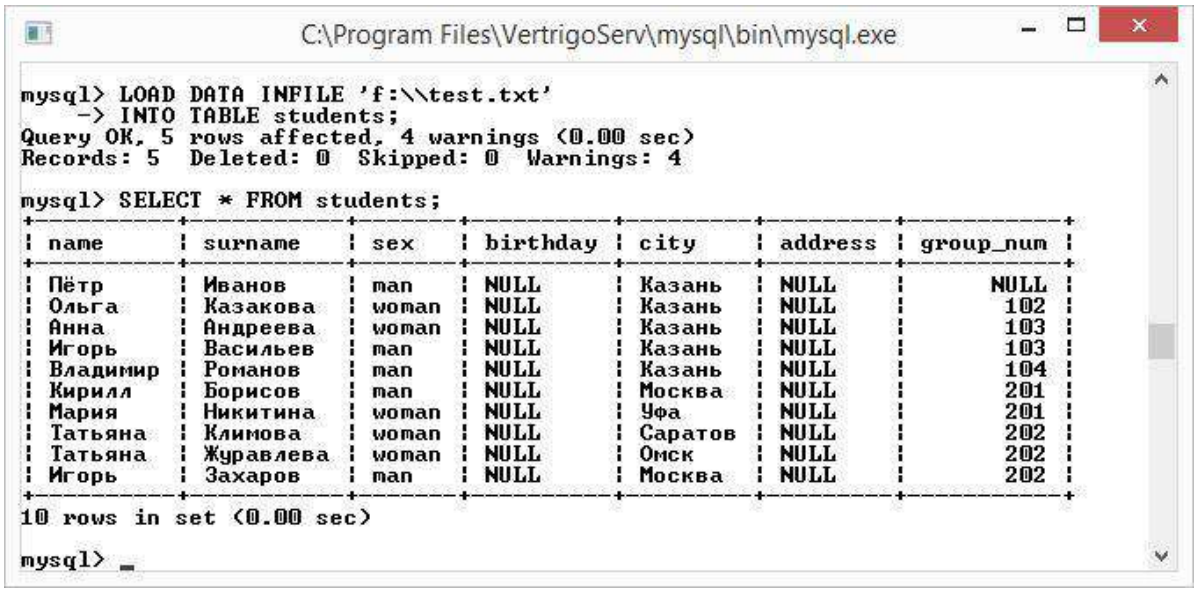
2.1.3. Пакетная загрузка данных

Упрощенный синтаксис команды:

```
LOAD DATA INFILE 'file_name';
```

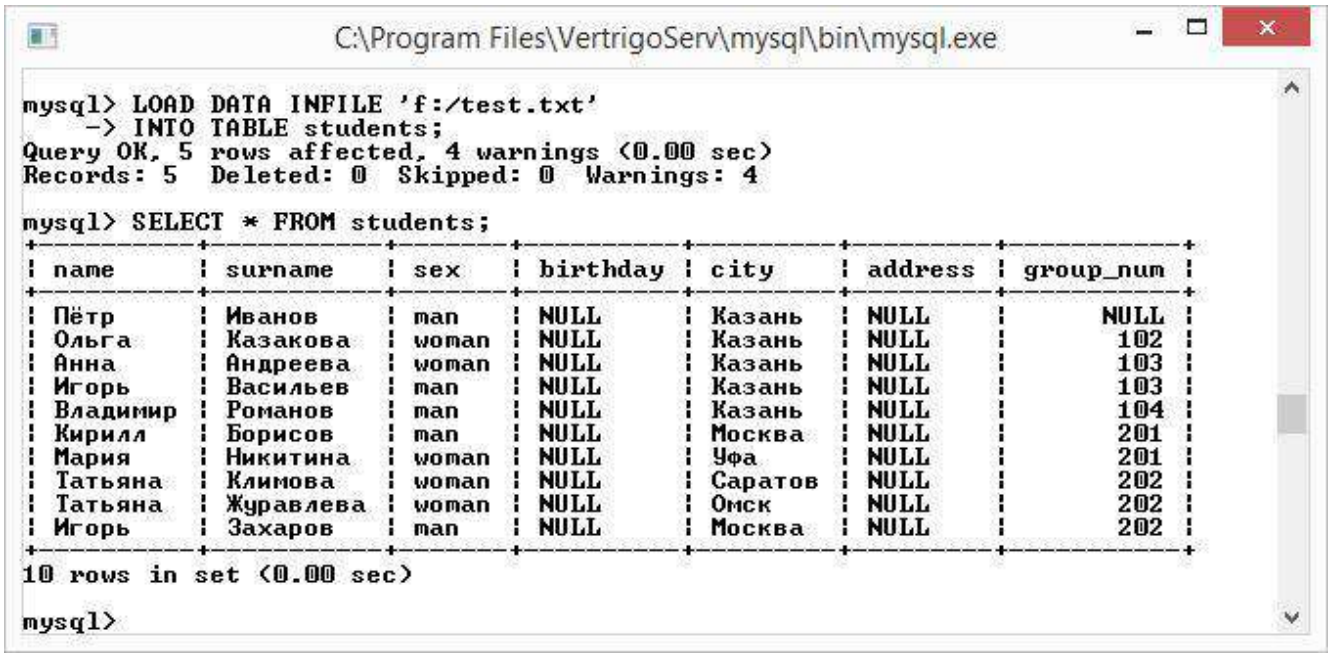
Поля в загружаемом файле должны быть разделены символом табуляции. Даже при наличии в файле только одного столбца с данными, в конце строки обязательно должен быть символ табуляции. Значение *NULL* обозначается как \N.

Пример 12. Загрузка из файла.



При формировании пути к файлу в операционной системе *Windows* обратные слеши необходимо экранировать, этого можно избежать, если использовать прямые слеши.

Пример 13. Загрузка из файла.



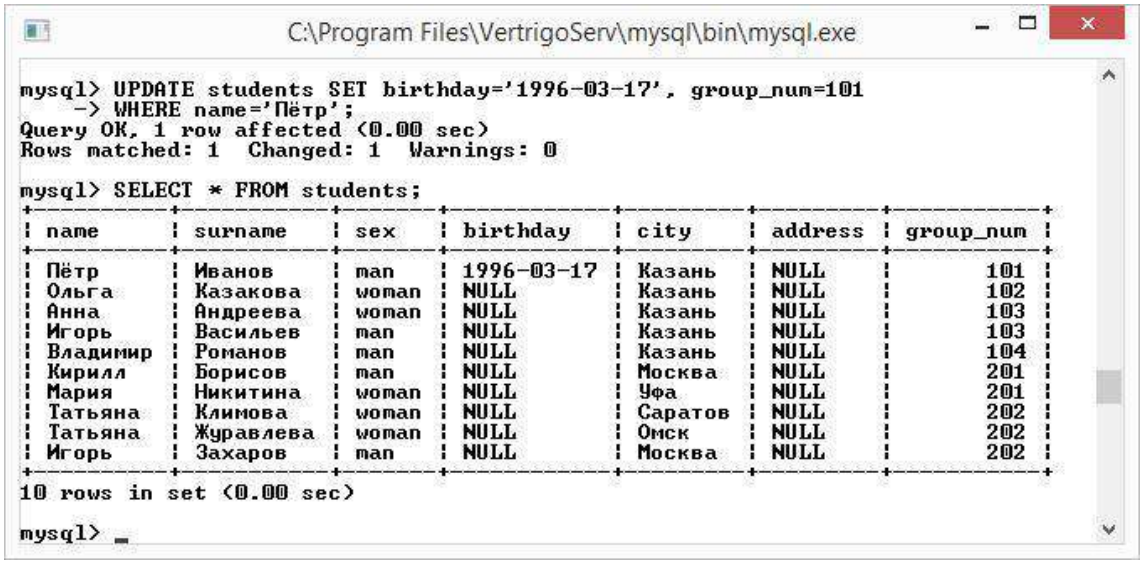
2.2. Изменение данных

Синтаксис команды:

UPDATE *tbl_name*
SET *col_name1*={*expr1*|DEFAULT} [, *col_name2*={*expr2*|DEFAULT}]
...
[WHERE *where_condition*]

Оператор UPDATE обновляет столбцы в соответствии с их новыми значениями в строках выбранной таблицы. В выражении SET указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении WHERE, если оно присутствует, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Каждое значение может быть задано как выражение, либо значением по умолчанию с помощью ключевого слова DEFAULT, чтобы установить столбец явным образом на значение по умолчанию. Предложение WHERE, если задано, определяет условия, которые определяют, какие строки будут обновляться. Без предложения WHERE, обновляются все строки.

Пример 14. Обновление одной строки в таблице *STUDENTS*.



2.3. Удаление данных

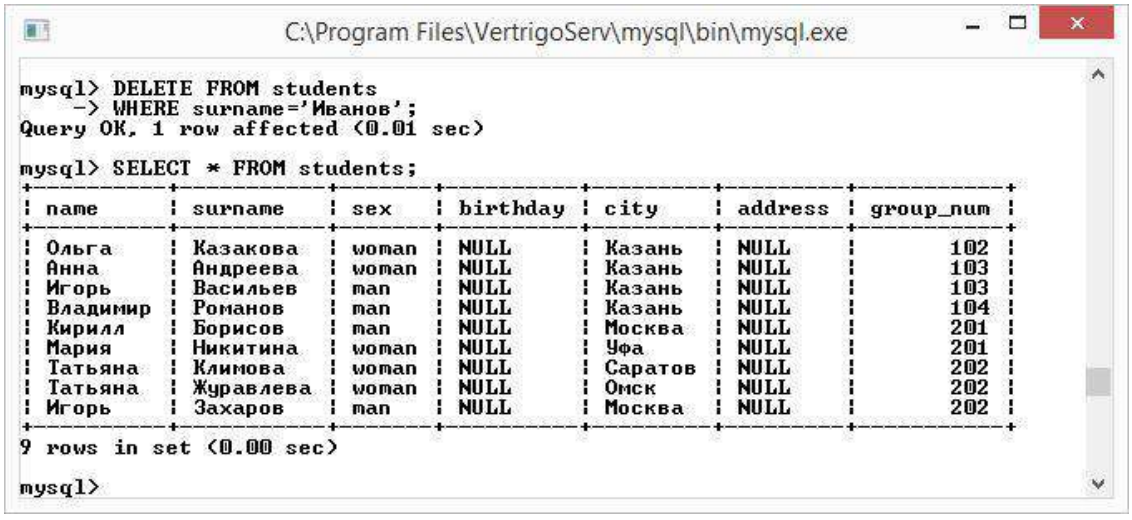
Синтаксис команды:

```
DELETE FROM tbl_name
    [WHERE where_condition]
```

Оператор DELETE удаляет из таблицы *tbl_name* строки, удовлетворяющие заданным в *where_definition* условиям, и возвращает число удаленных записей.

Если оператор DELETE запускается без определения WHERE, то удаляются все строки.

Пример 15. Удаление строки в таблице *STUDENTS*.



Для полного очищения таблицы используется оператор TRUNCATE TABLE *tbl_name*;

3. Использование основных операторов выборки, сортировки, ограничения выборки, группировки записей и сохранения результатов выборки во внешний файл

3.1. Оператор *SELECT*

Выборка из таблиц определенных данных с заданными условиями выполняется оператором SELECT.

Упрощённый синтаксис команды:

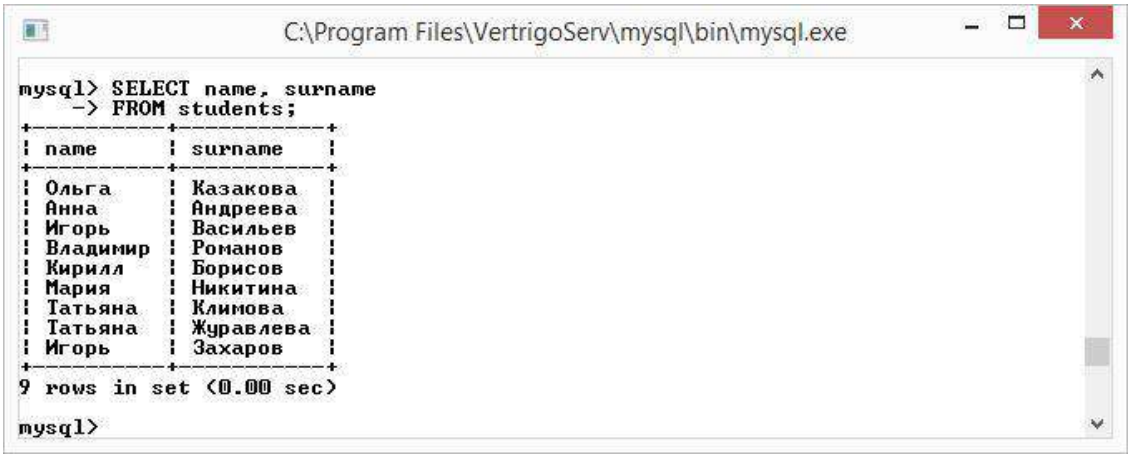
```
SELECT select_expr [, select_expr ...]
    FROM table_references
```

WHERE *where_condition*;

SELECT, используется для извлечения строк, выбранных из одной или нескольких таблиц. Выражение *select_expr* указывает на столбец, который необходимо получить. Должно быть, по крайней мере, одно выражение *select_expr*. Выражение *table_references* указывает на таблицу или таблицы, из которых для извлекаются строки. Ключевое слово WHERE указывает условие *where_condition* или условия, которым должны удовлетворять выбранные строки.

Оператор выбирает все строки, если нет WHERE.

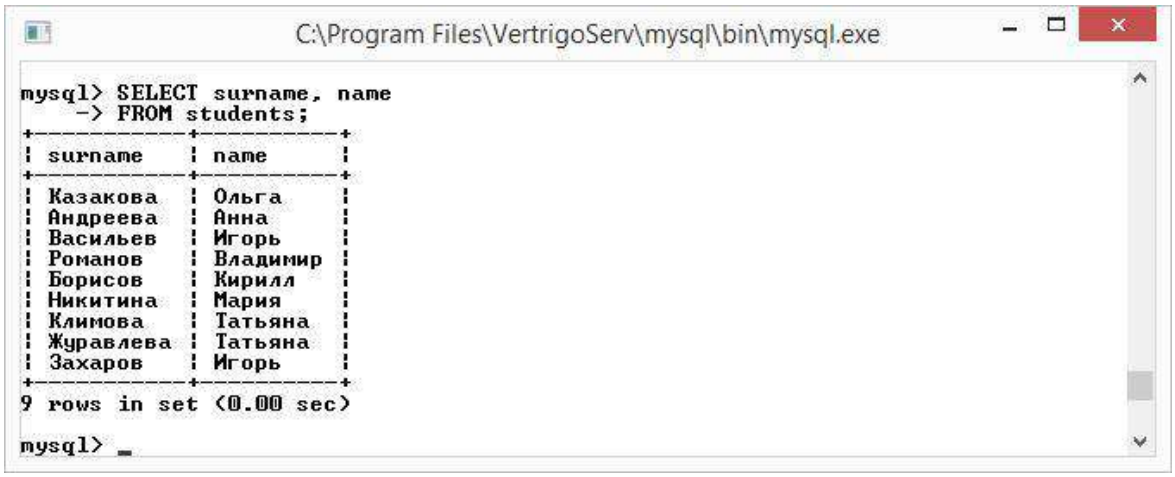
Пример 16. Выборка записей из таблицы.



В данном примере выводятся поля записей *name* и *surname.*

Данный оператор позволяет изменить порядок следования (вывода) полей, т.е. последовательность полей необязательно должна быть такой же, как она прописана в таблице.

Пример 17. Выборка записей из таблицы с измененной последовательностью полей.



3.2. Условия выборки

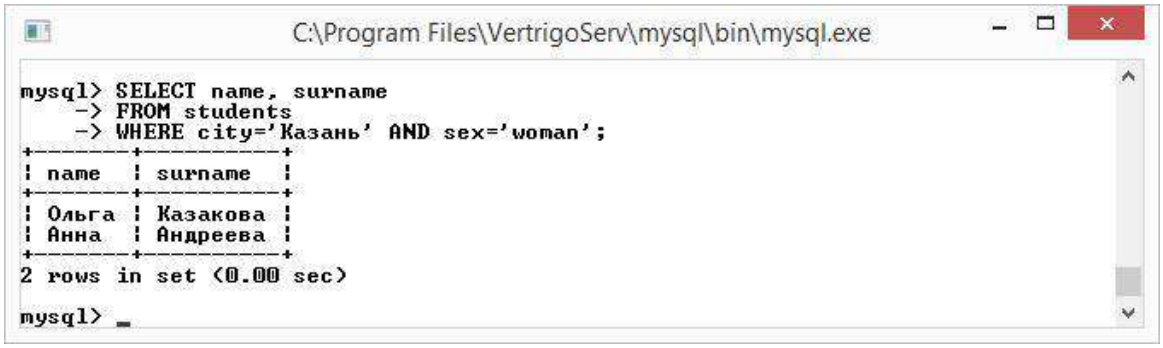
Операция, когда необходимо вывести записи из таблиц по одному или нескольким критериям используется чаще, чем выборка всех записей. Для ввода ограничений в операторе вводится ключевое слово WHERE, после которого следует логическое условие. Если запись удовлетворяет данному условию, она попадает в результат выборки, иначе такая запись отбрасывается.

Пример 18. Выборка строк из таблицы с условием.

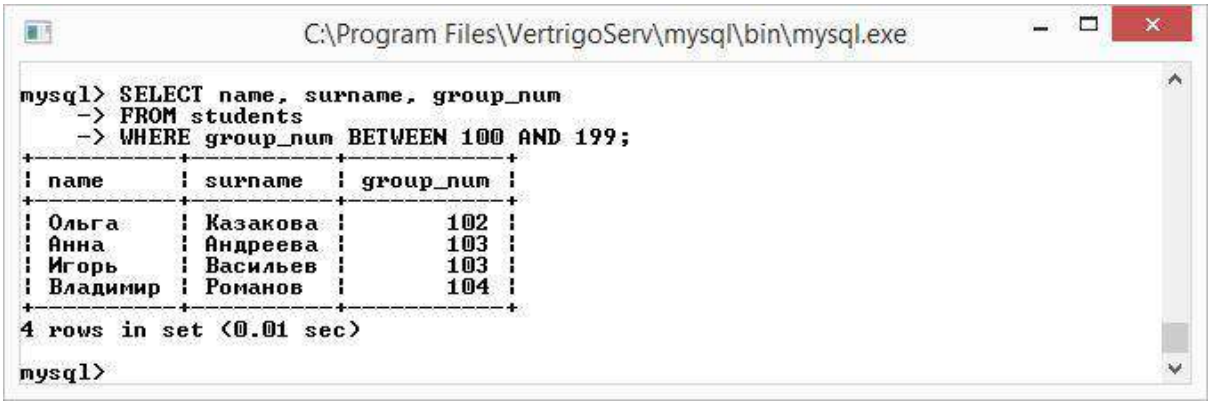
В данном примере выводятся поля name и surname для записей, для которых в поле city указано ‘Москва’.

Условие может быть составным и объединяться при помощи логических операторов.

Пример 19. Выборка строк из таблицы с составным условием.

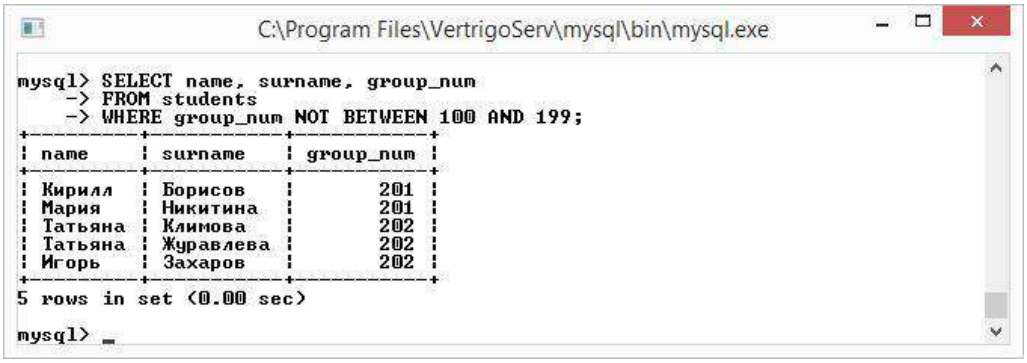


Пример 20. Использование конструкции *BETWEEN*.



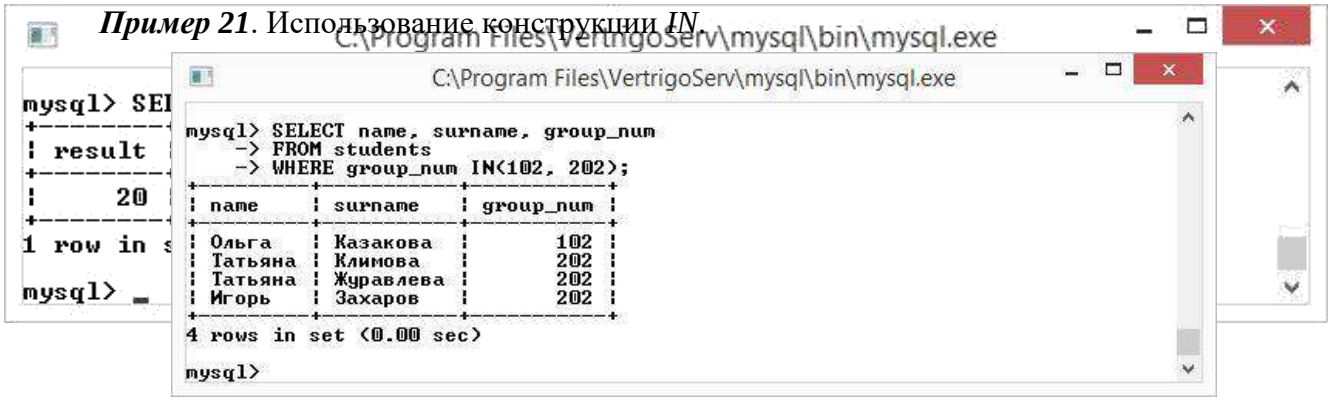
Существует противоположенная конструкция NOT BETWEEN, которая возвращает записи, не попавшие в интервал.

Пример 21. Использование конструкции *NOT BETWEEN*.



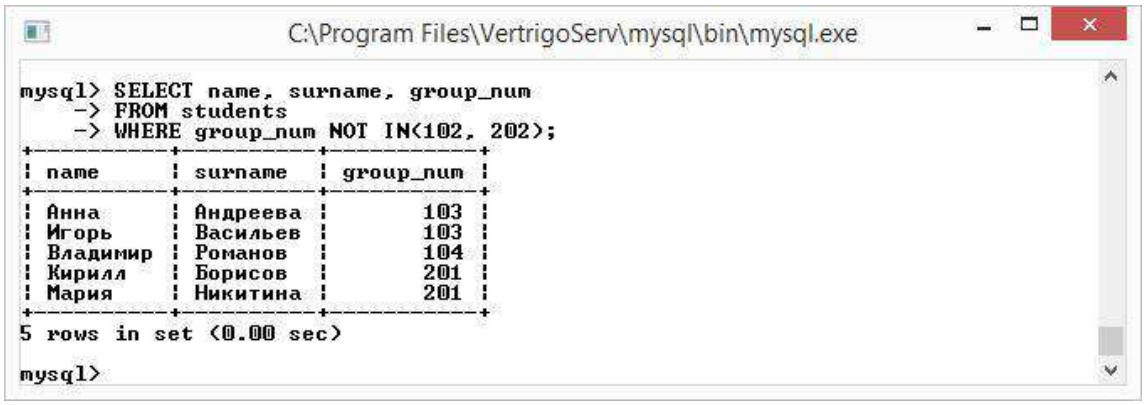
Есть возможность выбирать записи, соответствующие не диапазону, а некоторому списку в условии. Данный способ реализуется использованием ключевого слова IN.

Пример 21. Использование конструкции *IN*.



Существует противоположенная конструкция NOT IN, которая возвращает записи, не попавшие в список.

Пример 22. Использование конструкции *NOT IN*.



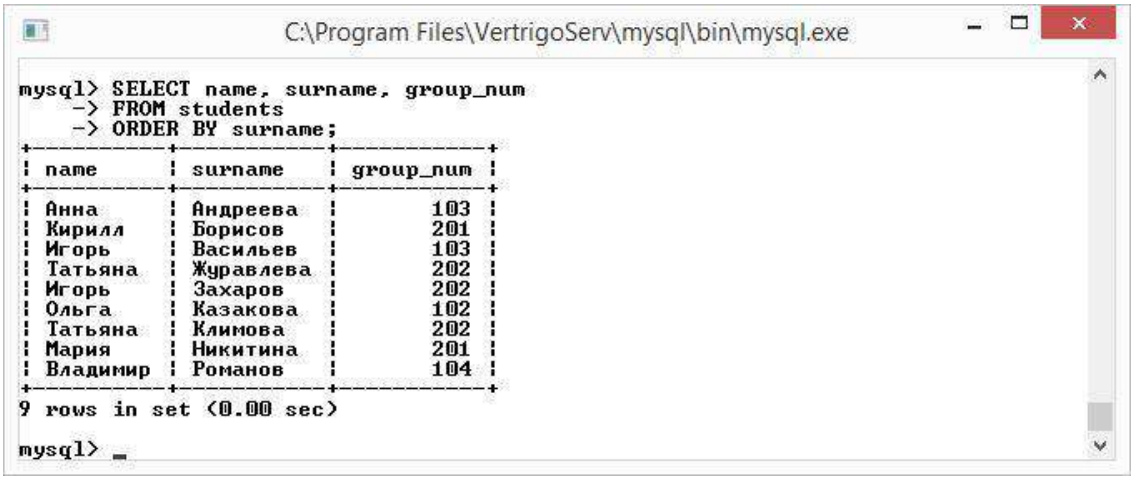
Оператор SELECT можно использовать для извлечения строк, вычисленных без ссылки на какую-либо таблицу.

Для более осмысленного названия результирующего столбца можно использовать спецификатор AS, после которого следуют псевдоним.

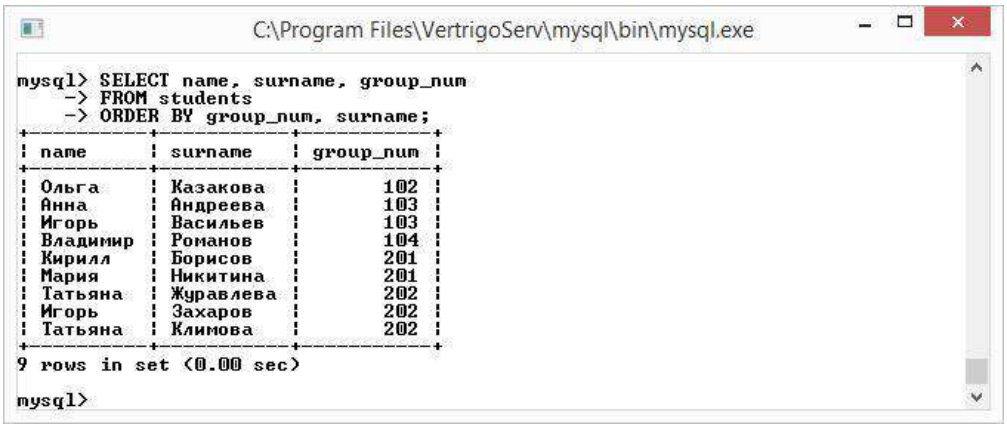
3.3. Сортировка

Очень часто интерес представляет результат выборки, отсортированный определенным образом по каким-либо столбцам. Это достигается применением конструкции ORDER BY. После этой конструкции следует имя столбца, по которому следует сортировать данные.

Пример 23. Сортировка по одному столбцу.

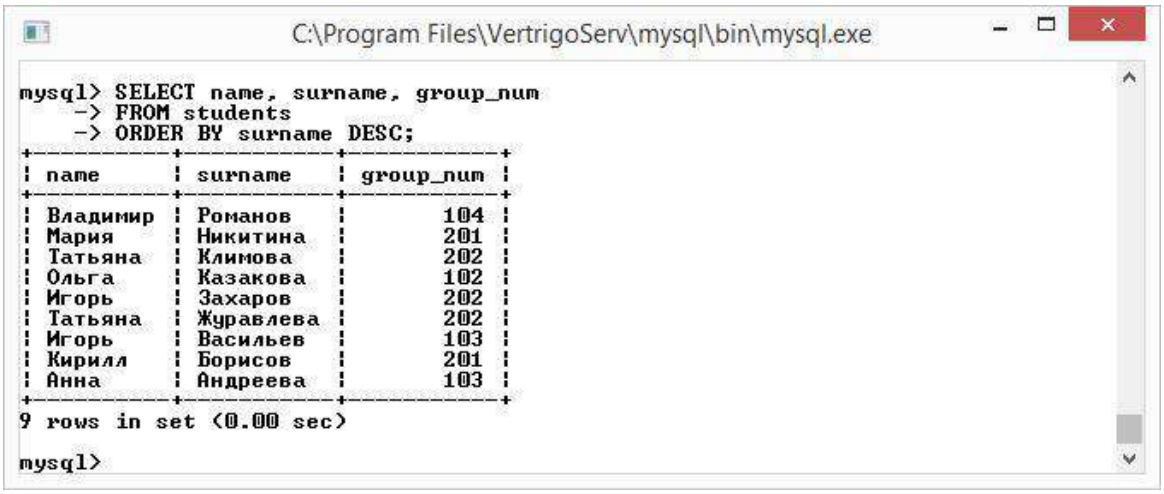


Пример 24. Сортировка по нескольким столбцам.



По умолчанию данные сортируются в прямом порядке. Порядок можно изменить на обратный при помощи ключевого слова DESC. Для прямой сортировки используется ключевое слово ASC.

Пример 25. Сортировка в обратном порядке.



3.4. Ограничение выборки

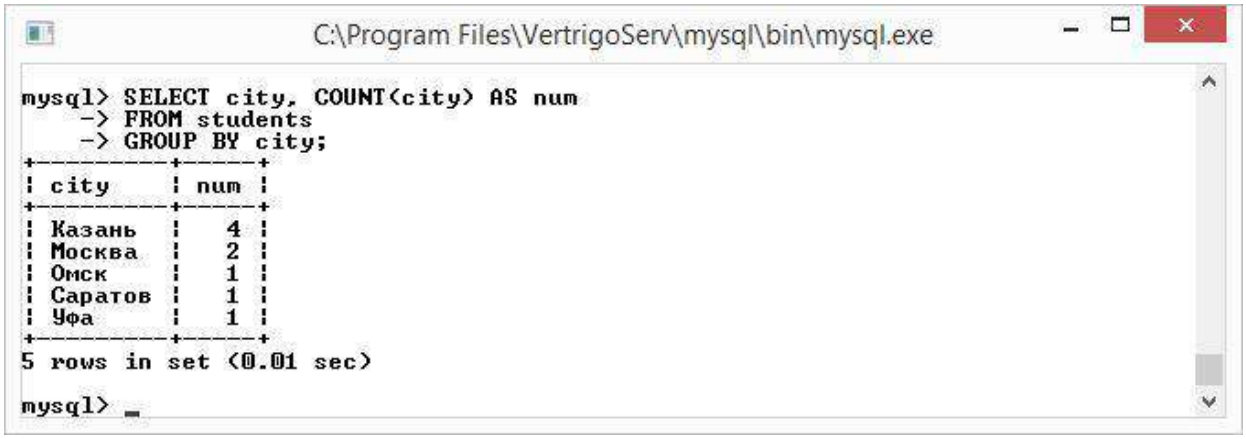
Для ограничения по количеству выводимых записей используется ключевое слово LIMIT start_index, count. На номер записи указывает первая цифра а вторая цифра count указывает на количество выводимых записей.

Пример 25. Ограничение выборки.

3.5. Группировка записей

Для подсчета числа записей с одинаковыми полями используется конструкция GROUP BY совместно с функцией COUNT().

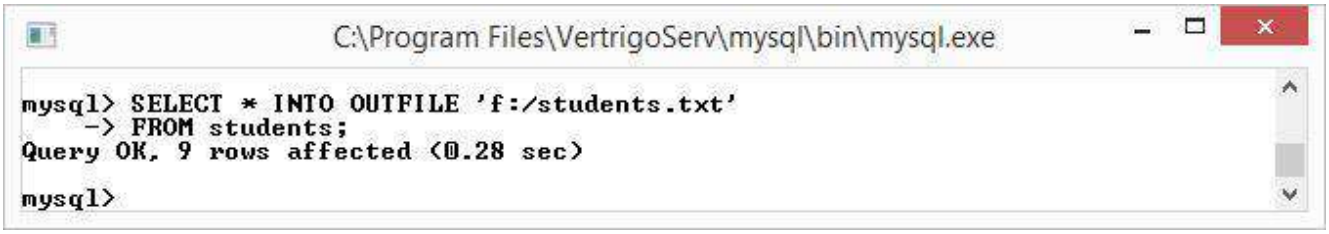
Пример 26. Совместное использование *GROUP BY* и *COUNT()*.



В данном примере подсчитывается число студентов по каждому городу в отдельности.

3.6. Сохранение результатов выборки во внешний файл

Результат, получаемый при выполнении оператора SELECT можно сохранить в текстовый файл.



4. Методика создания и использования хранимых процедур

4.1. Создание хранимой процедуры

Синтаксис команды:

```
CREATE PROCEDURE sp_name([param[, ...]])

        routine_body
```

Здесь *sp_name* имя хранимой процедуры. За именем следует список необязательных параметров, заключенных в круглые скобки. Если параметров несколько, то они перечисляются через запятую. Круглые скобки обязательны, даже если список параметров пустой. Пробел между именем процедуры и открывающейся круглой скобкой недопустим.

```
param:
[IN, OUT, INOUT] param_name type
```

Ключевые слова IN, OUT, INOUT слова используются для задания направления передачи данных:

IN – данные передаются строго в хранимую процедуру. Если параметру с данным модификатором внутри процедуры присваивается новое значение, по выходу из нее не сохраняется и параметр принимает значение, которое он имел до вызова процедуры.

OUT – данные передаются строго из хранимой процедуры. По выходу из процедуры параметр с данным модификатором будет иметь значение, которое он примет внутри процедуры.

INOUT – значение этого параметра принимаются во внимание внутри процедуры, и если параметр изменится внутри, то параметр по выходу из процедуры примет измененное значение.

Если ни один из модификаторов не указан, считается, что параметр объявлен с модификатором *IN*.

type – после имени параметра указывается его тип.

routine_body – тело процедуры.

Основное неудобство заключено в использовании в конце запроса символа точки с запятой ‘;’ воспринимаемое консольным клиентом как сигнал к выполнению запроса, т.е. отправке запроса на сервер. Обойти это можно переопределив символ разделителя запросов ‘;’ на последовательность символов “//” с помощью оператора DELIMITER.

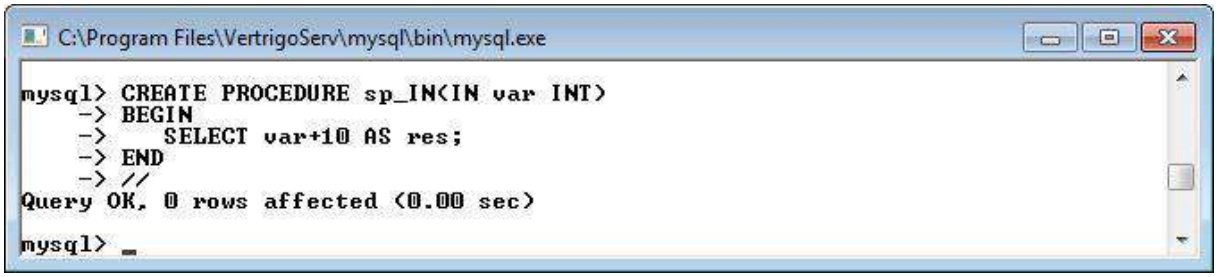
Пример 27. Переопределение символа разделителя запросов

4.2. Тело процедуры

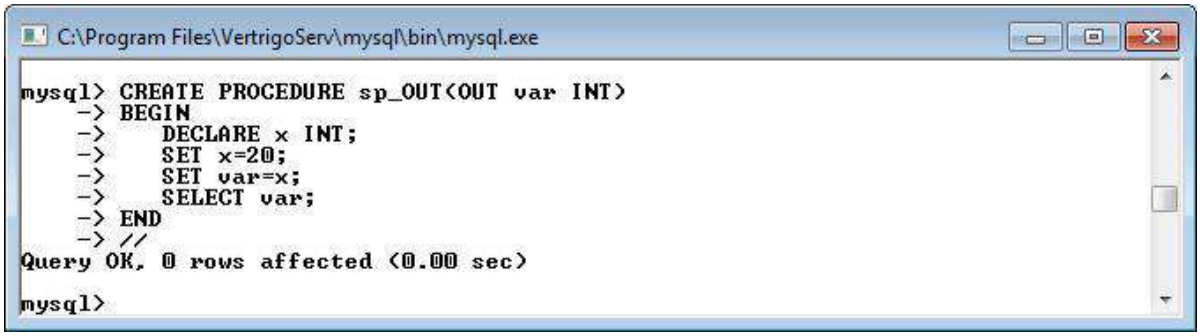
Тело процедуры состоит из составного оператора BEGIN ... END, внутри которого могут располагаться другие операторы, в том числе и другие операторы BEGIN ... END.

```
        [label:] BEGIN
            Statements
    END [label]
```

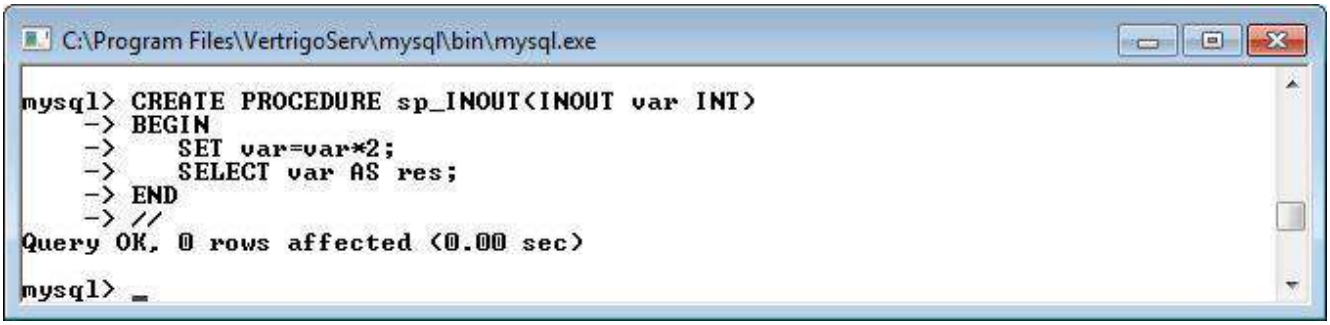
Пример 28. Параметр *IN*



Пример 29. Параметр *OUT*



Пример 30. Параметр *INOUT*



4.3. Переменные *SQL*

Существуют два вида переменных: пользовательские и хранимые. Пользовательские переменные хранят значение до завершения соединения с сервером и доступны внутри хранимого кода и имеют совершенно определенный тип данных. Объявление переменных начинается с символа ‘@’.

Переменные хранимого кода напротив – имеют ограниченную область видимости (в пределах процедуры). Они должны объявляться в хранимом коде с ключевым словом DECLARE без символа ‘@’.

Синтаксис объявления пользовательской переменной:

```
SELECT @var_name;
```

или объявление пользовательской переменной с инициализацией (с присвоением значения):

```
SELECT @var_name:=value;
```

или с помощью оператора:

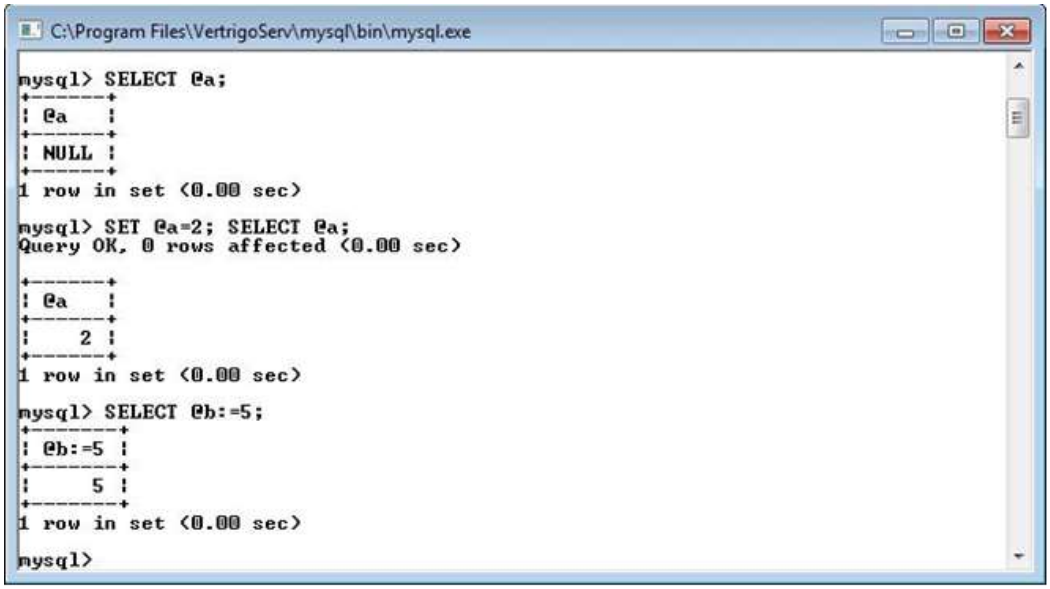
```
SET @var_name=value;
```

Синтаксис объявления хранимой переменной:

```
DECLARE var_name type DEFAULT def_value;
```

var_name – имя переменной, *value* – значение, *type* – тип переменной, *def_value* – значение по умолчанию.

Пример 31. Объявление и инициализация переменных

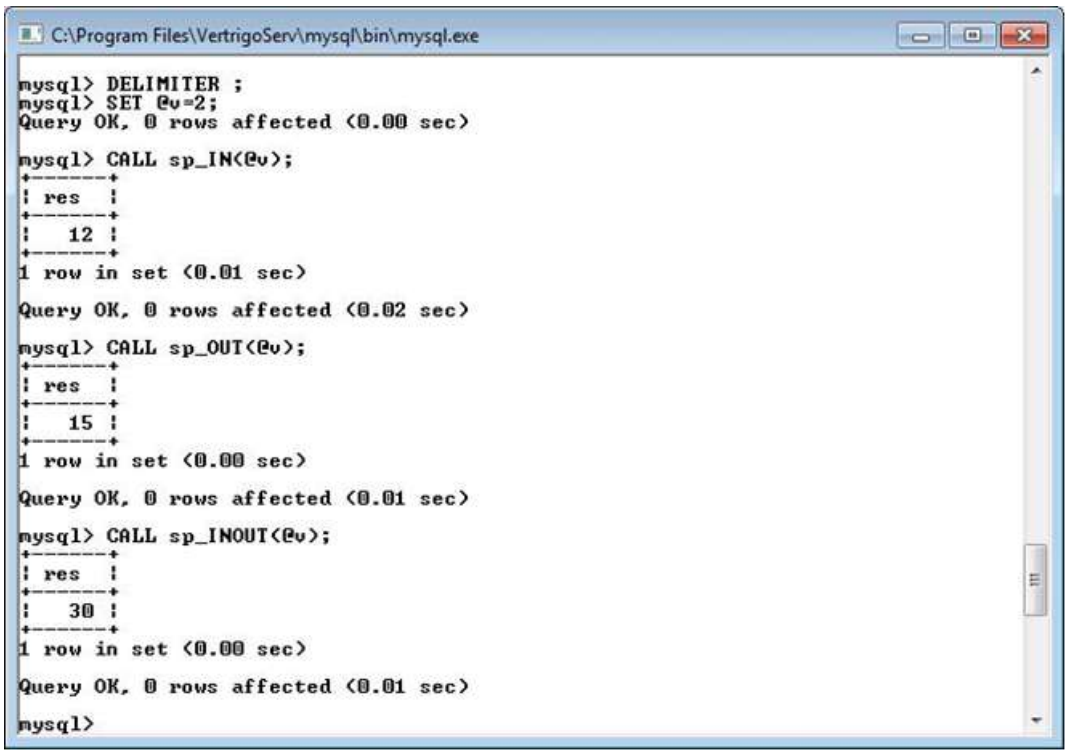


4.4. Вызов хранимой процедуры

Синтаксис команды:

```
CALL sp_name([param[, ...]]);
```

Пробел между именем процедуры и открывающейся круглой скобкой недопустим.



4.5. Удаление хранимой процедуры

Синтаксис команды:

```
DROP PROCEDURE [IF EXISTS] sp_name;
```

sp_name – имя удаляемой процедуры. Ключевое слово IF EXISTS предотвращает ошибку возникновения, если процедуры не существует.

Продолжение примера проектирования и создания БД MySQL (рассмотренного выше).

2 Физическое проектирование

2.1 Выбор СУБД и других программных средств

Реализовать разрабатываемую систему можно с использованием любой СУБД, в том числе — нереляционной (NoSQL). NoSQL базы данных, в свою очередь делятся на несколько типов:

- колоночные базы и базы «ключ-значение» призваны ускорить обработку данных за счет реализации особых схем хранения данных в памяти;
- документные базы позволяют хранить данные с разными полями (у разных объектов) и лучше подходят для параллельной обработки данных. Однако, медленно выполняют обновление данных.

В нашем случае база будет использоваться внутри библиотеки и, скорее всего, не будет требовать очень высокой производительности. Кроме того, структура таблиц меняться не должна. Поэтому будем использовать реляционные базы данных.

2.2 Составление и нормализация реляционных отношений

Таблица 1 — Схема отношения «Администраторы» (administrators):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный

Логин	login	VARCHAR(45)	Обязательное поле
Пароль	password	VARCHAR(45)	Обязательное поле

Таблица 2 — Схема отношения «Читатели» (readers):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный
Имя	*login*	*VARCHAR(45)*	*Обязательное поле*
Пароль	*password*	*VARCHAR(45)*	*Обязательное поле*
ФИО	name	VARCHAR(45)	Обязательное поле
Серия, номер паспорта	passport	VARCHAR(45)	Обязательное поле
Номер телефона	phone	VARCHAR(45)	Необязательное поле

Таблица 3 — Схема отношения «Библиотекари — Читальные залы» (librarian_rooms):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор библиотекаря	id_librarian	INT(4)	Первичный ключ (составной), внешний ключ к librarians, обязательное поле
Идентификатор читального зала	id_room	INT(4)	Первичный ключ (составной), внешний ключ к rooms, обязательное поле

Таблица 4 — Схема отношения «Карточки выдачи книг» (booking_cards):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный
Идентификатор читателя	id_reader	INT(4)	Внешний ключ к readers, обязательное поле
Идентификатор книги	id_book	INT(4)	Внешний ключ к books, обязательное поле
Идентификатор библиотекаря	id_librarian	INT(4)	Внешний ключ к librarians, обязательное поле
Время выдачи книги	time	DATETIME	Обязательное поле

Срок (время сдачи книги)	*period*	*DATETIME*	*Необязательное поле*

Таблица 5 — Схема отношения «Карточки бронирования книг» (issue_cards):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный
Идентификатор читателя	id_reader	INT(4)	Внешний ключ к readers, обязательное поле
Идентификатор книги	id_book	INT(4)	Внешний ключ к books, обязательное поле
Время создания карточки	time	DATETIME	Обязательное поле
Срок (время сдачи книги)	period	DATETIME	Необязательное поле

Таблица 6 — Схема отношения «Читальные залы» (rooms):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный
Название зала	name	VARCHAR(45)	Обязательное поле

Таблица 7 — Схема отношения Библиотекари (librarians):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный
Логин	login	VARCHAR(45)	Обязательное поле
Пароль	password	VARCHAR(45)	Обязательное поле

Таблица 8 — Схема отношения Книги (books):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор	id	INT(4)	Первичный ключ, уникальный
Автор	author	VARCHAR(45)	Обязательное поле
Год публикации	publication_year	INT(4)	Обязательное поле
Издательство	publisher	VARCHAR(45)	Обязательное поле
Название книги	name	VARCHAR(45)	Обязательное поле

ISBN	isbn	VARCHAR(45)	Необязательное поле

Таблица 9 — Схема отношения «Размещение книг» (book_places):

Содержание поля	Имя поля	Тип, длина	Примечания
Идентификатор книги	id_book	INT(4)	Внешний ключ к books, обязательное поле
Идентификатор зала	id_room	INT(4)	Внешний ключ к books, обязательное поле
Количество книг	quantity	**INT**(4)	Обязательное поле
Номер полки	shell_number	**INT**(4)	Обязательное поле

При разработке реляционных отношений, было обнаружено, что в ряд таблиц базы стоит обавить ряд новых полей — выделены в таблицах курсивом. Для хранения даты в MySQL используется тип данных **DATETIME** , объект которого занимает 8 байт.

Схема базы данных была создана в среде MySQL Workbench , в результате получена схема, показанная на рисунке 6.

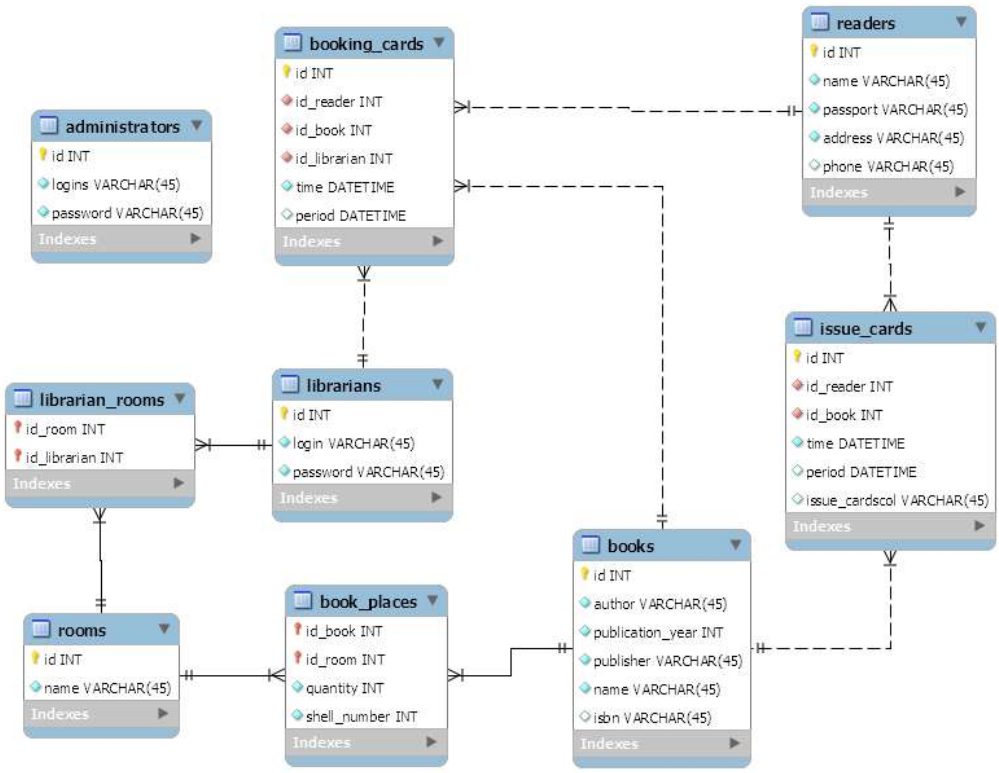


Рисунок 6.

2.3 Нормализация полученных отношений

Разработанная схема БД находится в:

- первой нормальной форме, так как в качестве доменов выступают только скалярные значения и информация в таблицах не дублируется. Почти во всех таблицах есть идентификатор (id), а в остальных — librarian_rooms и book_places в качестве первичного ключа выступает пара полей, так как нет смысла добавлять одного и того же библиотекаря или книгу дважды в один зал. При повторном добавлении книги (если произошла приемка точно таких же книг) — надо выполнить поиск и изменить число экземпляров в существующей записи;

- второй и третьей нормальных формах, каждый не ключевой атрибут неприводимо и нетранзитивно зависит от первичного ключа. Для всех таблиц нашей БД это очевидно — Логин и Пароль зависят от Id и их нельзя вывести иным образом; количество книг и номер полки зависят от id книги и id комнаты и их тоже нельзя вывести никак иначе.

Таким образом, схема базы данных показанная на рисунках 5 и 6 находится в нормальной форме Бойса-Кодда.

2.4 Определение требований к операционной обстановке

Очевидно, в библиотеке основной объем памяти будут занимать книги, пользователи и карточки выдачи/бронирования книг. Предположим, в библиотеку в месяц будет поступать 100 новых (разных) книг и записываться 200 пользователей. Тысяча пользователей возьмет по 3 книги. Сколько книг будет забронировано — не важно, т.к. карточки бронирования уничтожаются. Учитывая, что для хранения записи об одной книге требуется

45*4+4*2 = 188

байт, для хранения читателя 184 байта, а одна карточка выдачи книги занимает 32 байта можно определить примерный объем памяти, необходимый для базы данных библиотеки в течении одного месяца работы:

100*188 + 200*184 + 1000*3*32 = 18800 + 36800 + 96000 = 151600 байт = 148 Кб

Значит, за год объем базы не должен превысить 1,73Мб.

2.5 Описание групп пользователей и прав доступа

Администратор базы данных взаимодействует с базой посредством исполнения SQL-запросов. При этом он имеет доступ ко всем данных, может изменять структуру БД, устанавливает права доступа для остальных групп.

Администратор зала библиотеки имеет доступ по чтению и записи к отношениям Librarians , rooms, librarians_rooms. При необходимости работы с фондами библиотеки администратор входит в систему с учетной записью библиотекаря.

Библиотекарь имеет доступ:
• по чтению к отношениям: readers, issue_cards, librarians_rooms, и rooms ;
• по чтению и записи к отношениям: readers, booking_cards, book_places, books, issue_casrds.

Читатель библиотеки может взаимодействовать с системой через программу-клиент, установленную в зале библиотеки или извне библиотеки через веб-интерфейс. При этом, он имеет доступ по чтению к отношениям: books, book_places, rooms, booking_cards, issue_cards.

При работе через программу-клиент читатель имеет также доступ по записи к отношению issue_cards — он может из читального зала забронировать книгу.

3 Формирование запросов к СУБД

3.1 Создание таблиц в базе данных и установка индексов

Для создания таблиц в соответствии с заданной схемой БД в СУБД MySQL можно использовать запросы, сгенерированые автоматически по схеме базы данных в среде MySQL Workbench (тут база данных называется library:

```
DROP SCHEMA IF EXISTS `library` ;

CREATE SCHEMA IF NOT EXISTS `library` DEFAULT CHARACTER SET utf8 ;
SHOW WARNINGS;
USE `library` ;

DROP TABLE IF EXISTS `library`.`administrators` ;

CREATE TABLE IF NOT EXISTS `library`.`administrators` (
  `id` INT NOT NULL,
  `logins` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
```

```
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`readers` ;

CREATE TABLE IF NOT EXISTS `library`.`readers` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `passport` VARCHAR(45) NOT NULL,
  `address` VARCHAR(45) NOT NULL,
  `phone` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`rooms` ;

CREATE TABLE IF NOT EXISTS `library`.`rooms` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`librarians` ;

CREATE TABLE IF NOT EXISTS `library`.`librarians` (
  `id` INT NOT NULL,
  `login` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`books` ;

CREATE TABLE IF NOT EXISTS `library`.`books` (
  `id` INT NOT NULL,
  `author` VARCHAR(45) NOT NULL,
  `publication_year` INT NOT NULL,
  `publisher` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `isbn` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`librarian_rooms` ;

CREATE TABLE IF NOT EXISTS `library`.`librarian_rooms` (
  `id_room` INT NOT NULL,
  `id_librarian` INT NOT NULL,
  PRIMARY KEY (`id_room`, `id_librarian`),
  INDEX `id_librarian_idx` (`id_librarian` ASC),
  CONSTRAINT `id_lr_room`
    FOREIGN KEY (`id_room`)
    REFERENCES `library`.`rooms` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `id_lr_librarian`
    FOREIGN KEY (`id_librarian`)
    REFERENCES `library`.`librarians` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = tis620
COLLATE = tis620_bin;

DROP TABLE IF EXISTS `library`.`booking_cards` ;

CREATE TABLE IF NOT EXISTS `library`.`booking_cards` (
  `id` INT NOT NULL,
  `id_reader` INT NOT NULL,
  `id_book` INT NOT NULL,
  `id_librarian` INT NOT NULL,
  `time` DATETIME NOT NULL,
  `period` DATETIME NULL,
  PRIMARY KEY (`id`),
  INDEX `id_reader_idx` (`id_reader` ASC),
  INDEX `id_book_idx` (`id_book` ASC),
  INDEX `id_librarian_idx` (`id_librarian` ASC),
```

```
  CONSTRAINT `id_bc_reader`
    FOREIGN KEY (`id_reader`)
    REFERENCES `library`.`readers` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `id_bc_book`
    FOREIGN KEY (`id_book`)
    REFERENCES `library`.`books` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `id_bc_librarian`
    FOREIGN KEY (`id_librarian`)
    REFERENCES `library`.`librarians` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`issue_cards` ;

CREATE TABLE IF NOT EXISTS `library`.`issue_cards` (
  `id` INT NOT NULL,
  `id_reader` INT NOT NULL,
  `id_book` INT NOT NULL,
  `time` DATETIME NOT NULL,
  `period` DATETIME NULL,
  `issue_cardscol` VARCHAR(45) NULL,
  PRIMARY KEY (`id`),
  INDEX `id_reader_idx` (`id_reader` ASC),
  INDEX `id_book_idx` (`id_book` ASC),
  CONSTRAINT `id_ic_reader`
    FOREIGN KEY (`id_reader`)
    REFERENCES `library`.`readers` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `id_ic_book`
    FOREIGN KEY (`id_book`)
    REFERENCES `library`.`books` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

DROP TABLE IF EXISTS `library`.`book_places` ;

CREATE TABLE IF NOT EXISTS `library`.`book_places` (
  `id_book` INT NULL,
  `id_room` INT NOT NULL,
  `quantity` INT NOT NULL,
  `shell_number` INT NOT NULL,
  PRIMARY KEY (`id_book`, `id_room`),
  INDEX `id_room_idx` (`id_room` ASC),
  CONSTRAINT `id_bp_book`
    FOREIGN KEY (`id_book`)
    REFERENCES `library`.`books` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `id_bp_room`
    FOREIGN KEY (`id_room`)
    REFERENCES `library`.`rooms` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Приведенный запрос успешно выполнен на сервере, в результате — созданы таблицы. Тут стоит обратить внимание на именование ограничений для внешний ключей — они должны быть уникальными в базе данных, поэтому в имени ограничения кодируется имя отношения, для которого оно описано, например

CONSTRAINT `id_ic_reader`

задает ограничение на поле id_reader в отношении issue_cards.

Видно, что для всех ключевых полей в базе данных проставлены индексы — за счет этого записи упорядочиваются по этим полям и поиск выполняется быстрее (бинарный поиск вместо линейного).

3.2 Проектирование наиболее востребованных запросов

Перед созданием запросов был установлен и запущен MySQL Server, настроено подключение к этому серверу среды MySQL Workbench. В базу были добавлены данные для проверки корректности выполнения запросов. Добавление производилось с помощью MySQL Workbench, в результате были сгенерированы следующие запросы:

```
INSERT INTO `library`.`administrators` (`id`, `logins`, `password`) VALUES (1, 'lena',
'12345');
INSERT INTO `library`.`administrators` (`id`, `logins`, `password`) VALUES (2, 'petya',
'54321');

INSERT INTO `library`.`readers` (`id`, `name`, `passport`, `address`, `phone`) VALUES
(1, 'vasya', '0402 892322', 'Moskva, Kreml', '214 34 12');
INSERT INTO `library`.`readers` (`id`, `name`, `passport`, `address`, `phone`) VALUES
(2, 'kostya', '4561 455311', 'Spb, Mira 11', '8 909 999 99 99');

INSERT INTO `library`.`rooms` (`id`, `name`) VALUES (1, 'Зал C++');
INSERT INTO `library`.`rooms` (`id`, `name`) VALUES (2, 'Зал проектирование');

INSERT INTO `library`.`librarians` (`id`, `login`, `password`) VALUES (1, 'vova',
'11111');
INSERT INTO `library`.`librarians` (`id`, `login`, `password`) VALUES (2, 'sveta',
'22222');

INSERT INTO `library`.`books` (`id`, `author`, `publication_year`, `publisher`, `name`,
`isbn`) VALUES (1, ' Э. Гамма, Р. Хелм, Р. Джонсон', 2009, 'СПб.: Питер', 'Приемы ОО-
проектирования', NULL);
INSERT INTO `library`.`books` (`id`, `author`, `publication_year`, `publisher`, `name`,
`isbn`) VALUES (2, 'Джейсон Мак-Колм Смит', 2013, 'Вильямс', 'Элементарные шаблоны
проектирования', NULL);
INSERT INTO `library`.`books` (`id`, `author`, `publication_year`, `publisher`, `name`,
`isbn`) VALUES (3, 'Стивен Прата', 2020, 'Вильямс', 'Язык программирования C++ (C++11).
', NULL);
INSERT INTO `library`.`books` (`id`, `author`, `publication_year`, `publisher`, `name`,
`isbn`) VALUES (4, 'Мейерс С.', 2014, 'ДМК Пресс', 'Эффективное использование C++',
NULL);
INSERT INTO `library`.`books` (`id`, `author`, `publication_year`, `publisher`, `name`,
`isbn`) VALUES (5, 'Андрей Александреску', 2002, 'Вильямс', 'Современное проектирование
на C++.', NULL);

INSERT INTO `library`.`librarian_rooms` (`id_room`, `id_librarian`) VALUES (1, 2);
INSERT INTO `library`.`librarian_rooms` (`id_room`, `id_librarian`) VALUES (2, 1);

INSERT INTO `library`.`book_places` (`id_book`, `id_room`, `quantity`, `shell_number`)
VALUES (1, 1, 10, 555);
INSERT INTO `library`.`book_places` (`id_book`, `id_room`, `quantity`, `shell_number`)
VALUES (1, 2, 5, 333);
INSERT INTO `library`.`book_places` (`id_book`, `id_room`, `quantity`, `shell_number`)
VALUES (2, 1, 4, 111);
INSERT INTO `library`.`book_places` (`id_book`, `id_room`, `quantity`, `shell_number`)
VALUES (3, 2, 60, 222);

INSERT INTO `library`.`booking_cards` (`id`, `id_reader`, `id_book`, `id_librarian`,
`time`, `period`) VALUES (1, 1, 1, 2, '2019-10-20', '2019-11-20');
```

При подготовке этих данных выявилась проблема разработанной базы — не все названия книг умещаются в 45 символов. Пришлось сокращать названия. Также, из-за того, что пользователи распределены по трем таблицам — то при создании нового пользователя надо выполнять проверку отсутствия логина во всех таблицах.

Для получения книг по фильтру должны выполняться запросы на подобии такого:

```
select * from books where name like '%C++%'
```

В данном случае выводятся книги, в названии которых есть подстрока «C++». Результат выполнения запроса приведен на рисунке 7.

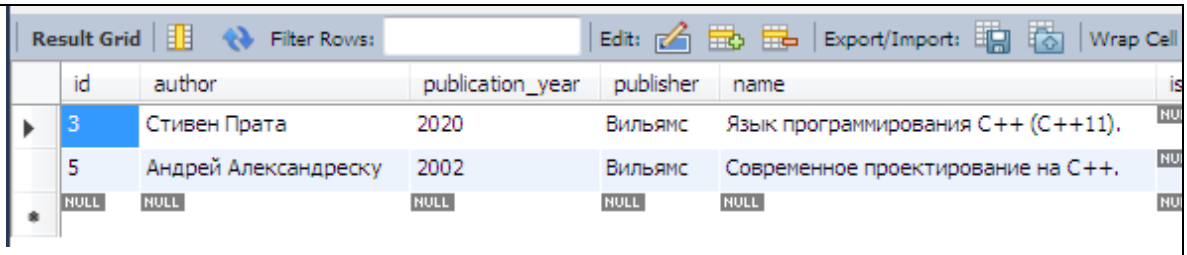


Рисунок 7.

Для поиска должников можно выполнить такой запрос:

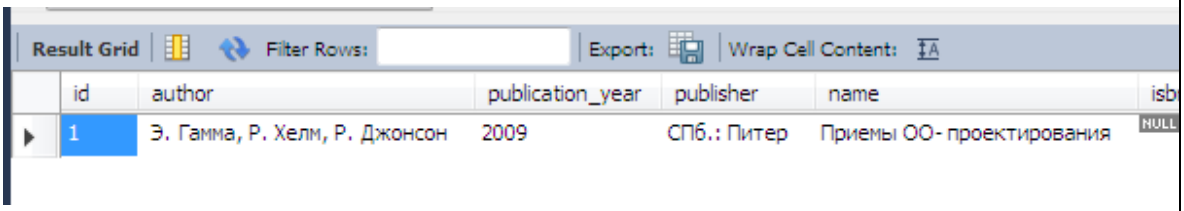
```
select rd.* from readers rd, booking_cards bc
where rd.id = bc.id_reader and bc.period < '2021-10-20';
```

Вместо константы должна поставляться текущая дата.

Для отображения книг, которые задолжал конкретный пользователь можно выполнить такой запрос:

```
select bk.* from booking_cards bc, books bk
where
bk.id = bc.id and
bc.period < '2021-10-20' and
bc.id_reader = 1;
```

Для этого запроса информационная система должна сначала находить пользователя в базе (получать его id) и подставлять это значение вместо 1, вместо константы даты должна подставляться текущая дата. Результат выполнения запроса приведена на рисунке 8



.

Рисунок 8.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Создать базу данных предметной области (согласно методическим рекомендациям и индивидуальному варианту, приведенному в приложении А).

Задание 2. Научиться заполнять, изменять и удалять данные в таблицах. Заполнить таблицы базы данных (согласно методическим рекомендациям и индивидуальному варианту, приведенному в приложении А).

Задание 3. Изучить основные операторы выборки, сортировки, ограничения выборки, группировки записей и сохранения результатов выборки во внешний файл (согласно методическим рекомендациям и индивидуальному варианту, приведенному в приложении А).

Задание 4. Научиться создавать, вызывать и удалять хранимые процедуры, используя примеры методических рекомендаций и наработки согласно методическим рекомендациям и индивидуальному варианту, приведенному в приложении А и другие, ранее созданные базы данных.

Задание 5. Оформить отчет, содержащий подробное описание выполненной работы, скрины диалоговых окон (команд) и вывод по лабораторной работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение понятиям База данных и Система управления базами данных.
2. Назовите команды создания и удаления базы данных. В чем разница создания базы данных и таблицы?
3. Какие типы данных используются при работе с MySQL?
4. Чем отличается реляционная модель данных от объектно-ориентированной модели?
5. Какое количество таблиц может содержать реляционная модель данных?
6. Какую команду выполняет однострочный оператор INSERT?
7. Для чего используется ключевое слово SET?
8. При помощи, какой команды осуществляется изменение данных?
9. При помощи, какой команды осуществляется удаление данных?
10. Дайте определение понятию Хранимые процедуры.
11. Какие типы функций поддерживает MySQL?
12. Какие существуют классы пользовательских функций?
13. Какие существуют ограничения на пользовательские функции?
14. Переменные, каких типов данных нельзя передавать в пользовательскую функцию?
15. Какие операторы используются для создания и вызова хранимых процедур?
16. При помощи, какой команды можно удалить пользовательскую функцию?
17. Какой оператор используется для выборки записей из таблицы?
18. Для чего используются конструкции BETWEEN и NOT BETWEEN?
19. Опишите конструкции IN и NOT IN, в чем их разница?
20. Какие виды сортировок существуют?
21. С какой целью используется конструкция GROUP BY совместно с функцией COUNT()?
22. Как осуществить ограничение выборки по количеству выводимых записей?

Лабораторная работа №3
«ОРГАНИЗАЦИЯ ЛОКАЛЬНОЙ СЕТИ. НАСТРОЙКА ЛОКАЛЬНОЙ СЕТИ»

Цель работы: получить теоретические знания и практические навыки организации локальной сети, настройки локальной сети.

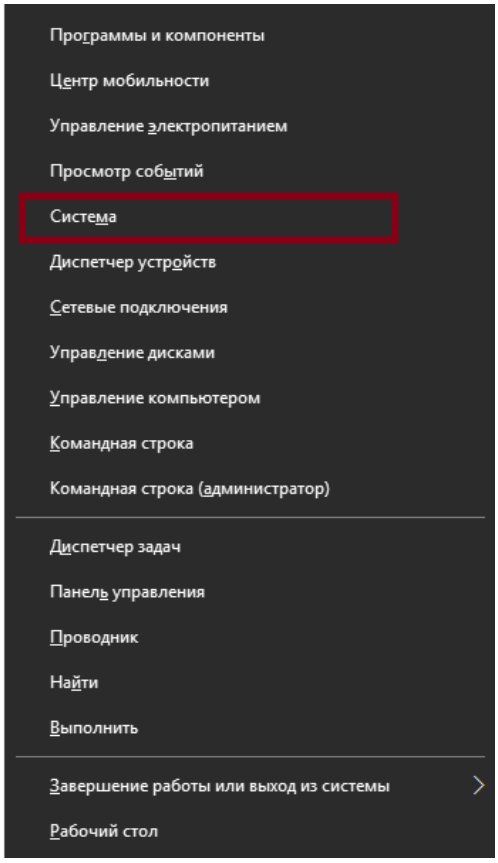
Теоретические сведения

Есть два основных способа как вы можете создать локальную сеть. Это создание беспроводной локальной сети через wi-fi, или же создание локальной сети, используя сетевой кабель.

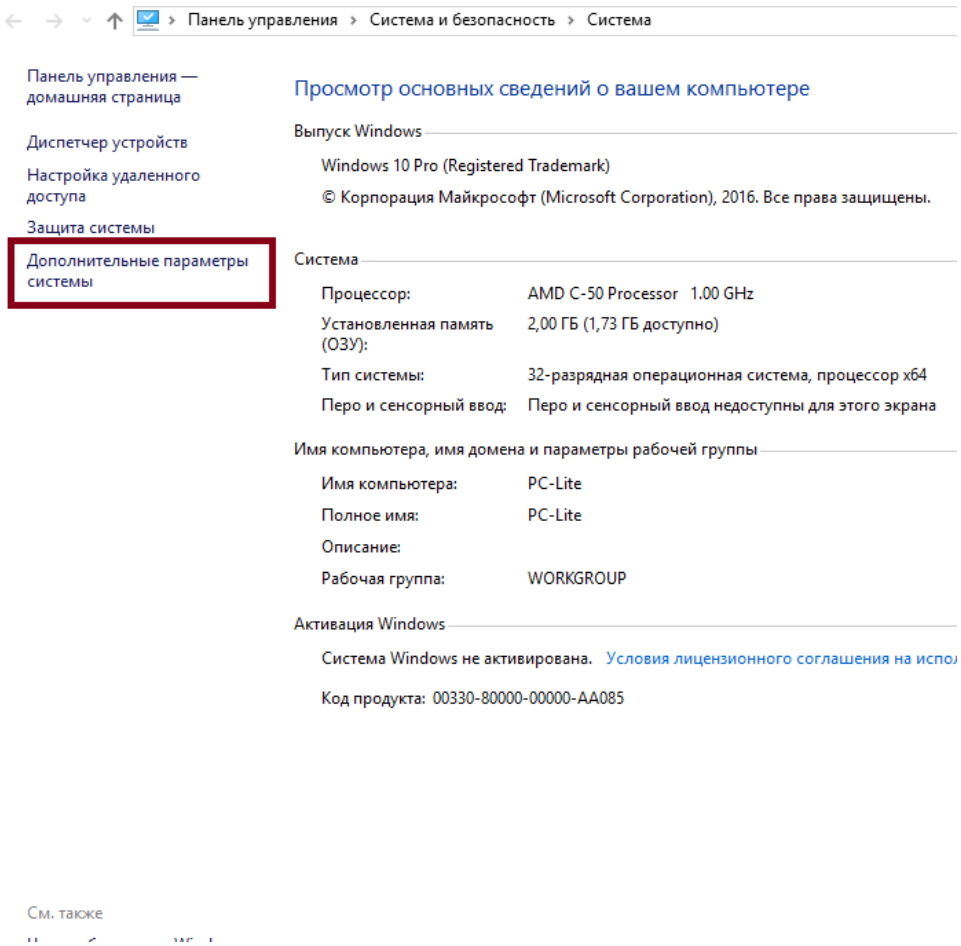
1. Создание ЛС с помощью кабеля

Стоит сразу указать, что у вас дома присутствует wi-fi роутер и все компьютеры подключены к нему, никаких дополнительных усилий для создания сети не требуется. Между вашими компьютерами уже есть связь, так что подключать их дополнительно друг к другу нет нужды. Но в большинстве случаев потребность в создании локальной сети возникает тогда, когда такого подключения нет. В этом случае вам понадобится соединить компьютеры напрямую друг с другом (современные модели без труда соединяются простым интернет-кабелем). Затем, первым делом следует убедиться, что параметру «Рабочая группа» присвоено одно и то же название на каждом из подключаемых устройств. Для этого стоит проделать следующие действия:

1) Нажмите сочетание клавиш Win+X и выберите раздел «Система» из списка (также можете кликнуть правой кнопкой мыши в левом нижнем углу экрана для вызова этого списка).

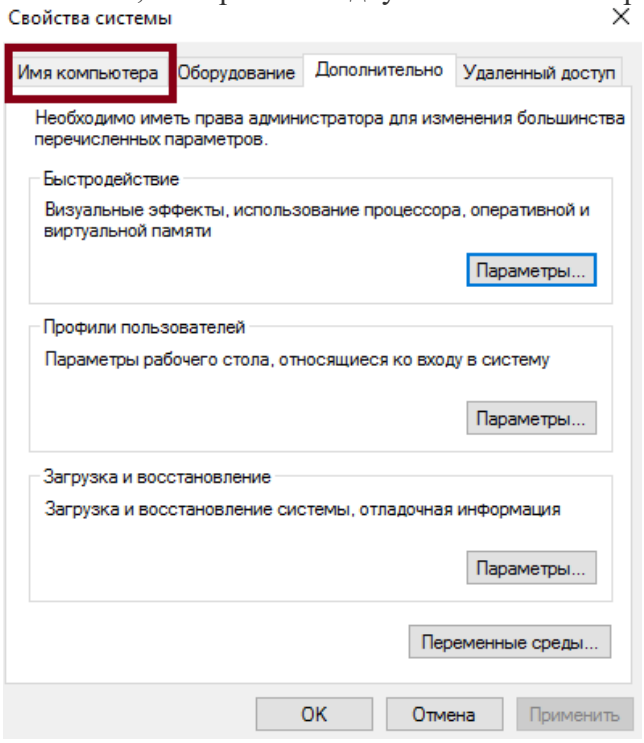


2) Затем, нажмите «Дополнительные параметры системы».



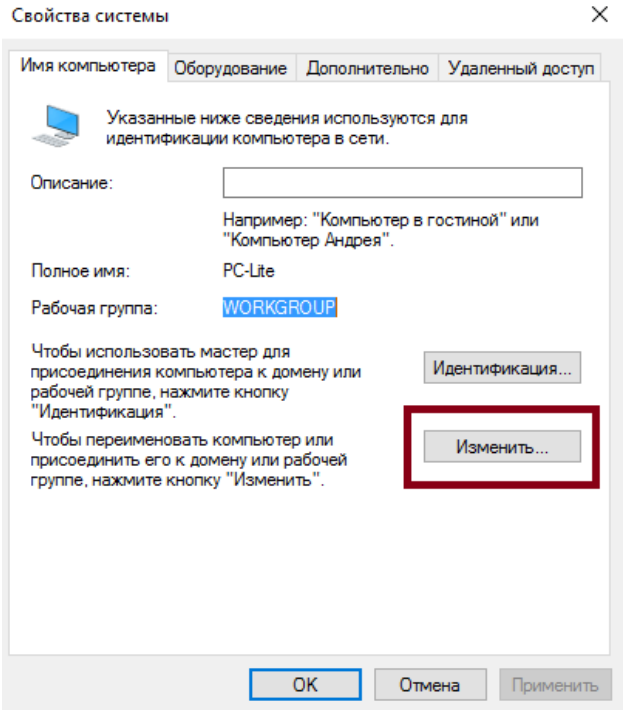
Выберите пункт «Дополнительные параметры системы

3) И в появившемся окне, выберите вкладку «Имя компьютера».



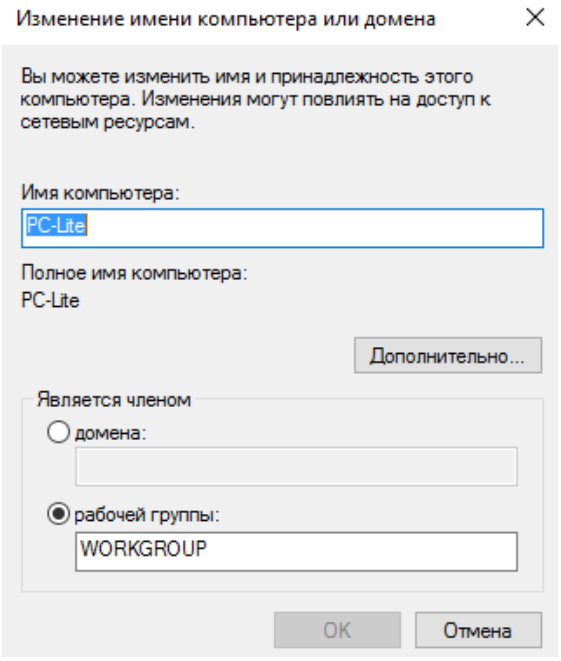
Выберите вкладку «Имя компьютера»

4) Здесь будет указано текущее название рабочей группы, установленное по умолчанию (если вы не меняли его до этого). Нажмите кнопку «Изменить» для смены названия.



Нажмите кнопку «Изменить»

5) Вы можете ввести любое имя, лишь бы оно было одинаковым на всех устройствах которые вы желаете объединить в общую локальную сеть.



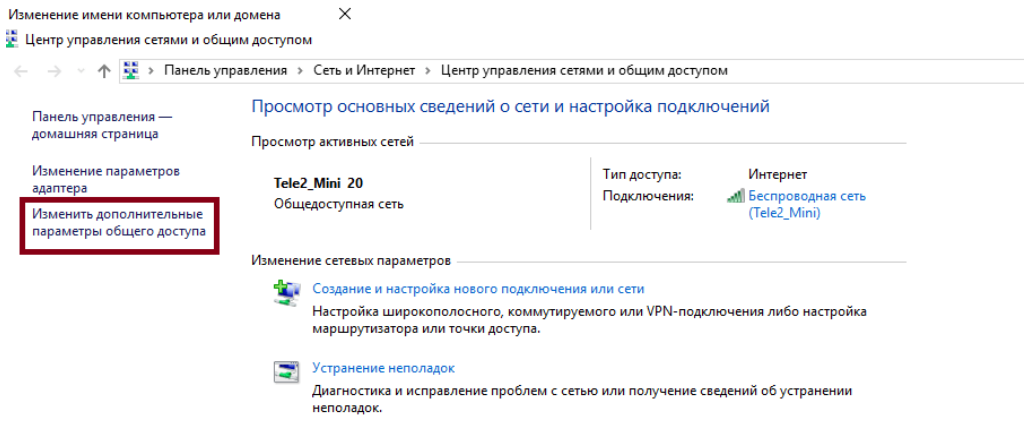
Введите имя группы, одинаковое для всех устройств

Далее, необходимо задать настройки сетевого обнаружения. Включённое сетевое обнаружение позволит взаимодействовать с вашим компьютером в локальной сети, в то время, как отключённое, заблокирует эту возможность. Хорошо уметь переключать этот параметр, в целях безопасности. Делается это следующим образом:

1. Кликнете правой кнопкой мыши на значок подключения к сети в трее (правый нижний угол экрана).

2. Выберите пункт «Центр управления сетями и общим доступом».

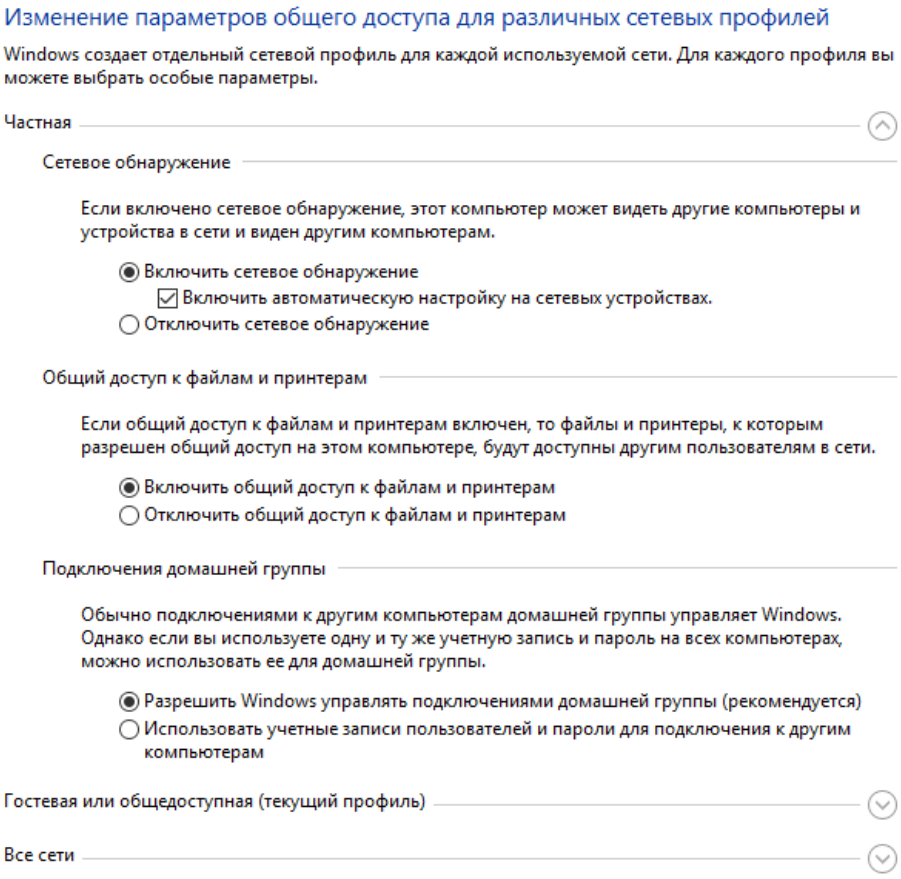
3. Слева от основного окна, следует выбрать «изменение дополнительных параметров».



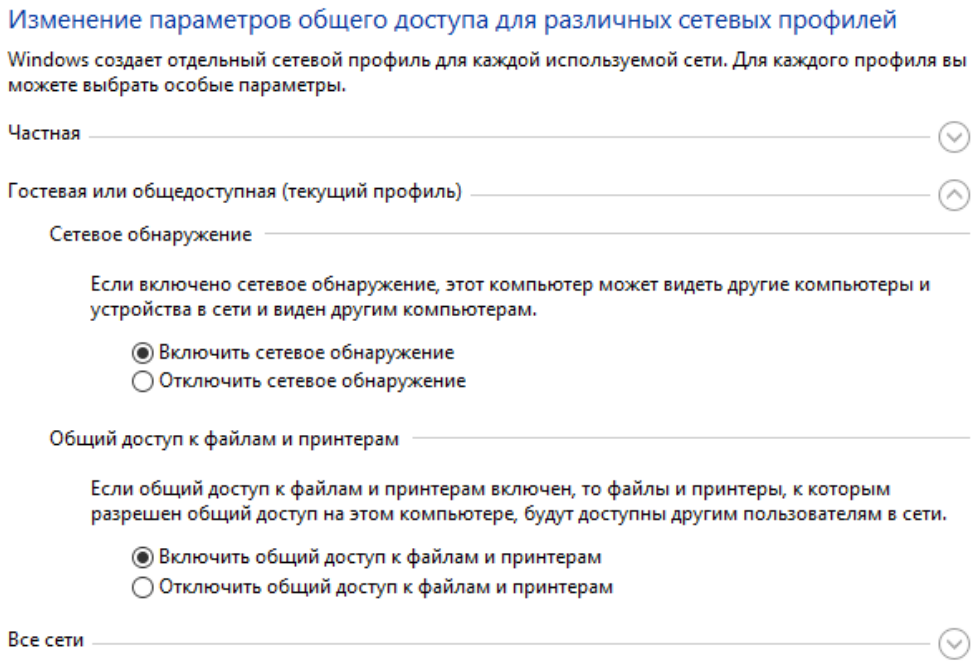
Выберите «Изменение дополнительных параметров…»

4. В открывшемся окне, необходимо задать определённые настройки. В каждом из профилей дайте доступ ко всем возможным параметрам, кроме доступа с парольной защитой. Его необходимо отключить. Если в дальнейшем вам понадобится выключить сетевое обнаружение устройства просто проделайте обратные действия в этом окне.

Включите и Разрешите все пункты



Включите оба пункта

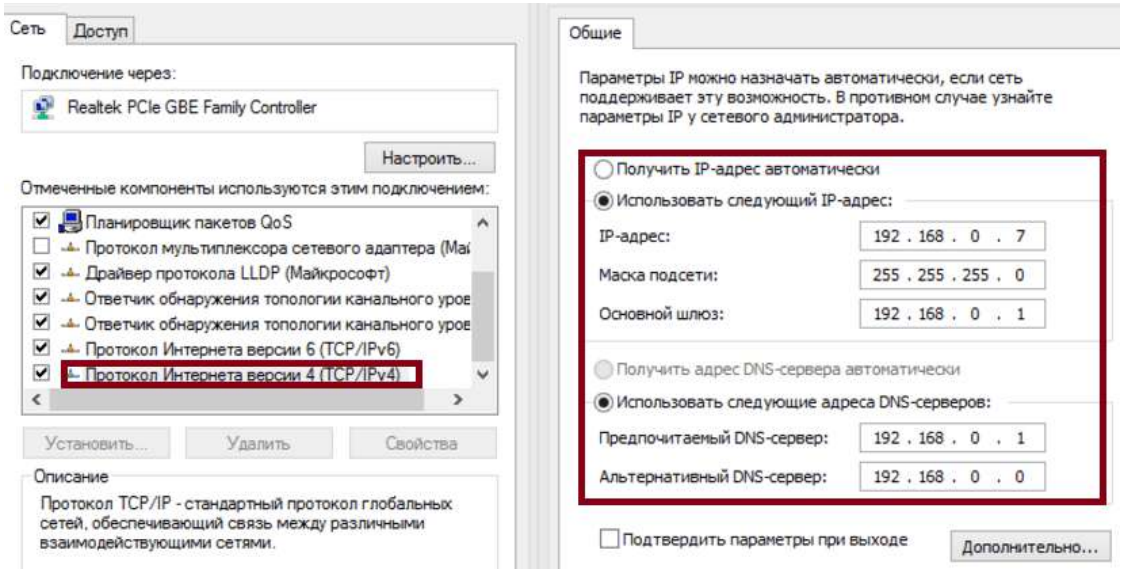


Включите всё, кроме раздела «Общий доступ с парольной защитой»

5.Убедитесь, что настройки выставлены так же как и на изображениях выше.
6.Сохраните внесённые настройки и закройте это окно.

Этого уже достаточно если компьютеры соединены кабелями через роутер. Но если же они соединены напрямую, следуют провести ряд дополнительных настроек. Делаем следующее:

1. Нажимаем сочетание клавиш Win+X.
2. Выбираем пункт «Сетевые подключения» в появившемся списке.
3. Выбираем подключение к сети через ваше устройство и вызываем контекстное меню, нажав правую кнопку мыши на него.
4. Заходим в «Свойства» устройства.
5. Далее, открываем свойства компонента (TCP/iPc4)
6. И задаём настройки как на изображении ниже, при учёте того что последняя цифра IP адреса должна отличаться у каждого компьютера в локальной сети.



Установите значения как на изображении

7. Принимаем заданные настройки.

Таким образом, мы создали и настроили локальную сеть через сетевой кабель. Компьютеры имеют общий доступ к файлам и принтеру друг друга, что удовлетворят нашим целям.

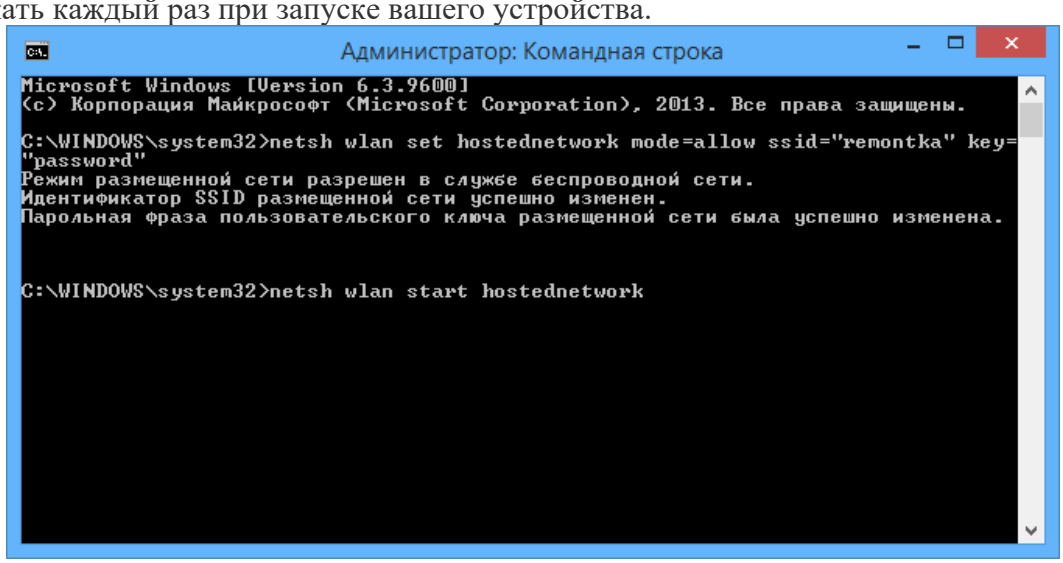
2. Как создать и настроить беспроводную сеть через Wi-Fi соединение

Беспроводное соединение является более удобным для большинства пользователей, хоть связь по нему и может быть менее стабильной чем хотелось бы. Для его создания в Windows 10 придётся использовать командную строку. Но для упрощения процесса который необходимо повторять каждый раз при включении компьютера, мы сразу рассмотрим создание исполняемого файла, что будет воспроизводить эту команду. Для этого создаём текстовый файл и вводим туда следующий блок команд:

```
netsh wlan set hostednetwork mode=allow ssid=”имя-сети” key=”пароль-для-подключения”
netsh wlan start hostednetwork
```

При этом название сети и пароль от неё должны быть введены без кавычек.

Далее, при сохранении файла меняем формат .txt на .bat просто сменив подпись формата файла, после точки в его наименовании. Исполняемый файл готов. Его стоит запускать каждый раз при запуске вашего устройства.

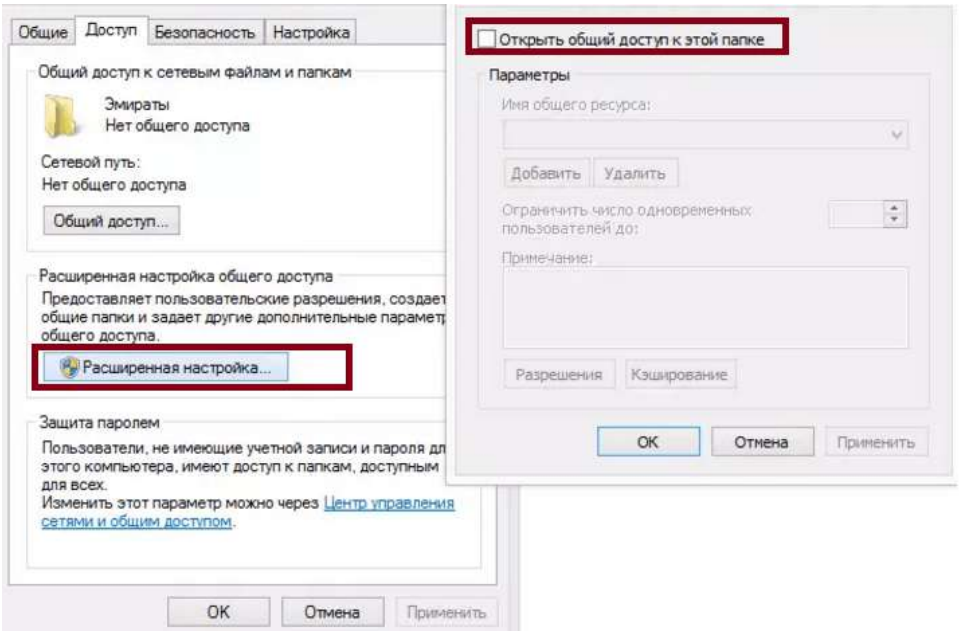


После того как сеть будет запущена этой серией команд, другое устройство, может подключиться к ней используя установленные вами пароль для подключения и название сети.

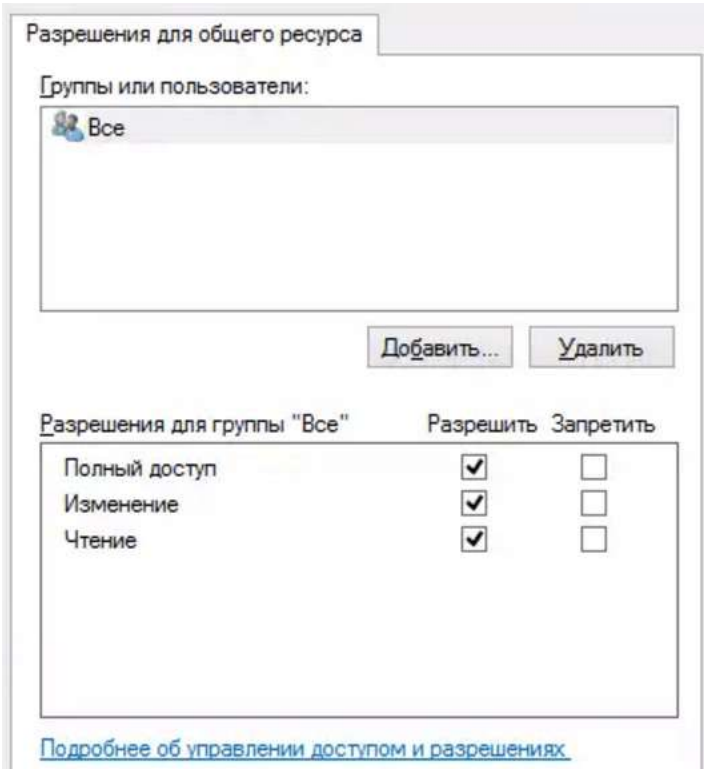
Настройка доступ к папкам в созданной сети

Теперь когда локальная сеть между двумя компьютерами установлена, разберёмся как открыть общий доступ к тем или иным папкам. Общий доступ можно настроить как к любой отдельной папке, так и ко всему диску — в зависимости от ваших целей и уровня доверия пользователю другого компьютера. Для этого:

1. Нажмите правую кнопку мыши на любой папке, у которой хотите изменить настройки доступа и вберите раздел «Свойства».
2. Перейдите на пункт «Доступ» и выберите расширенные настройки доступа.
3. У вас будет доступен только пункт для открытия общего доступа к этой папке. Отметьте его.
4.



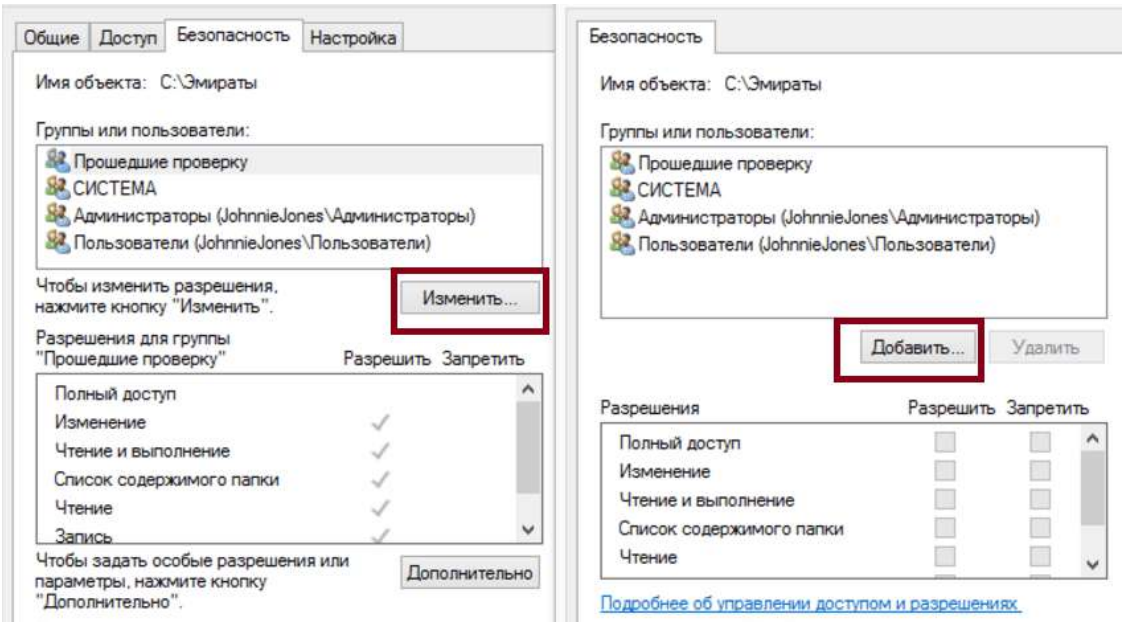
4. Далее, выбираем раздел «Разрешения» и настраиваем права общего доступа к папке. В верхней части окна указывается конкретный пользователь или группа пользователей которая получает доступ, а в нижней — доступ какого рода им будет предоставлен.



1. Тут указанны группы пользователей и права, которые им даны

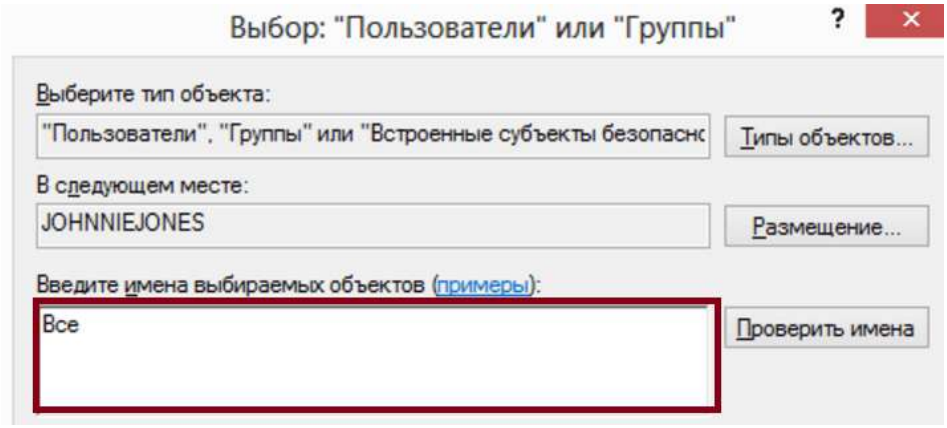
Теперь остаётся лишь провести настройки безопасности. Для этого там же, в свойствах папки переходим в соответствующую вкладку и делаем следующее:

1. В меню разрешений для отдельных групп нажимаем «Изменить».
2. В следующем, выбираем добавление новой группы.
3.



Нажмите кнопку «Добавить»

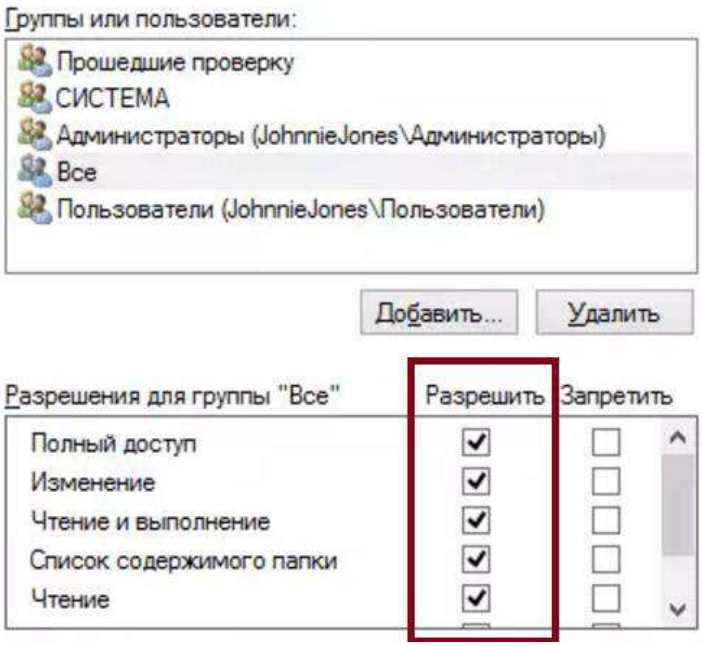
4. В окно ввода имени выбираемых объектов вводим «Все», как показано на изображении.



Введите слово «Все» в соответствующее окно

5. Указываем, на группу которую мы только что создали.

6. Ставим галочки разрешений напротив каждого из пунктов (или напротив тех, что вам необходимы).



Задаём права безопасности для созданной группы

7. Принимаем внесённые изменения и перезагружаем компьютер.

Таким образом, был настроен доступ для пользователей к указанной папке. Вы можете сделать это для любого числа папок и пользователей.

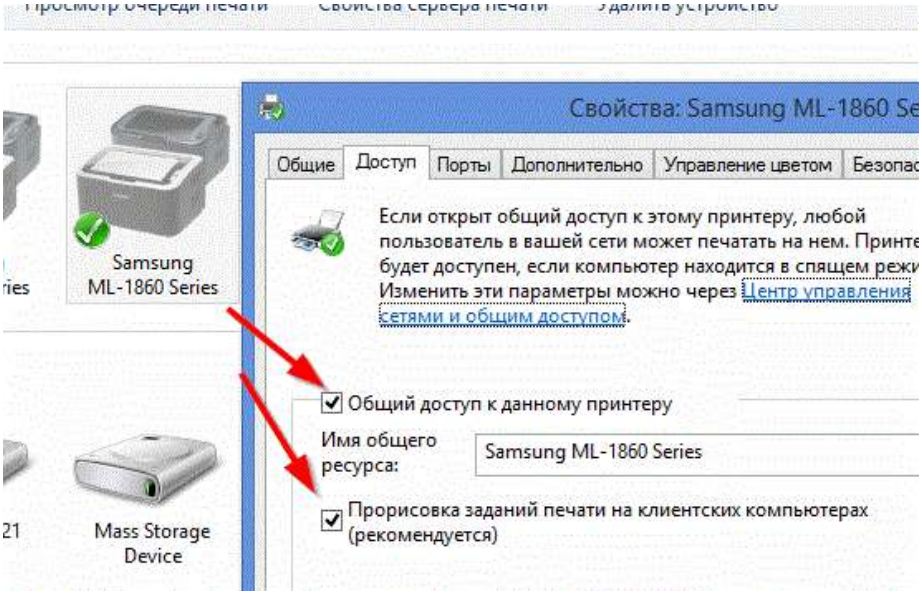
Добавление нового устройства

Если вы уже настроили локальную сеть по инструкции выше, то добавить новое устройство для общего использования не составит никакого труда. Для этого достаточно открыть панель управления вашего компьютера, и перейти в раздел «Оборудование и звук», а затем выбрать «Устройства и принтеры». В этой папке вы увидите все подключённые устройства. Делаем следующее:

1. Выбираем устройство, для которого необходимо задать общий доступ. Например, это может быть принтер.

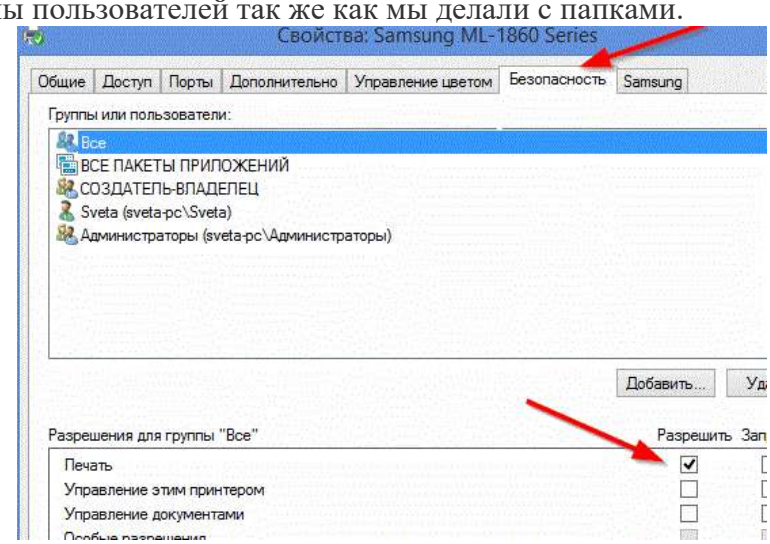
2. Нажатием правой кнопки, вызываем контекстное меню этого устройства и выбираем раздал «Свойства принтера».

3. Там, переходим, как и ранее, во вкладку «Доступ» и находим пункт настроек общего доступа к данному принтеру. Ставим галочки, чтобы дать этот доступ.



Поставьте галочки в соответствующих пунктах

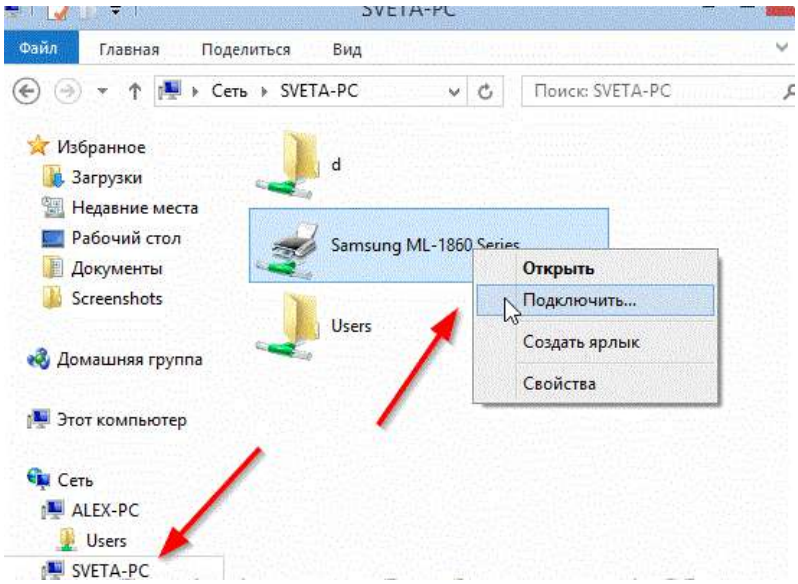
4. Остаётся лишь зайти в настройки безопасности и там указать права для группы пользователей так же как мы делали с папками.



Выставьте требуемые права, например на «Печать»

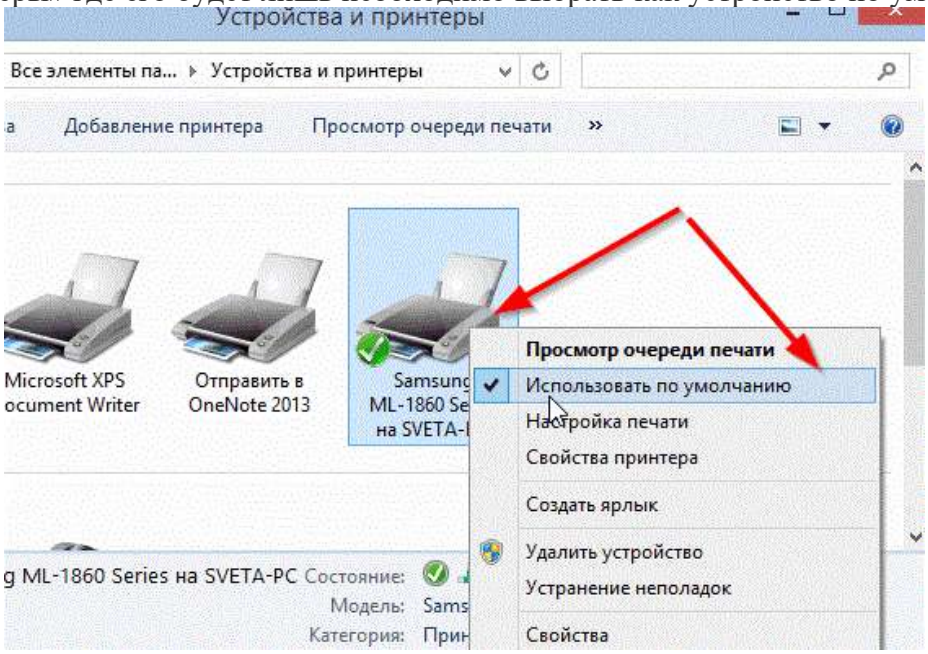
Теперь, когда общий доступ устройству открыт, требуется лишь подключить его действуя через компьютер в локальной сети. Делается это так:

1. Выберите пользователя устройством которого хотите воспользоваться и вы должны увидеть устройство в общем доступе.
2. Нажмите правой кнопкой мыши по нему и выберите «Подключить».



Нажмите «Подключить»

3. После этого устройство отобразится в вашем разделе «Устройство и Принтеры» где его будет лишь необходимо выбрать как устройство по умолчанию.



Выберите устройство и установите его как устройство по умолчанию

4. Если всё сделано правильно, вы сможете использовать подключённый через локальную сеть принтер (или другое устройство) без всяких проблем.

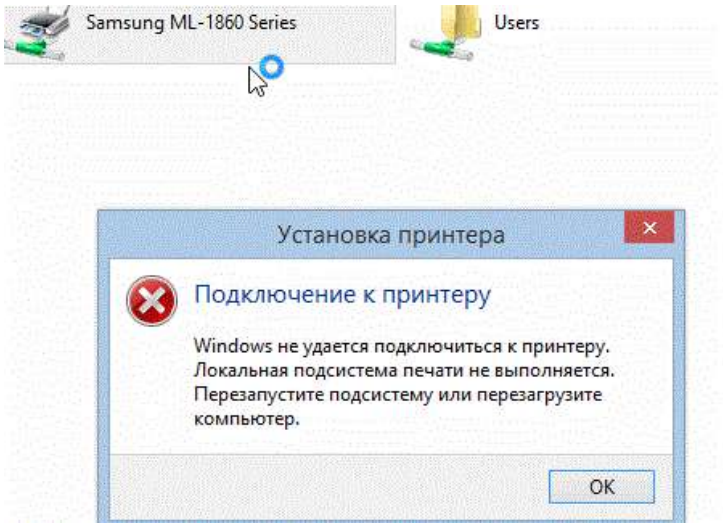
Проблемы подключения в Windows 10

Если вы правильно создали и настроили локальную сеть, у вас не должно быть особых проблем. Просто убедитесь, что:

- Вы правильно вводите ключ безопасности, при подключении к локальной сети через wi-fi
- Кабель подключён надёжно к обоим компьютерам.
- Вы дали все необходимые права доступа и безопасности.

- Все подключённые устройства имеют правильный IP адрес, если он не задаётся автоматически.
- В настройках включено сетевое обнаружение вашего устройства.

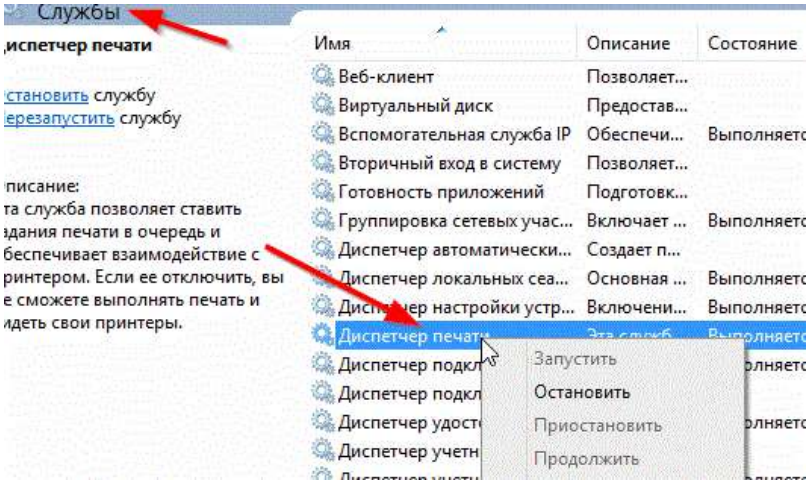
Кроме этого, есть ряд специфических проблем при подключении устройств.



Если у вас вылезла ошибка подобная этой, необходимо перезапустить службу печати

К примеру, если при попытке подключиться к устройству вы получили стандартную ошибку Windows о невозможности этого действия, следует предпринять следующие шаги:

1. Нажмите Win+X
2. В появившемся списке выберите «Управление компьютером»
3. Далее, вам надо перейти в раздел «Службы» и найти в списке «Диспетчер печати».
4. Отключите эту службу, перезагрузите компьютер и включите службу вновь. Скорее всего, ваша проблема будет решена.



Отключите службу, а затем снова включите её после перезагрузки

Удаление ЛС в Windows 10

Несмотря на то, что в большинстве случаев достаточно отключить обнаружение устройства в локальной сети методом указанным выше в этой статье, нет никакой нужды держать на компьютере локальные подключения которые не используются.

Поэтому рассмотрим как удалить уже созданную локальную сеть которая нам не нужна.

Сделать это в Windows 10 возможно через реестр. Для его вызова нажмите Win+R и в появившемся окне введите команду regedit.

Внимание, любое неверное изменение реестру может нанести вред вашему компьютеру. Вы совершаете эти действия на свой страх и риск.

В реестре, проследуйте по этому пути:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles

Там вы обнаружите множество следов былых сетевых подключений с конкретными именами. Просто удалите те подразделы, которые указывают вам на ненужные больше сетевые подключения.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Создайте и настройте локальную сеть.

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая сеть является локальной? Дайте ее описание.
2. Как создать локальную сеть?
3. Как настроить локальную сеть?

Лабораторная работа №4
«УСТАНОВКА И НАСТРОЙКА SQL-СЕРВЕРА»

Цель работы: получить теоретические знания и практические навыки установки и настройки SQL Server.

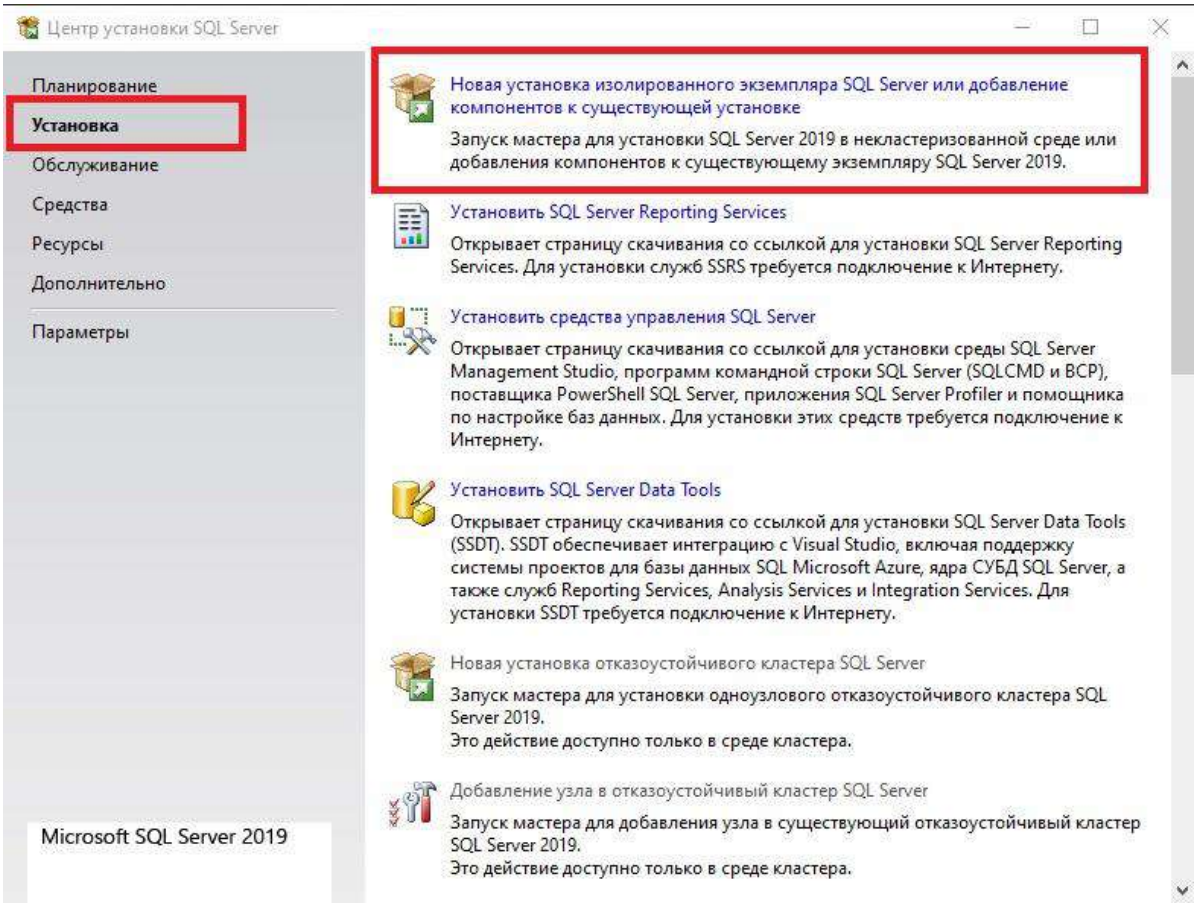
Теоретические сведения

1. Установка и настройка SQl Server

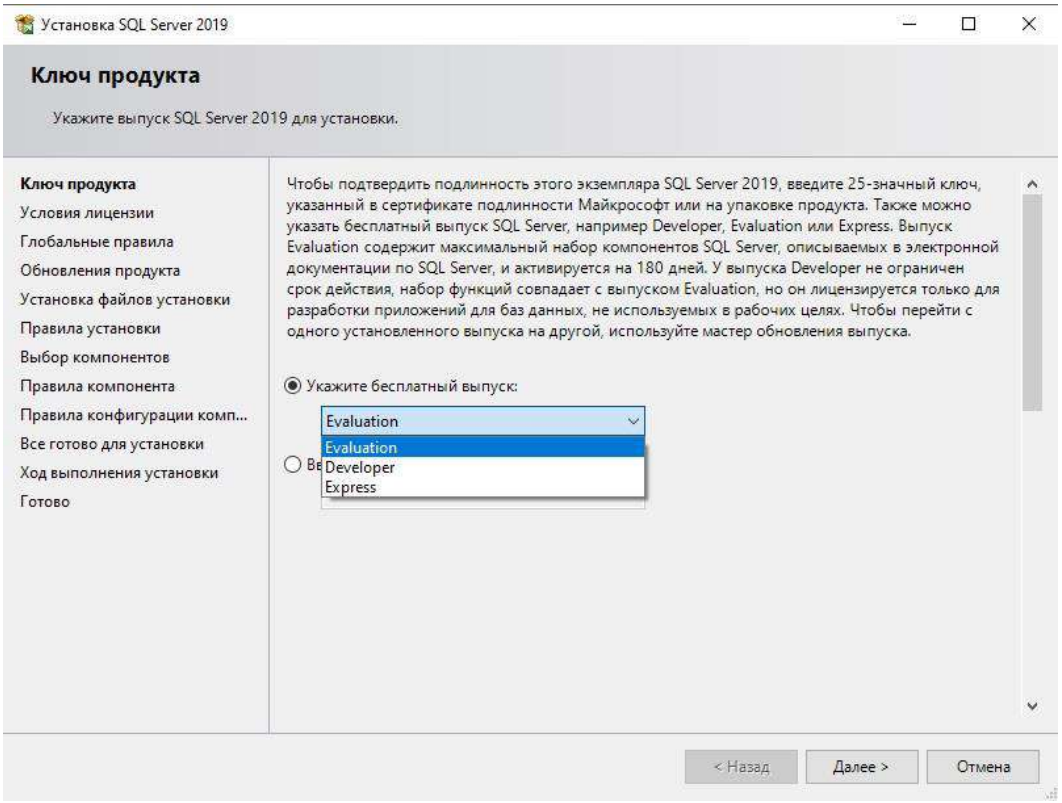
Рассмотрим как происходит установка SQL Server 2019, а так же начальные настройки приложения, которые можно задать при установке.

1) Скачиваем дистрибутив Microsoft SQL Server 2019.

2) Далее открываем установщик и начинаем инсталляцию приложения.

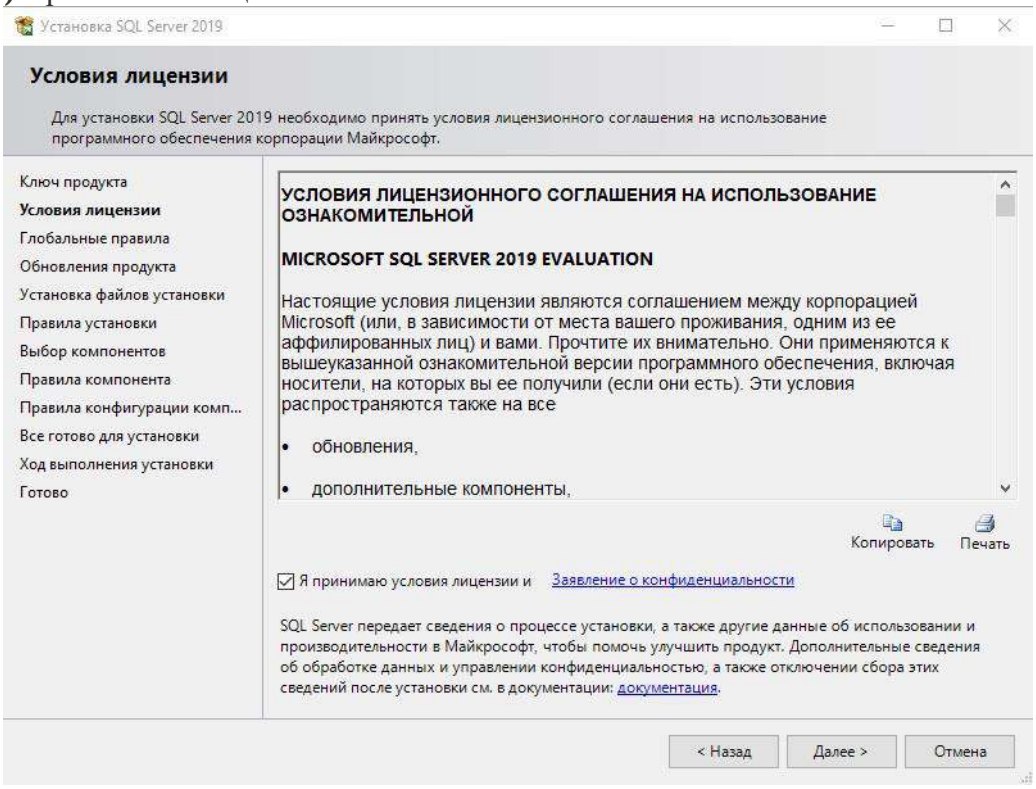


3) Далее, у нас запросят выбрать версию установки из бесплатных, к примеру Evaluation (ознакомительня версия). Бесплатные ознакомительные версии работают только 180 дней пробного периода, далее приложение закрывает свой функционал.

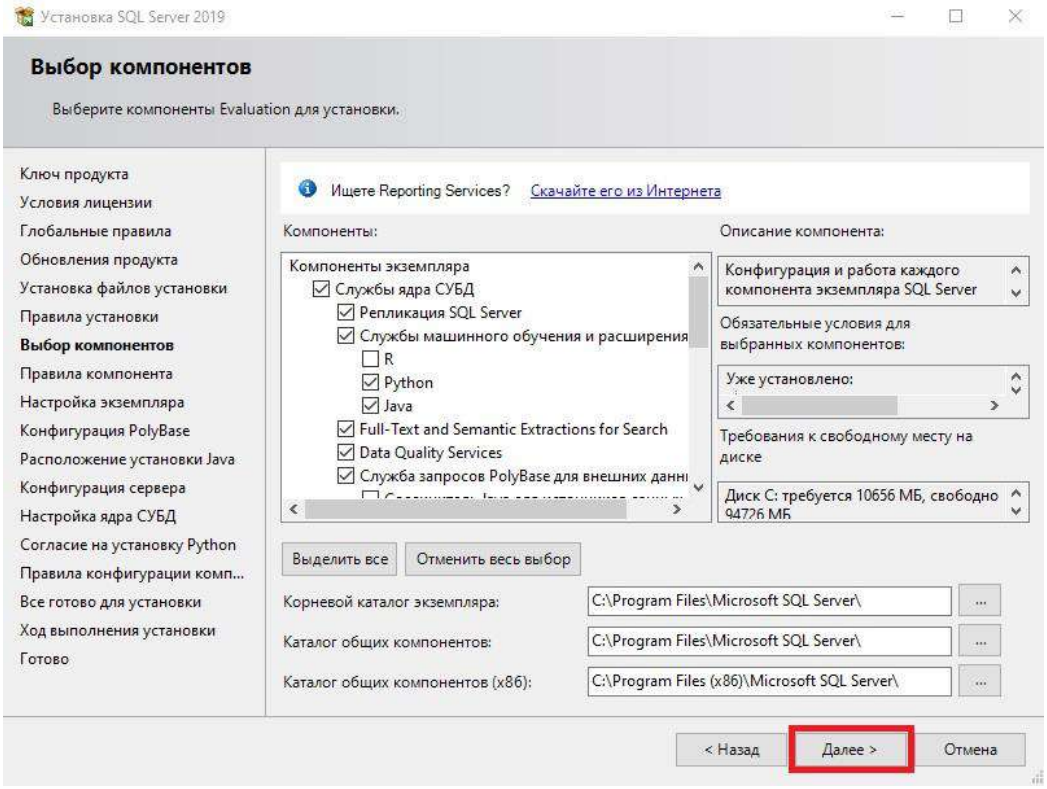


Либо если у Вас есть ключ активации от полнофункциональной версии, к примеру SQL Server 2019 Standard, то в нижнее поле можете ввести его.

4) Принимаете лицензионное соглашение.

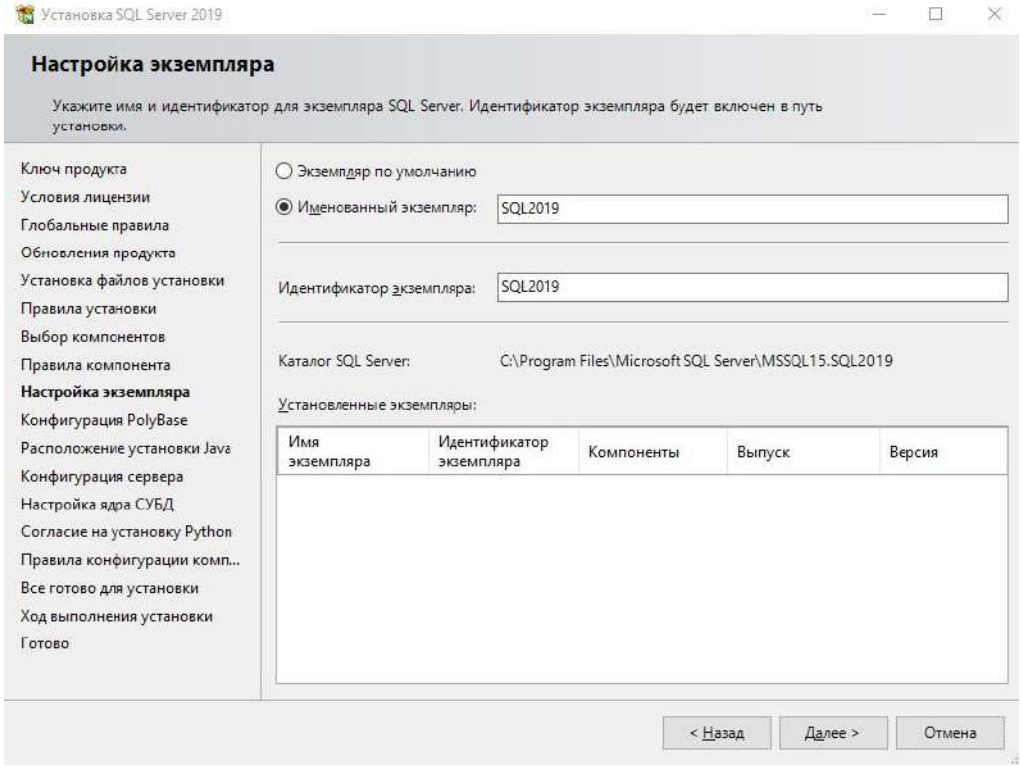


5) Далее, Вам предлагается выбрать компоненты для установки в SQL Server, выбираете нужные под Ваши задачи и жмем далее.

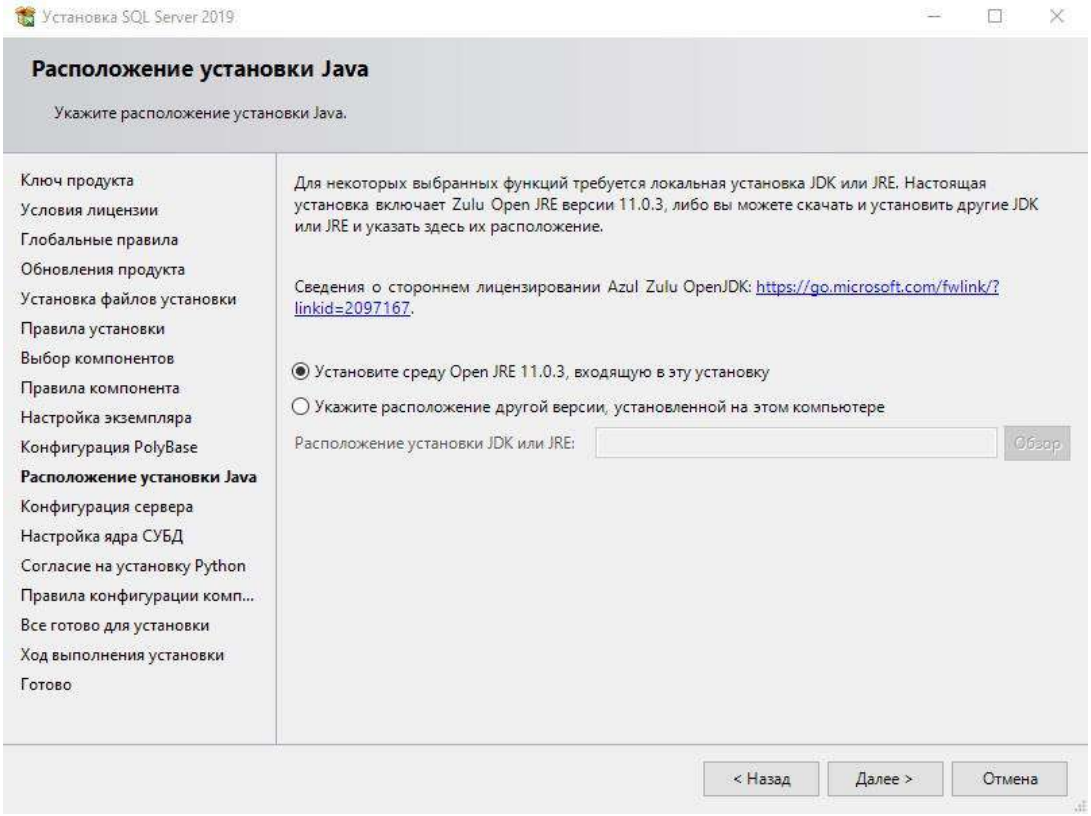


6) Далее задаем имя интерфейса. Если Вы планируете использовать только один экземпляр, то можете выбрать пункт "Экземпляр по умолчанию", в этом случае подключение будет осуществляться к этому экземпляру.

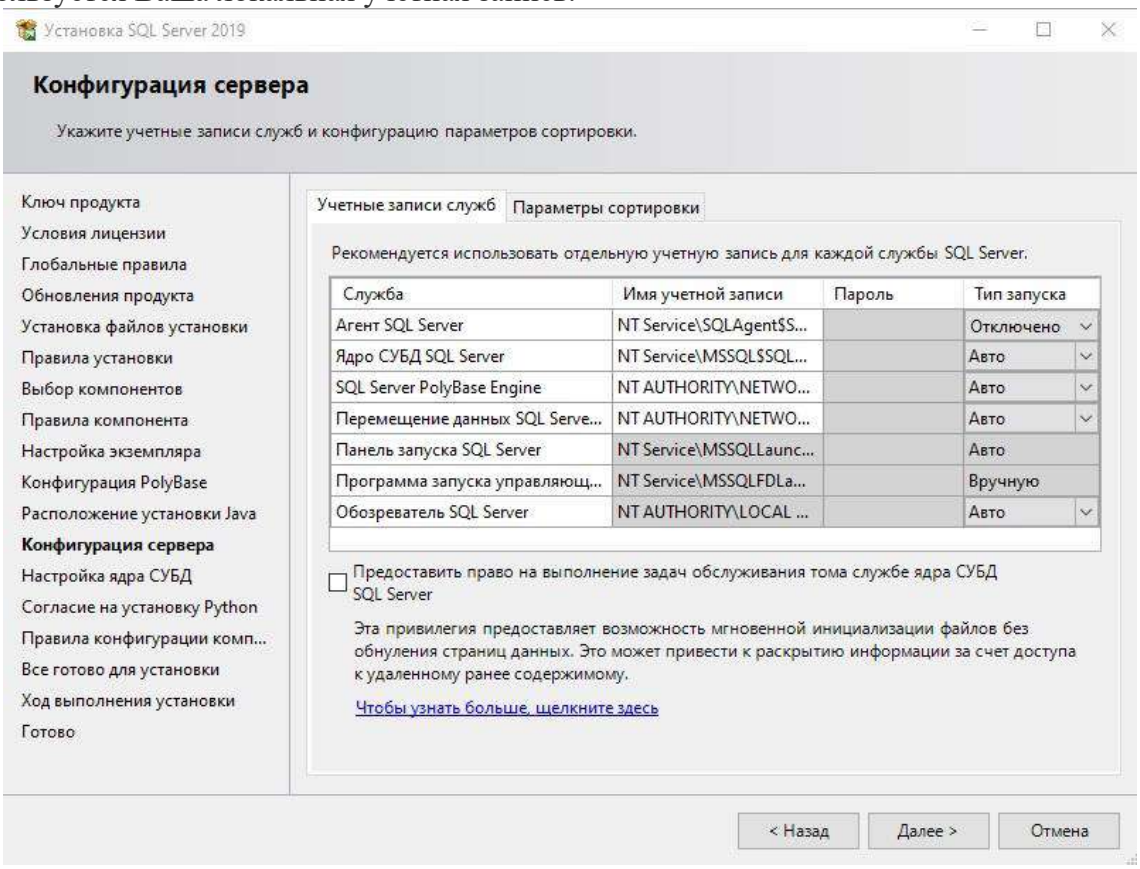
В нашем случае сделаем именованный и зададим ему имя.



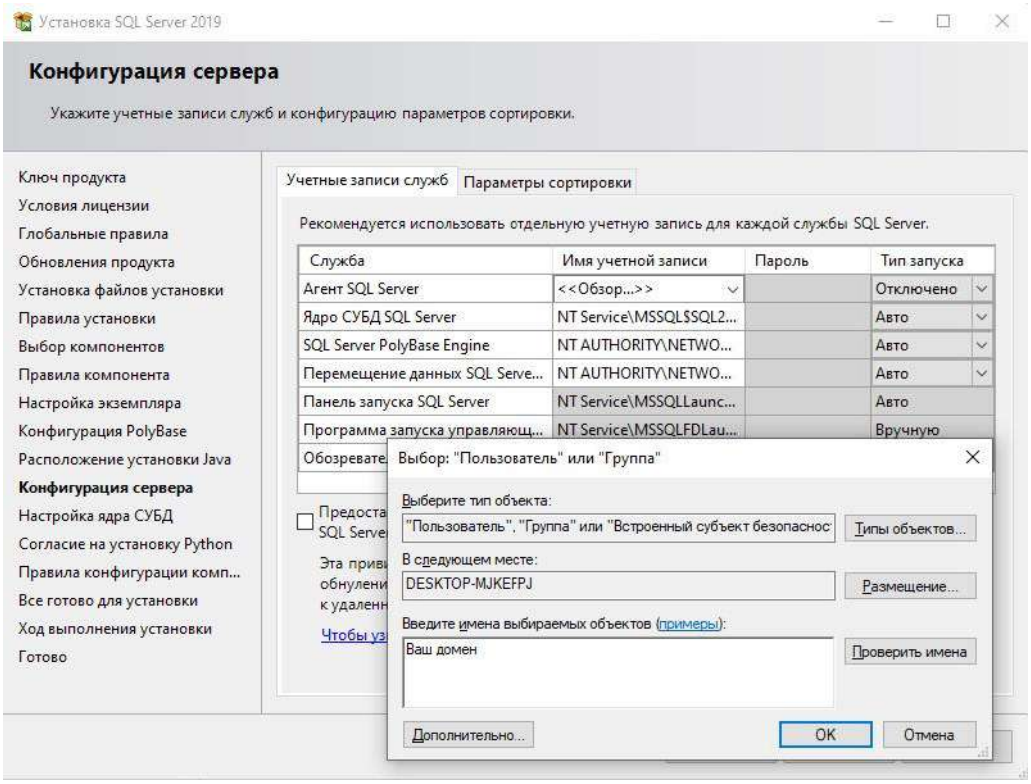
7) Для поддержки Java установим среду JRE. Но если Вам требуется еще JDK или JRE другой версии, то Вы можете самостоятельно скачать их и выбрать установщики в меню "*Расположение установки ...*"



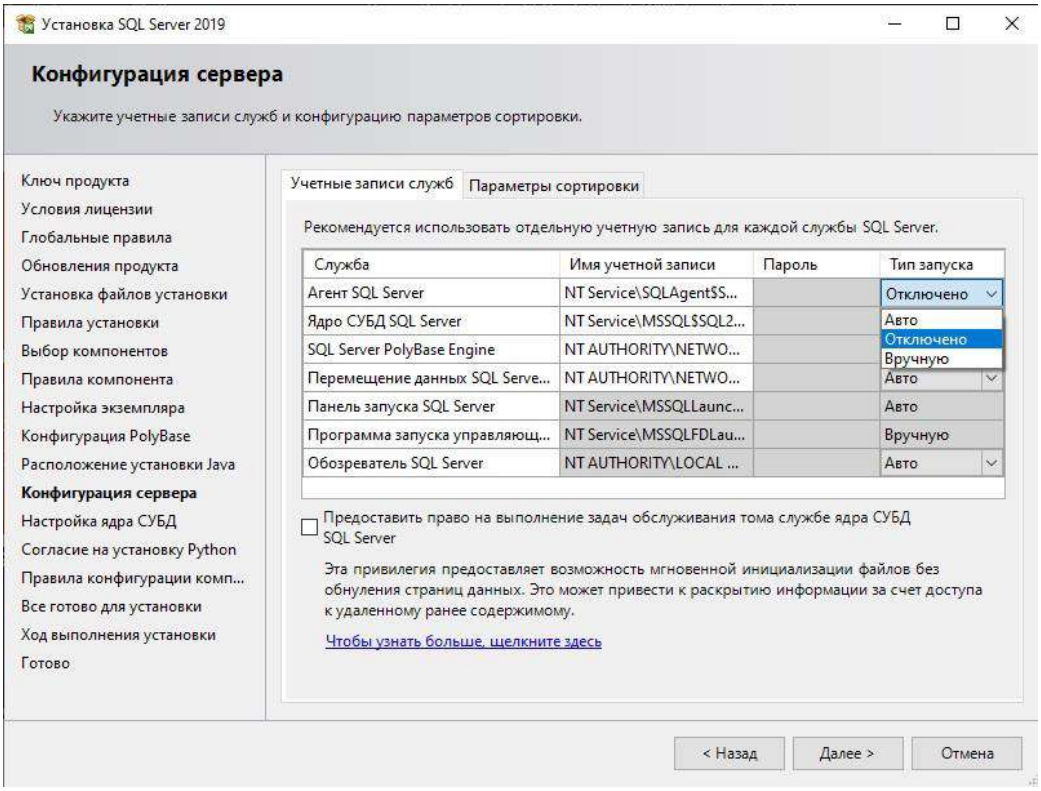
8) Далее, переходим к настройкам конфигурации сервера, обратим внимание на раздел "Имя учетной записи" - это те учетные записи из под которых будут стартовать службы SQL Server Agent, SQL Server Database, SQL Server Browser. По умолчанию используется Ваша локальная учетная запись.



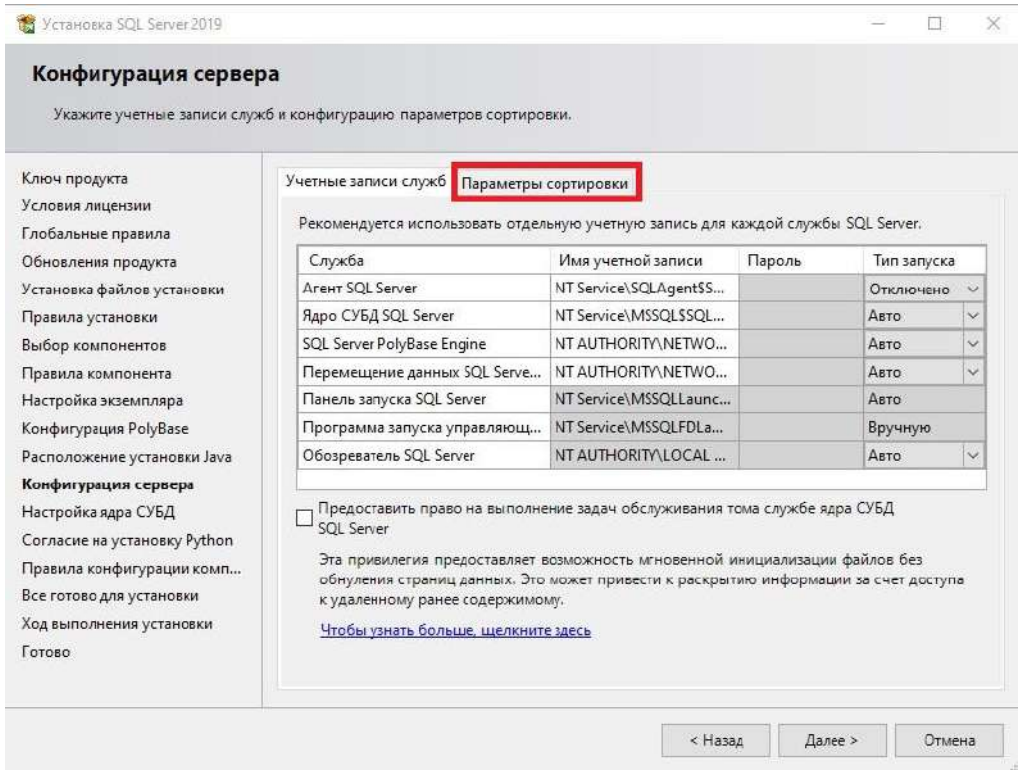
Но Вы в настройках можете выбрать учетную запись Вашего домена, если Ваш сервер входит в домен.

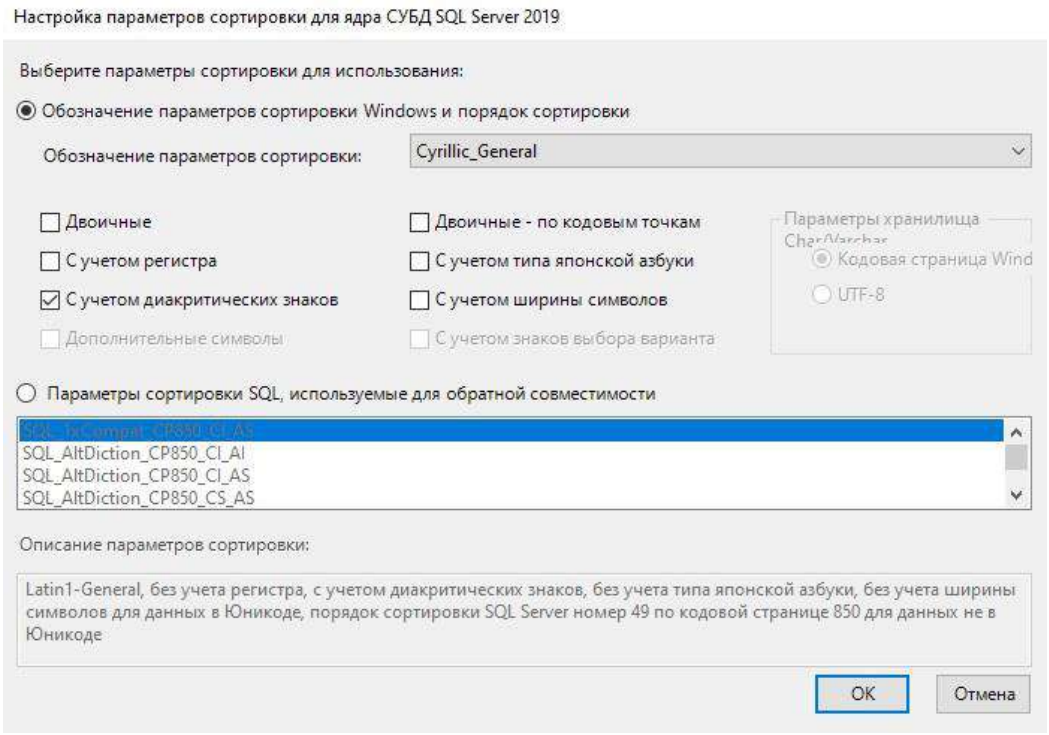


Так же, Вы можете задать тип запуска какой-либо службы. Поставить ее на автозапуск, в ручную, или вообще отключить, если Вам данная служба не нужна под Ваши задачи.



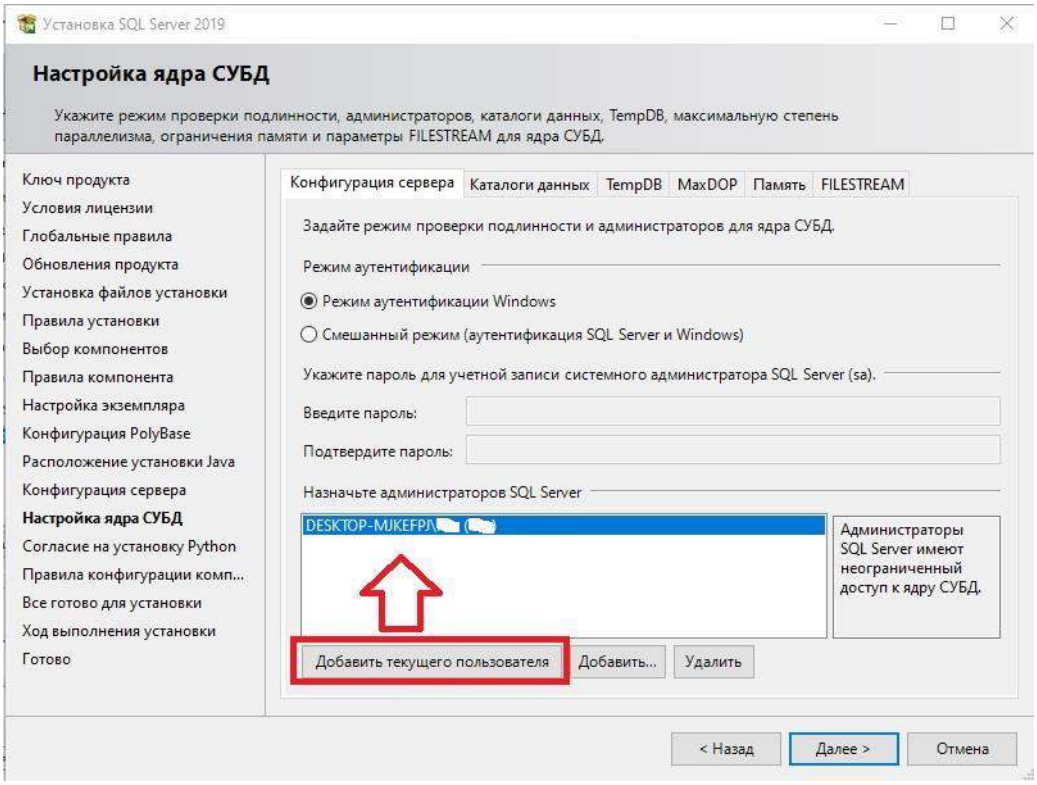
Так же можем зайти в меню "Параметры сортировки" - Это настройки таблицы кодировок. А так же, выполнять сортировку, как учитывать верхний и нижний регистр, как реагировать на символы, и т.п.



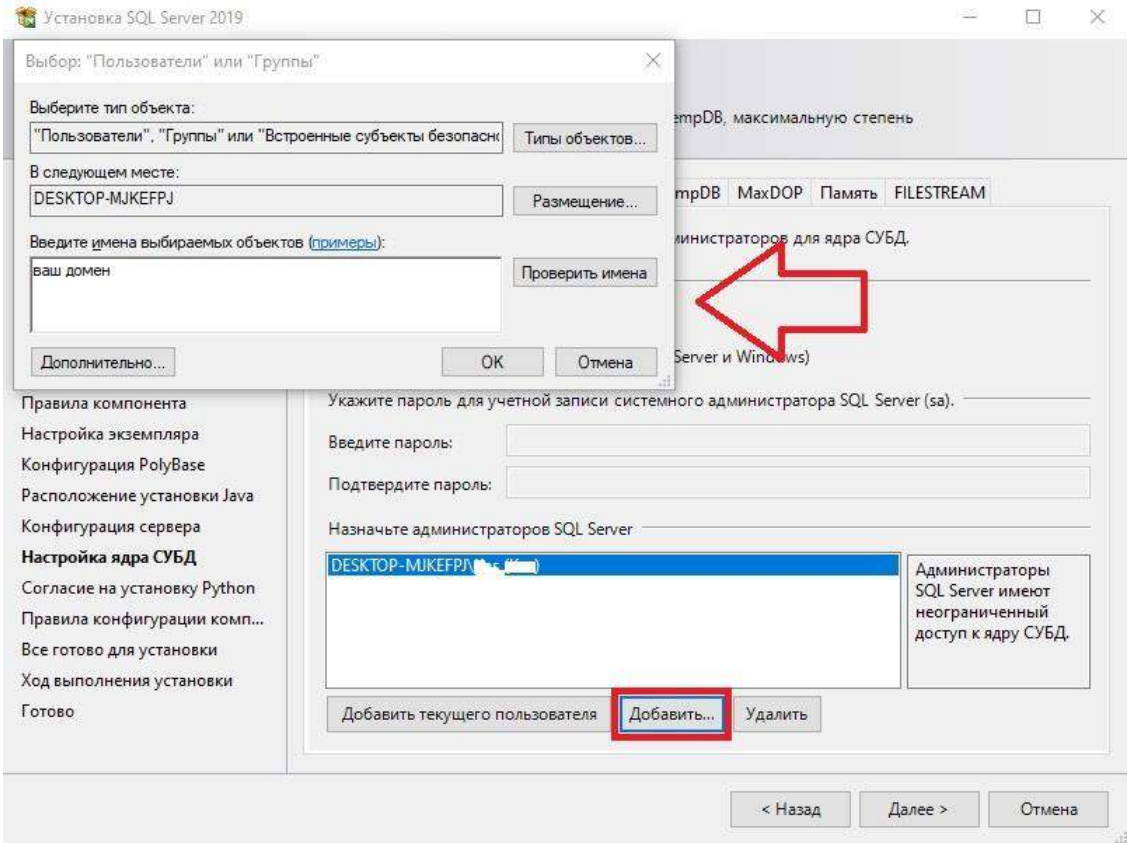


9) Далее переходим к настройкам учетных записей для подключения к SQL Server 2019. Нам предлагают выбор режим входа под учетными записями Windows, либо смешанный режим, т.е возможность входа под учетной записью Windows и под учетной записью SQL Server, если выбрать смешанную, то Вам предложится создать учетную запись SQL Server.

На примере мы выберем "Режим аутентификации Windows => нажимаем на кнопку "Добавить текущего пользователя" и добавляем его.

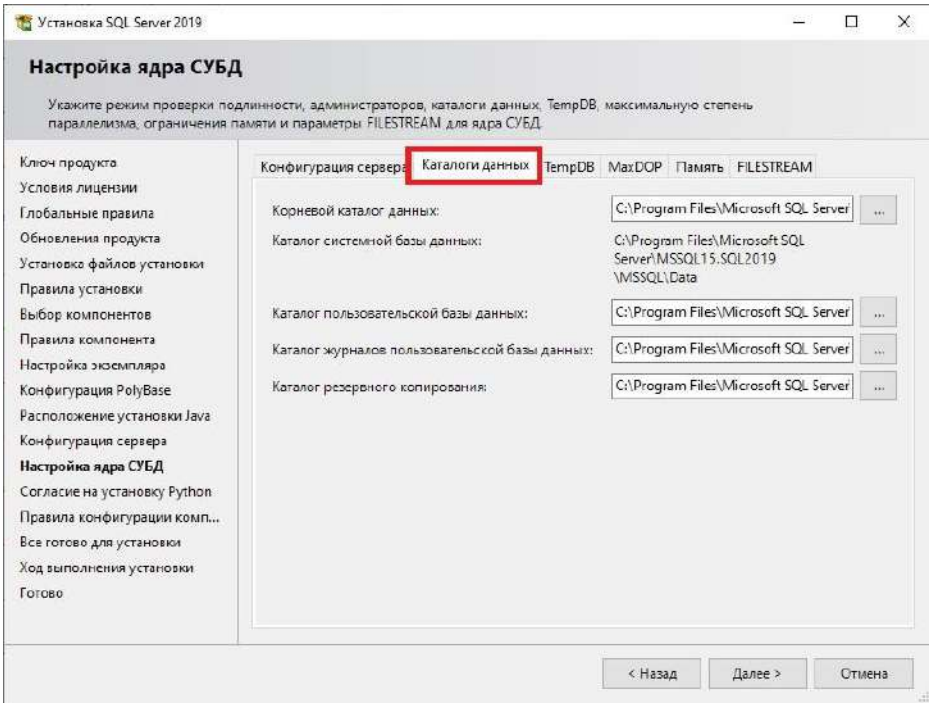


Так же, если Ваш сервер входит в домен, то можете добавить авторизацию с домена.



Далее можем перейти в вкладку "Каталоги данных". В этом пункте вы можете выбрать места расположения для корневого каталога, каталога системной базы данных,

пользовательской БД, каталог журналов пользовательской БД и каталог для бекапа. Рекомендуется все месторасположения указывать на разных жестких дисках / разделах.



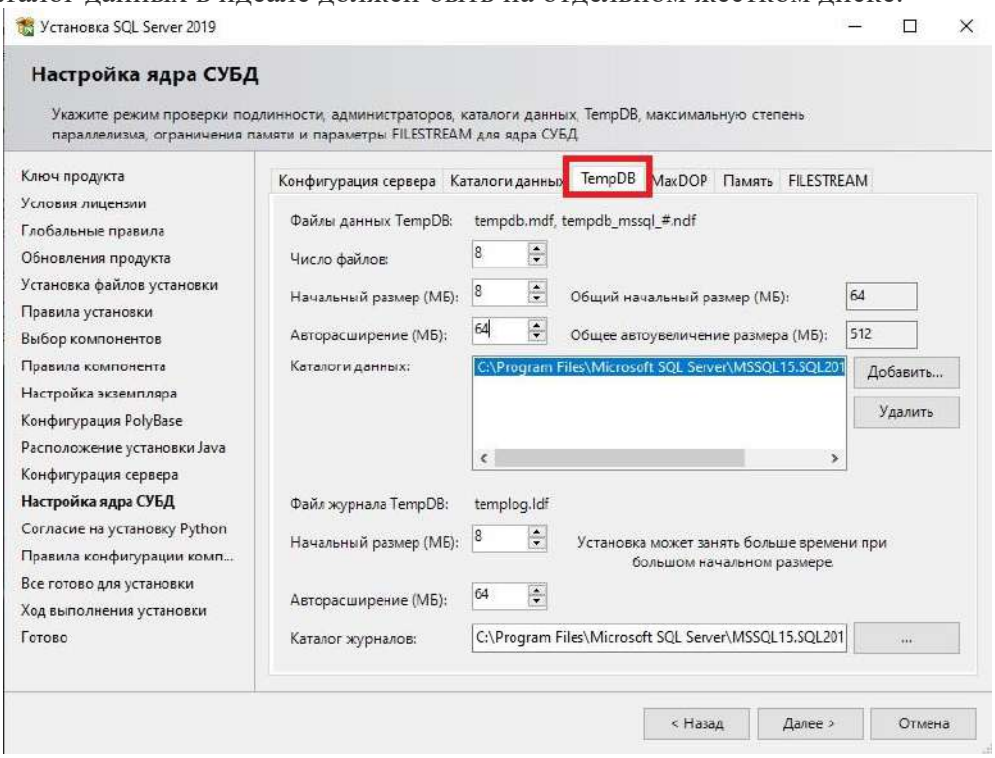
В следующей вкладке "TempDB".

Число файлов - прописано по умолчанию в зависимости от количества ядер.

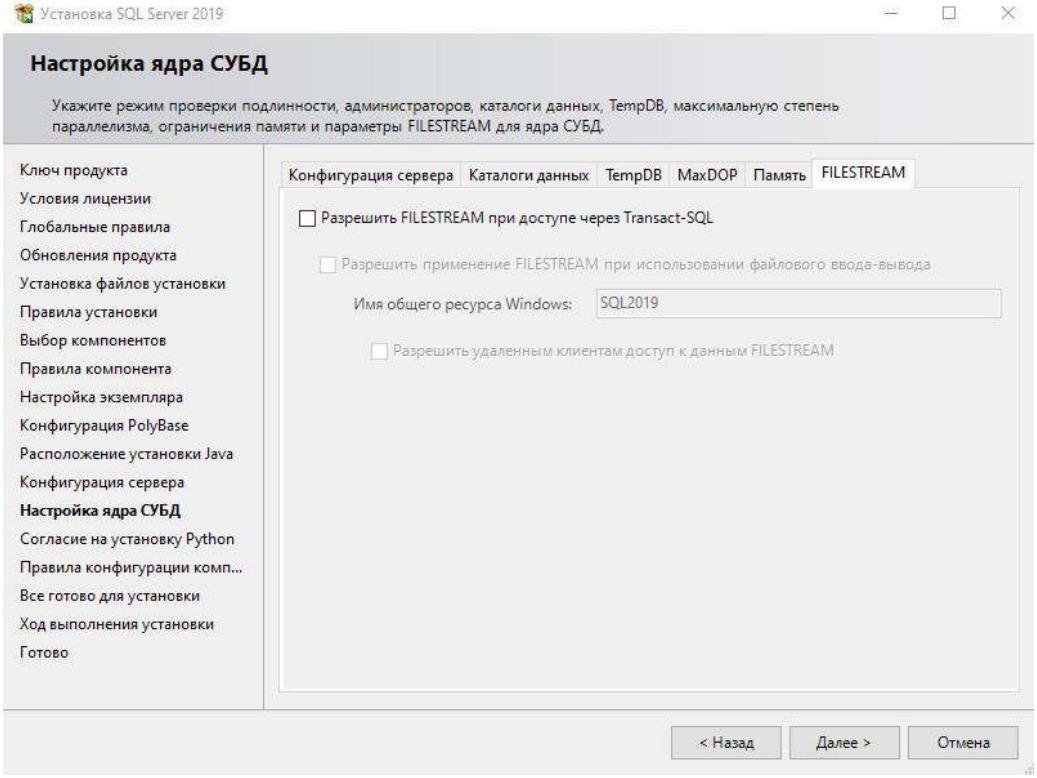
Начальный размер - рекомендуется прибавить размер хотя бы до 16, или 32 МБ.

Авторасширение - Это количество МБ, которое будет приращиваться, если файл будет заполнен на 100%. Количество указывают обычно в зависимости от задач и использования SQL Server, по умолчанию 64 МБ, но рекомендуется добвлять от 1гб, чтобы не было нагрузки на производительность SQL Server

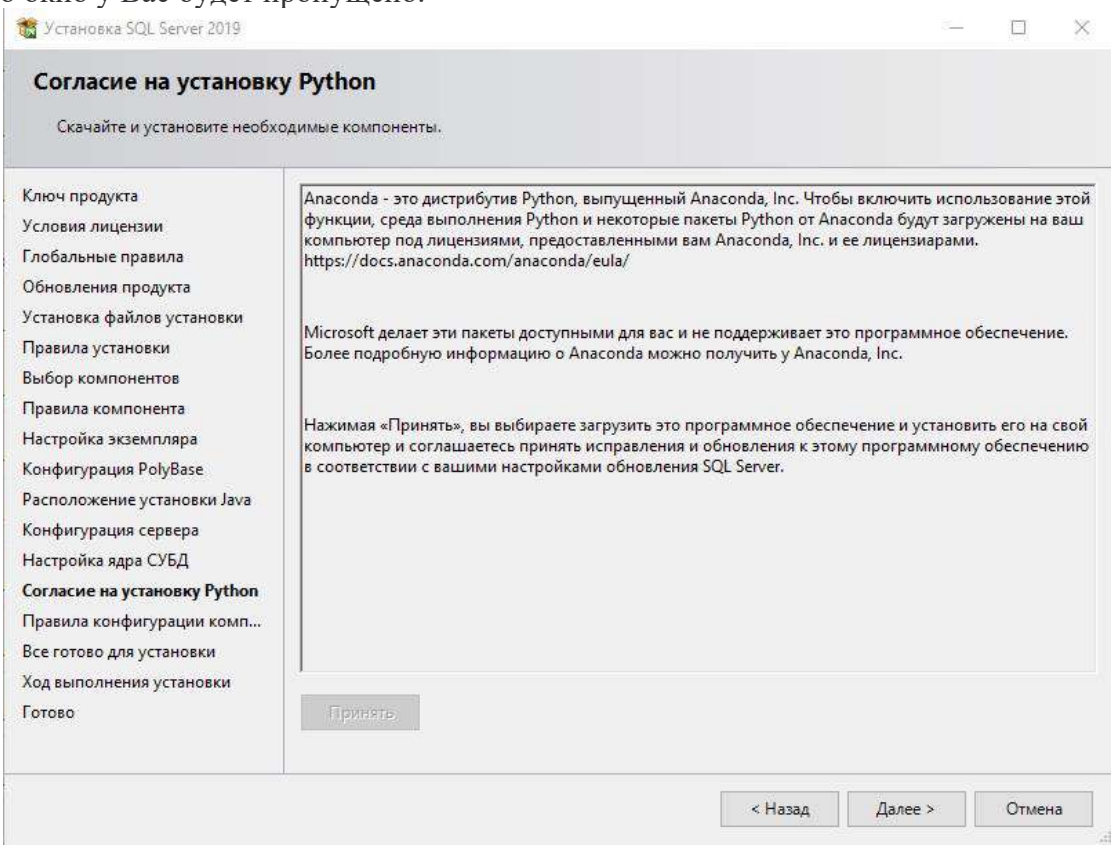
Каталог данных в идеале должен быть на отдельном жестком диске.



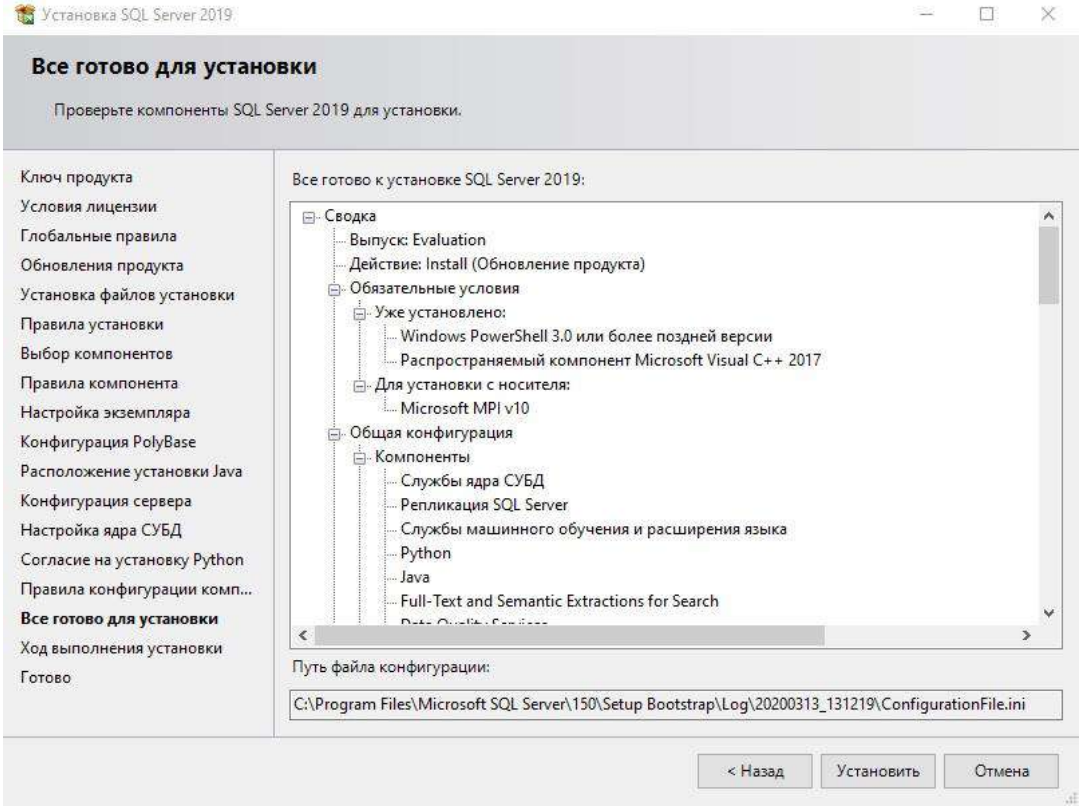
Следующая вкладка это настройки Filestream — это настройка, которая позволяет хранить файлы в файловой системе NTFS, она может быть как активирована, так и нет, опционально.



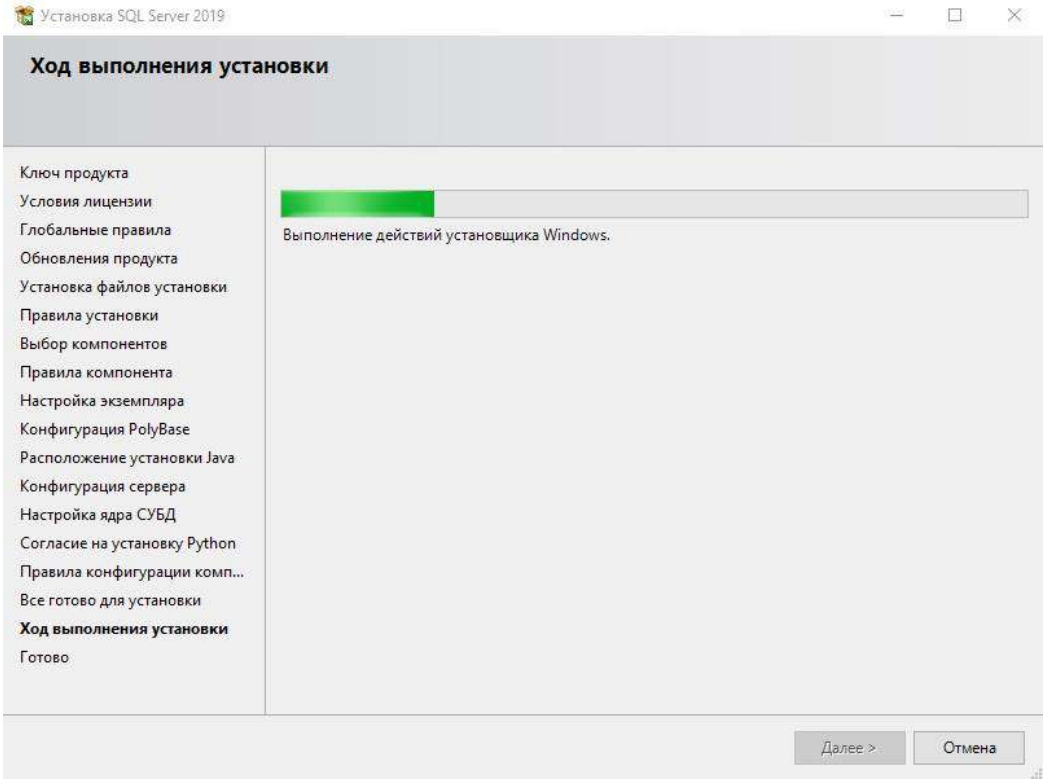
10) Если вы в начале установке в меню "Компоненты" выбирали установку Pyton, то принимаем загрузку ПО следующим шагом. Если Вы не выбирали данный компонент, то это окно у Вас будет пропущено.



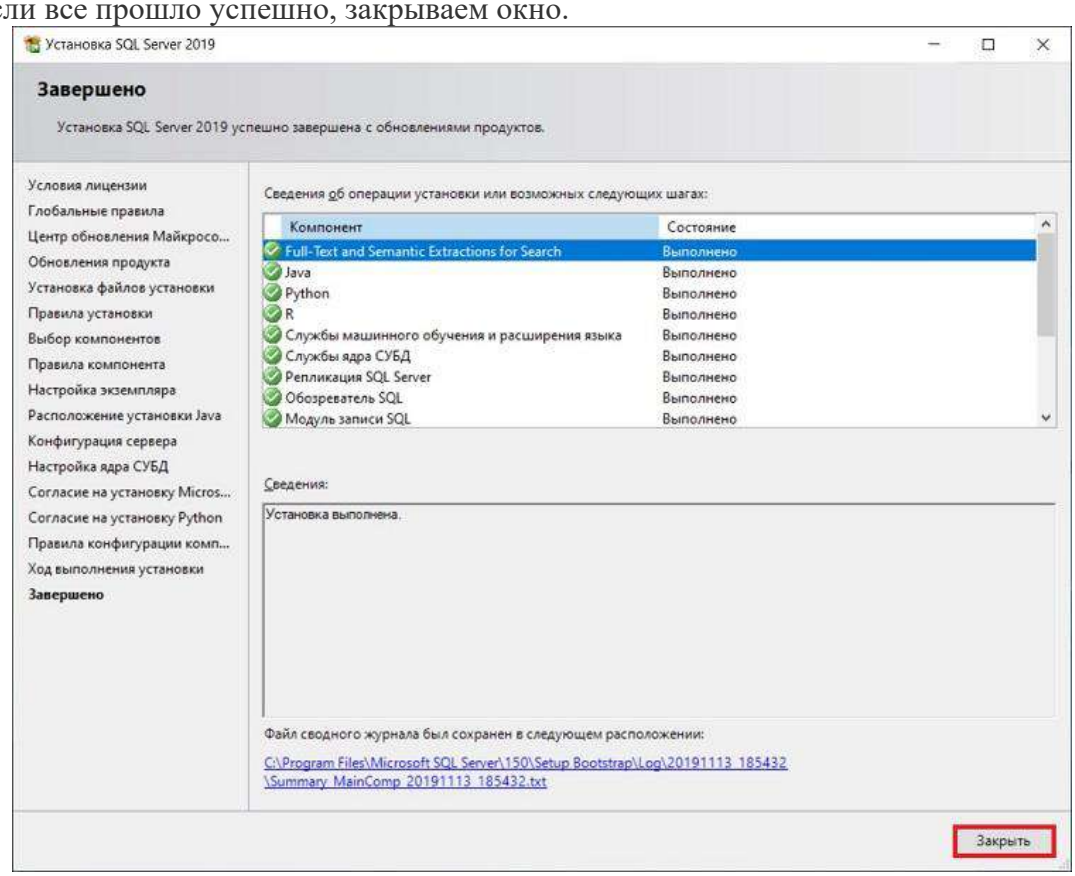
11) Теперь в следующем окне Вы можете наблюдать все установки и настройки, которые вы ранее задавали. Нажимаем "Установить".



12) Ожидаем завершения хода установки.

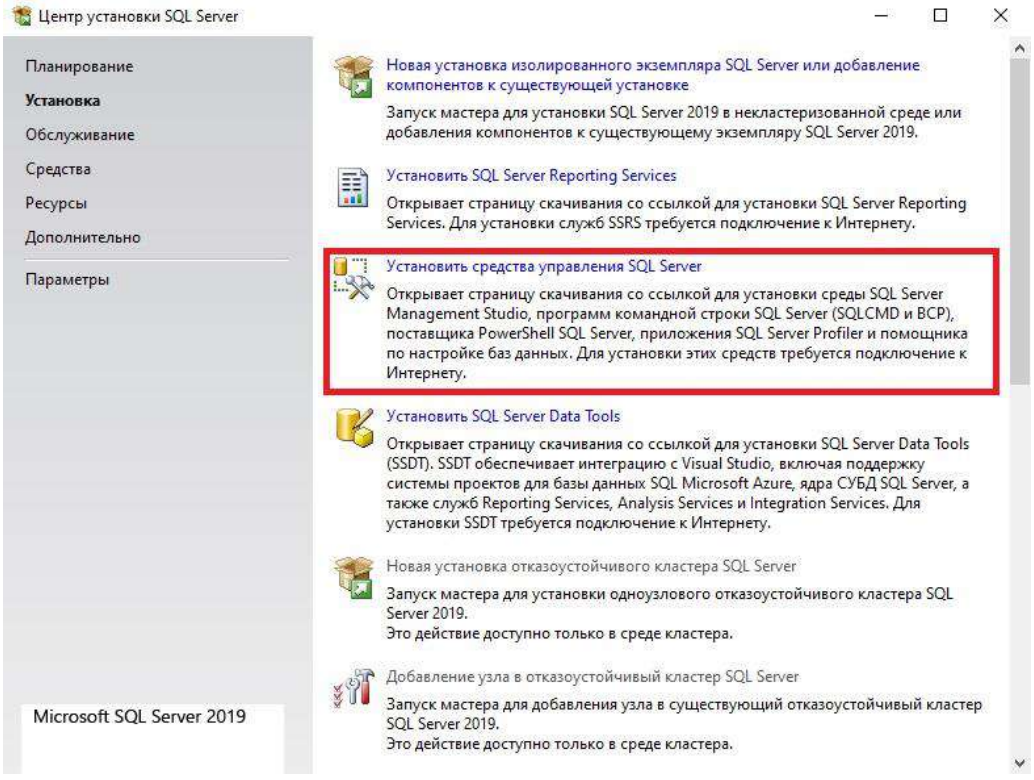


13) Если все прошло успешно, закрываем окно.



14) После того, как установка SQL Server 2019 завершена, нам нужно установить приложение, с помощью которого мы будем подключаться к серверу баз данных. Это приложение SQL Server Management Studio (SSMS).

Заходим снова в центр установки SQL Server и наживаем "Установить средства управления SQL Server".

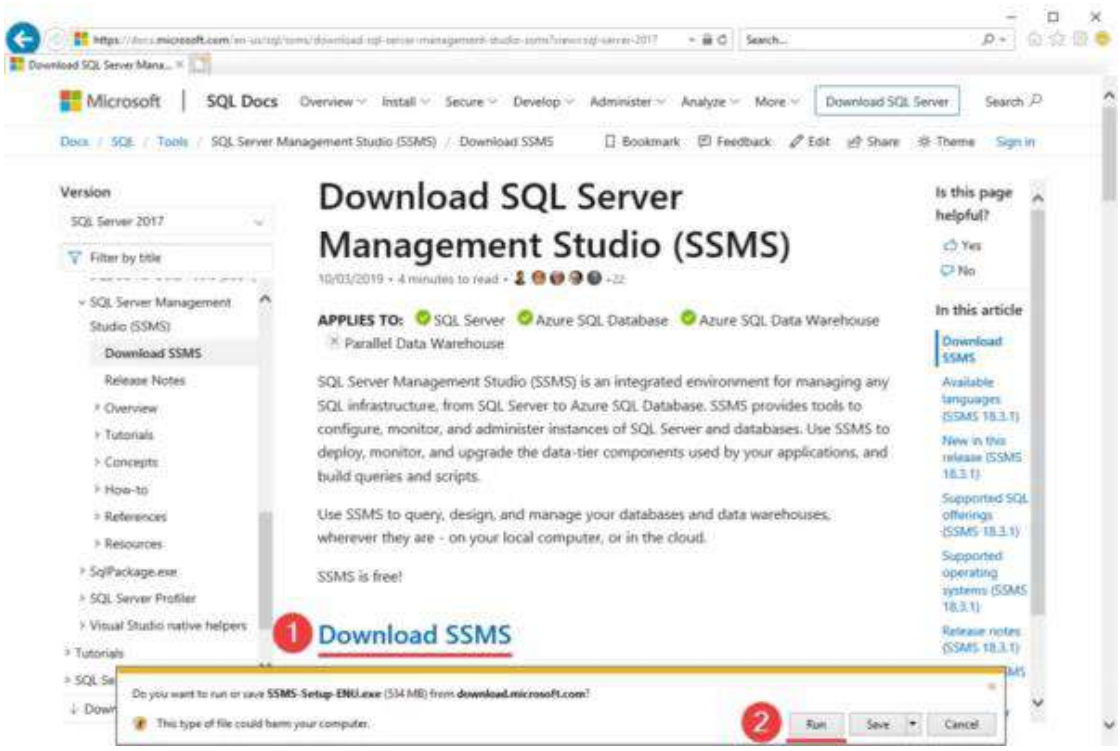


15) При нажатии у нас откроется сайт и нам нужно будет скачать и установить SSMS.

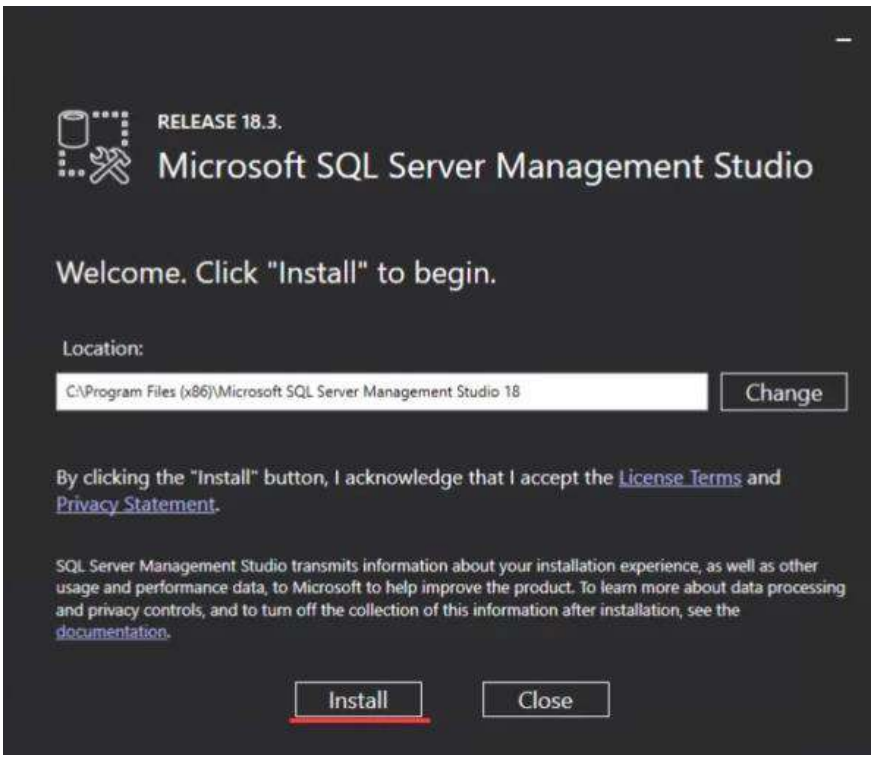
2. Установка и настройка SQL Server Management Studio

SQL Server Management Studio (SSMS) — это программа с графическим интерфейсом, которая позволяет быстро и легко управлять базами данных, создавать пользователей, устанавливать разрешения для баз данных, создавать резервные копии и многое другое. Данная статья поможет установить программу и выполнить простой запрос в базу данных для проверки корректности установки.

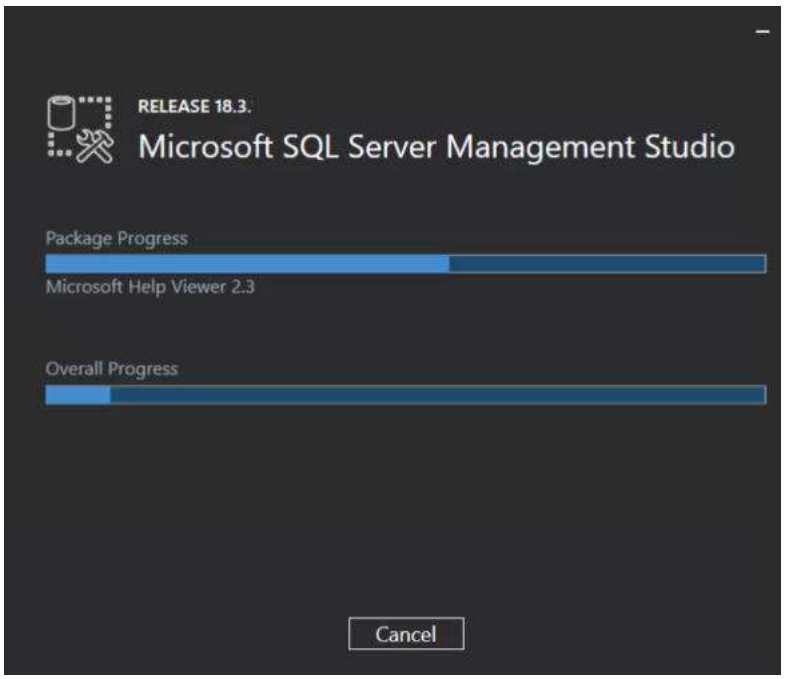
SQL Server Management Studio доступен отдельно от Microsoft SQL Server. Для его установки посетите страницу загрузки Microsoft и скачайте последний бинарный файл мастера установки.



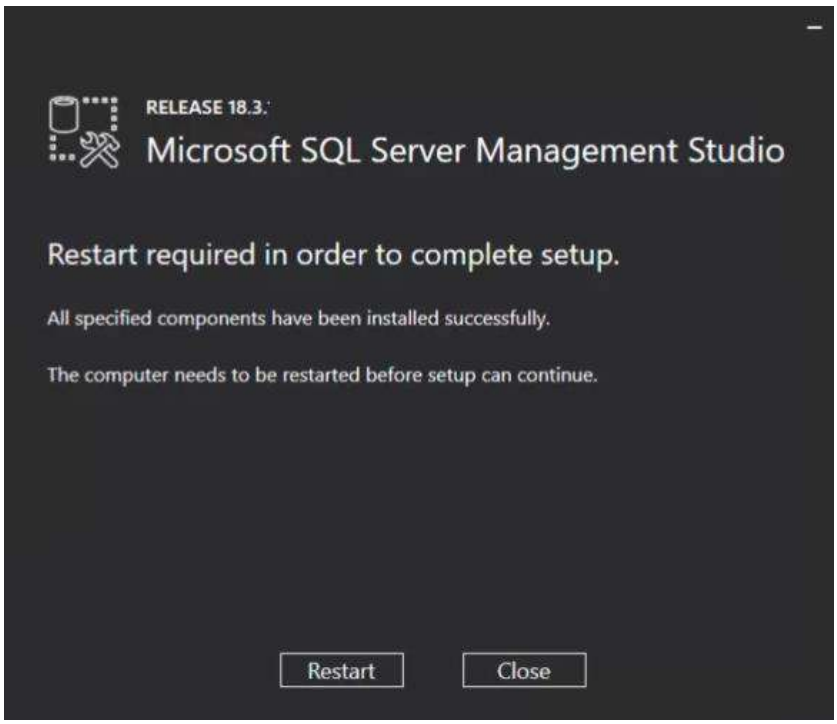
Следующим шагом выбираем место установки студии управления и нажимаем **Install:**



После нажатия на кнопку Install процедура установки начнется автоматически:



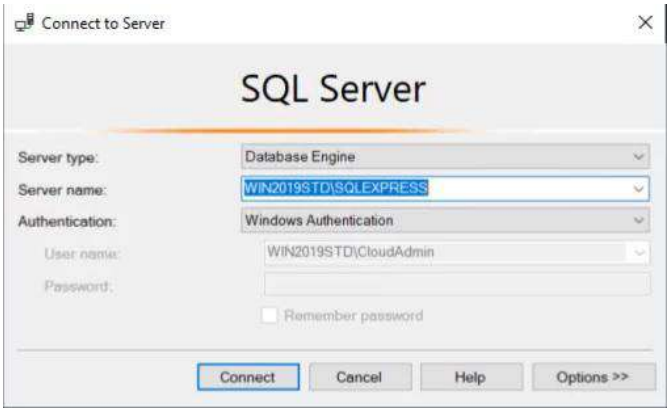
По завершении установки появится экран, предлагающий выполнить перезагрузку. Перезагружаемся, нажав на **Restart**:



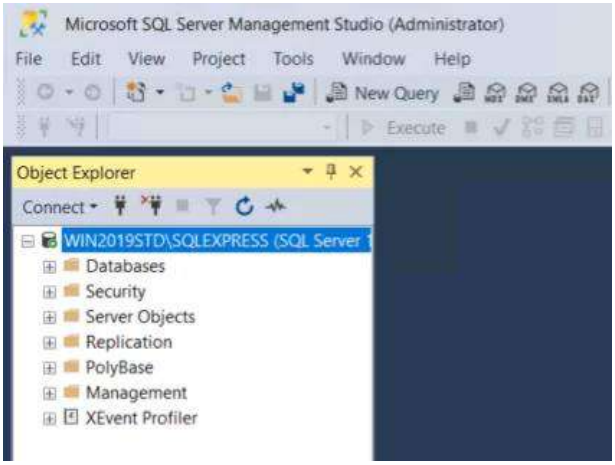
Программа SQL Server Management Studio готова к запуску. Чтобы подключиться к SQL Server с помощью Microsoft SQL Server Management Studio, выполните следующие действия:

1. Запустите Microsoft SQL Server Management Studio из меню «Пуск»:

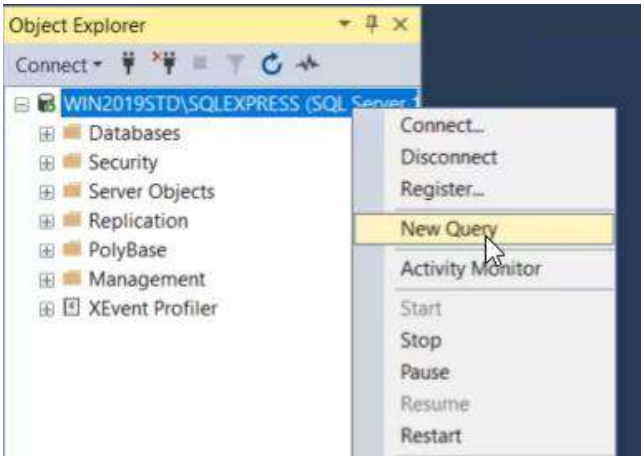
2. Откроется окно аутентификации. Вы можете выбрать опцию, на основе которой настроили аутентификацию при установке MS SQL сервера. Далее нажмите **Connect**:



3. Если соединение установлено успешно, вы увидите панель Object Explorer — обозревателя объектов:



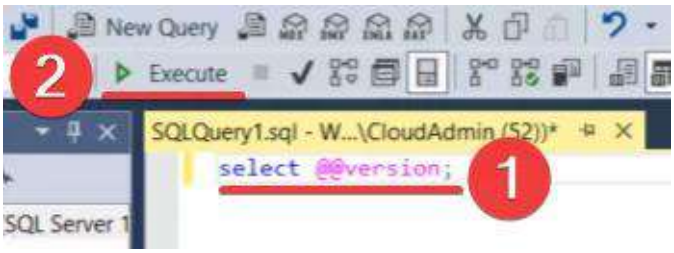
4. Чтобы выполнить запрос в базу данных, кликните правой кнопкой мыши по узлу сервера (в нашем примере это **WIN2019STDSQLEXPRESS**), и выберите пункт меню **New Query**:



5. Введите в редакторе следующий запрос:

```
select @@version;
```

Этот запрос возвращает версию SQL Server. После чего нажмите **Execute** (Выполнить):



6. В окне «**Results**» отобразится версия SQL Server, как показано на снимке экрана ниже. Быстрый способ выполнить запрос — нажать клавишу **F5**:

Теперь можно подключаться к SQL серверу и выполнять запросы из студии управления сервером базы данных.

3. Использование диспетчера конфигурации SQL Server

Включение или отключение протокола клиента

1. В диспетчере конфигурации SQL Server разверните узел **Конфигурация SQL Server Native Client**, щелкните правой кнопкой мыши элемент **Клиентские протоколы** и выберите пункт **Свойства**.
2. Выберите протокол в окне **Отключенные протоколы** и нажмите **Включить,** чтобы разрешить его работу.
3. Выберите протокол в окне **Включенные протоколы** и нажмите кнопку **Отключить**, чтобы запретить его работу.

Изменение протокола по умолчанию или порядка задействования протоколов для компьютера клиента

1. В диспетчере конфигурации SQL Server разверните узел **Конфигурация SQL Server Native Client**, щелкните правой кнопкой мыши элемент **Клиентские протоколы** и выберите пункт **Свойства**.
2. В окне **Включенные протоколы** нажмите кнопку **Вверх** или **Вниз** для изменения порядка, в котором задействуются протоколы при попытке соединения с SQL Server. Верхний протокол в окне **Разрешенные протоколы** является протоколом по умолчанию.

Важно!

Диспетчер конфигурации SQL Server создает параметры реестра для конфигураций псевдонимов сервера и клиентскую сетевую библиотеку по умолчанию. Однако приложение не устанавливает какие-либо клиентские сетевые библиотеки SQL Server или сетевые протоколы. Клиентские сетевые библиотеки SQL Server устанавливаются во время установки SQL Server; сетевые протоколы — во время установки Microsoft Windows (или через элемент **Сеть** на **панели управления**). Конкретный сетевой протокол может быть недоступен при установке Windows. Дополнительные сведения об установке этих сетевых протоколов см. в документации поставщика.

Настройка клиента для использования TCP/IP

1. В диспетчере конфигурации SQL Server разверните узел **Конфигурация SQL Server Native Client**, щелкните правой кнопкой мыши элемент **Клиентские протоколы** и выберите пункт **Свойства**.
2. В поле **Включенные протоколы** используйте кнопки со стрелками вниз и вверх, чтобы изменить порядок применения протоколов при попытках установить соединение с SQL Server. Верхний протокол в окне **Разрешенные протоколы** является протоколом по умолчанию.

Протокол общедоступной памяти активируется отдельно, установкой флажка **Включенный протокол общей памяти**.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Установите и настройте SQL сервер.

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Укажите назначение SQl Server.
2. Укажите назначение SQL Server Management Studio.
3. Опишите процесс установки и настройки SQl Server.
4. Опишите процесс установки и настройки SQL Server Management Studio.

Лабораторная работа №4
«ЭКСПОРТ ДАННЫХ БАЗЫ В ДОКУМЕНТЫ ПОЛЬЗОВАТЕЛЯ»

Цель работы: получить теоретические знания и практические навыки экспорта данных базы в документы пользователя

Теоретические сведения

Рассмотрим, как подключаться к источникам данных **MySQL** со страницы **Выбор источника данных** или **Выбор назначения** в мастере импорта и экспорта SQL Server. Для подключения к MySQL можно использовать ряд поставщиков данных.

Подключение к MySQL с помощью поставщика данных платформы .NET Framework для MySQL

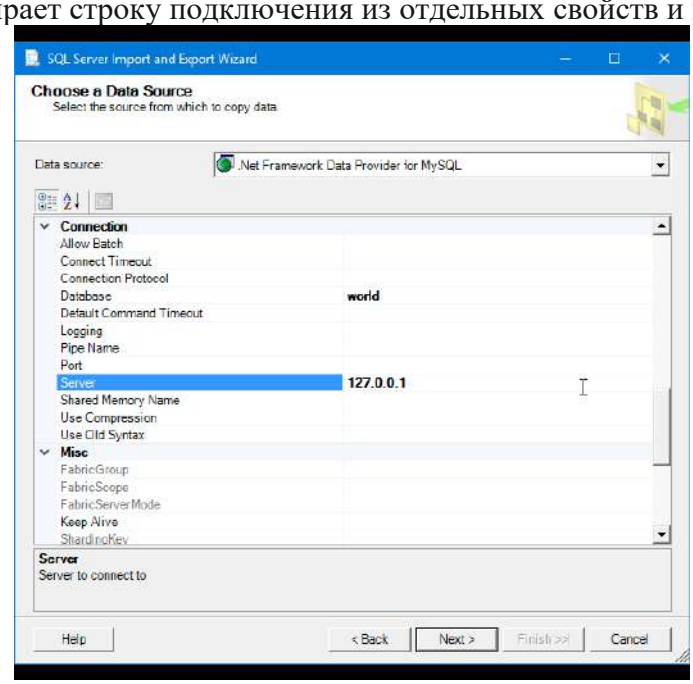
После выбора элемента **Поставщик данных .NET Framework для MySQL** на странице **Выбор источника данных** или **Выбор назначения** мастера появится сгруппированный список параметров для поставщика. Многие из них могут быть вам незнакомы или иметь непонятные имена. Однако вам достаточно указать лишь несколько параметров. Остальные параметры можно пропустить.

Примечание

Параметры подключения для этого поставщика данных одинаковы независимо от того, является ли MySQL источником или назначением. Таким образом, на страницах **Выбор источника данных** и **Выбор назначения** мастера отображаются одинаковые параметры.

Необходимые сведения	Поставщик данных .NET Framework для свойства MySQL
Имя сервера	Server
Имя базы данных	База данных
Сведения для проверки подлинности (имя входа)	Идентификатор пользователя и пароль

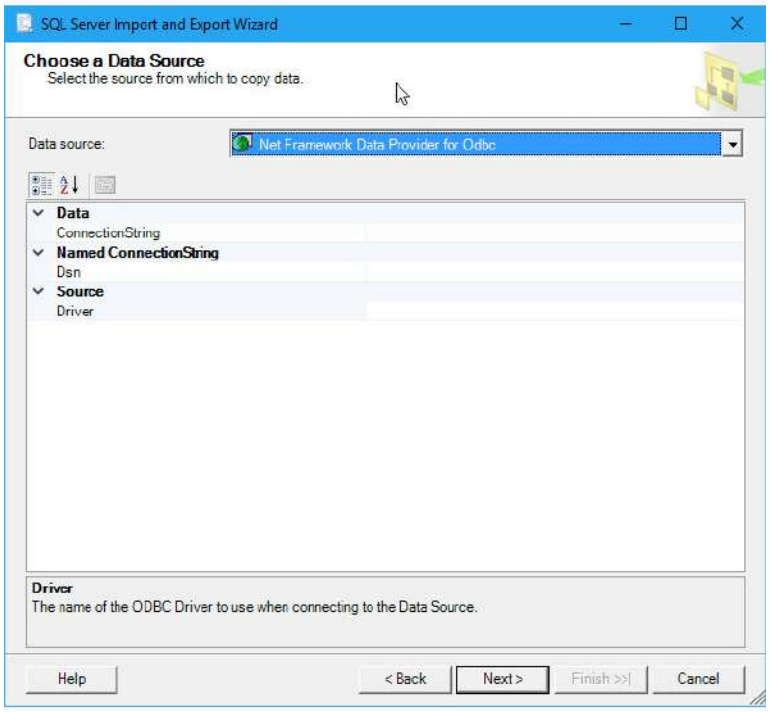
Вам не нужно вводить строку подключения в поле **ConnectionString** списка. После ввода отдельных значений для имени сервера MySQL (**сервера**) и информации для входа мастер собирает строку подключения из отдельных свойств и их значений.



Подключение к MySQL с помощью драйвера ODBC для MySQL

Драйверы ODBC не приводятся в раскрывающемся списке источников данных. Чтобы подключиться с помощью драйвера ODBC, сначала выберите **поставщик данных .NET Framework для ODBC** в качестве источника данных на странице **Выбор источника данных** или **Выбор назначения**. Этот поставщик служит оболочкой для драйвера ODBC.

Ниже показан экран, который появляется сразу после выбора поставщика данных .NET Framework для ODBC.



Указываемые параметры (драйвер ODBC для MySQL)

Примечание

Параметры подключения для этого поставщика данных и драйвера ODBC одинаковы независимо от того, является ли сервер MySQL источником или назначением. Таким образом, на страницах **Выбор источника данных** и **Выбор назначения** мастера отображаются одинаковые параметры.

Чтобы подключиться к MySQL с помощью драйвера ODBC для MySQL, соберите строку подключения, используя указанные ниже параметры и

их значения. Полный формат строки подключения приведен после списка параметров.

Совет

Вы можете получить помощь в построении строки подключения. Кроме того, вместо указания строки подключения вы можете предоставить существующее имя DSN (имя источника данных) или создать новое.

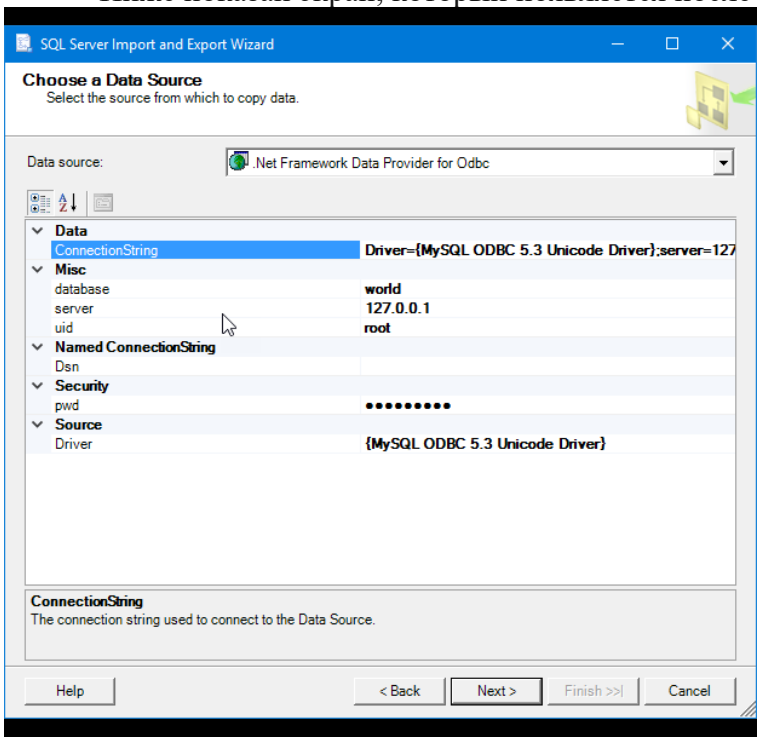
Драйвер
Имя драйвера ODBC.
Server
Имя сервера MySQL.
База данных
Имя базы данных MySQL.
UID и **PWD**
Идентификатор пользователя и пароль для подключения.

Формат строки подключения
Ниже приведен формат типичной строки подключения.
Driver={MySQL ODBC 5.3 Unicode Driver};Server=<server>;Database=<database>;UID=<user id>;PWD=<password>

Ввод строки подключения
Введите строку подключения в поле **ConnectionString** либо введите имя DSN в поле **Dsn** на странице **Выбор источника данных** или **Выбор назначения**. После того как вы введете строку подключения, мастер проанализирует ее и отобразит отдельные свойства и их значения в списке.

В приведенном ниже примере используется следующая строка подключения:
Driver={MySQL ODBC 5.3 Unicode Driver};Server=127.0.0.1;Database=world;UID=root;PWD=

Ниже показан экран, который появляется после ввода строки подключения.



Примеры экспорта-импорта

Запустите мастер импорта и экспорта SQL Server одним из описанных здесь способов, чтобы импортировать данные из любого поддерживаемого источника данных и экспортировать данные в него.

Варианты запуска мастера:

- Из меню "Пуск".
- Из командной строки.
- Из SQL Server Management Studio (SSMS).
- Из Visual Studio с SQL Server Data Tools (SSDT).

Если вы хотите запустить мастер, но на вашем компьютере не установлен Microsoft SQL Server, мастер импорта и экспорта SQL Server можно установить с помощью SQL Server Data Tools (SSDT).

Примечание

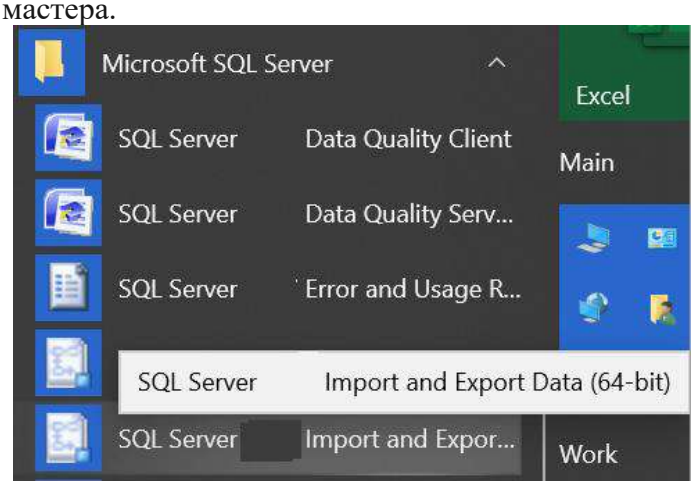
Чтобы использовать 64-разрядную версию мастера экспорта и импорта SQL Server, нужно установить SQL Server. SQL Server Data Tools (SSDT) и SQL Server Management Studio (SSMS) являются 32-разрядными приложениями и устанавливают только 32-разрядные файлы, включая 32-разрядную версию мастера.

Меню «Пуск»

Запуск мастера экспорта и импорта SQL Server из меню "Пуск"

1. В меню **Пуск** найдите и разверните узел **Microsoft SQL Server 20xx**.
2. Выберите один из следующих параметров:
   - **Импорт и экспорт данных в SQL Server 20xx (64-разрядная версия)**
   - **Импорт и экспорт данных в SQL Server 20xx (32-разрядная версия)**

Если источнику данных не требуется 32-разрядный поставщик данных, выбирайте 64-разрядную версию мастера.



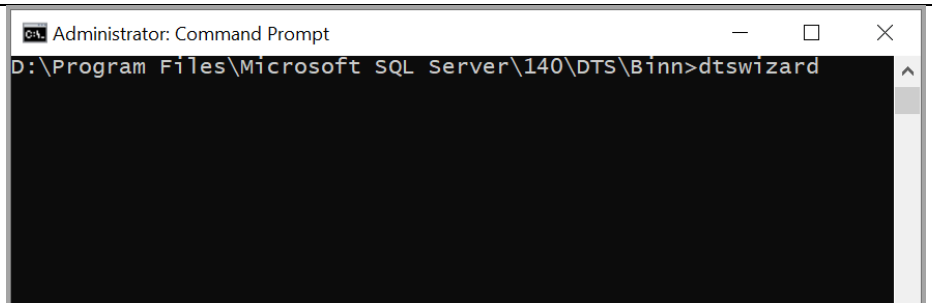
С помощью командной строки

Запуск мастера экспорта и импорта SQL Server из командной строки

В окне командной строки запустите **DTSWizard.exe** в одном из следующих каталогов:

- **C:\Program Files\Microsoft SQL Server\140\DTS\Binn** для 64-разрядной версии.
  - *140* = SQL Server 2017. Это значение зависит от вашей версии SQL Server.
- **C:\Program Files (x86)\Microsoft SQL Server\140\DTS\Binn** для 32-разрядной версии.
  - *140* = SQL Server 2017. Это значение зависит от вашей версии SQL Server.

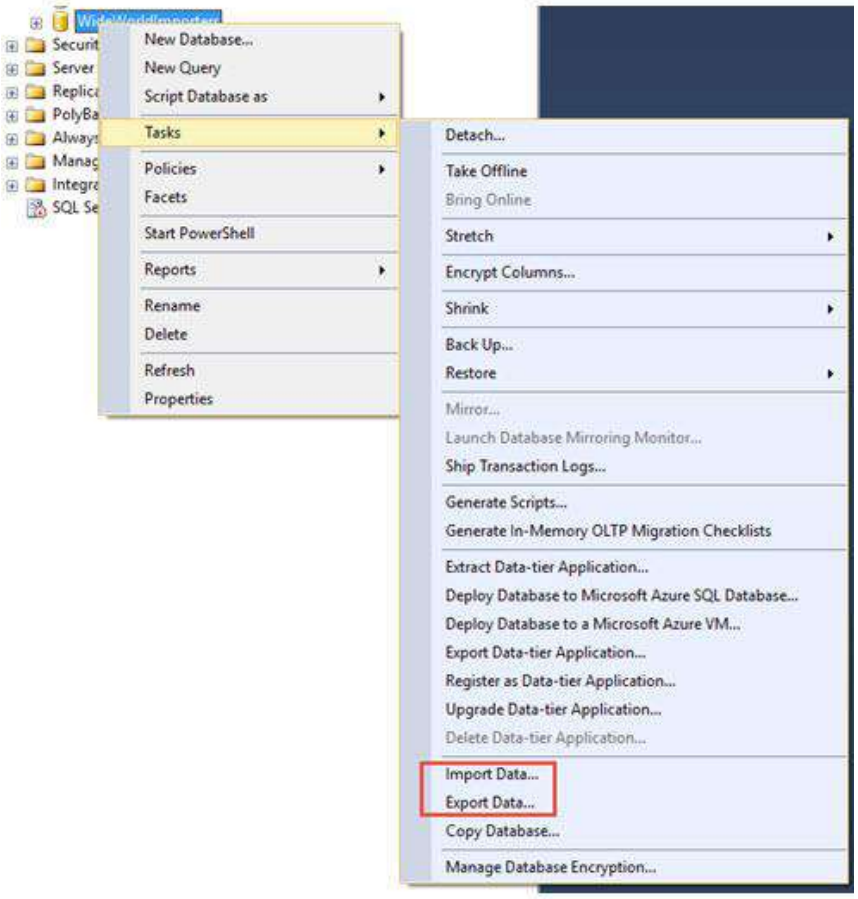
Если источнику данных не требуется 32-разрядный поставщик данных, выбирайте 64-разрядную версию мастера.



SQL Server Management Studio (SSMS)

Запуск мастера экспорта и импорта SQL Server из среды SQL Server Management Studio (SSMS)

1. В SQL Server Management Studio подключитесь к экземпляру SQL Server Компонент Database Engine.
2. Разверните узел **Базы данных**.
3. Щелкните базу данных правой кнопкой мыши.
4. Наведите указатель мыши на пункт **Задачи**.
5. Выберите один из следующих параметров:
   - **Импорт данных**
   - **Экспорт данных**

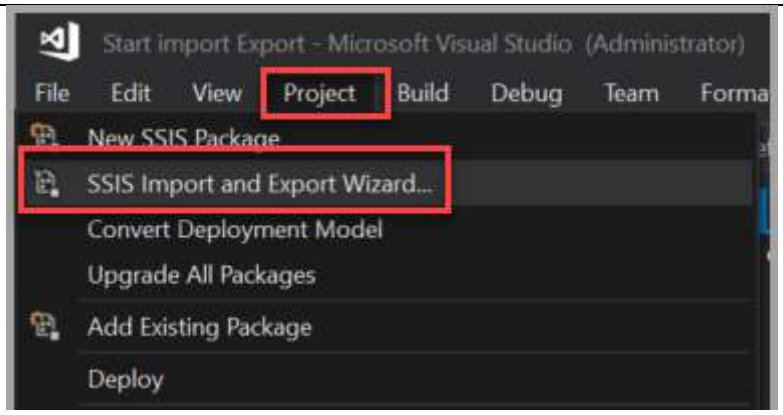


Visual Studio

Запуск мастера экспорта и импорта SQL Server из Visual Studio с SQL Server Data Tools (SSDT)

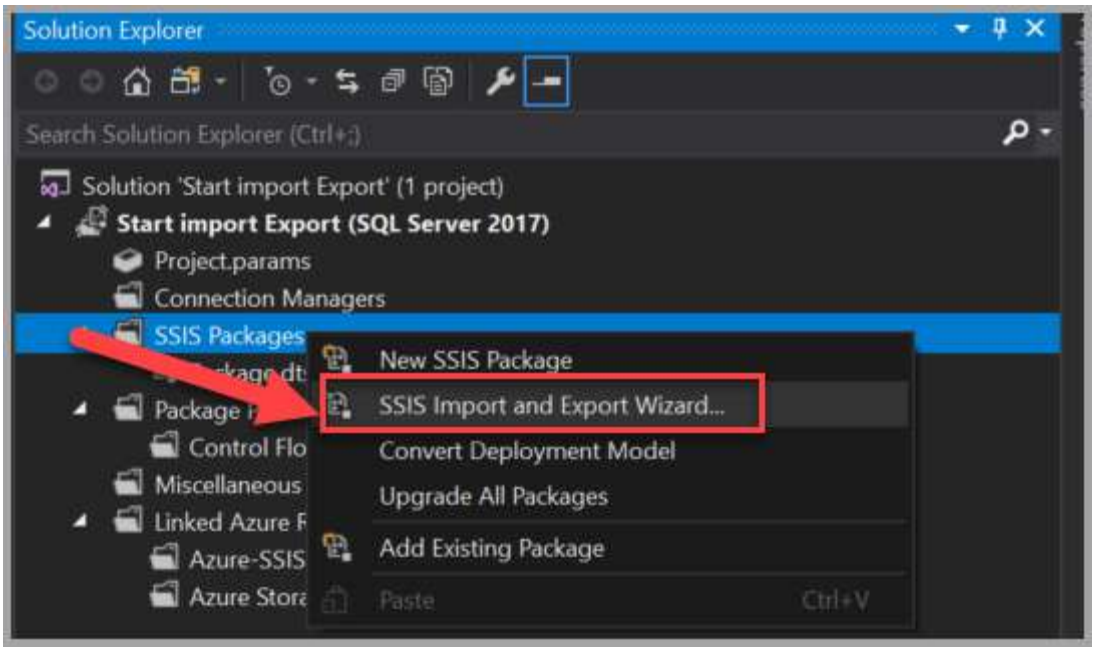
В Visual Studio с SQL Server Data Tools (SSDT)откройте проект Integration Services и выполните одно из описанных ниже действий.

- В меню **Проект** выберите пункт **Мастер импорта и экспорта служб SSIS**.



- или -

- В обозревателе решений щелкните правой кнопкой мыши папку **Пакеты служб SSIS** и выберите **Мастер импорта и экспорта служб SSIS**.



Получение мастера

Если вы хотите запустить мастер, но на вашем компьютере не установлен Microsoft SQL Server, мастер импорта и экспорта SQL Server можно установить с помощью SQL Server Data Tools (SSDT).

Получение справки во время работы мастера

Совет

Нажмите клавишу F1 при просмотре любой страницы или диалогового окна, чтобы открыть документацию по текущей странице мастера.

Дальнейшие действия

При запуске мастера открывается страница **Мастер импорта и экспорта SQL Server**. На этой странице никакие действия не требуются.

ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Осуществите экспорт данных базы в различные по формату документы пользователя.

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите процесс подключения к MySQL с помощью драйвера ODBC для MySQL
2. Какие Указываемые параметры (драйвер ODBC для MySQL) используются при настройке?
3. Опишите процесс подключения к MySQL с помощью поставщика данных платформы .NET Framework для MySQL**.**

Лабораторная работа №6
«ИМПОРТ ДАННЫХ ПОЛЬЗОВАТЕЛЯ В БАЗУ ДАННЫХ»

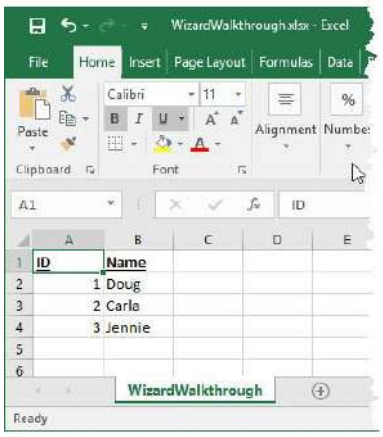
Цель работы: получить теоретические знания и практические навыки импорта данных пользователя в базу данных.

Теоретические сведения

Пример мастера импорта-экспорта

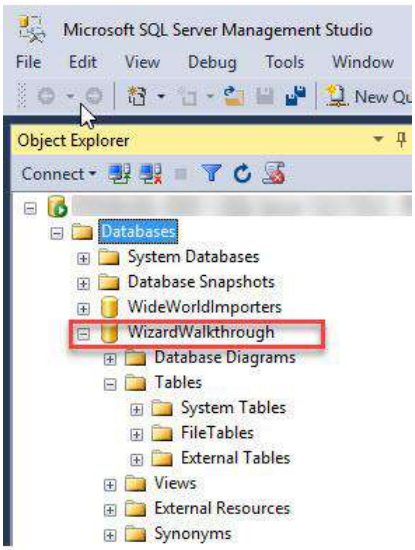
Исходные данные для этого примера Excel

Это исходные данные, которые вы собираетесь скопировать, — небольшая таблица из двух столбцов на листе WizardWalkthrough в книге WizardWalkthrough.xlsx Excel.



Целевая база данных для этого примера

Это целевая база данных SQL Server (или MySQL) (в SQL Server Management Studio), куда вы собираетесь скопировать исходные данные. Целевая таблица не существует, поэтому вы позволите мастеру создать ее для вас.



Шаг 1. Запуск мастера

Запустите мастер из группы "Microsoft SQL Server" в меню "Пуск" Windows.

Примечание

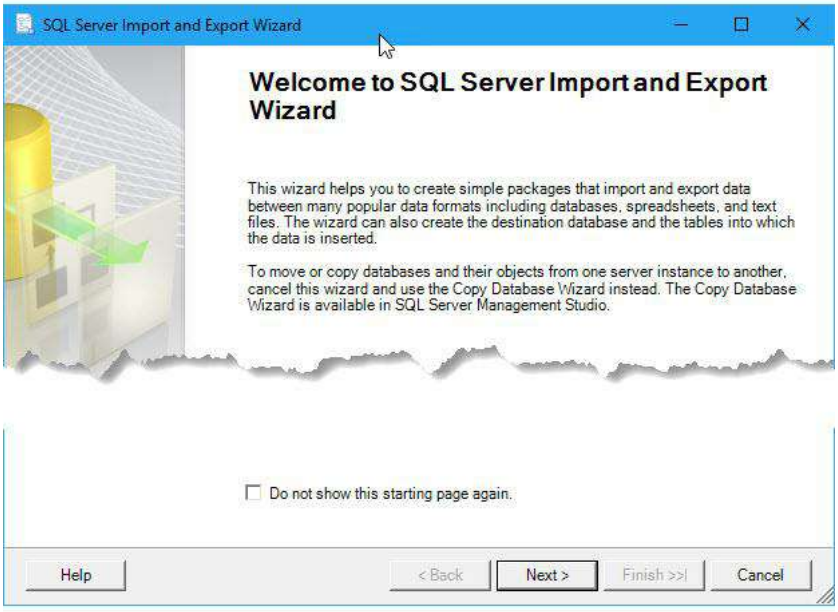
Для этого примера выберите 32-разрядный мастер, так как у вас установлена 32-разрядная версия Microsoft Office. Поэтому нужно использовать 32-разрядный поставщик данных для подключения к Excel. Для многих других источников данных в большинстве случаев можно выбрать 64-разрядный мастер.

Чтобы использовать 64-разрядную версию мастера экспорта и импорта SQL Server, нужно установить SQL Server. SQL Server Data Tools (SSDT) и SQL Server Management Studio (SSMS) являются 32-разрядными приложениями и устанавливают только 32-разрядные файлы, включая 32-разрядную версию мастера.

Шаг 2. Просмотр страницы приветствия

Первая страница мастера — это страница **приветствия**.

Если вы не хотите, чтобы она открывалась снова, установите флажок **Больше не показывать это окно**.

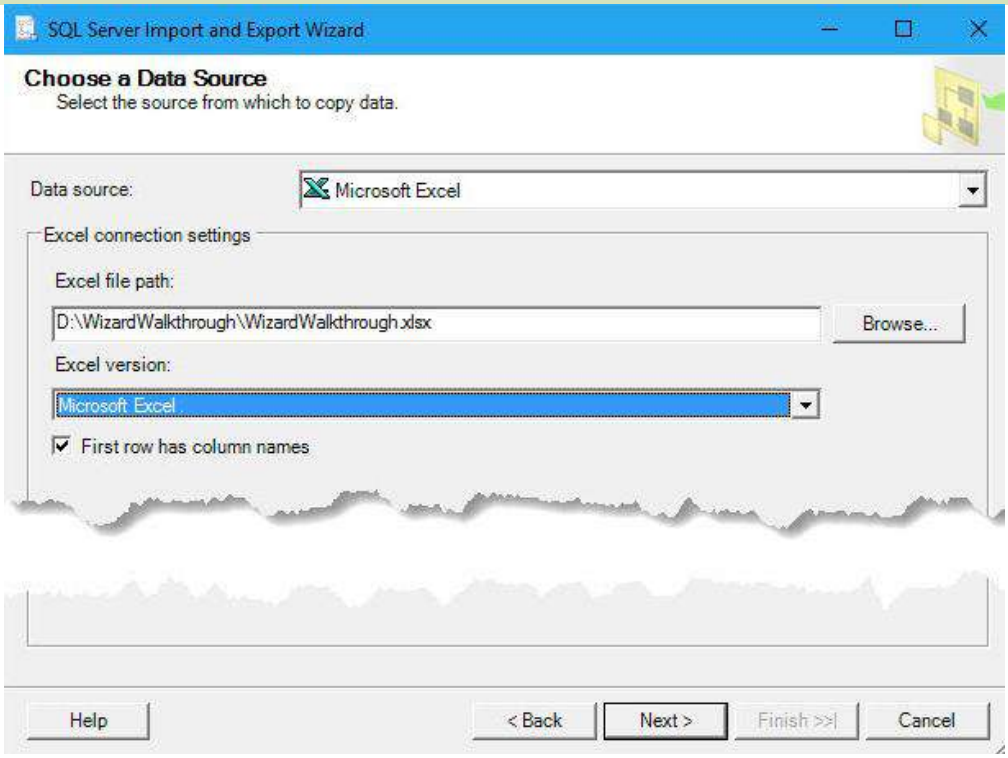


Шаг 3. Выбор Excel в качестве источника данных

На следующей странице **Выбор источника данных** выберите Microsoft Excel в качестве источника данных. Затем перейдите к файлу Excel и выберите его. Наконец, укажите версию Excel, которую вы использовали для создания файла.

Важно!

Дополнительные сведения о подключении к файлам Excel, а также об ограничениях и известных проблемах, связанных с загрузкой данных в файлы этого приложения и из них.



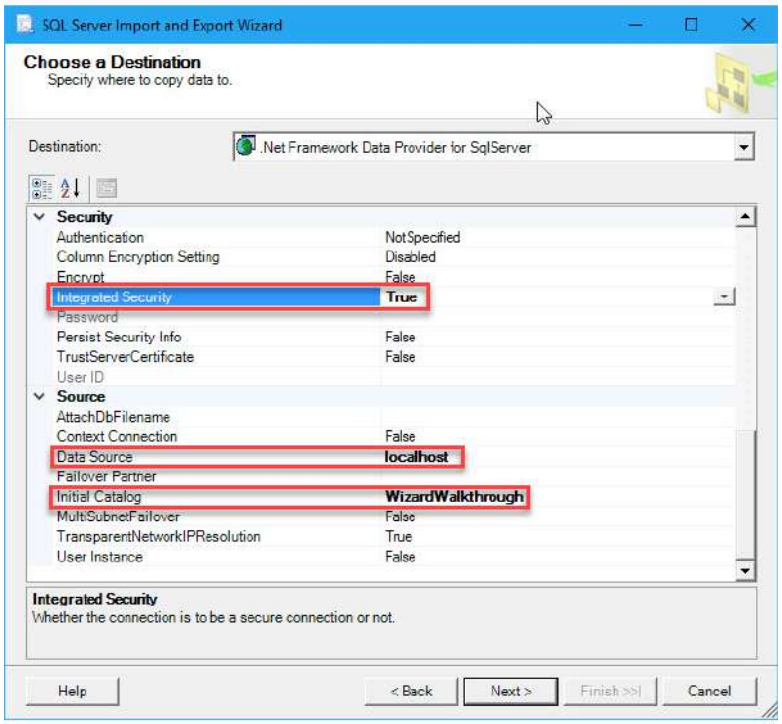
Шаг 4. Выбор SQL Server в качестве назначения

На следующей странице **Выбор назначения** выберите Microsoft SQL Server в качестве назначения, выбрав в списке одного из поставщиков данных, подключающегося к SQL Server. В этом примере выберите **Поставщик данных .Net Framework для SQL Server**.

На странице отображается список свойств поставщика. Многие из них могут быть вам незнакомы или иметь непонятные имена. К счастью, для подключения к любой корпоративной базе данных, как правило, достаточно указать лишь три параметра. Остальные параметры можно пропустить.

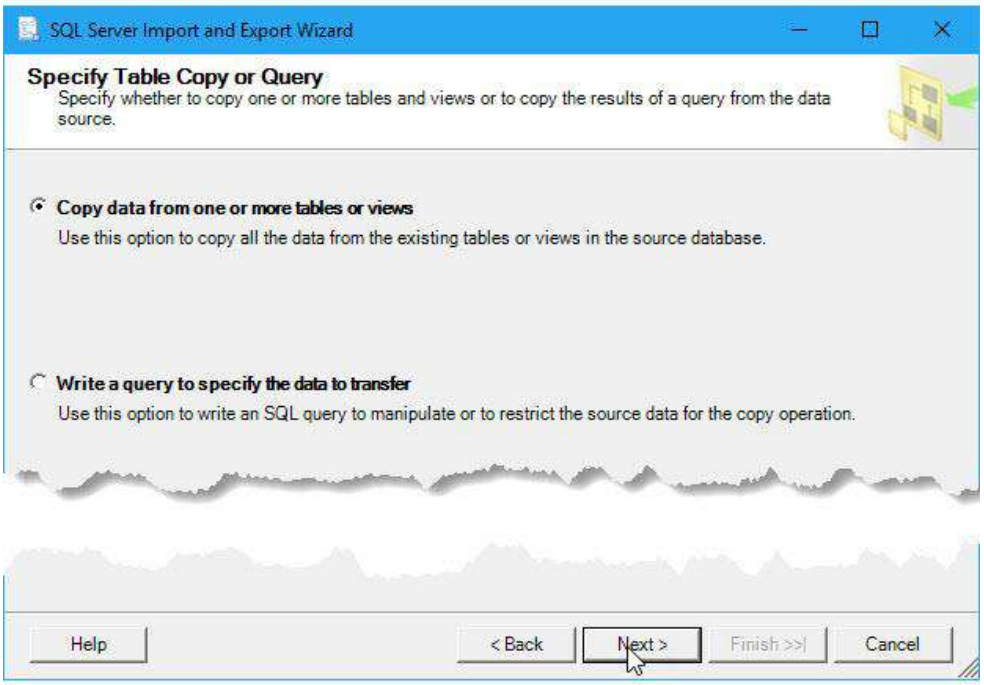
ШАГ 4. ВЫБОР SQL SERVER В КАЧЕСТВЕ НАЗНАЧЕНИЯ

Необходимые сведения	**Свойство "Поставщик данных .NET Framework для SQL Server"**
Имя сервера	**Источник данных**
Сведения для проверки подлинности (имя входа)	**Встроенная система безопасности** или **Идентификатор пользователя** и **Пароль** Чтобы открыть раскрывающийся список баз данных на сервере, сначала нужно указать действительные данные для входа.
Имя базы данных	**Исходный каталог**



Шаг 5. Копирование таблицы вместо написания запроса

На следующей странице **Выбор копирования таблицы или запроса** укажите, что нужно скопировать всю таблицу исходных данных. Вам не нужно создавать запрос на языке SQL, чтобы выбрать данные для копирования.

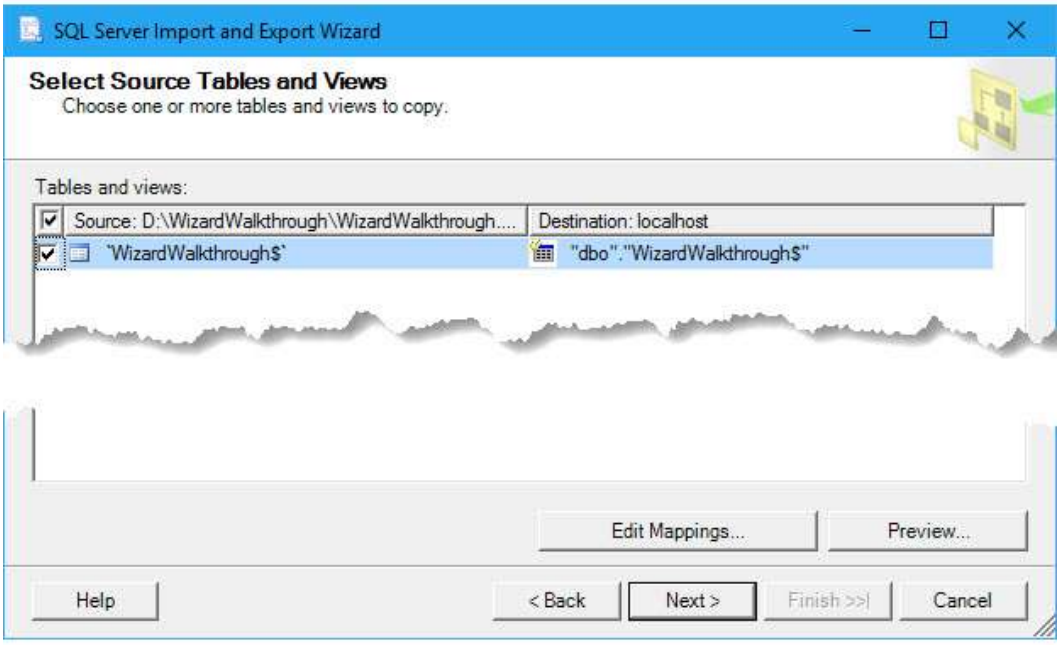


Шаг 6. Выбор копируемой таблицы

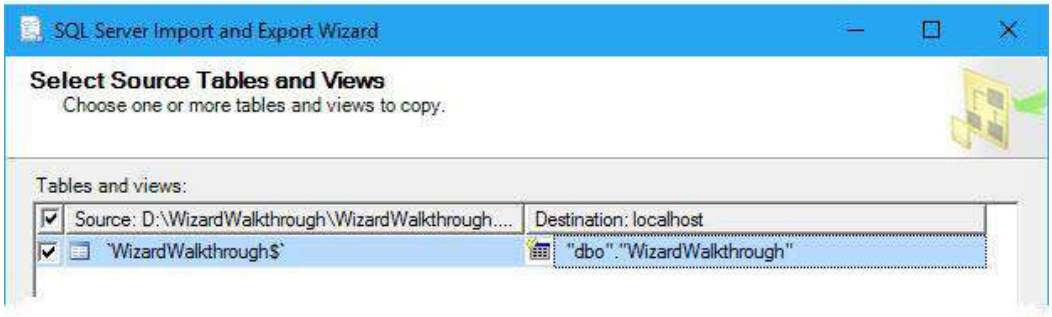
На следующей странице **Выбор исходных таблиц и представлений** выберите таблицу или таблицы, которые необходимо копировать из источника данных. Затем следует сопоставить каждую из выбранных исходных таблиц с новой или существующей целевой таблицей.

В этом примере по умолчанию мастер сопоставил лист **WizardWalkthrough$** в столбце **Source** с новой таблицей с таким же именем в назначении SQL Server. (Книга Excel содержит всего один рабочий лист.)

- Знак доллара ($) на имени исходной таблицы обозначает лист Excel. (Именованный диапазон в Excel представлен лишь своим именем.)
- Звезда на значке целевой таблицы указывает, что мастер создаст новую целевую таблицу.

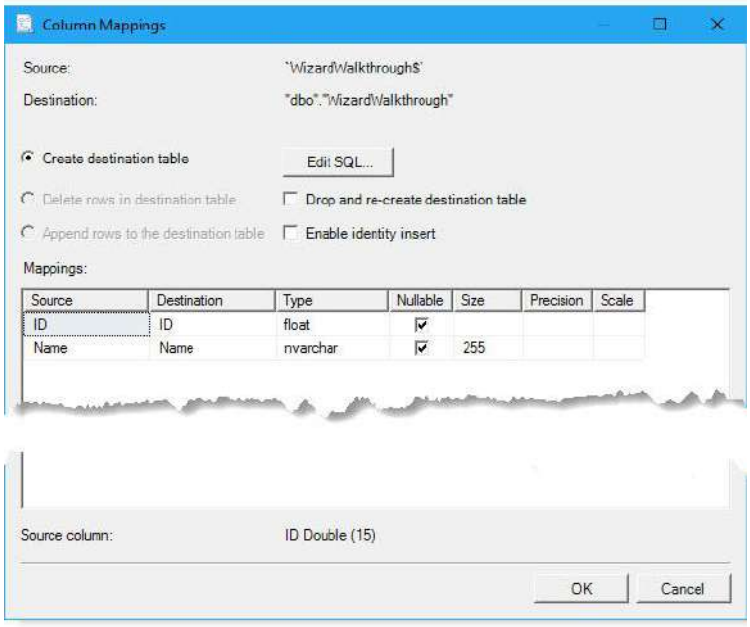


Может потребоваться удалить знак доллара ($) из имени новой целевой таблицы.



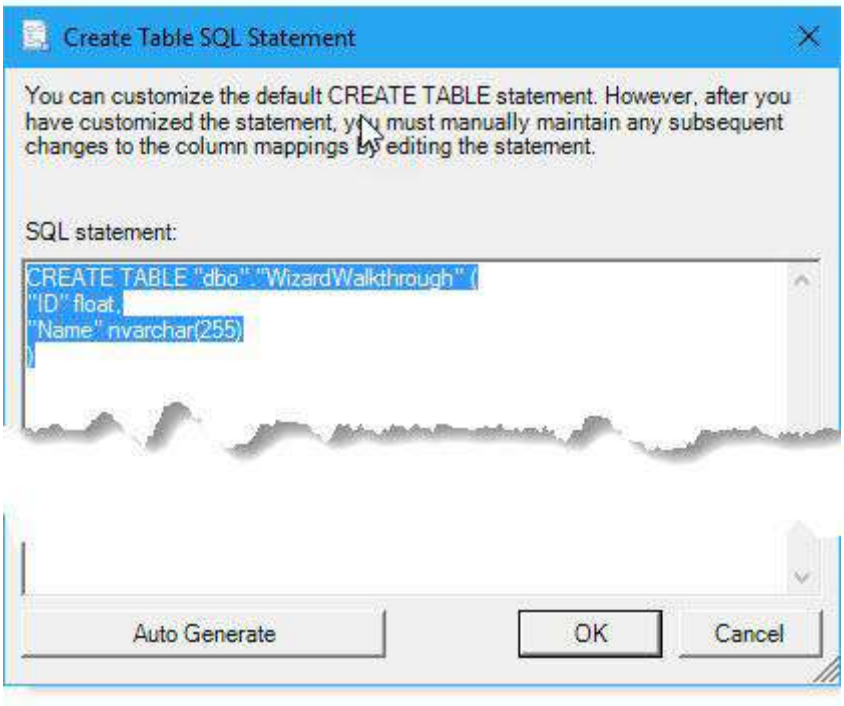
Необязательный шаг 7. Проверка сопоставлений столбцов

Прежде чем закрыть страницу **Выбор исходных таблиц и представлений**, вы можете нажать кнопку **Изменить сопоставления**, чтобы открыть диалоговое окно **Сопоставления столбцов**. Здесь в таблице **Сопоставления** видно, как мастер будет сопоставлять столбцы на исходном листе со столбцами в целевой таблице.



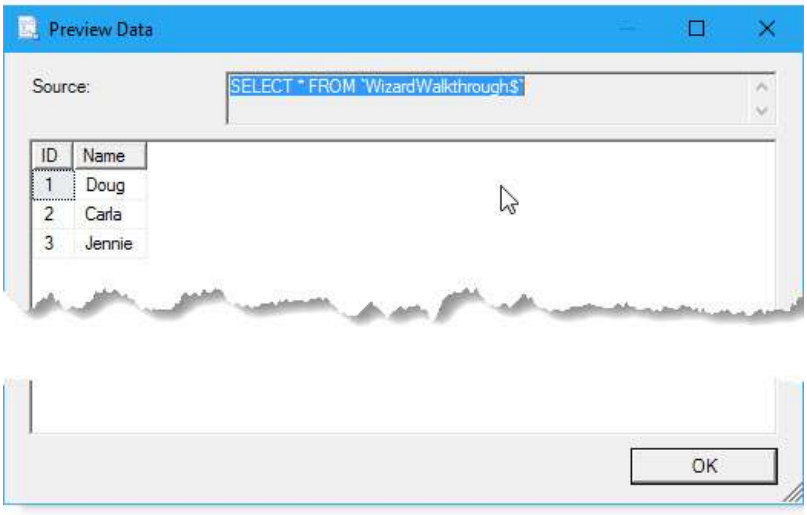
Необязательный шаг 8. Просмотр инструкции CREATE TABLE

Хотя диалоговое окно **Сопоставления столбцов** открыто, вы можете нажать кнопку **Изменить SQL**, чтобы открыть диалоговое окно **Инструкция SQL Create Table**. Вы увидите инструкцию **CREATE TABLE**, созданную мастером для создания целевой таблицы. Обычно менять эту инструкцию не требуется.



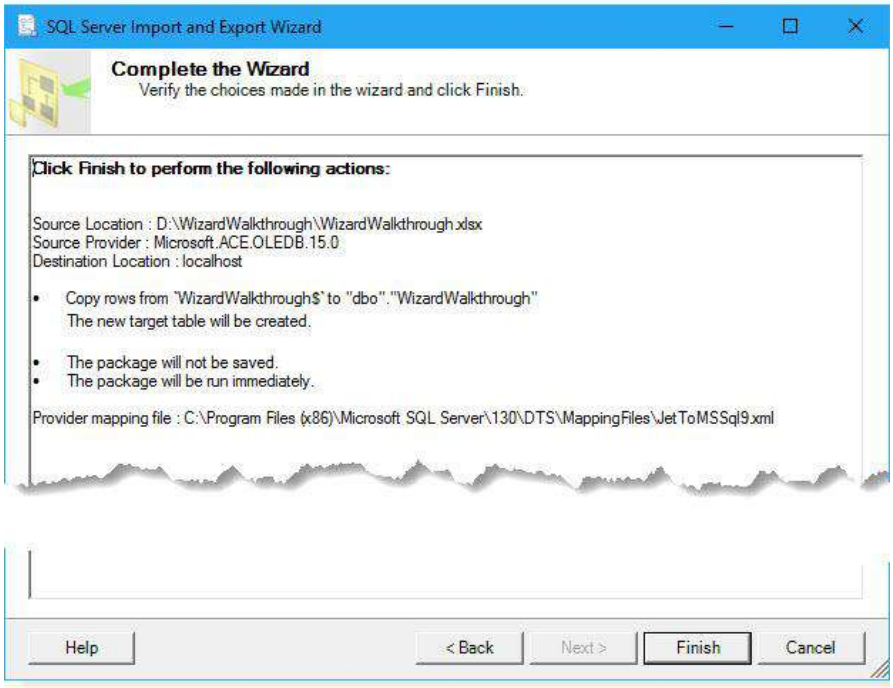
Необязательный шаг 9. Просмотр данных для копирования

Нажав кнопку **ОК** для закрытия диалогового окна **Инструкция SQL Create Table** и кнопку **ОК** для закрытия диалогового окна **Сопоставления столбцов**, вы вернетесь на страницу **Выбор исходных таблиц и представлений**. При необходимости нажмите кнопку **Предварительный просмотр**, чтобы просмотреть образец данных, которые скопирует мастер. В этом примере он выглядит правильно.



Шаг 10. Требуется запустить операцию импорта-экспорта

На следующей странице **Сохранение и запуск пакета** оставьте параметр **Запустить немедленно** включенным, чтоб скопировать данные сразу после нажатия кнопки **Готово** на следующей странице. Либо вы можете пропустить следующую страницу, нажав кнопку **Готово** на странице **Сохранение и запуск пакета**.

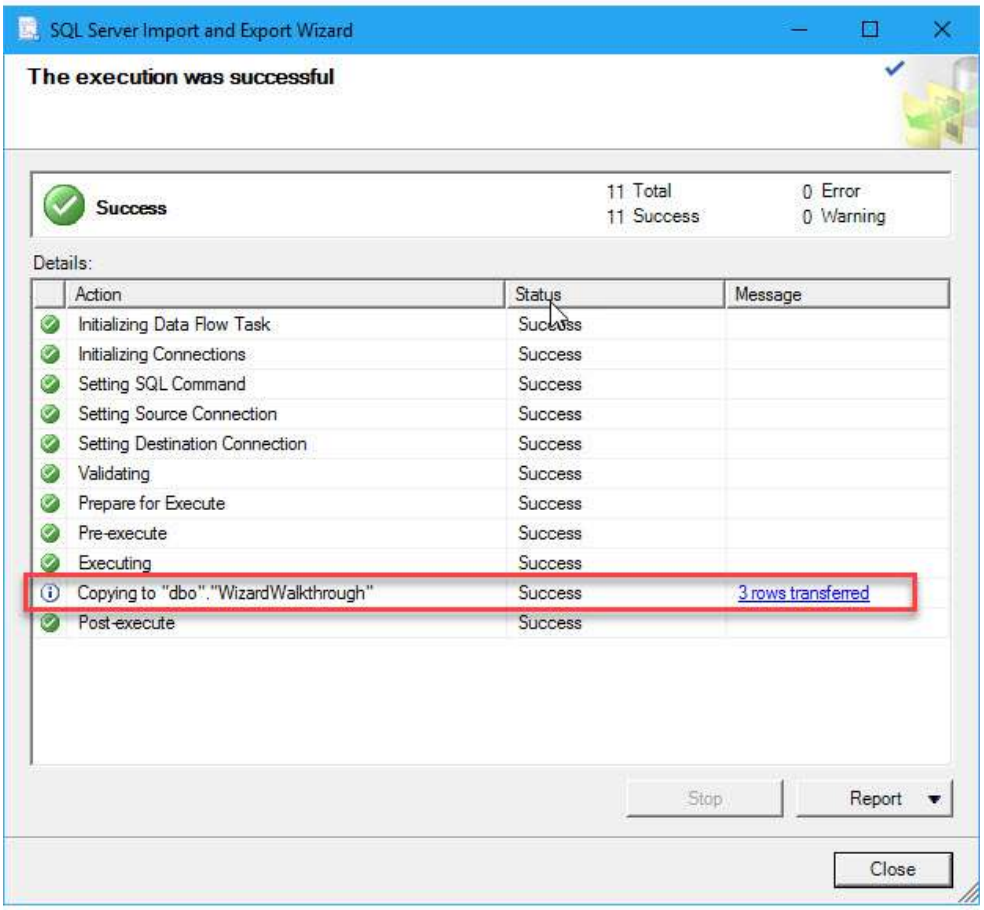


Шаг 11. Завершение работы с мастером и запуск операции импорта-экспорта

Если вы нажали кнопку **Далее** вместо **Готово** на странице **Сохранение и запуск пакета**, то на следующей странице **Завершение работы мастера** выводится сводка о дальнейших действиях мастера. Нажмите кнопку **Готово**, чтобы запустить операцию импорта-экспорта.

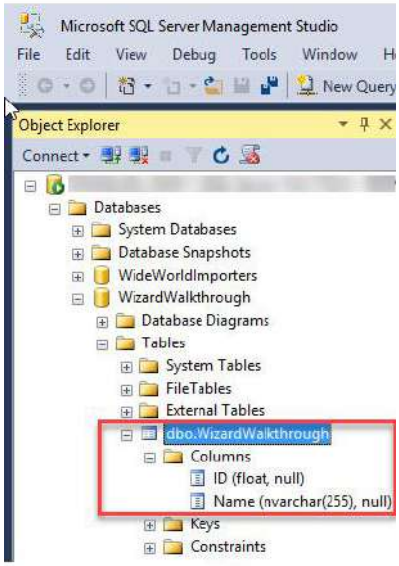
Шаг 12. Просмотр результатов работы мастера

На последней странице наблюдайте, как мастер выполняет каждую задачу, а затем просмотрите результаты. Выделенная строка указывает, что мастер успешно скопировал ваши данные. Готово!

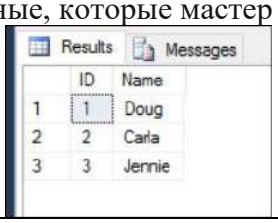


Новая таблица данных, скопированная в SQL Server

Здесь видна новая целевая таблица (В SQL Server Management Studio), созданная мастером в SQL Server.



Здесь (опять в SSMS) видны данные, которые мастер скопировал в SQL Server.



ЗАДАНИЕ НА РАБОТУ

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Осуществите импорта различных по формату данных пользователя в созданную ранее базу данных с учетом предметной области.

Задание 3. Оформите подробный отчет по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите процесс подключения к MySQL с помощью драйвера ODBC для MySQL
2. Какие Указываемые параметры (драйвер ODBC для MySQL) используются при настройке?
3. Опишите процесс подключения к MySQL с помощью поставщика данных платформы .NET Framework для MySQL**.**

Лабораторная работа №7
«ВЫПОЛНЕНИЕ НАСТРОЕК ДЛЯ АВТОМАТИЗАЦИИ ОБСЛУЖИВАНИЯ БАЗЫ ДАННЫХ»

Цель работы: получить теоретические знания и практических выполнения настроек для автоматизации обслуживания базы данных

Теоретические сведения

Рассмотрим создание плана обслуживания одного или нескольких серверов с помощью мастера планов обслуживания в SQL Server. Это позволяет выполнять различные задачи администрирования базы данных, включая резервное копирование, проверки целостности базы данных или обновление статистики базы данных через указанные интервалы времени.

Ограничения

- Для создания многосерверного плана требуется необходимо настроить многосерверную среду, содержащую один главный сервер и один или несколько целевых серверов. Следует создать и поддерживать многосерверные планы обслуживания на главном сервере. Вы можете просматривать планы на целевых серверах.
- Члены ролей **db_ssisadmin** и **dc_admin** могут повышать свои права доступа до **sysadmin**. Такое повышение прав доступа может происходить, поскольку эти роли могут изменять пакеты служб Службы Integration Services , а эти пакеты могут выполняться SQL Server при помощи контекста безопасности **sysadmin** агента SQL Server.

Чтобы предотвратить такое повышение прав доступа при выполнении планов обслуживания, наборов элементов сбора данных и других пакетов служб Службы Integration Services , настройте задания агента SQL Server , запускающие пакеты, на использование учетной записи-посредника с ограниченными правами доступа или добавьте в роли **db_ssisadmin** и **dc_admin** только членов роли **sysadmin** .

Permissions

Для создания планов обслуживания и работы с ними пользователь должен быть членом предопределенной роли сервера **sysadmin** . В обозревателе объектов узел **Планы обслуживания** отображается только для пользователей, являющихся членами предопределенной роли сервера **sysadmin** .

Использование мастера планов обслуживания

Запуск мастера

1. Разверните сервер, где нужно создать план управления.
2. Разверните папку **Управление** .
3. Щелкните правой кнопкой мыши папку **Планы обслуживания** и выберите пункт **Мастер планов обслуживания**.
4. На странице **Мастер планов обслуживания SQL Server** нажмите кнопку **Далее**.
5. На странице **Выбор свойств плана** выполните следующие действия:
1. В поле **Имя** введите имя создаваемого плана обслуживания.
2. В поле **Описание** кратко опишите план обслуживания.
3. В списке **Выполнить от имени** укажите учетные данные, используемые агентом Microsoft SQL Server при выполнении плана обслуживания.
4. Выберите вариант **Отдельные расписания для каждой задачи** или **Единое расписание для всего плана или без расписания** , чтобы указать повторяющееся расписание плана обслуживания.

ПРИМЕЧАНИЕ. Если выбран вариант **Отдельные расписания для каждой задачи**, необходимо выполнить шаги, приведенные далее в пункте **Д** , для каждой задачи в плане обслуживания.

5. Если выбран вариант **Единое расписание для всего плана или без расписания**, то нажмите кнопку **Изменить** в разделе **Расписание**.

1. В диалоговом окне **Создание расписания задания** в поле **Имя** введите имя расписания задания.

2. В списке **Тип расписания** выберите тип расписания:

- **Запускать автоматически при запуске агента SQL Server**
- **Запускать при бездействии процессоров**
- **Повторяющееся**. Это параметр выбирается по умолчанию.
- **Однократно**

3. Установите или снимите флажок **Включен** , чтобы включить или отключить расписание.

4. При выборе **Повторяющееся**:

1. В разделе **Частота** в списке **Выполняется** укажите частоту выполнения:

- При выборе **Ежедневно** в поле **Выполняется каждые** укажите частоту повторного выполнения расписания задания в днях.
- При выборе **Еженедельно** в поле **Выполняется каждые** укажите частоту повторного выполнения расписания задания в неделях. Выберите день недели, в которые выполняется расписание задания.
- При выборе **Ежемесячно** щелкните **День** или **Определенный**.
- При выборе **День** введите дату месяца, в которую должно выполняться расписание задания, и укажите частоту повторного выполнения расписания задания в месяцах. Например, если требуется, чтобы расписание задания выполнялось 15 числа каждого второго месяца, выберите **День** и введите в первом поле 15 и 2 — во втором поле. Обратите внимание, что число, введенное во втором поле, не должно превышать 99.
- При выборе **Определенный** выберите определенный день недели в месяце, в котором должно выполняться расписание задания, и укажите частоту повторного выполнения расписания задания в месяцах. Например, если требуется, чтобы расписание задания выполнялось в последний день недели каждого второго месяца, выберите **День,** затем **последний** в первом списке и **рабочий день** во втором списке, а потом введите "2" во втором поле. Также можно выбрать **первый**, **второй**, **третий** или **четвертый**, а также конкретные дни недели (например: воскресенье или среду) в первых двух списках. Обратите внимание, что число, введенное в последнем поле, не должно превышать 99.

2. В поле **Сколько раз в день** укажите частоту повторного выполнения расписания задания в день запуска расписания задания:

- При выборе **Выполнять раз в** укажите определенное время дня для запуска расписания задания в поле **Выполнять раз в** . Укажите время дня: час, минуту и секунду.
- При выборе **Выполняется каждые** укажите частоту выполнения задания в выбранный день в поле **Частота**. Например, если требуется, чтобы расписание задания выполнялось каждые 2 часа в день запуска расписания задания, выберите **Выполняется кажд.** , введите "2" в первом поле, а затем выберите в списке **часы**. В этом списке также можно выбрать **минуты** и **секунды**. Обратите внимание, что число, введенное в первом поле, не должно превышать 100.

В поле **Начинать в** введите время для начала запуска расписания задания. В поле **Заканчивать в** введите время для завершения повторного выполнения расписания задания. Укажите время дня: час, минуту и секунду.

3. В разделе **Длительность**, в области **Дата начала** введите дату начала запуска расписания задания. Выберите **Дата окончания** или **Без даты окончания** , чтобы указать

дату завершения выполнения расписания задания. При выборе **Дата окончания** введите дату завершения запуска расписания задания.

5. При выборе значения **Однократно** в **Однократное выполнение** в поле **Дата** введите дату запуска расписания задания. В поле **Время** введите время запуска расписания задания. Укажите время дня: час, минуту и секунду.

6. В разделе **Сводка** в **Описание** проверьте правильность всех параметров расписания задания.

7. Нажмите кнопку **ОК**.

6. Щелкните **Далее**.

6. На странице **Выбор целевых серверов** выберите серверы, на которых необходимо запустить план обслуживания. Эта страница видна только на экземплярах SQL Server , сконфигурированных как главный сервер.

ПРИМЕЧАНИЕ. Чтобы создать многосерверный план обслуживания, необходимо настроить многосерверную среду, содержащую один главный сервер и один или несколько целевых серверов, при этом локальный сервер должен быть настроен в качестве главного. В многосерверной среде на этой странице отображается **(локальный)** главный сервер и все соответствующие целевые серверы.

7. На странице **Выбор задач по обслуживанию** выберите одну или несколько задач обслуживания для добавления к плану. После выбора всех необходимых задач нажмите кнопку **Далее**.

ПРИМЕЧАНИЕ. Выбранные здесь задачи определяют, параметры на каких страницах понадобится заполнить после страницы **Выбор порядка задач по обслуживанию**.

8. На странице **Выбор порядка задач по обслуживанию** выберите задачу и измените ее порядок выполнения кнопками **Вверх...** и **Вниз...** . После определения порядка задач нажмите кнопку **Далее**.

ПРИМЕЧАНИЕ. Если выбран вариант **Отдельные расписания для каждой задачи** на странице **Выбор свойств плана**, то изменение порядка задач по обслуживанию невозможно.

Определение проверки целостности базы данных (CHECKDB)

На странице **Определение задачи по проверке целостности базы данных** выберите базы данных, в которых будут проверяться размещение и структурная целостность пользовательских и системных таблиц и индексов. Выполнение инструкции DBCC CHECKDBTransact-SQL в этой задаче гарантирует обнаружение всех проблем целостности с базой данных, что позволяет системному администратору или владельцу базы данных устранить проблемы позднее. Дополнительные сведения см. в разделе DBCC CHECKDB (Transact-SQL).По завершении нажмите кнопку **Далее**.

На этой странице доступны следующие параметры.

Список **Базы данных**

Укажите базы данных, для которых должна выполняться эта задача.

- **Все базы данных**

Создайте план обслуживания, который запускает эту задачу для всех баз данных Microsoft SQL Server, за исключением **tempdb**.

Системные базы данных

- Создайте план обслуживания, который запускает эту задачу для системных баз данных SQL Server , за исключением **tempdb** и созданных пользователем баз.

Все пользовательские базы данных (кроме master, model, msdb, tempdb)

- Создать план обслуживания, который производит запуск этой задачи по отношению ко всем базам данных, созданных пользователем. Для системных баз данных SQL Server задачи обслуживания выполняться не будут.

Следующие базы данных

- Создать план обслуживания, который производит запуск этой задачи по отношению только к выбранным базам данных. Если выбран этот параметр, необходимо выбрать в списке хотя бы одну базу данных.

Флажок **Включить индексы**

- Проверка целостности всех страниц индекса, а также табличных страниц данных.

Только физические

- Ограничивает проверку лишь проверкой целостности физической структуры страниц и заголовков записей и последовательности выделения пространства в базе данных. Указание этого параметра может снизить время выполнения инструкции DBCC CHECKDB для больших баз данных, поэтому рекомендуется для частого использования в рабочих системах.

Tablock

- Указание значения аргумента приводит к получению инструкцией DBCC CHECKDB блокировок вместо использования внутреннего моментального снимка базы данных. Это включает краткосрочное использование монопольной блокировки (X) на базу данных. Использование этого параметра ускорит выполнение инструкции DBCC CHECKDB на базе данных, находящейся под интенсивной нагрузкой, однако уменьшит возможности одновременной работы пользователей с базой данных во время выполнения инструкции DBCC CHECKDB.

Определение задач по сжатию базы данных

1. На странице **Определение задачи «Сжатие базы данных»** создайте задачу, которая уменьшает размер выбранных баз данных, используя инструкцию DBCC SHRINKDATABASE с параметром NOTRUNCATE или TRUNCATEONLY .

После завершения нажмите кнопку **Далее**.

ПРЕДУПРЕЖДЕНИЕ! Данные, перемещаемые в процессе сжатия файла, могут быть разбросаны по любым доступным местам в файле. Это вызывает фрагментацию индекса и может увеличить время выполнения запросов, выполняющих поиск в диапазоне индекса. Чтобы устранить фрагментацию, предусмотрите возможность перестроения индексов файла после сжатия.

На этой странице доступны следующие параметры.

Список **Базы данных**

Укажите базы данных, для которых должна выполняться эта задача. Дополнительные сведения о параметрах, доступных в этом списке, см. выше в шаге 9.

Поле **Сжать базу данных при превышении ею размера**
Укажите размер в мегабайтах, по достижении которого будет выполняться задача.

Поле **Объем свободного места после сжатия**
Сжатия останавливается, когда свободное место в файлах базы данных достигает этого объема (в процентах).

Удерживать свободное место в файлах базы данных
База данных сжимается до состояния непрерывных страниц, но страницы не освобождаются, и файлы базы данных не сжимаются. С помощью этого параметра можно предусмотреть ситуацию, когда в будущем ожидается повторное увеличение базы данных, и поэтому не нужно перераспределять пространство. Этот параметр не позволяет достичь максимальной степени сжатия файлов базы данных. В этом случае используется параметр NOTRUNCATE.

Возвращать освободившееся место в операционную систему
База данных сжимается до состояния непрерывных страниц, и страницы возвращаются

обратно в операционную систему, где они могут использоваться другими программами. В этом случае используется параметр TRUNCATEONLY. Это параметр по умолчанию.

Определение задач индекса

1. На странице **Определение задачи «Реорганизация индекса»** выберите серверы, на которые будут перемещаться страницы индекса для более эффективного порядка поиска. Эта задача использует инструкцию ALTER INDEX ... REORGANIZE. После завершения нажмите кнопку **Далее**.

На этой странице доступны следующие параметры.

Список **Базы данных.**

Укажите базы данных, для которых должна выполняться эта задача. Дополнительные сведения о параметрах, доступных в этом списке, см. выше в шаге 9.

Список **Объект.**

В списке **Выбор** можно отображать только таблицы, только представления или таблицы и представления. Этот список доступен, если в списке **База данных** выше выбрана только одна база данных.

Список **Выбор.**

Укажите таблицы или индексы, которые должны обрабатываться этой задачей. Недоступно, если в диалоговом окне «Объект» выбран тип **Таблицы и представления** .

Флажок **Сжимать большие объекты.**

По возможности освобождается пространство для таблиц и представлений. Этот параметр использует инструкцию ALTER INDEX ... LOB_COMPACTION = ON.

2. На странице **Определение задачи "Перестроение индекса"** выберите базы данных, в которых будут повторно создаваться несколько индексов. Эта задача использует инструкцию ALTER INDEX ... REBUILD PARTITION. Дополнительные сведения см. в разделе ALTER INDEX (Transact-SQL). После завершения нажмите кнопку **Далее**.

На этой странице доступны следующие параметры.

Список **Базы данных.**

Укажите базы данных, для которых должна выполняться эта задача. Дополнительные сведения о параметрах, доступных в этом списке, см. выше в шаге 9.

Список **Объект.**

В списке **Выбор** можно отображать только таблицы, только представления или таблицы и представления. Этот список доступен, если в списке **База данных** выше выбрана только одна база данных.

Список **Выбор.**

Укажите таблицы или индексы, которые должны обрабатываться этой задачей. Недоступно, если в диалоговом окне «Объект» выбран тип **Таблицы и представления** .

Область **Параметры свободного места.**

Представляет параметры для применения коэффициента заполнения к индексам и таблицам.

Свободное пространство по умолчанию на странице.

Реорганизует страницы с использованием объема свободного места по умолчанию. Индексы таблиц в базе данных будут удалены и созданы повторно с коэффициентом заполнения, указанным при создании индексов. Это параметр по умолчанию.

Поле **Изменить долю свободного места на странице.**

Удалите индексы таблиц в базе данных и создайте их повторно с новым, автоматически вычисляемым коэффициентом заполнения, резервирующим указанный объем свободного пространства на страницах индекса. Чем выше процентное значение, тем больше свободного места резервируется на страницах индекса и тем больше будет размер индекса. Допустимые значения: от 0 до 100. Используется параметр FILLFACTOR .

Область **Дополнительные параметры.**

Представляет дополнительные параметры для сортировки индексов и повторного индексирования.

Флажок **Сортировать результаты в tempdb.**Использует параметр SORT_IN_TEMPDB , определяющий место временного хранения промежуточных результатов сортировки, сформированных при создании индекса. Если операция сортировки не требуется или сортировка может быть выполнена в памяти, параметр SORT_IN_TEMPDB не обрабатывается.

Флажок **Разредить индекс.**

Используется параметр PAD_INDEX .

Флажок **Сохранять индекс в режиме "в сети" в процессе переиндексирования.**

Использует параметр ONLINE , который дает пользователям возможность получать доступ к базовой таблице или данным кластеризованного индекса, а также к любым, связанным с ними некластеризованным индексам при операциях с индексами. При выборе этого параметра активируются другие параметры для перестроения индексов, которые не допускают перестроения в режиме "в сети": **Не перестраивать индексы** и **Перестроение индекса в автономном режиме**.

При выборе этого параметра также активируется использование низкого приоритета, для которого используется параметр WAIT_AT_LOW_PRIORITY . Операции перестроения индекса в режиме "в сети" будут ожидать блокировки с низким приоритетом в течение MAX_DURATION мин, позволяя выполняться другим операциям, пока операция перестроения индекса в режиме "в сети" находится в состоянии ожидания.

ПРИМЕЧАНИЕ. Операции с индексами в сети доступны не во всех выпусках SQL Server.

Флажок **MAXDOP**

Переопределяет параметр конфигурации, задающий максимальный уровень параллелизма, в sp_configure для DBCC CHECKDB.

Определение задачи «Обновление статистики»

1. На странице **Определение задачи «Обновление статистики»** определите базы данных, в которых будет обновляться статистика индекса и таблицы. Эта задача использует инструкцию UPDATE STATISTICS. Дополнительные сведения см. в разделе UPDATE STATISTICS (Transact-SQL). По завершении нажмите кнопку **Далее**.

На этой странице доступны следующие параметры.

Список **Базы данных.**

Укажите базы данных, для которых должна выполняться эта задача. Дополнительные сведения о параметрах, доступных в этом списке, см. выше в шаге 9.

Список **Объект.**

В списке **Выбор** можно отображать только таблицы, только представления или таблицы и представления. Этот список доступен, если в списке **База данных** выше выбрана только одна база данных.

Список **Выбор.**

Укажите таблицы или индексы, которые должны обрабатываться этой задачей. Недоступно, если в диалоговом окне «Объект» выбран тип **Таблицы и представления** .

Вся собранная статистика.

Обновить статистику столбцов и статистику индексов.

Только статистика по столбцам.

Обновить только статистику столбцов. Используется параметр WITH COLUMNS .

Только статистика индексов.

Обновить только статистику индексов. Используется параметр WITH INDEX .

Тип просмотра.

Тип просмотра, который используется для сбора обновленной статистики.

Полный просмотр.

Считывает все строки в таблице или представлении для сбора статистики.

Пример.

Укажите процент таблицы или индексированного представления либо количество строк для выборки, если статистика собирается для больших таблиц или представлений.

Определение задачи «Очистка журнала»

1. На странице **Определение задачи «Очистка журнала»** определите базы данных, в которых нужно удалить журнал задач. Эта задача использует инструкции EXEC sp_purge_jobhistory, EXEC sp_maintplan_delete_logи EXEC sp_delete_backuphistory для удаления сведений журнала из таблиц **msdb** . По завершении нажмите кнопку **Далее**.

На этой странице доступны следующие параметры.

Выберите данные журнала для удаления
Выберите тип удаляемых данных задачи

Журнал резервного копирования и восстановления
Хранение записей о том, когда были созданы последние резервные копии, может помочь SQL Server в создании плана восстановления, когда потребуется восстановить базу данных. Срок хранения должен как минимум соответствовать частоте полных резервных копирований базы данных.

Журнал заданий агента SQL Server
Этот журнал может помочь устранить ошибки, возникшие при выполнении заданий, или определить, почему были выполнены операции с базой данных.

Журнал плана обслуживания
Этот журнал может помочь устранить ошибки, возникшие при выполнении заданий плана обслуживания, или определить, почему были выполнены операции с базой данных.

Удалить из журнала записи старше
Укажите возраст элементов, которые следует удалить. Можно указать срок в **ч**, **дн.** , **нед.** (по умолчанию), **мес.** или **годах**.

Определение задачи «Выполнение задания агента»

1. На странице **Определение задачи «Выполнение задания агента»** в разделе **Доступные задания агента SQL Server** выберите задания для выполнения. Этот параметр не будет доступен, если нет заданий агента SQL Server. Эта задача использует инструкцию EXEC sp_start_job. По завершении нажмите кнопку **Далее**.

Определение задач резервного копирования

1. На странице **Определение задачи "Резервное копирование базы данных (полное)"** выберите базы данных, для которых создается полная резервная копия. Эта задача использует инструкцию BACKUP DATABASE. Дополнительные сведения см. в разделе BACKUP (Transact-SQL). По завершении нажмите кнопку **Далее**.

На этой странице доступны следующие параметры.

Список **Тип резервного копирования**

Отображает тип выполняемой резервной копии. Доступно только для чтения.

Список **Базы данных**

Укажите базы данных, для которых должна выполняться эта задача. Дополнительные сведения о параметрах, доступных в этом списке, см. выше в шаге 9.

Компонент резервного копирования

Выберите **База данных** для создания резервной копии всей базы данных. Выберите **Файл и файловые группы** для создания резервной копии части базы данных. При выборе этого значения укажите имя файла или файловой группы. Если в окне **Базы**

данных выбрано несколько баз данных, то в области **Компонент резервного копирования** выберите только **Базы данных**. Для резервного копирования файлов или файловых групп создайте задачу для каждой базы данных. Эти параметры доступны, если в списке **База данных** выше выбрана только одна база данных.

Флажок **Срок действия резервного набора данных истекает**

Указывает время, по истечении которого резервный набор данных для этой резервной копии может быть перезаписан. Выберите параметр **После** и введите число дней до истечения срока действия или выберите параметр **Вкл** и укажите дату истечения срока. Этот параметр будет отключен, если в качества назначения резервного копирования выбран параметр **URL-адрес** .

Создать резервную копию на

Задает носитель, на котором создается резервная копия базы данных. Выберите **Диск**, **Лента** или **URL-адрес**. Доступны только ленточные устройства, подключенные к компьютеру, на котором находится база данных.

Создание резервных копий баз данных в одном или нескольких файлах

Нажмите кнопку **Добавить** , чтобы открыть диалоговое окно **Выбор места расположения резервной копии** . Этот параметр будет отключен, если в качества назначения резервного копирования выбран параметр «URL-адрес».

Нажмите кнопку **Удалить** , чтобы удалить файл из поля.

Нажмите кнопку **Содержимое** , чтобы прочитать заголовок файла и отобразить текущее содержимое резервной копии файла.

Диалоговое окно **Выбор места расположения резервной копии**

Выберите файл, ленточный накопитель или устройство резервного копирования в качестве назначения резервной копии. Этот параметр будет отключен, если в качества назначения резервного копирования выбран параметр «URL-адрес».

Список **Если файлы резервной копии существуют**

Укажите, как обрабатывать существующие резервные копии. Выберите пункт **Добавить** для добавления новых резервных копий после уже существующих в файле или на ленте. Выберите пункт **Перезаписать** , чтобы удалить старое содержимое из файла или с ленты и записать на его место новую резервную копию.

Создать файл резервной копии для каждой базы данных

Создавать файл резервной копии в расположении, указанном в соответствующем окне папки. Создается один файл для каждой выбранной базы данных. Этот параметр будет отключен, если в качества назначения резервного копирования выбран параметр «URL-адрес».

Флажок **Создавать вложенный каталог для каждой базы данных**

Создать внутри указанного каталога на диске отдельный вложенный каталог для резервной копии каждой базы данных, создаваемой как часть плана обслуживания.

ВАЖНО! Этот вложенный каталог будет наследовать разрешения от родительского. Ограничьте разрешения для предотвращения несанкционированного доступа.

Поле **Папка**

Укажите папку, в которой будут автоматически создаваться файлы баз данных. Этот параметр будет отключен, если в качества назначения резервного копирования выбран параметр «URL-адрес».

Учетные данные SQL

Учетные данные SQL, используемые для проверки подлинности в службе хранилища Azure. Если у вас нет существующих учетных данных SQL, нажмите кнопку **Создать** , чтобы создать новые учетные данные SQL.

ВАЖНО! В диалоговом окне, которое открывается при нажатии кнопки **Создать** , необходимо ввести сертификат управления или профиль публикации подписки. Если у вас нет доступа к сертификату управления или профилю публикации, можно создать учетные данные SQL, указав имя учетной записи хранилища и сведения ключа доступа при помощи Transact-SQL или SQL Server Management Studio. Также можно в среде SQL Server Management Studio, из экземпляра компонента database engine, щелкнуть правой кнопкой мыши **Безопасность** и выбрать пункт **Создать**, а затем **Учетные данные**. Укажите имя учетной записи хранения в поле **Идентификатор** и ключ доступа в поле **Пароль** .

Контейнер хранилища Azure

Укажите имя контейнера службы хранилища Azure.

Префикс URL-адреса:

Этот элемент создается автоматически на основе данных учетной записи хранения в учетных данных SQL и заданного имени контейнера Azure. Мы рекомендуем не изменять данные в этом поле, если только не используется домен в формате, отличном от **<storage account>.blob.core.windows.net**.

Поле **Расширение файла резервной копии**

Укажите расширение, которое должно использоваться для файлов резервных копий. Значение по умолчанию — BAK.

Флажок **Проверять целостность резервной копии**

Проверка полноты резервного набора данных и читаемости всех томов.

Флажок **Perform checksum** (Вычислить контрольную сумму)

Проверка контрольной суммы и наличия разрывов на каждой странице (если эти проверки включены и доступны), а также создание контрольной суммы для всей резервной копии.

Флажок **Продолжить при возникновении ошибки**

Определяет, что инструкция BACKUP должна продолжить выполнение, несмотря на возникновение таких ошибок, как неверные контрольные суммы или разрывы страницы.

Шифрование резервной копии

Для создания шифрованной резервной копии установите флажок **Зашифровать файл резервной копии** . Выберите алгоритм шифрования для шага шифрования и выберите сертификат или асимметричный ключ из списка существующих сертификатов или асимметричных ключей. Доступны следующие алгоритмы шифрования:

- AES 128
- AES 192
- AES 256
- Triple DES

Параметр шифрования отключен, если вы решили присоединить резервную копию к существующему резервному набору данных.

Рекомендуется регулярно создавать резервные копии сертификата или ключей и сохранять их в местоположении, отличном от местоположения шифруемой резервной копии.

Поддерживаются только ключи, относящиеся к расширенному управлению ключами.

Флажок **Размер блока** , список

Указывает размер физического блока в байтах. Этот параметр обычно влияет на производительность только при записи на ленточные устройства, в массивы RAID или SAN.

Флажок **Max transfer size** (Максимальный размер передачи), список

Указывает наибольший объем пакета данных в байтах для обмена данными между SQL Server и носителем резервных копий.

Список **Сжимать резервные копии**

ТАБЛИЦА 1

Значение	Описание
Использовать параметр сервера по умолчанию	Щелкните для использования настроек уровня сервера, установленных по умолчанию. Значения по умолчанию устанавливаются в параметре конфигурации сервера **backup compression default** .
Сжимать резервные копии	Щелкните для сжатия резервной копии, независимо от уровня сервера по умолчанию. ** **Важно!** ** По умолчанию сжатие существенно повышает загрузку ЦП. Дополнительная нагрузка на ЦП может помешать выполнению параллельных операций. Поэтому лучше создавать сжатые резервные копии с низким приоритетом в сеансах, доступ которых к ЦП ограничивается регулятором ресурсов.
Не сжимать резервные копии	Щелкните для создания резервной копии без сжатия, независимо от уровня сервера по умолчанию.

2. На странице **Определение задачи "Резервное копирование базы данных (разностное)"** выберите базы данных, для которых создается частичная резервная копия. Дополнительные сведения о параметрах, доступных на этой странице, см. выше в списке определений на шаге 16. Эта задача использует инструкцию BACKUP DATABASE ... WITH DIFFERENTIAL. По завершении нажмите кнопку **Далее**.

3. На странице **Определение задачи "Резервное копирование базы данных (журнал транзакций)"** выберите базы данных, для которых создается резервная копия журнала транзакций. Дополнительные сведения о параметрах, доступных на этой странице, см. выше в списке определений на шаге 16. Эта задача использует инструкцию BACKUP LOG. По завершении нажмите кнопку **Далее**.

Определение задач «Очистка после обслуживания»

1. На странице **Определение задачи «Очистка после обслуживания»** укажите типы файлов, которые удаляются в рамках плана обслуживания, в том числе текстовые отчеты, созданные планами обслуживания, и файлы резервных копий базы данных. Эта задача использует инструкцию EXEC xp_delete_file. По завершении нажмите кнопку **Далее**.

ВАЖНО! Эта задача автоматически не удаляет файлы во вложенных папках указанного каталога. Это снижает вероятность злоумышленного проникновения с использованием задачи «Очистка после обслуживания» для удаления файлов. Если нужно удалять файлы во вложенных папках первого уровня, установите флажок **Включить вложенные папки первого уровня**.

На этой странице доступны следующие параметры.

Удалить файлы следующего типа

Укажите тип файлов для удаления.

Файлы резервных копий

Удаляет файлы резервной копии базы данных.

Текстовые отчеты плана обслуживания

Удаляет текстовые отчеты предыдущих запусков планов обслуживания.

Размещение файла

Укажите путь к удаляемым файлам.

Удалить определенный файл

Удаляет файл, указанный в текстовом поле **Имя файла** .

Удалить из папки файлы с определенным расширением

Удаляет все файлы с определенным расширением в указанной папке. Позволяет удалить несколько файлов сразу, например все файлы резервных копий с расширением BAK в папке «Вторник».

Поле **Папка**

Путь к папке, содержащей удаляемые файлы, и ее имя.

Поле **Расширение файла**

Введите расширение удаляемых файлов. Для удаления сразу несколько файлов, например всех файлов резервных копий с расширением BAK из папки «Вторник», укажите BAK.

Флажок **Включить вложенные папки первого уровня**

Удаляет файлы с расширением, заданным в поле **Расширение файла** , из вложенных папок первого уровня относительно папки, заданной в поле **Папка**.

Флажок **Удалить файлы на основе возраста во время выполнения задачи**

Задайте минимальный возраст удаляемых файлов, указав числовое значение и единицу времени в поле **Удалить все файлы старше чем** .

Удалить все файлы старше чем

Задайте минимальный возраст удаляемых файлов, указав числовое значение и единицу времени (**час**, **день**, **неделя**, **месяц** или **год**). Файлы, старше указанного временного интервала, будут удалены.

Выбор параметров отчета

1. На странице **Выбор параметров отчета** выберите параметры для сохранения или распространения отчета о действиях плана обслуживания. Эта задача использует инструкцию EXEC sp_notify_operator. По завершении нажмите кнопку **Далее.**

На этой странице доступны следующие параметры.

Флажок **Записать отчет в текстовый файл**

Сохраните отчет в файл.

Поле **Расположение папки**

Задайте местонахождения файла, который будет содержать отчет.

Флажок **Отчет по электронной почте**

Отправляет сообщение электронной почты, если задача завершается ошибкой. Для использования этой задачи необходимо включить и правильно настроить компонент Database Mail с MSDB в качестве базы данных обслуживания почты, а также использовать оператор агента MicrosoftSQL Server с допустимым электронным адресом.

Оператор агента

Указать получателя электронного письма.

Профиль электронной почты

Задайте профиль, определяющий отправителя сообщения по электронной почте.

Завершение работы мастера

1. На странице **Завершение работы мастера** проверьте параметры, выбранные на предыдущих страницах, и нажмите кнопку **Готово**.

2. Страница **Ход работы мастера планов обслуживания** позволяет следить за сведениями о состоянии действий, выполняемых мастером планов обслуживания. В зависимости от действий, выбранных в мастере, страница выполнения может содержать одно или несколько действий. В верхнем поле показано общее состояние мастера и число полученных им сообщений о состоянии, предупреждений и сообщений об ошибках.

На странице **Ход работы мастера планов обслуживания** доступны следующие параметры.

Сведения

Сведения о событии, состоянии и любых сообщениях, которые возвращены в результате действий мастера.

Действие

Задает тип и имя каждого действия.

Состояние

Указывает, вернуло ли действие мастера в целом значение **Успешно** или **Ошибка**.

Сообщение

Любые сообщения об ошибках или предупреждения от процесса.

Отчет

Создание отчета, содержащего результаты мастера создания секций. Доступные параметры: **Просмотреть отчет**, **Сохранить отчет в файл**, **Копировать отчет в буфер обмена** и **Отправить отчет по электронной почте**.

Просмотреть отчет

Открытие диалогового окна **Просмотр отчета** , которое содержит текстовый отчет о работе мастера создания секций.

Сохранить отчет в файл

Открытие диалогового окна **Сохранить отчет как** .

Копировать отчет в буфер обмена

Копирование результатов отчета о работе мастера в буфер обмена.

Отправить отчет по электронной почте

Копирование результатов отчета о состоянии мастера в сообщение электронной почты.

Таким образом, вы можете выполнять следующие задачи в рамках плана обслуживания базы данных:

- Сокращение базы данных.
- Резервное копирование базы данных.
- Выполнение уведомления оператора.
- Обновление статистики базы данных.
- Проверка целостности базы данных.
- Очистка оставшихся файлов обслуживания.
- Выполнение задания агента SQL Server.
- Выполнение инструкции Transact-SQL.
- Восстановление индекса.
- Реорганизация индекса.
- Очистка истории баз данных.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями, используя созданную в предыдущих работах БД, разработайте план обслуживания БД и выполните:
   1) обновление статистики базы данных;
   2) проверку целостности базы данных.
   3) очистку оставшихся файлов обслуживания;
   4) очистка истории баз данных.
3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Опишите процесс выполнения «Сокращение базы данных».
2. Опишите процесс выполнения «Резервное копирование базы данных».
3. Опишите процесс выполнения «Выполнение уведомления оператора».
4. Опишите процесс выполнения «Обновление статистики базы данных».
5. Опишите процесс выполнения «Проверка целостности базы данных».

6. Опишите процесс выполнения «Очистка оставшихся файлов обслуживания».
7. Опишите процесс выполнения «Выполнение задания агента SQL Server».
8. Опишите процесс выполнения «Выполнение инструкции Transact-SQL».
9. Опишите процесс выполнения «Восстановление индекса».
10. Опишите процесс выполнения «Реорганизация индекса».
11. Опишите процесс выполнения «Очистка истории баз данных».

Лабораторная работа №8
«МОНИТОРИНГ РАБОТЫ СЕРВЕРА»

Цель работы: получить теоретические знания и практические навыки выполнения мониторинга работы сервера.

Теоретические сведения

1. Открытие монитора активности в среде SQL Server Management Studio (SSMS)

Монитор активности выполняет запросы в отслеживаемом экземпляре, чтобы получать данные для панелей отображения монитора активности. Если установлен интервал обновления менее 10 секунд, то время, затрачиваемое на выполнение этих запросов, может повлиять на производительность сервера.

1.1. Permissions

Для просмотра фактической активности вам нужно разрешение VIEW SERVER STATE. Для просмотра раздела ввода-вывода в файл данных монитора активности, кроме разрешения на VIEW SERVER STATE, необходимо иметь также разрешения на CREATE DATABASE, ALTER ANY DATABASE или VIEW ANY DEFINITION.

Чтобы вызвать инструкцию KILL для процесса, пользователь должен быть членом предопределенных ролей сервера sysadmin или processadmin.

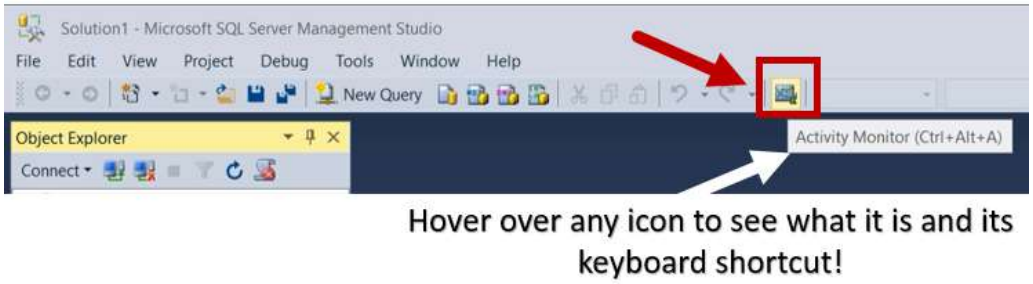
1.12 Открытие монитора активности

Обозреватель объектов

Щелкните правой кнопкой мыши объект верхнего уровня для подключения SQL Server и выберите пункт **Монитор активности**.

Панель инструментов

На стандартной панели инструментов нажмите на значок **Монитор активности**. Он находится в середине, справа от кнопок отмены и повтора.

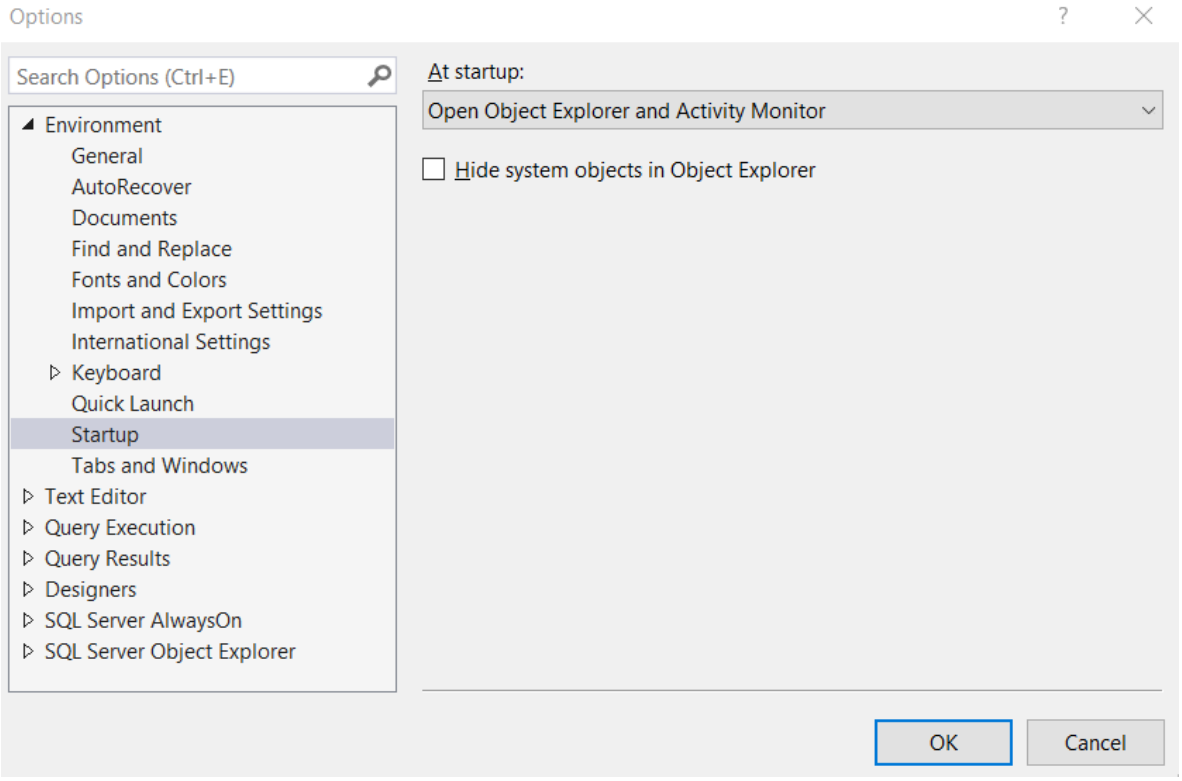


Заполните данные в диалоговом окне **Соединение с сервером** , если вы еще не подключены к экземпляру SQL Server, которые требуется отслеживать.

1.3. Открытие монитора активности и обозревателя объектов при запуске

1. В меню **Сервис** выберите **Параметры**.
2. В диалоговом окне **Параметры** разверните узел **Среда** и выберите **Запуск**.
3. В раскрывающемся списке **При запуске** выберите **Открыть обозреватель объектов и монитор активности**.

4. Щелкните **ОК**.



1.4 Задание интервала обновления монитора активности

1. Откройте монитор активности.
2. Щелкните правой кнопкой мыши пункт **Обзор**, выберите пункт **Интервал обновления**, а затем — интервал, в котором монитор активности будет получать новые данные экземпляра.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями проведите мониторинг используемого сервера.
3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте характеристику монитору активности.
2. Какие параметры позволяет отслеживать монитор активности?

Лабораторная работа №9
«ВЫПОЛНЕНИЕ РЕЗЕРВНОГО КОПИРОВАНИЯ»

Цель работы: получить теоретические знания и практические навыки выполнения резервного копирования.

Теоретические сведения

1. Создание полной резервной копии базы данных

Рассмотрим создание полной резервной копии базы данных в SQL Server с помощью SQL Server Management Studio (Transact-SQL,PowerShell)

Ограничения

- Инструкция BACKUP не допускается в явных и неявных транзакциях.
- Резервные копии, созданные более поздними версиями SQL Server , не могут быть восстановлены в более ранних версиях SQL Server.

Рекомендации

- По мере увеличения размера базы данных полное резервное копирование занимает больше времени и требует больше дискового пространства. Для больших баз данных может потребоваться, кроме полных резервных копий, создавать также и разностные резервные копии баз данных.
- Размер полной резервной копии базы данных вы можете вычислить с помощью системной хранимой процедуры sp_spaceused .
- По умолчанию каждая успешная операция резервного копирования добавляет запись в журнал ошибок служб SQL Server и в журнал системных событий. Если резервные копии создаются слишком часто, сообщения об успешном завершении очень быстро накапливаются. Это приводит к увеличению журналов ошибок, затрудняя поиск других сообщений. Если работа существующих скриптов не зависит от записей журнала резервного копирования, то их можно отключить с помощью флага трассировки 3226.

Безопасность

Для резервной копии базы данных свойству **TRUSTWORTHY** присваивается значение OFF. Начиная с версии SQL Server 2012 (11.x), параметры **PASSWORD** и **MEDIAPASSWORD** при создании резервных копий не поддерживаются. Все еще вы можете восстанавливать резервные копии, созданные с паролями.

Permissions

Разрешения BACKUP DATABASE и BACKUP LOG по умолчанию назначаются участникам предопределенной роли сервера **sysadmin** и предопределенным ролям базы данных **db_owner** и **db_backupoperator**.

Проблемы, связанные с владельцем и разрешениями у физических файлов на устройстве резервного копирования, могут помешать операции резервного копирования. Служба SQL Server выполняет операции чтения и записи на устройстве. Учетная запись, под которой работает служба SQL Server, должна иметь разрешения на запись на устройстве резервного копирования. Однако процедура sp_addumpdevice, добавляющая запись для устройства резервного копирования в системные таблицы, не проверяет разрешения на доступ к файлу. Проблемы с физическим файлом устройства резервного копирования могут не проявиться до тех пор, пока эта резервная копия не будет применена или не будет выполнена попытка восстановления.

Использование среды SQL Server Management Studio

Примечание

При создании задания резервного копирования с помощью среды SQL Server Management Studio вы можете сформировать соответствующий скрипт Transact-SQL **BACKUP**, нажав кнопку **Скрипт** и выбрав назначение скрипта.

1. После подключения к соответствующему экземпляру Microsoft Компонент SQL Server Database Engine в **обозревателе объектов** разверните дерево сервера.
2. Разверните узел **Базы данных** и выберите пользовательскую базу данных или разверните узел **Системные базы данных** и выберите системную базу данных.
3. Щелкните правой кнопкой мыши базу данных, резервную копию которой вы намерены создать, наведите указатель на пункт **Задачи** и выберите команду **Создать резервную копию...** .
4. В диалоговом окне **Резервное копирование базы данных** выбранная база данных приводится в раскрывающемся списке (ее можно изменить на любую другую базу данных на сервере).
5. В раскрывающемся списке **Тип резервной копии** выберите нужный вариант (по умолчанию выбран тип **Полная**).

Важно!

Перед тем как выполнять разностное резервное копирование или резервное копирование журналов транзакций, необходимо произвести по крайней мере одно полное резервное копирование базы данных.

6. В разделе **Компонент резервного копирования** выберите **База данных**.
7. В разделе **Назначение** проверьте расположение по умолчанию для файла резервной копии (в папке ../mssql/data).

Чтобы выбрать другое устройство, можно использовать раскрывающийся список **Создать резервную копию на**. Щелкните **Добавить**, чтобы добавить объекты резервного копирования и (или) целевые объекты. Резервный набор данных можно перераспределить между несколькими файлами, чтобы повысить скорость резервного копирования.

Чтобы удалить целевой объект резервного копирования, выберите его и щелкните **Удалить**. Чтобы просмотреть содержимое существующего целевого объекта резервного копирования, выберите его и щелкните **Содержимое**.

8. (Необязательно) Просмотрите другие доступные параметры на страницах **Параметры носителя** и **Параметры резервного копирования**.
9. Чтобы начать резервное копирование, нажмите кнопку **ОК**.
10. После успешного завершения резервного копирования щелкните **ОК**, чтобы закрыть диалоговое окно SQL Server Management Studio.

Дополнительные сведения

- После создания полной резервной копии базы данных можно создавать разностные резервные копии или резервные копии журналов транзакций.
- Также можно установить флажок **Резервная копия только для копирования**, чтобы создать резервную копию только для копирования. *Резервная копия только для копирования* — это резервная копия SQL Server, которая не зависит от обычной последовательности создания традиционных резервных копий SQL Server. Резервная копия только для копирования недоступна для типа резервной копии **Разностная**.
- При резервном копировании на URL-адрес параметр **Перезаписать носитель** на странице **Параметры носителя** недоступен.

Примеры

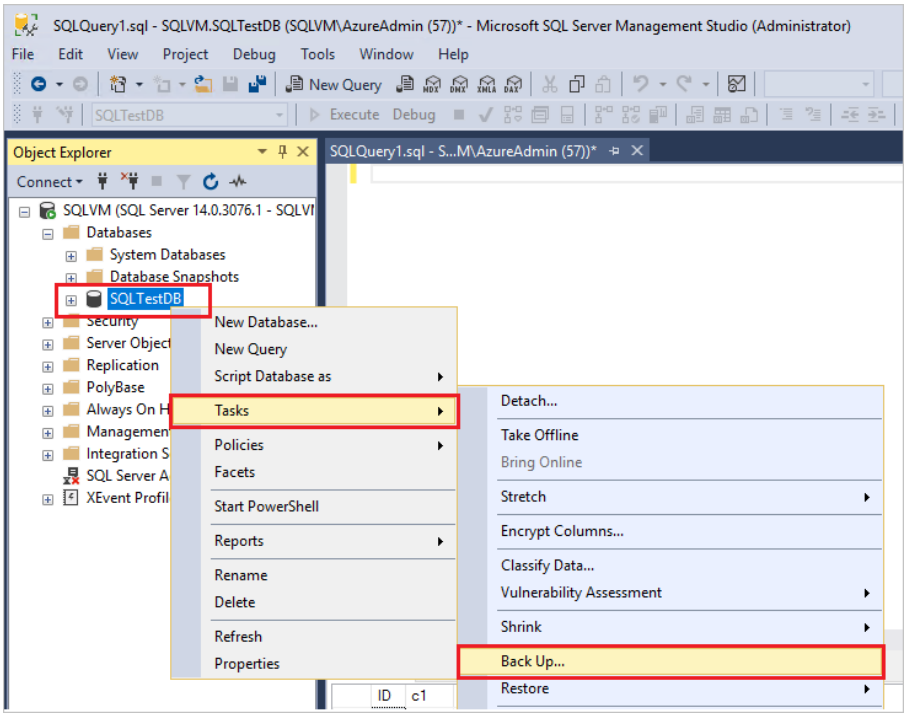
Для следующих примеров создайте тестовую базу данных со следующим кодом Transact-SQL:

```
USE [master]
GO

CREATE DATABASE [SQLTestDB]
GO

USE [SQLTestDB]
GO
CREATE TABLE SQLTest
  (
    ID INT NOT NULL PRIMARY KEY,
    c1 VARCHAR(100) NOT NULL,
    dt1 DATETIME NOT NULL DEFAULT getdate()
  );
GO

USE [SQLTestDB]
GO

INSERT INTO SQLTest (ID, c1) VALUES (1, 'test1')
INSERT INTO SQLTest (ID, c1) VALUES (2, 'test2')
INSERT INTO SQLTest (ID, c1) VALUES (3, 'test3')
INSERT INTO SQLTest (ID, c1) VALUES (4, 'test4')
INSERT INTO SQLTest (ID, c1) VALUES (5, 'test5')
GO

SELECT * FROM SQLTest
GO
```

A. Полное резервное копирование на диск в расположение по умолчанию

В этом примере база данных SQLTestDB будет заархивирована на диск в папку резервных копий по умолчанию.

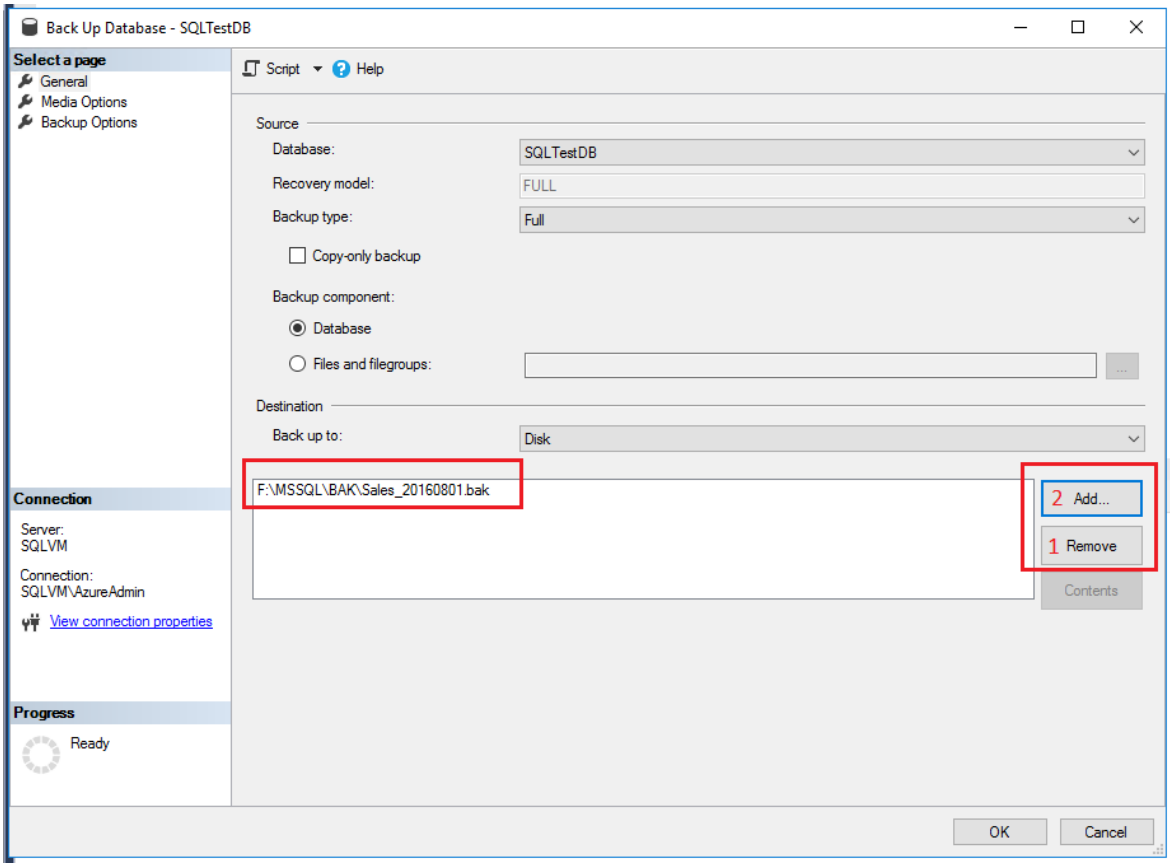
1. После подключения к соответствующему экземпляру Microsoft Компонент SQL Server Database Engine в **обозревателе объектов** разверните дерево сервера.
2. Разверните элемент **Базы данных**, щелкните SQLTestDB правой кнопкой мыши, наведите указатель на пункт **Задачи** и выберите действие **Создать резервную копию...** .
3. Щелкните **ОК**.
4. После успешного завершения резервного копирования щелкните **ОК**, чтобы закрыть диалоговое окно SQL Server Management Studio.



Б. Полное резервное копирование на диск в нестандартное расположение

В этом примере база данных SQLTestDB будет заархивирована на диск в выбранную вами папку.

1. После подключения к соответствующему экземпляру Microsoft Компонент SQL Server Database Engine в **обозревателе объектов** разверните дерево сервера.
2. Разверните элемент **Базы данных**, щелкните SQLTestDB правой кнопкой мыши, наведите указатель на пункт **Задачи** и выберите действие **Создать резервную копию...** .
3. На странице **Общие** в разделе **Назначение** выберите **Диск** в раскрывающемся списке **Создать резервную копию на:** .
4. Щелкайте элемент **Удалить**, пока не будут удалены все существующие файлы резервных копий.
5. Нажмите кнопку **Добавить**, чтобы открыть диалоговое окно **Выбор места расположения резервной копии**.
6. Введите допустимый путь и имя файла в текстовом поле **Имя файла** и используйте расширение **.bak**, чтобы упростить классификацию файла.
7. Щелкните **ОК**, а затем еще раз щелкните **ОК**, чтобы начать резервное копирование.
8. После успешного завершения резервного копирования щелкните **ОК**, чтобы закрыть диалоговое окно SQL Server Management Studio.



B. Создание зашифрованной резервной копии

В этом примере база данных SQLTestDB будет заархивирована с шифрованием в папку резервных копий по умолчанию.

1. После подключения к соответствующему экземпляру Microsoft Компонент SQL Server Database Engine в **обозревателе объектов** разверните дерево сервера.

2. Разверните узел **Базы данных** и узел **Системные базы данных**, щелкните правой кнопкой мыши базу данных master и выберите действие **Создать запрос**, чтобы открыть окно запроса с подключением к базе данных SQLTestDB.

3. Выполните приведенные ниже команды, чтобы создать **главный ключ базы данных** и **сертификат** в базе данных master.

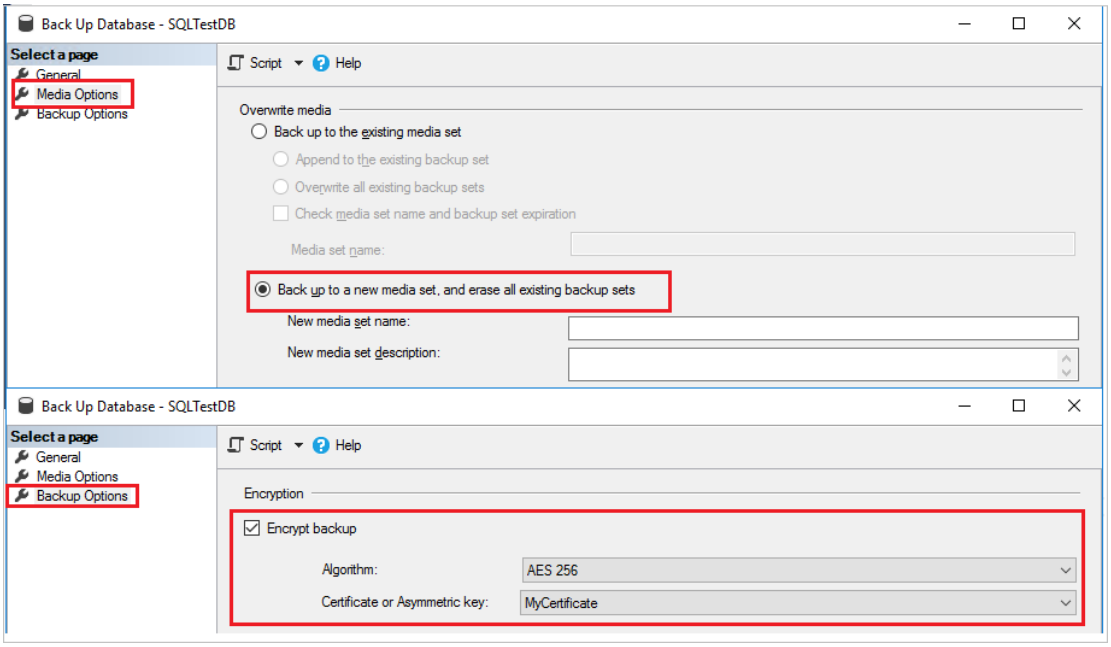
```
-- Create the master key
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '23987hxJ#KL95234nl0zBe';

-- If the master key already exists, open it in the same session that you create the certificate (see next step)
OPEN MASTER KEY DECRYPTION BY PASSWORD = '23987hxJ#KL95234nl0zBe'

-- Create the certificate encrypted by the master key
CREATE CERTIFICATE MyCertificate
WITH SUBJECT = 'Backup Cert', EXPIRY_DATE = '20201031';
```

4. В **обозревателе объектов** в узле **Базы данных** щелкните правой кнопкой мыши базу данных SQLTestDB, наведите указатель на пункт **Задачи** и выберите действие **Создать резервную копию...** .

5. На странице **Параметры носителя** в разделе **Перезапись носителя** выберите **Создать резервную копию в новом наборе носителей и удалить все существующие резервные наборы данных**.

6. На странице **Параметры резервного копирования** в разделе **Шифрование** установите флажок **Зашифровать резервную копию** .

7. В раскрывающемся списке "Алгоритм" выберите **AES 256**.
8. В раскрывающемся списке **Сертификат или асимметричный ключ** выберите MyCertificate.
9. Щелкните **ОК**.



Г. Резервное копирование в службу хранилища BLOB-объектов Azure

В приведенном ниже примере создается полная резервная копия базы данных SQLTestDB в службе "Хранилище BLOB-объектов Azure". В этом примере предполагается, что у вас уже есть учетная запись хранения с контейнером BLOB-объектов. В примере создается подписанный URL-адрес, и если у контейнера уже есть подписанный URL-адрес, операция завершится сбоем.

Если у вас нет контейнера BLOB-объектов Azure в учетной записи хранения, создайте его перед продолжением работы. Дополнительные сведения см. в статье Создание учетной записи хранения и разделе Создание контейнера.

1. После подключения к соответствующему экземпляру Microsoft Компонент SQL Server Database Engine в **обозревателе объектов** разверните дерево сервера.
2. Разверните элемент **Базы данных**, щелкните SQLTestDB правой кнопкой мыши, наведите указатель на пункт **Задачи** и выберите действие **Создать резервную копию...** .
3. На странице **Общие** в разделе **Назначение** выберите **URL-адрес** в раскрывающемся списке **Создать резервную копию на:** .
4. Нажмите кнопку **Добавить**, чтобы открыть диалоговое окно **Выбор места расположения резервной копии**.
5. Если ранее вы зарегистрировали контейнер службы хранилища Azure, который хотите использовать с SQL Server Management Studio, то выберите его. В противном случае щелкните **Создать контейнер**, чтобы зарегистрировать новый контейнер.
6. В диалоговом окне **Соединение с подпиской Майкрософт** войдите в свою учетную запись.
7. В текстовом поле с раскрывающимся списком **Выберите учетную запись хранения** выберите свою учетную запись хранения.
8. В текстовом поле с раскрывающимся списком **Выбрать контейнер BLOB-объектов** выберите контейнер больших двоичных объектов.
9. В поле календаря с раскрывающимся списком **Политика срока действия подписанных URL-адресов** выберите дату окончания срока действия для политики общего доступа, создаваемой в этом примере.

10. Щелкните **Создать учетные данные**, чтобы создать подписанный URL-адрес и учетные данные в SQL Server Management Studio.

11. Щелкните **ОК**, чтобы закрыть диалоговое окно **Соединение с подпиской Майкрософт**.

12. В текстовом поле **Файл резервной копии** при необходимости измените имя файла резервной копии.

13. Щелкните **ОК**, чтобы закрыть диалоговое окно **Выбор места назначения резервной копии**.

14. Чтобы начать резервное копирование, нажмите кнопку **ОК**.

15. После успешного завершения резервного копирования щелкните **ОК**, чтобы закрыть диалоговое окно SQL Server Management Studio.

Использование Transact-SQL

Создайте полную резервную копию базы данных, выполнив инструкцию BACKUP DATABASE для создания полной резервной копии базы данных и указав следующее:

- имя базы данных для создания резервной копии;
- устройство резервного копирования, на которое записывается полная резервная копия базы данных.

Базовая структура синтаксиса Transact-SQL для полного резервного копирования базы данных:

```
BACKUP DATABASE database TO backup_device [ , ...n ] [
WITH with_options [ , ...o ] ];
```

Параметр	Описание
database	База данных для резервного копирования.
backup_device [, ...*n*]	Указывает список от 1 до 64 устройств резервного копирования, используемых для создания резервной копии. Можно указать как физическое устройство резервного копирования, так и соответствующее логическое устройство, если оно уже определено. Для указания физического устройства резервного копирования используйте параметр DISK или TAPE. { DISK | TAPE } = *physical_backup_device_name* Дополнительные сведения см. в разделе Устройства резервного копирования (SQL Server).
WITH *with_options* [, ...*o*]	Используется для указания одного или нескольких параметров, *o*. Сведения о некоторых основных параметрах см. в пункте 2.

При необходимости укажите один параметр **WITH** или несколько. Здесь описываются некоторые основные параметры **WITH**.

Основные параметры **WITH** резервного набора данных:

- **{ COMPRESSION | NO_COMPRESSION }** : Только в версии SQL Server 2008 Enterprise и выше указано, выполняется ли команда backup compression для этой резервной копии, переопределяя значение по умолчанию на уровне сервера.
- **ENCRYPTION (ALGORITHM, SERVER CERTIFICATE | ASYMMETRIC KEY)** : Только для SQL Server 2014 и выше укажите используемый алгоритм шифрования, а также сертификат или асимметричный ключ для шифрования.
- **DESCRIPTION** = { ' *text* ' | *@ text_variable* }: Задает произвольное текстовое описание резервного набора данных. В этой строке может содержаться до 255 символов.
- **NAME** = **{ *имя_резервного_набора_данных* | *@ переменная_резервного_набора_данных* }** : Указывает имя резервного набора данных. Длина имени не может превышать 128 символов. Если имя не указано, оно остается пустым.

По умолчанию команда BACKUP добавляет резервную копию в существующий набор носителей, сохраняя существующие резервные наборы данных. Чтобы явно задать значение, используйте параметр NOINIT.

Чтобы отформатировать носитель резервной копии, используйте параметр **FORMAT**:

```
FORMAT [ , MEDIANAME = { media_name | @ media_name_variable } ]
[ , MEDIADESCRIPTION = { text | @ text_variable } ]
```

Используйте предложение **FORMAT** при первом обращении к носителю или при необходимости перезаписать все существующие данные. При необходимости назначьте новому носителю имя и описание.

Важно!

Будьте предельно осторожны, используя предложение **FORMAT** инструкции BACKUP, так как оно удаляет все резервные копии, сохраненные ранее на носителе резервных копий.

Примеры

Для следующих примеров создайте тестовую базу данных со следующим кодом Transact-SQL:

```
USE [master]
GO

CREATE DATABASE [SQLTestDB]
GO

USE [SQLTestDB]
GO
CREATE TABLE SQLTest (
  ID INT NOT NULL PRIMARY KEY,
  c1 VARCHAR(100) NOT NULL,
  dt1 DATETIME NOT NULL DEFAULT GETDATE()
)
GO

USE [SQLTestDB]
GO

INSERT INTO SQLTest (ID, c1) VALUES (1, 'test1')
INSERT INTO SQLTest (ID, c1) VALUES (2, 'test2')
INSERT INTO SQLTest (ID, c1) VALUES (3, 'test3')
INSERT INTO SQLTest (ID, c1) VALUES (4, 'test4')
INSERT INTO SQLTest (ID, c1) VALUES (5, 'test5')
GO

SELECT * FROM SQLTest
GO
```

A. Резервное копирование на дисковое устройство

В следующем примере производится резервное копирование всей базы данных SQLTestDB на диск и создание нового набора носителей с помощью параметра FORMAT .

```
USE SQLTestDB;
GO
BACKUP DATABASE SQLTestDB
TO DISK = 'c:\tmp\SQLTestDB.bak'
```

```
  WITH FORMAT,
    MEDIANAME = 'SQLServerBackups',
    NAME = 'Full Backup of SQLTestDB';
GO
```

Б. Резервное копирование на ленточное устройство

В следующем примере создается полная резервная копия базы данных SQLTestDB на ленте в дополнение к предыдущим резервными копиям.

```
USE SQLTestDB;
GO
BACKUP DATABASE SQLTestDB
  TO TAPE = '\\.\Tape0'
  WITH NOINIT,
    NAME = 'Full Backup of SQLTestDB';
GO
```

В. Резервное копирование на логическое ленточное устройство

В следующем примере создается логическое устройство резервного копирования для ленточного накопителя. Затем показано, как производится полное резервное копирование базы данных SQLTestDB на этот накопитель.

```
-- Create a logical backup device,
-- SQLTestDB_Bak_Tape, for tape device \\.\tape0.
USE master;
GO
EXEC sp_addumpdevice 'tape', 'SQLTestDB_Bak_Tape', '\\.\tape0'; USE SQLTestDB;
GO
BACKUP DATABASE SQLTestDB
  TO SQLTestDB_Bak_Tape
  WITH FORMAT,
    MEDIANAME = 'SQLTestDB_Bak_Tape',
    MEDIADESCRIPTION = '\\.\tape0',
    NAME = 'Full Backup of SQLTestDB';
GO
```

Использование PowerShell

Используйте командлет **Backup-SqlDatabase** . Чтобы явно указать, что это полная резервная копия базы данных, задайте параметр **-BackupAction** со значением **Database**, которое используется по умолчанию. Данный параметр является необязательным для полных резервных копий баз данных.

Примечание

Для этих примеров требуется модуль SqlServer. Чтобы определить, установлен ли он, выполните команду Get-Module -Name SqlServer. Чтобы установить его, выполните команду Install-Module -Name SqlServer в сеансе PowerShell с правами администратора.

Важно!

При открытии окна PowerShell из SQL Server Management Studio для подключения к установке SQL Server учетные данные можно опустить, так как для установки подключения между PowerShell и экземпляром SQL Server автоматически используются ваши учетные данные в SSMS.

Примеры

А. Полная резервная копия (локальная)

В следующем примере создается полная резервная копия базы данных <myDatabase> в заданном по умолчанию расположении резервного копирования на экземпляре

сервера Computer\Instance. Дополнительно в этом примере указывается параметр **-BackupAction Database**.

Полные примеры синтаксиса см. в документации по Backup-SqlDatabase.

```
$credential = Get-Credential

Backup-SqlDatabase -ServerInstance Computer[\Instance] -Database <myDatabase> -BackupAction Database -Credential $credential
```

Б. Полная резервная копия в Azure

В следующем примере создается полная резервная копия базы данных <myDatabase> в экземпляре <myServer> службы хранилища больших двоичных объектов Azure. Хранимая политика доступа была создана с правами на чтение, запись и составление списков. Учетные данные SQL Server, https://<myStorageAccount>.blob.core.windows.net/<myContainer>, были созданы с использованием подписанного URL-адреса, который связан с хранимой политикой доступа. Команда PowerShell использует параметр **BackupFile** для указания расположения (URL-адреса) и имени файла резервной копии.

```
$credential = Get-Credential
$container = 'https://<myStorageAccount>blob.core.windows.net/<myContainer>'
$fileName = '<myDatabase>.bak'
$server = '<myServer>'
$database = '<myDatabase>'
$backupFile = $container + '/' + $fileName

Backup-SqlDatabase -ServerInstance $server -Database $database -BackupFile $backupFile -Credential $credential
```

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями выполните резервное копирование созданной Вами ранее базы данных.
3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какое копирование является резервным?
2. Как можно осуществить резервное копирование?

Лабораторная работа №10
«ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ ИЗ РЕЗЕРВНОЙ КОПИИ»

Цель работы: получить теоретические знания и практические навыки выполнения восстановления базы данных из резервной копии

Теоретические сведения

Рассмотрим процесс восстановления полной резервной копии базы данных с использованием среды SQL Server Management Studio.

Важно!

Перед восстановлением базы данных по модели полного восстановления или восстановления с неполным протоколированием, возможно, необходимо будет выполнить резервное копирование активного журнала транзакций (который называется заключительным фрагментом журнала).

Чтобы восстановить зашифрованную базу данных, потребуется доступ к сертификату или асимметричному ключу, использовавшемуся для шифрования этой базы данных. Без сертификата или асимметричного ключа вы не сможете восстановить базу данных. Сохраняйте сертификат, который использовали для шифрования ключа шифрования базы данных, в течение всего периода хранения резервной копии. Если восстановить базу данных более старой версии до SQL Server 2019 (15.x), эта база данных будет автоматически обновлена до SQL Server 2019 (15.x). Это исключает возможность использования базы данных с более старой версией Компонент Database Engine. Но это относится к обновлению метаданных и не влияет на уровень совместимости базы данных. Если уровень совместимости пользовательской базы данных до обновления был 100 или выше, после обновления он останется таким же. Если уровень совместимости до обновления был 90, в обновленной базе данных он устанавливается в 100, что является минимально поддерживаемым уровнем совместимости в SQL Server 2019 (15.x).

Как правило, база данных сразу становится доступной. Однако если база данных SQL Server 2005 (9.x) содержит полнотекстовые индексы, то в процессе обновления будет произведен их импорт, сброс или перестроение в зависимости от установленного значения свойства сервера **Режим обновления полнотекстового каталога** . Если выбран режим обновления **Импортировать** или **Перестроить**, то полнотекстовые индексы во время обновления будут недоступны. В зависимости от объема индексируемых данных процесс импорта может занять несколько часов, а повторная сборка — до десяти раз дольше.

Если выбран режим обновления **Импорт**, а полнотекстовый каталог недоступен, то связанные с ним полнотекстовые индексы будут перестроены.

Примеры

А. Восстановление полной резервной копии базы данных

1. В **обозревателе объектов** подключитесь к экземпляру компонента Компонент SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**
3. Чтобы указать источник и расположение восстанавливаемых резервных наборов данных, используйте страницу **Общие** , раздел **Источник** . Выберите один из следующих вариантов.
   - **База данных**

Выберите из раскрывающегося списка базу данных для восстановления. Данный список содержит только базы данных, резервное копирование которых было выполнено в соответствии с журналом резервного копирования **msdb** .

Примечание

Если резервная копия была получена с другого сервера, на целевом сервере не будет журнала резервного копирования для указанной базы данных. В этом случае щелкните пункт **Устройство** , чтобы вручную указать файл или устройство для восстановления.

- **Устройство**

Нажмите кнопку обзора (**...**), после чего откроется диалоговое окно **Выбор устройств резервного копирования** .

- Диалоговое окно **Выбор устройств резервного копирования** .

Носитель данных резервной копии

Выберите тип носителя в раскрывающемся списке **Тип носителя данных резервной копии** . Примечание. Параметр **Лента** появляется только в случае, если на компьютере установлен ленточный накопитель, а параметр **Устройство резервного копирования** — только в случае, если имеется хотя бы одно устройство резервного копирования.

Добавление

В зависимости от типа носителя данных, выбранного в поле **Носитель резервной копии** , при нажатии кнопки **Добавить** открывается одно из следующих диалоговых окон. (Если список в поле со списком **Тип носителя резервной копии** заполнен, кнопка **Добавить** недоступна.)

Тип носителя данных	.	Описание
Файл	Локальный файл резервной копии	В данном диалоговом окне можно выбрать локальный файл из дерева или указать удаленный файл, используя его полное имя в формате UNC.
Устройство	Выбор устройства резервного копирования	В данном диалоговом окне из списка можно выбрать логические устройства резервного копирования, определенные на экземпляре сервера.
Лента	Выбор ленты с резервной копией	В данном диалоговом окне из списка можно выбрать ленточные накопители, физически подключенные к компьютеру, на котором запущен экземпляр SQL Server.
URL-адрес	Выберите расположение файла резервной копии	В этом диалоговом окне можно выбрать существующие учетные данные SQL Server или контейнер хранилища Azure, добавить новый контейнер хранилища Azure с подписанным URL-адресом или сформировать подписанный URL-адрес и учетные данные SQL Server для уже существующего контейнера хранилища.

Удалить

Удаляет один или несколько выбранных файлов, лент или устройств резервного копирования.

Содержание

Отображает содержимое носителя выбранного файла, ленты или устройства резервного копирования. Эта кнопка может не работать, если тип носителя — **URL-адрес**.

Тип носителя резервной копии

Перечисляет выбранные носители.

После добавления нужных устройств в списке **Носитель резервной копии** нажмите кнопку **ОК** для возвращения на страницу **Общие** .

В списке **Источник > Устройство > База данных** выберите имя базы данных, которую нужно восстановить.

Примечание

Данный список доступен, только если выбран параметр **Устройство** . Будут выбраны только те базы данных, резервные копии которых доступны на выбранном устройстве.

4. В разделе **Назначение** , в поле **База данных** автоматически появится имя базы данных для восстановления. Для изменения имени базы данных введите новое имя в окно **База данных** .

5. В поле **Восстановить до** оставьте значение по умолчанию **До последней выбранной резервной копии** или щелкните **Временную шкалу** , чтобы перейти в диалоговое окно **Временная шкала резервной копии** и выбрать вручную конкретное пороговое время восстановления.

6. В сетке **Резервные наборы данных для восстановления** выберите нужные резервные наборы. В этой сетке отображаются резервные копии, доступные в указанном месте. По умолчанию предлагается план восстановления. Чтобы переопределить предложенный план восстановления, можно изменить выбранные элементы в сетке. Выбор всех резервных копий, которые зависят от восстановления более ранних копий, отменяется автоматически, как только отменяется выбор более ранних копий.

7. При необходимости нажмите кнопку **Файлы** на панели **Выбор страницы** и перейдите в диалоговое окно **Файлы** . Отсюда можно восстановить базу данных в новое расположение, определив новое место восстановления для каждого файла в сетке **Восстановить файлы базы данных как** .

8. Чтобы просмотреть или выбрать дополнительные параметры, на странице **Параметры** на панели **Параметры восстановления** вберите любой из следующих параметров, если он подходит к данной ситуации.

1. Параметры **WITH** (необязательно):
   - **Перезаписать существующую базу данных (WITH REPLACE)**
   - **Сохранить параметры репликации (WITH KEEP_REPLICATION)**
   - **Ограничить доступ к восстановленной базе данных (WITH RESTRICTED_USER)**
2. Выберите параметр в поле **Состояние восстановления** . В данном окне определяется состояние базы данных после операции восстановления.
   - По умолчанию используется схема **RESTORE WITH RECOVERY**, которая выполняет откат незафиксированных транзакций и после завершения оставляет базу данных работоспособной. Дополнительные журналы транзакций восстановить невозможно. Выберите этот вариант, если выполняется восстановление сразу всех необходимых резервных копий.
   - Схема **RESTORE WITH NORECOVERY** оставляет базу данных в нерабочем состоянии и не выполняет откат незафиксированных транзакций. Можно восстановить дополнительные журналы транзакций. Вы не сможете использовать эту базу данных, пока она не будет восстановлена.
   - Схема **RESTORE WITH STANDBY** оставляет базу данных в режиме только для чтения. С помощью данного параметра можно отменить незафиксированные транзакции и сохранить отмененные действия в резервном файле, чтобы результаты восстановления можно было отменить.
3. **Создайте резервную копию заключительного фрагмента журнала до восстановления** Не для всех сценариев восстановления требуется резервная копия заключительного фрагмента журнала.

4. Если имеются активные соединения с базой данных, то операция восстановления может завершиться ошибкой. Проверьте окно **Закрыть существующие соединения** и убедитесь, что все активные соединения между Среда Management Studio и базой данных закрыты. Эта настройка переводит базу данных в однопользовательский режим перед началом процедуры восстановления, а затем возвращает в многопользовательский режим после ее завершения.

5. Установите флажок **Выдавать запрос перед восстановлением каждой резервной копии** , если хотите отследить каждую операцию восстановления. Это не требуется, если не нужно наблюдать за состоянием операции восстановления для базы данных большого объема.

9. Нажмите кнопку **ОК**.

Б. Восстановление более ранней резервной копии диска поверх существующей базы данных

В следующем примере восстанавливается более ранняя резервная копия диска из базы данных Sales и перезаписывается существующая база данных Sales.

1. В **обозревателе объектов** подключитесь к экземпляру компонента Компонент SQL Server Database Engine и разверните его.

2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**

3. На странице **Общие** выберите пункт **Устройство** в разделе **Источник** .

4. Нажмите кнопку обзора (**...**), после чего откроется диалоговое окно **Выбор устройств резервного копирования** . Щелкните **Добавить** и перейдите к резервной копии. Щелкните **ОК**, когда выберите один или несколько файлов резервной копии диска.

5. Нажмите кнопку **ОК** , чтобы вернуться на страницу **Общие** .

6. Нажмите кнопку **Параметры** на панели **Выбрать страницу** .

7. в разделе **Параметры восстановления** установите флажок **Перезаписать существующую базу данных (WITH REPLACE)** .

Примечание

Если этот параметр не выбран, может отобразиться следующее сообщение об ошибке: "System.Data.SqlClient.SqlError: резервный набор данных содержит резервную копию базы данных, отличающуюся от существующей базы данных Sales. (Microsoft.SqlServer.SmoExtended)"

8. В разделе **Резервная копия заключительного фрагмента журнала** снимите флажок **Делать резервную копию заключительного фрагмента журнала перед восстановлением**.

Примечание

Не для всех сценариев восстановления требуется резервная копия заключительного фрагмента журнала. Резервная копия заключительного фрагмента журнала не нужна, если точка восстановления содержится в более ранней резервной копии журнала. Кроме того, резервная копия заключительного фрагмента журнала не требуется при перемещении или замещении (перезаписи) базы данных, при котором не нужно восстанавливать ее на определенный момент времени после создания ее последней резервной копии.

Этот параметр недоступен для баз данных при использовании ПРОСТОЙ модели восстановления.

9. В разделе **Соединения с сервером** установите флажок **Закрыть существующие подключения к целевой базе данных**.

Примечание

Если вы не выберете этот параметр, может отобразиться следующее сообщение об ошибке: "System.Data.SqlClient.SqlError: Не удалось получить монопольный доступ, так как база данных используется. (Microsoft.SqlServer.SmoExtended)"

10. Нажмите кнопку **ОК**.

В. Восстановление более ранней резервной копии диска с новым именем базы данных при условии, что исходная база данных по-прежнему существует

В следующем примере восстанавливается более ранняя резервная копия диска из базы данных Sales и создается новая база данных с именем SalesTest. При этом исходная база данных, Sales, все еще существует на сервере.

1. В **обозревателе объектов** подключитесь к экземпляру компонента Компонент SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**
3. На странице **Общие** выберите пункт **Устройство** в разделе **Источник** .
4. Нажмите кнопку обзора (**...**), после чего откроется диалоговое окно **Выбор устройств резервного копирования** . Щелкните **Добавить** и перейдите к резервной копии. Щелкните **ОК**, когда выберите один или несколько файлов резервной копии диска.
5. Нажмите кнопку **ОК** , чтобы вернуться на страницу **Общие** .
6. В разделе **Назначение** , в поле **База данных** автоматически появится имя базы данных для восстановления. Для изменения имени базы данных введите новое имя в окно **База данных** .
7. Нажмите кнопку **Параметры** на панели **Выбрать страницу** .
8. В разделе **Резервная копия заключительного фрагмента журнала** снимите флажок **Делать резервную копию заключительного фрагмента журнала перед восстановлением**.

Важно!

Если оставить этот флажок установленным, существующая база данных Salesсменит состояние на состояние восстановления.

9. Нажмите кнопку **ОК**.

Примечание

Если вы получаете следующее сообщение об ошибке: "System.Data.SqlClient.SqlError: резервная копия заключительного фрагмента журнала для базы данных "Sales" не создана. Если журнал содержит работу, потеря которой нежелательна, создайте резервную копию с помощью инструкции BACKUP LOG WITH NORECOVERY.

Скорее всего, вы не ввели название новой базы данных из шага 6 выше. Восстановление обычно не допускает случайной перезаписи базы данных другой базой данных. Если указанная в инструкции RESTORE база данных уже существует на текущем сервере, а идентификатор GUID семейства для заданной базы данных отличается от идентификатора GUID семейства для базы данных, записанного в резервном наборе данных, то ее восстановление не будет выполнено. Это является важной защитной мерой.

Г. Восстановление до точки во времени

В следующем примере база данных восстанавливается в состояние на 1:23:17 PM``May 30, 2016 и демонстрируется операция восстановления, использующая несколько резервных копий журналов. Указанная база данных не существует на сервере.

1. В **обозревателе объектов** подключитесь к экземпляру компонента Компонент SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**
3. На странице **Общие** выберите пункт **Устройство** в разделе **Источник** .
4. Нажмите кнопку обзора (**...**), после чего откроется диалоговое окно **Выбор устройств резервного копирования** . Щелкните **Добавить** и перейдите к полной

резервной копии и всем соответствующим резервным копиям журнала транзакций. Нажмите кнопку **ОК** , выбрав файлы резервной копии диска.

5. Нажмите кнопку **ОК** , чтобы вернуться на страницу **Общие** .
6. В разделе **Назначение** щелкните **Временная шкала** , чтобы получить доступ к диалоговому окну **Временная шкала резервного копирования** и вручную выбрать момент остановки восстановления.
7. Выберите **Указанные дата и время**.
8. В раскрывающемся списке **Интервал временной шкалы** поменяйте значение на **Час** (необязательно).
9. Переместите ползунок в нужное время.
10. Нажмите кнопку **ОК**, чтобы вернуться на страницу "Общие".
11. Нажмите кнопку **ОК**.

Д. Восстановление резервной копии с помощью службы хранилища Microsoft Azure

Общие шаги

В двух примерах ниже выполняется восстановление базы данных Sales из резервной копии, расположенной в службе хранилища Microsoft Azure. Имя учетной записи хранилища — mystorageaccount. Контейнер называется myfirstcontainer. Для краткости первые шесть шагов перечислены здесь однократно, а все примеры начинаются с **шага 7**.

1. В **обозревателе объектов** подключитесь к экземпляру компонента SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**
3. На странице **Общие** выберите пункт **Устройство** в разделе **Источник** .
4. Нажмите кнопку обзора (...), после чего откроется диалоговое окно **Выбор устройств резервного копирования** .
5. Выберите **URL-адрес** в раскрывающемся списке **Тип носителя резервной копии:** .
6. Нажмите кнопку **Добавить**, чтобы открыть диалоговое окно **Выберите расположение файла резервной копии**.

E1. Восстановление чередующейся резервной копии поверх существующей базы данных при наличии подписанного URL-адреса.

Хранимая политика доступа была создана с правами на чтение, запись, удаление и составление списков. Подписанный URL-адрес, связанный с хранимой политикой доступа, был создан для контейнера https://mystorageaccount.blob.core.windows.net/myfirstcontainer. Шаги, в основном, одинаковы, если учетные данные SQL Server уже существуют. База данных Sales в настоящее время существует на сервере. Файлы резервной копии — Sales_stripe1of2_20160601.bak и Sales_stripe2of2_20160601.bak.

1. Выберите https://mystorageaccount.blob.core.windows.net/myfirstcontainer из раскрывающегося списка **Контейнер хранилища Azure:** , если учетные данные SQL Server уже существуют. В противном случае введите имя контейнера вручную https://mystorageaccount.blob.core.windows.net/myfirstcontainer.
2. Введите подписанный URL-адрес в поле форматированного текста **Подписанный URL-адрес:** .
3. Нажмите кнопку **ОК** , чтобы открыть диалоговое окно **Поиск файла резервной копии в Microsoft Azure** .
4. Разверните узел **Контейнеры** и перейдите к https://mystorageaccount.blob.core.windows.net/myfirstcontainer.
5. Удерживая клавишу ctrl, выберите файлы Sales_stripe1of2_20160601.bak и Sales_stripe2of2_20160601.bak.

6. Нажмите кнопку **ОК**.
7. Нажмите кнопку **ОК** , чтобы вернуться на страницу **Общие** .
8. Нажмите кнопку **Параметры** на панели **Выбрать страницу** .
9. в разделе **Параметры восстановления** установите флажок **Перезаписать существующую базу данных (WITH REPLACE)** .
10. В разделе **Резервная копия заключительного фрагмента журнала** снимите флажок **Делать резервную копию заключительного фрагмента журнала перед восстановлением**.
11. В разделе **Соединения с сервером** установите флажок **Закрыть существующие подключения к целевой базе данных**.
12. Нажмите кнопку **ОК**.

E2. Подписанный URL-адрес не существует.

В нашем примере база данных Sales не существует на сервере.

1. Щелкните **Добавить** , чтобы открыть диалоговое окно **Соединение с подпиской Microsoft** .
2. Выполните все действия в диалоговом окне **Соединение с подпиской Microsoft** и нажмите кнопку **ОК** , чтобы вернуться в диалоговое окно **Выбор расположения файла резервной копии** .
3. Нажмите кнопку **ОК** в диалоговом окне **Выбор расположения файла резервной копии** , чтобы открыть диалоговое окно **Поиск файла резервной копии в Microsoft Azure** .
4. Разверните узел **Контейнеры** и перейдите к https://mystorageaccount.blob.core.windows.net/myfirstcontainer.
5. Выберите файл резервной копии и нажмите кнопку **ОК**.
6. Нажмите кнопку **ОК** , чтобы вернуться на страницу **Общие** .
7. Нажмите кнопку **ОК**.

E. Восстановление локальной резервной копии в хранилище Microsoft Azure (URL)

База данных Sales будет восстановлена в контейнер хранилища Microsoft Azure https://mystorageaccount.blob.core.windows.net/myfirstcontainer из резервной копии, расположенной по адресу E:\MSSQL\BAK. Учетные данные SQL Server для контейнера Azure уже созданы. Учетные данные SQL Server для целевого контейнера уже должны быть созданы, поскольку создать их, выполнив задачу **Восстановить** , невозможно. База данных Sales в настоящее время не существует на сервере.

1. В **обозревателе объектов** подключитесь к экземпляру компонента SQL Server Database Engine и разверните его.
2. Щелкните правой кнопкой мыши узел **Базы данных** и выберите команду **Восстановить базу данных...**
3. На странице **Общие** выберите пункт **Устройство** в разделе **Источник** .
4. Нажмите кнопку обзора (...), после чего откроется диалоговое окно **Выбор устройств резервного копирования** .
5. Выберите **Файл** в раскрывающемся списке **Тип носителя резервной копии:** .
6. Нажмите кнопку **Добавить** , откроется диалоговое окно **Поиск файла резервной копии** .
7. Перейдите к E:\MSSQL\BAK, выберите файл резервной копии и нажмите кнопку **ОК**.
8. Нажмите кнопку **ОК** , чтобы вернуться на страницу **Общие** .
9. На панели **Выбор страницы** нажмите кнопку **Файлы** .
10. Установите флажок **Переместить все файлы в папку**.

11. Укажите контейнер, https://mystorageaccount.blob.core.windows.net/myfirstcontainer, в текстовых полях **Папка файла данных:** и **Папка файлов журнала:** .

12. Нажмите кнопку **ОК**.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями выполните восстановление базы данных из резервной копии, созданной в лабораторной работе №9.
3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какими способами восстановить базу данных их резервной копии?
2. Какие инструменты можно использовать для резервного копирования и восстановления базы данных?

Лабораторная работа №11
«РЕАЛИЗАЦИЯ ДОСТУПА ПОЛЬЗОВАТЕЛЕЙ К БАЗЕ ДАННЫХ»

Цель работы: получить теоретические знания и практические навыки организации и реализации доступа пользователей к базе данных

Теоретические сведения

1. Настройка разрешений для объектов базы данных

Предоставление пользователю доступа к базе данных включает три шага. Вначале создается имя входа. Имя входа дает пользователю возможность подключиться к компоненту Компонент SQL Server Database Engine. Затем имя входа настраивается как пользователь в заданной базе данных. Наконец, предоставляются пользовательские разрешения на объекты базы данных. В этом занятии показаны все три шага, а также создание представления и хранимой процедуры в виде объекта.

Предварительные требования

Для работы с этим руководством необходима среда SQL Server Management Studio и доступ к экземпляру SQL Server.

- Установите SQL Server Management Studio.

Если у вас нет доступа к экземпляру SQL Server, выберите свою платформу в следующих ссылках. При выборе проверки подлинности SQL используйте учетные данные SQL Server.

Создает вход

Чтобы получить доступ к компоненту Компонент Database Engine, необходимо иметь имя входа. Имя входа может идентифицировать пользователя как учетную запись Windows или как члена группы Windows, или имя входа может быть именем входа SQL Server , которое существует только в SQL Server. При возможности используйте проверку подлинности Windows.

По умолчанию администраторы компьютера имеют полный доступ к SQL Server. Для этого занятия нужно иметь пользователя с меньшим правом доступа; следовательно, вы создадите новую локальную учетную запись проверки подлинности Windows на компьютере. Чтобы сделать это, нужно быть администратором на своем компьютере. После этого нужно предоставить новому пользователю доступ к SQL Server.

Создание учетной записи Windows

1. Нажмите кнопку **Пуск**, выберите **Выполнить**, в поле **Открыть** введите **%SystemRoot%\system32\compmgmt.msc /s** и нажмите кнопку **OK** , чтобы открыть программу "Управление компьютером".
2. В пункте **Служебные программы** откройте **Локальные пользователи и группы**, щелкните правой кнопкой мыши элемент **Пользователи** и выберите пункт **Новый пользователь**.
3. В поле **Имя пользователя** введите **Mary**.
4. В полях **Пароль** и **Подтверждение пароля** введите надежный пароль и нажмите кнопку **Создать** , чтобы создать нового локального пользователя Windows.

Создание имени для входа SQL

В окне редактора запросов среды SQL Server Management Studioввведите и выполните следующий исходный код, заменив computer_name на имя компьютера. FROM WINDOWS указывает, что Windows проверит подлинность пользователя. Необязательный аргумент DEFAULT_DATABASE соединяет Mary с базой данных TestData , если только в

ее строке соединения не указана другая база данных. Эта инструкция рассматривает точку с запятой в виде необязательного завершения инструкции языка Transact-SQL .

```
CREATE LOGIN [computer_name\Mary]
   FROM WINDOWS
   WITH DEFAULT_DATABASE = [TestData];
GO
```

Этим авторизируется имя пользователя Mary, проверенное компьютером, чтобы получить доступ к экземпляру SQL Server. Если на компьютере находится более одного экземпляра SQL Server , нужно создать имя входа для каждого экземпляра, к которому Mary должна иметь доступ.

Примечание

Поскольку Mary не является доменной учетной записью, это имя пользователя может быть принято только на данном компьютере.

Предоставление доступа к базе данных

Теперь Mary имеет доступ к данному экземпляру SQL Server, но не имеет разрешения на доступ к базе данных. У нее даже нет доступа к своей базе данных по умолчанию **TestData** , пока вы не авторизируете ее в качестве пользователя базы данных.

Чтобы предоставить Mary доступ, переключитесь на базу данных **TestData** и при помощи инструкции CREATE USER сопоставьте ее имя входа с именем пользователя «Mary».

Создание пользователя в базе данных

Введите и выполните следующие инструкции (заменяя computer_name на имя компьютера), чтобы предоставить пользователю Mary доступ к базе данных TestData .

```
USE [TestData];
GO

CREATE USER [Mary] FOR LOGIN [computer_name\Mary];
GO
```

Теперь пользователь Mary имеет доступ к SQL Server и к базе данных TestData .

Создание представлений и хранимых процедур

Будучи администратором, можно выполнять инструкцию SELECT из таблицы **Products** и представления **vw_Names**, а также выполнять процедуру **pr_Names**; однако Мэри всего этого не может. Чтобы предоставить Mary необходимые разрешения, воспользуйтесь инструкцией GRANT.

Предоставление разрешений на хранимые процедуры

Выполните следующую инструкцию, чтобы предоставить Mary разрешение на EXECUTE для хранимой процедуры pr_Names .

```
GRANT EXECUTE ON pr_Names TO Mary;
GO
```

В данном сценарии Mary имеет доступ только к таблице **Products** посредством хранимой процедуры. Если Mary нужно выполнять инструкцию SELECT к представлению, нужно выполнить инструкцию GRANT SELECT ON vw_Names TO Mary. Чтобы удалить доступ к объектам базы данных, воспользуйтесь инструкцией REVOKE.

Примечание

Если таблицей, представлением или хранимой процедурой не владеет та же схема, процесс предоставления прав становится более сложным.

Об инструкции GRANT

Нужно иметь разрешение на EXECUTE, чтобы выполнить хранимую процедуру. Нужно иметь разрешения на SELECT, INSERT, UPDATE и DELETE, чтобы получить доступ к данным и изменять их. Инструкция GRANT также используется для других разрешений, например для разрешений на создание таблиц.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями выполните установку прав доступа пользователей к созданной Вами ранее базе данных.
3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как Вы понимаете «права доступа к базе данных»?
2. Какие права доступа к базе данных можно установить?

Лабораторная работа №12
«МОНИТОРИНГ БЕЗОПАСНОСТИ РАБОТЫ С БАЗАМИ ДАННЫХ»

Цель работы: получить теоретические знания и практические навыки мониторинга безопасности работы с базами данных.

Теоретические сведения

1. Обеспечение безопасности SQL Server

Защиту SQL Server можно рассматривать как последовательность шагов, затрагивающих четыре области: платформу, проверку подлинности, объекты (включая данные) и приложения, получающие доступ к системе. В приведенных ниже разделах описано создание и реализация эффективного плана обеспечения безопасности.

Безопасность платформы и сети

Платформа для SQL Server включает в себя физическое оборудование и сетевые компьютеры, с помощью которых клиенты соединяются с серверами базы данных, а также двоичные файлы, применяемые для обработки запросов базы данных.

Физическая безопасность

Рекомендуется строго ограничивать доступ к физическим серверам и компонентам оборудования. Например, оборудование сервера базы данных и сетевые устройства должны находиться в закрытых охраняемых помещениях. Доступ к резервным носителям также следует ограничить. Для этого их рекомендуется хранить в отдельных охраняемых помещениях.

Реализация физической сетевой безопасности начинается с запрета доступа неавторизованных пользователей к сети.

Безопасность операционной системы

В состав пакетов обновления и отдельных обновлений для операционной системы входят важные дополнения, позволяющие усилить безопасность. Все обновления для операционной системы необходимо устанавливать только после их тестирования с приложениями базы данных.

Кроме того, эффективную безопасность можно реализовать с помощью брандмауэров. Брандмауэр, распределяющий или ограничивающий сетевой трафик, можно настроить в соответствии с корпоративной политикой информационной безопасности. Использование брандмауэра повышает безопасность на уровне операционной системы, обеспечивая узкую область, на которой можно сосредоточить меры безопасности.

Уменьшение контактной зоны является мерой безопасности, предполагающей остановку или отключение неиспользуемых компонентов. Уменьшение контактной зоны повышает уровень безопасности за счет уменьшения числа возможных способов атаковать систему. Важную роль в ограничении контактной зоны SQL Server играет запуск необходимых служб по принципу «минимума прав доступа», согласно которому службам и пользователям предоставляются только необходимые для работы права.

Если в системе SQL Server используются службы IIS, необходимы дополнительные действия для обеспечения безопасности контактной зоны платформы.

Безопасность файлов операционной системы SQL Server

SQL Server использует файлы операционной системы для работы и хранения данных. Оптимальным решением для обеспечения безопасности файлов будет ограничение доступа к ним.

SQL Server позволяют повысить безопасность.

С помощью приведенного ниже скрипта можно определить установленный в системе пакет обновления.

```
SELECT CONVERT(char(20), SERVERPROPERTY('productlevel'));
```

Безопасность участников и объектов базы данных

Участники — это отдельные пользователи, группы и процессы, которым предоставлен доступ к ресурсам SQL Server. Защищаемые объекты — это сервер, база данных и объекты, которые содержит база данных. У каждого из них существует набор разрешений, с помощью которых можно уменьшить контактную зону SQL Server .

Шифрование и сертификаты

Шифрование не решает проблемы управления доступом. Однако оно повышает безопасность, ограничивая потерю данных даже в тех редких случаях, когда средства управления доступом удается обойти. Например, если главный компьютер, на котором установлена база данных, был настроен неправильно, и злонамеренный пользователь смог получить конфиденциальные данные (например, номера кредитных карточек), то украденная информация будет бесполезна, если она была предварительно зашифрована.

Сертификаты — это совместно используемые на двух серверах программные «ключи», которые позволяют обеспечить безопасную передачу данных с помощью надежной проверки подлинности. SQL Server позволяет создавать и использовать сертификаты для повышения безопасности объектов и соединений.

Безопасность приложений

Клиентские программы

SQL Server рекомендуется разрабатывать защищенные клиентские приложения.

Управление приложениями в Защитнике Windows (WDAC)

Управление приложениями в Защитнике Windows (WDAC) предотвращает попытки несанкционированного выполнения кода. WDAC является эффективным способом устранения угрозы со стороны вредоносных программ с исполняемыми файлами.

Средства, программы, представления и функции безопасности SQL Server

SQL Server предусмотрены средства, программы, представления и функции, которые используются для настройки и управления безопасностью.

Средства и программы безопасности SQL Server

Следующая таблица содержит сведения о средствах и программах SQL Server , с помощью которых можно настраивать и администрировать безопасность.

Сведения о	См.
Соединение с SQL Server	Использование среды SQL Server Management Studio
Соединение с SQL Server и запуск запросов из командной строки	Программа sqlcmd
Настройка сети и управление SQL Server	Диспетчер конфигурации SQL Server
Включение и отключение компонентов с помощью средства управления на основе политики	Администрирование серверов с помощью управления на основе политик
Управление симметричными ключами для сервера отчетов	Программа rskeymgmt (SSRS)

Представления каталога и функции безопасности SQL Server

В компоненте Компонент Database Engine сведения о безопасности доступны через несколько представлений и функций, оптимизированных для наибольшей производительности и полезности. Следующая таблица содержит сведения о представлениях и функциях безопасности.

Сведения о	См.
SQL Server представления каталога безопасности, которые возвращают сведения о разрешениях, участниках, ролях и других сущностях уровня базы данных и сервера. Кроме того, существуют представления каталога, содержащие сведения о ключах шифрования, сертификатах и учетных данных.	Представления каталога безопасности (Transact-SQL)
SQL Server , которые возвращают сведения о текущем пользователе, разрешениях и схемах.	Функции безопасности (Transact-SQL)
SQL Server .	Динамические представления управления и функции, связанные безопасностью (Transact-SQL)

2. Рекомендации по обеспечению безопасности автономных баз данных

Угрозы, связанные с пользователями

Пользователям автономной базы данных, имеющим разрешение **ALTER ANY USER**, таким как члены предопределенных ролей базы данных **db_owner** и **db_accessadmin**, может быть предоставлен доступ к базе данных без ведома или разрешения администратора SQL Server. Предоставление пользователям доступа к автономной базе данных увеличивает контактную зону возможной атаки на всем экземпляре SQL Server . Администраторы должны понимать это делегирования контроля доступа и должны быть очень осторожными при предоставлении пользователям в автономной базе данных разрешения **ALTER ANY USER** . Все владельцы базы данных обладают разрешением **ALTER ANY USER** . SQL Server должны периодически проводить аудит пользователей в автономной базе данных.

Доступ к другим базам данных с использованием гостевой учетной записи

Владельцы и пользователи базы данных, имеющие разрешение **ALTER ANY USER** , могут создавать пользователей автономной базы данных. После соединения с автономной базой данных в экземпляре SQL Serverпользователь автономной базы данных имеет доступ к другим базам данных в Компонент Database Engine, если в других базах данных включена **гостевая** учетная запись.

Создание повторяющегося пользователя в другой базе данных

Для некоторых приложений может оказаться необходимым, чтобы пользователь имел доступ к более чем одной базе данных. Это можно сделать путем создания идентичных пользователей автономной базы данных в каждой базе данных. При создании второй учетной записи пользователя, защищенной паролем, используйте параметр идентификатора безопасности. В следующем примере создается два идентичных пользователя в двух базах данных.

```
USE DB1;
GO
CREATE USER Carlo WITH PASSWORD = '<strong password>';
-- Return the SID of the user
```

```
SELECT SID FROM sys.database_principals WHERE name = 'Carlo';

-- Change to the second database
USE DB2;
GO
CREATE USER Carlo WITH PASSWORD = '<same password>', SID = <SID from DB1>;
GO
```

Для выполнения межбазового запроса необходимо установить параметр **TRUSTWORTHY** в вызывающей базе данных. Например, если определенный выше пользователь (Carlo) находится в базе данных DB1 для выполнения запроса SELECT * FROM db2.dbo.Table1; , то параметр **TRUSTWORTHY** должен быть включен для базы данных DB1. Чтобы включить параметр **TRUSTWORTHY** , выполните следующий код:

```
ALTER DATABASE DB1 SET TRUSTWORTHY ON;
```

Создание пользователя с повторяющимся именем входа

Если при создании пользователя автономной базы данных с паролем в качестве имени пользователя было использовано имя входа SQL Server и пользователь с именем входа SQL Server в запросе на подключение указывает автономную базу данных в качестве исходного каталога, то пользователь с именем входа SQL Server подключиться не сможет. Соединение будет оцениваться как пользователь (участник) автономной базы данных с паролем автономной базы данных, а не пользователь на основе имени входа SQL Server . Это может вызвать намеренный или случайный отказ в обслуживании для имени входа SQL Server .

- Согласно рекомендациям, члены предопределенной роли сервера **sysadmin** должны всегда рассматривать соединение без использования параметра исходного каталога. Это вызывает подключение имени входа к базе данных master и предотвращает попытки владельца базы данных неправильно использовать попытку входа. Затем администратор может выполнить переключение на автономную базу данных с помощью инструкции **USE** <*database*> . Можно также установить в качестве базы данных по умолчанию автономную базу данных, которая выполняет вход в базу данных **master**, а затем переносит вход на автономную базу данных.
- Рекомендуется не создавать пользователей автономной базы данных с паролями, имена которых совпадают с именами входа SQL Server .
- Если существует дублированное имя входа, выполните подключение к базе данных **master** без указания исходного каталога, а затем выполните команду **USE** , чтобы переключиться в автономную базу данных.
- При наличии автономных баз данных пользователи баз данных, не являющихся автономными, должны подключаться к Компонент Database Engine , указывая в качестве исходного каталога имя неавтономной базы данных или вообще не указывая исходный каталог. Это предотвращает соединение с автономной базой данных, которая находится под меньшим контролем администратора компонента Компонент Database Engine .

Увеличение доступа путем изменения состояния включения базы данных

Имена входа, которые имеют разрешение **ALTER ANY DATABASE** , такие как члены предопределенной роли сервера **dbcreator** и пользователи неавтономной базы данных, с разрешением **CONTROL DATABASE** , такие как члены предопределенной роли базы данных **db_owner** , могут изменять параметр включения базы данных. При изменении параметра включения базы данных с **NONE** на **PARTIAL** или **FULL** пользователю может быть предоставлен доступ путем создания пользователей автономной базы данных с

паролями. В результате можно предоставить доступ без ведома или согласия администраторов SQL Server . Чтобы предотвратить появление автономных баз данных, присвойте параметру **проверка подлинности автономной базы данных** компонента Компонент Database Engine значение 0. Для предотвращения подключения пользователей автономной базы данных с паролями к выбранным автономным базам данных, используйте триггеры входа для отмены попыток входа пользователей автономной базы данных с паролями.

Присоединение автономной базы данных

При присоединении автономной базы данных администратор может предоставить нежелательным пользователям доступ к экземпляру компонента Компонент Database Engine. Для предотвращения такого риска администратор может перевести базу данных в состояние "в сети" в режиме **RESTRICTED USER** , в результате чего пользователи автономной базы данных с паролями не смогут пройти проверку подлинности. Доступ к компоненту Компонент Database Enginесмогут получить только участники, авторизованные через имена входа.

Пользователи создаются с учетом требований к паролю, действительных в момент их создания, и при присоединении базы данных повторная проверка паролей не производится. При присоединении автономной базы данных, допускающей простые пароли к системе с более строгой политикой паролей, администратор может разрешить пароли, которые не соответствуют текущей политике паролей присоединяемого компонента Компонент Database Engine. Администраторы могут запретить сохранение простых паролей с помощью требования переустановки всех паролей в присоединенной базе данных.

Политики паролей

Можно потребовать, чтобы пароли в базе данных были надежными, но защитить их надежными политиками паролей нельзя. По возможности используйте проверку подлинности Windows, чтобы использовать преимущества более сложных политик паролей, доступных в Windows.

Проверка подлинности Kerberos

Пользователи автономной базы данных с паролями не могут использовать проверку подлинности Kerberos. По возможности используйте проверку подлинности Windows, чтобы использовать преимущества таких возможностей Windows, как Kerberos.

Атака вне сети перебором по словарю

Хэши паролей для пользователей автономной базы данных с паролями хранятся в автономной базе данных. Любое лицо с доступом к базе данных может выполнить атаку перебором по словарю против пользователей автономной базы данных с паролями в системе без аудита. Чтобы устранить эту угрозу, ограничьте доступ к файлам базы данных или разрешите подключение к автономным базам данных только с использованием проверки подлинности Windows.

Экранирование автономной базы данных

Если база данных является частично автономной, то администраторы SQL Server должны периодически выполнять аудит возможностей пользователей и модулей в автономных базах данных.

Отказ в обслуживании через параметр AUTO_CLOSE

Не настраивайте автономную базу данных на автоматическое закрытие. Если закрыть базу данных, ее открытие для проверки подлинности пользователя будет связано с

потреблением дополнительных ресурсов, что может стать объектом атаки типа «отказ в обслуживании».

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями проведите мониторинг безопасности работы к созданной Вами ранее базе данных.
3. Составьте рекомендации по повышению безопасности работы с базой данных. Реализуйте из.
4. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем заключается обеспечение безопасности SQL Server?
2. Какие существуют рекомендации по обеспечению безопасности автономных баз данных?

Лабораторная работа №13
«УСТАНОВКА ПРИОРИТЕТОВ»

Цель работы: получить теоретические знания и практические навыки установки приоритетов.

Теоретические сведения

CREATE BROKER PRIORITY (Transact-SQL) - задает уровень приоритета и набор критериев для определения диалогов компонента. Компонент Service Broker, которым нужно назначить уровень приоритета. Уровень приоритета назначается любой конечной точке диалога, использующей то же сочетание контрактов и служб, которое указано в приоритете диалога.

Приоритеты должны находиться в диапазоне от 1 (низкий) до 10 (высокий). Значение по умолчанию — 5.

Синтаксис

```
CREATE BROKER PRIORITY ConversationPriorityName
FOR CONVERSATION
[ SET ( [ CONTRACT_NAME = {ContractName | ANY } ]
      [ [ , ] LOCAL_SERVICE_NAME = {LocalServiceName | ANY } ]
      [ [ , ] REMOTE_SERVICE_NAME = {'RemoteServiceName' | ANY } ]
      [ [ , ] PRIORITY_LEVEL = {PriorityValue | DEFAULT } ]
     )
]
[;]
```

Аргументы

ConversationPriorityName

Задает имя данного приоритета диалога. Имя должно быть уникальным внутри текущей базы данных и должно соответствовать правилам для идентификаторов компонента Компонент Database Engine.

SET

Задает критерий для определения применимости приоритета к заданному диалогу. Если указано, SET должен содержать хотя бы один критерий: CONTRACT_NAME, LOCAL_SERVICE_NAME, REMOTE_SERVICE_NAME или PRIORITY_LEVEL. Если аргумент SET не указан, устанавливаются значения по умолчанию для всех трех критериев.

CONTRACT_NAME = {*ContractName* | **ANY**}

Указывает имя контракта, который будет использоваться в качестве критерия, определяющего применимость приоритета к диалогу. Аргумент *ContractName* — это идентификатор компонента Компонент Database Engine, который должен указывать имя контракта в текущей базе данных.

ContractName

Указывает, что приоритет может применяться только к диалогам, в которых инструкция BEGIN DIALOG, начинающая диалог, содержит параметр ON CONTRACT *ContractName*.

ANY

Указывает, что приоритет может применяться к любому диалогу, независимо от используемого контракта.

Значение по умолчанию — ANY.

LOCAL_SERVICE_NAME = {*LocalServiceName* | **ANY**}

Указывает имя службы, которая будет использоваться в качестве критерия для определения применимости приоритета к конечной точке диалога.

LocalServiceName — это идентификатор Компонент Database Engine. Должно быть указано имя службы в текущей базе данных.

LocalServiceName

Указывает, что объектом применения приоритета диалога может быть:

- любая конечная точка-инициатор диалога, имя вызывающей службы которой соответствует аргументу *LocalServiceName*;
- любая целевая конечная точка диалога, имя целевой службы которой соответствует аргументу *LocalServiceName*.

ANY

Указывает, что приоритет может применяться к любой конечной точке диалога, независимо от имени локальной службы, используемой точкой.

Значение по умолчанию — ANY.

REMOTE_SERVICE_NAME = {'*RemoteServiceName*' | **ANY**}

Указывает имя службы, которая будет использоваться в качестве критерия для определения применимости приоритета к конечной точке диалога.

RemoteServiceName — это литерал типа **nvarchar(256)**. Компонент Компонент Service Broker производит побайтовое сравнение при поиске соответствия строке *RemoteServiceName*. При сравнении учитывается регистр и не применяются текущие параметры сортировки. Целевая служба может располагаться в текущем экземпляре компонента Компонент Database Engine или в удаленном экземпляре компонента Компонент Database Engine.

'*RemoteServiceName*'

Указывает, что объектом применения приоритета диалога может быть:

- любая конечная точка-инициатор диалога, для которой имя целевой службы совпадает с параметром *RemoteServiceName*;
- любая целевая конечная точка диалога, для которой имя вызывающей службы совпадает с параметром *RemoteServiceName*.

ANY

Указывает, что приоритет диалога может быть применен к любой конечной точке диалога, независимо от имени удаленной службы, связанной с конечной точкой.

Значение по умолчанию — ANY.

PRIORITY_LEVEL = { *PriorityValue* | **DEFAULT** }

Указывает приоритет, который назначается любой конечной точке диалога, использующей контракты и службы, указанные в приоритете диалога. Аргумент *PriorityValue* должен быть целочисленным литералом в диапазоне от 1 (наименьший приоритет) до 10 (наибольший приоритет). Значение по умолчанию — 5.

Комментарии

Компонент Компонент Service Broker назначает уровни приоритета конечным точкам диалога. Уровни приоритета управляют приоритетом операций, связанных с конечной точкой. У каждого диалога есть две конечные точки:

- Конечная точка — инициатор диалога связывает одну сторону диалога со службой и очередью инициатора. Конечная — точка-инициатор диалога создается во время выполнения инструкции BEGIN DIALOG. С конечной точкой — инициатором диалога связаны следующие операции:
  - отправка сообщений вызывающей службой;
  - получение сообщений из очереди инициатора;
  - получение следующей группы сообщений из очереди инициатора.
- Целевая конечная точка диалога связывает противоположную сторону диалога с целевой службой и очередью. Целевая конечная точка диалога создается, когда диалог используется для отправки сообщения в очередь цели. С целевой конечной точкой диалога связаны следующие операции:
  - получение сообщений из очереди цели;
  - отправка сообщений целевой службы;
  - получение следующей группы сообщений из очереди целевой стороны.

Во время создания конечных точек диалога компонент Компонент Service Broker назначает уровни приоритета диалога. Конечная точка диалога поддерживает уровень приоритета до окончания диалога. К незавершенным диалогам не применяются новые приоритеты и изменения в текущих приоритетах.

Компонент Компонент Service Broker назначает конечной точке диалога уровень приоритета, критерии контракта и службы которого более всего совпадают со свойствами конечной точки. В следующей таблице показан порядок соответствия.

Контракт операции	**Локальная служба операции**	**Удаленная служба операции**
ContractName	*LocalServiceName*	*RemoteServiceName*
ContractName	*LocalServiceName*	ANY
ContractName	ANY	*RemoteServiceName*
ContractName	ANY	ANY
ANY	*LocalServiceName*	*RemoteServiceName*
ANY	*LocalServiceName*	ANY
ANY	ANY	*RemoteServiceName*
ANY	ANY	ANY

Компонент Компонент Service Broker в первую очередь ищет приоритет, в котором указаны контракт, локальная служба и удаленная служба, совпадающие с используемыми операцией. Если такой приоритет не найден, компонент Компонент Service Broker ищет приоритет, в котором контракт и локальная служба совпадают с используемыми операцией, а удаленная служба указана как ANY. Процесс продолжается для всех вариантов, перечисленных в таблице порядка соответствия. Если совпадения не обнаружено, операции назначается приоритет по умолчанию — 5.

Компонент Компонент Service Broker независимо назначает уровни приоритета каждой конечной точке диалога. Чтобы компонент Компонент Service Broker назначал уровни приоритета и для конечной точки вызывающей стороны, и для целевой конечной точки диалога, необходимо, чтобы приоритеты диалога распространялись на обе конечные

точки. Если конечная точка-инициатор диалога и целевая конечная точка диалога расположены в разных базах данных, необходимо создать приоритеты диалога в каждой базе данных. Обычно обеим конечным точкам диалога назначается одинаковый уровень приоритета, но можно указать и различные уровни приоритета.

Уровни приоритетов всегда применяются к операциям, получающим из очереди сообщения или идентификаторы групп сообщений. Уровни приоритета также применяются при передаче сообщений от одного экземпляра компонента Компонент Database Engine другому.

Уровни приоритета не используются при передаче сообщений:

- из базы данных, в которой параметр базы данных HONOR_BROKER_PRIORITY установлен в значение OFF.
- между службами в одном экземпляре компонента Database Engine.
- Всем операциям компонента Компонент Service Broker в базе данных назначаются приоритеты по умолчанию 5, если в базе данных не создано приоритетов диалога.

Разрешения

Разрешение на создание приоритета диалога по умолчанию предоставляется членам предопределенных ролей базы данных db_ddladmin и db_owner, а также членам предопределенной роли сервера sysadmin. Необходимо разрешение ALTER на базу данных.

Примеры

А. Назначение уровня приоритета обоим направлениям диалога.

Эти два приоритета диалога обеспечивают назначение всем операциям, в которых используется контракт SimpleContract между службами TargetService и InitiatorAService, уровня приоритета 3.

```
CREATE BROKER PRIORITY InitiatorAToTargetPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SimpleContract,
        LOCAL_SERVICE_NAME = InitiatorServiceA,
        REMOTE_SERVICE_NAME = N'TargetService',
        PRIORITY_LEVEL = 3);
CREATE BROKER PRIORITY TargetToInitiatorAPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SimpleContract,
        LOCAL_SERVICE_NAME = TargetService,
        REMOTE_SERVICE_NAME = N'InitiatorServiceA',
        PRIORITY_LEVEL = 3);
```

Б. Назначение уровня приоритета всем диалогам, которые используют контракт

Назначает уровень приоритета 7 всем операциям, которые используют контракт с именем SimpleContract. Предполагается, что не существует других приоритетов, которые в которых указаны одновременно контракт SimpleContract и локальная или удаленная служба.

```
CREATE BROKER PRIORITY SimpleContractDefaultPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SimpleContract,
        LOCAL_SERVICE_NAME = ANY,
```

```
      REMOTE_SERVICE_NAME = ANY,
      PRIORITY_LEVEL = 7);
```

В. Установка базового уровня приоритета для базы данных.

Определяет приоритеты диалога для двух конкретных служб, а затем определяет приоритет диалога, который будет соответствовать всем другим конечным точкам диалога. Это не заменяет приоритет по умолчанию, который всегда имеет значение 5, но уменьшает число элементов, которые назначаются по умолчанию.

```
CREATE BROKER PRIORITY [//Adventure-Works.com/Expenses/ClaimPriority]
  FOR CONVERSATION
  SET (CONTRACT_NAME = ANY,
      LOCAL_SERVICE_NAME = //Adventure-Works.com/Expenses/ClaimService,
      REMOTE_SERVICE_NAME = ANY,
      PRIORITY_LEVEL = 9);
CREATE BROKER PRIORITY [//Adventure-Works.com/Expenses/ApprovalPriority]
  FOR CONVERSATION
  SET (CONTRACT_NAME = ANY,
      LOCAL_SERVICE_NAME = //Adventure-Works.com/Expenses/ClaimService,
      REMOTE_SERVICE_NAME = ANY,
      PRIORITY_LEVEL = 6);
CREATE BROKER PRIORITY [//Adventure-Works.com/Expenses/BasePriority]
  FOR CONVERSATION
  SET (CONTRACT_NAME = ANY,
      LOCAL_SERVICE_NAME = ANY,
      REMOTE_SERVICE_NAME = ANY,
      PRIORITY_LEVEL = 3);
```

Г. Создание трех уровней приоритета для целевой службы с использованием служб

Поддерживает систему, обеспечивающую три уровня производительности: золотой (высокий), серебряный (средний) и бронзовый (низкий). Имеется один контракт, но каждый уровень располагает отдельной вызывающей службой. Все вызывающие службы связываются с центральной целевой службой.

```
CREATE BROKER PRIORITY GoldInitToTargetPriority
  FOR CONVERSATION
  SET (CONTRACT_NAME = SimpleContract,
      LOCAL_SERVICE_NAME = GoldInitiatorService,
      REMOTE_SERVICE_NAME = N'TargetService',
      PRIORITY_LEVEL = 6);
CREATE BROKER PRIORITY GoldTargetToInitPriority
  FOR CONVERSATION
  SET (CONTRACT_NAME = SimpleContract,
      LOCAL_SERVICE_NAME = TargetService,
      REMOTE_SERVICE_NAME = N'GoldInitiatorService',
      PRIORITY_LEVEL = 6);
CREATE BROKER PRIORITY SilverInitToTargetPriority
  FOR CONVERSATION
  SET (CONTRACT_NAME = SimpleContract,
      LOCAL_SERVICE_NAME = SilverInitiatorService,
      REMOTE_SERVICE_NAME = N'TargetService',
```

```
        PRIORITY_LEVEL = 4);
CREATE BROKER PRIORITY SilverTargetToInitPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SimpleContract,
        LOCAL_SERVICE_NAME = TargetService,
        REMOTE_SERVICE_NAME = N'SilverInitiatorService',
        PRIORITY_LEVEL = 4);
CREATE BROKER PRIORITY BronzeInitToTargetPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SimpleContract,
        LOCAL_SERVICE_NAME = BronzeInitiatorService,
        REMOTE_SERVICE_NAME = N'TargetService',
        PRIORITY_LEVEL = 2);
CREATE BROKER PRIORITY BronzeTargetToInitPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SimpleContract,
        LOCAL_SERVICE_NAME = TargetService,
        REMOTE_SERVICE_NAME = N'BronzeInitiatorService',
        PRIORITY_LEVEL = 2);
```

Д. Создание трех уровней приоритета для нескольких служб с использованием контрактов

Поддерживает систему, обеспечивающую три уровня производительности: золотой (высокий), серебряный (средний) и бронзовый (низкий). У каждого уровня имеется отдельный контракт. Эти приоритеты применяются к любым службам, на которые ссылаются диалоги, использующие контракты.

```
CREATE BROKER PRIORITY GoldPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = GoldContract,
        LOCAL_SERVICE_NAME = ANY,
        REMOTE_SERVICE_NAME = ANY,
        PRIORITY_LEVEL = 6);
CREATE BROKER PRIORITY SilverPriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = SilverContract,
        LOCAL_SERVICE_NAME = ANY,
        REMOTE_SERVICE_NAME = ANY,
        PRIORITY_LEVEL = 4);
CREATE BROKER PRIORITY BronzePriority
   FOR CONVERSATION
   SET (CONTRACT_NAME = BronzeContract,
        LOCAL_SERVICE_NAME = ANY,
        REMOTE_SERVICE_NAME = ANY,
        PRIORITY_LEVEL = 2);
```

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями определите и установите уровни приоритета для различных объектов и субъектов к созданной Вами ранее базе данных.
3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под «приоритетом»?
2. В каком диапазоне должны находиться приоритеты?
3. Какие приоритеты Вы знаете?

Лабораторная работа №14
«РАЗВЕРТЫВАНИЕ КОНТРОЛЛЕРОВ ДОМЕНА»

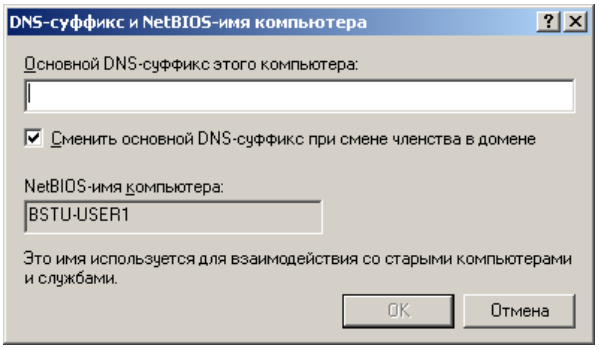
Цель работы: получить теоретические знания и практические навыки развертывания контроллеров домена и проверки его работоспособности.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Существование контроллера домена невозможно без службы каталогов Active Directory. Повышение роли рядового сервера до контроллера домена осуществляется установкой на него службы AD. Инструмент, который используется для установки (или удаления) Active Directory на сервер, называется Active Directory Installation Wizard (мастер установки Active Directory) – dcpromo.exe.

Сервер, на который осуществляется установка AD, должен удовлетворять целому ряду требований, перечисленных ниже:

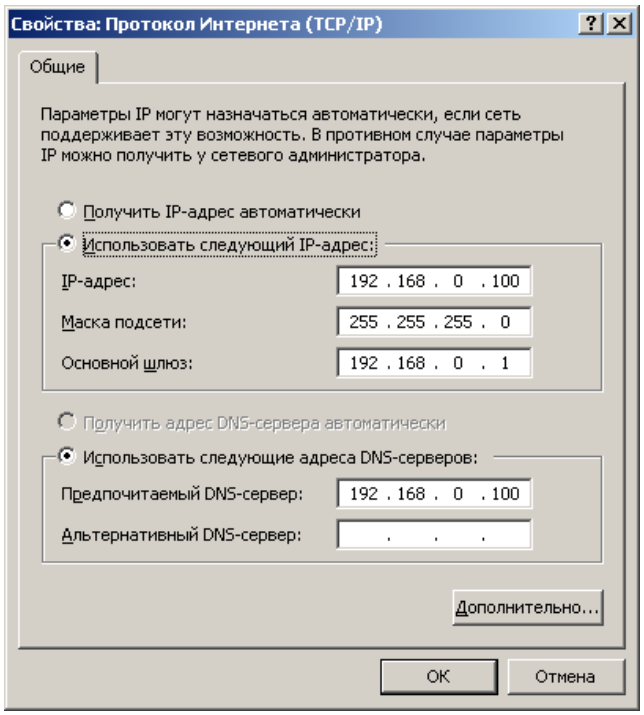
- Перед установкой на сервере должен быть установлен стек протоколов TCP/IP и для каждого из интерфейсов сервера выделен статический IP-адрес. Впоследствии администратор может изменить этот адрес и заново зарегистрировать в базе данных DNS доменное имя с уже новым адресом.
- Для сервера должен быть установлен DNS-суффикс, соответствующий имени домена, для которого будет устанавливаться контроллер домена. Последнее требование является необязательным, если установлен флажок «Сменить основной DNS-суффикс при смене членства в домене» изменения имени компьютера в свойствах системы (Мой компьютер).



- В этом случае система автоматически определит DNS-суффикс при включении сервера в состав некоторого домена.
- Служба каталога может быть установлена на раздел диска с файловой системой NTFS. Это требование обусловлено соблюдением должного уровня безопасности, требующего разграничения доступа к файлам, непосредственно на уровне файловой системы (скажем, файловая система FAT не предоставляет такой возможности). Кроме того, раздел, предназначенный для установки службы каталога, должен иметь как минимум 250 Мбайт свободного дискового пространства. С целью повышения производительности службы каталога администратор может разместить файлы хранилища каталога и журнала транзакций на отдельные физические диски. Это позволит избежать конкуренции операции ввода/вывода. Разумеется, в этом случае каждый из задействованных разделов должен быть отформатирован под NTFS.
- Операция установки контроллера домена требует наличия у выполняющего ее пользователя определенных полномочий. Установка первого контроллера домена в лесе осуществляется на одиночном сервере, не являющимся частью какого-либо домена. В этой ситуации пользователь должен обладать полномочиями локального администратора на том сервере, на котором происходит установка. Если происходит установка первого

контроллера в домене (в рамках уже существующего леса доменов), пользователь должен являться членом группы Enterprise Admins (Администраторы корпорации). В случае установки дополнительного контроллера в домене пользователь должен быть либо членом уже упомянутой группы, либо членом группы Domain Admins (Администраторы домена).

Прежде чем запустить на сервере мастер установки Active Directory, администратор должен проверить настройки стека протоколов TCP/IP для данного компьютера, обратив, в первую очередь, внимание на параметры службы DNS-клиента. Одним из важнейших параметров в этой ситуации является адрес предпочитаемого DNS-сервера (preferred DNS server).



Именно указанный в этом параметре сервер будет использоваться мастером установки для диагностики пространства имен DNS, предшествующего созданию нового домена Active Directory, и поиска существующих носителей копий каталога. Этот же DNS-сервер впоследствии будет использоваться для регистрации доменного имени сервера. Ошибки, допущенные на этом этапе, могут привести к тому, что по окончании процедуры установки контроллер домена окажется неработоспособным. Типичны следующие ошибки:

- настройки сервера, выбранного на роль контроллера домена, не содержат сведений о предпочитаемом DNS-сервере;
- DNS-сервер, указанный в настройках будущего контроллера домена в качестве предпочтительного, не является носителем требуемой зоны. Кроме того, возможна и другая ситуация, когда указан сервер, являющийся дополнительным носителем зоны. Как следствие, этот DNS-сервер не может быть использован для динамической регистрации доменных имен.

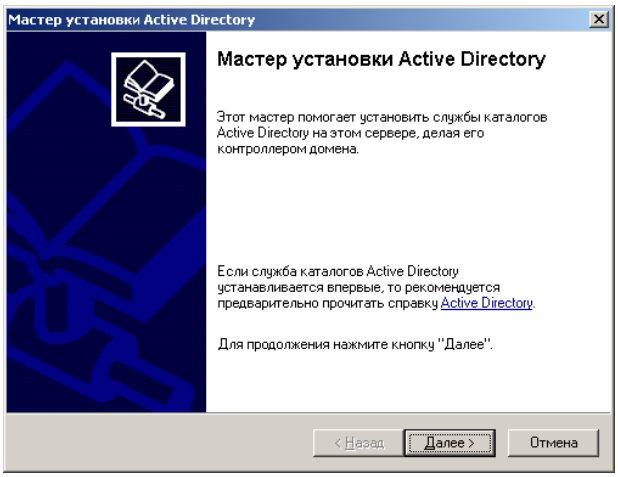
Если вы осуществляете установку первого контроллера домена в лесу доменов (фактически это означает первый этап развертывания в сети службы каталога), то вы должны предоставить возможность мастеру установки Active Directory установить на сервере службу DNS и произвести ее последующее конфигурирование. При этом в настройках стека протоколов TCP/IP данного сервера параметр Preferred DNS Server (Предпочитаемый DNS-сервер) должен указывать непосредственно на сам сервер. То есть после установки контроллера домена все ассоциированные с ним ресурсные записи будут зарегистрированы службой DNS, функционирующей на этом же сервере. Все последующие

контроллеры домена должны указывать уже на существующие DNS-серверы (например, на первый установленный DNS-сервер).

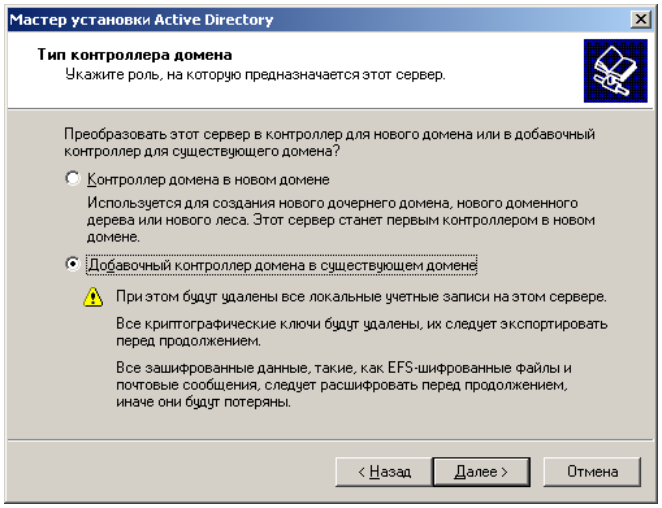
Кроме того, необходимо проверить доступность DNS-сервера с компьютера, который выбран на роль контроллера домена. Различные проблемы с сетевыми компонентами могут привести к тому, что хотя DNS-сервер функционирует и успешно используется другими хостами, вновь устанавливаемый контроллер домена не имеет с ним сетевого соединения.

Процедура установки контроллера домена выполняется с помощью мастера установки Active Directory. Для запуска мастера можно воспользоваться командой dcpromo, которая запускается из меню Пуск|Выполнить (Start|Run).

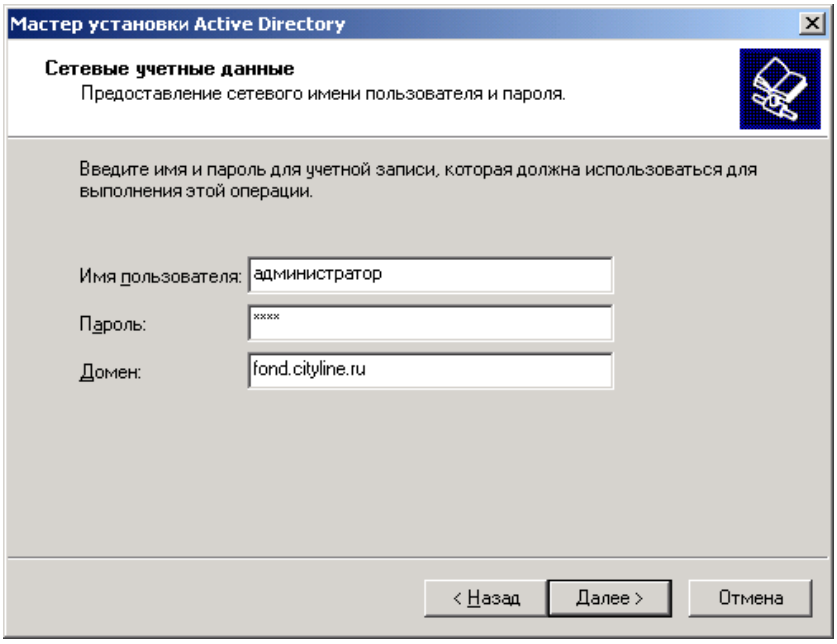
Альтернативный вариант - выбрать команду Пуск|Программы|Администрирование|Мастер настройки сервера или Управление данным сервером (Пуск|Все программы|Администрирование |Управление данным сервером), в открывшемся окне последовательно нажать на ссылку «Добавить или удалить роль», затем – установка Active Directory.



Поскольку мы устанавливаем дополнительные контроллеры домена в существующий домен, в появившемся окне мастера установки для данного варианта установки следует установить переключатель **Добавочный контроллер домена в существующем домене (Additional domain controller for an existing domain)** и нажать кнопку Далее.

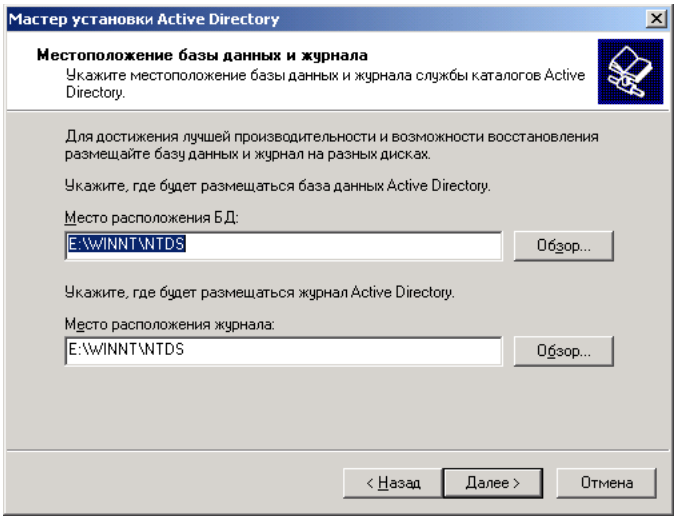


Введите имя, пароль и полное DNS-имя домена для пользовательской записи с административными правами в домене (это может быть член группы Администраторы или пользователь, имеющий права на подключение компьютеров к домену).



Введите полное DNS-имя существующего домена; при этом можно выбрать домен из списка существующих, нажав кнопку Обзор (Browse).

В следующих окнах мастера нужно указать дополнительные параметры (местоположение базы данных Active Directory, журналов регистрации событий, реплицируемого системного тома, а также пароль администратора для восстановления службы каталогов).



В появляющемся окне сводки проверьте правильность параметров и нажмите кнопку **Далее** и начнется процесс повышения роли сервера.

После перезагрузки компьютер будет работать как один из контроллеров указанного домена.

Примечание

Если на сервере до начала процесса повышения роли была установлена служба DNS, то она полностью конфигурируется с использованием записей основного DNS-сервера.

Таким образом, легко получить резервный DNS-сервер, повысив отказоустойчивость сети. При этом предпочтительнее, если зоны DNS хранятся в Active Directory.

В процессе установки контроллера домена происходит наполнение каталога. В случае установки первого контроллера домена в лесе все содержимое каталога создается

непосредственно мастером установки. Если в лесе создается новый домен, то мастер установки создает исключительно доменный раздел. Раздел схемы и каталога копируется с уже существующих контроллеров домена. В ситуации, когда администратор создает дополнительный контроллер в уже существующем домене, имеется два варианта наполнения каталога:

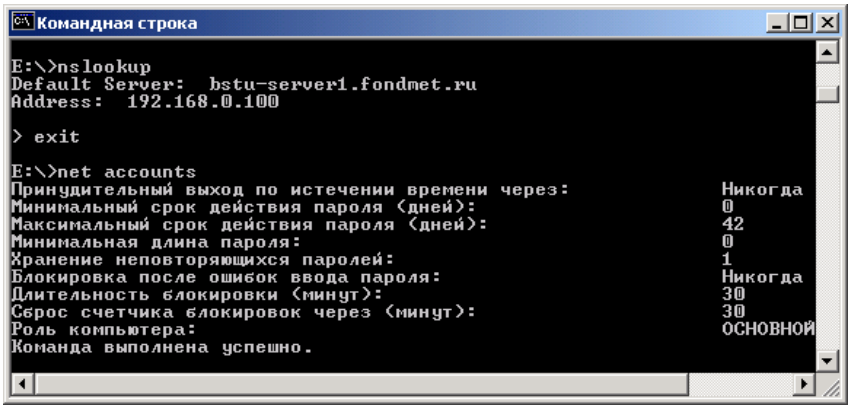
- все содержимое каталога копируется с уже существующего контроллера домена;
- содержимое каталога воссоздается из резервной копии каталога. Этот вариант целесообразно использовать в удаленных филиалах, соединенных с первичным контроллером домена низкоскоростными каналами связи.

Довольно часто возникает необходимость убедиться в том, процесс перехода сервера в новое качество успешно завершен. Если, например, служба репликации файлов (FRS – File Replication Service) не может успешно стартовать, она не инициализирует системный том, в результате чего служба Netlogon не может сделать общей (shared) системную папку SYSVOL. Как следствие папка NETLOGON также становится недоступной для общего доступа. Это приводит к возникновению проблем с выполнением групповых политик, а также многих других проблем, в частности, с репликацией и аутентификацией. При возникновении неисправностей в Active Directory, перед тем, как выявлять проблемы, касающиеся соединений между контроллерами домена, аутентификации и т. д., вы в обоих случаях должны убедиться в том, что серверы Windows Server действительно являются контроллерами домена.

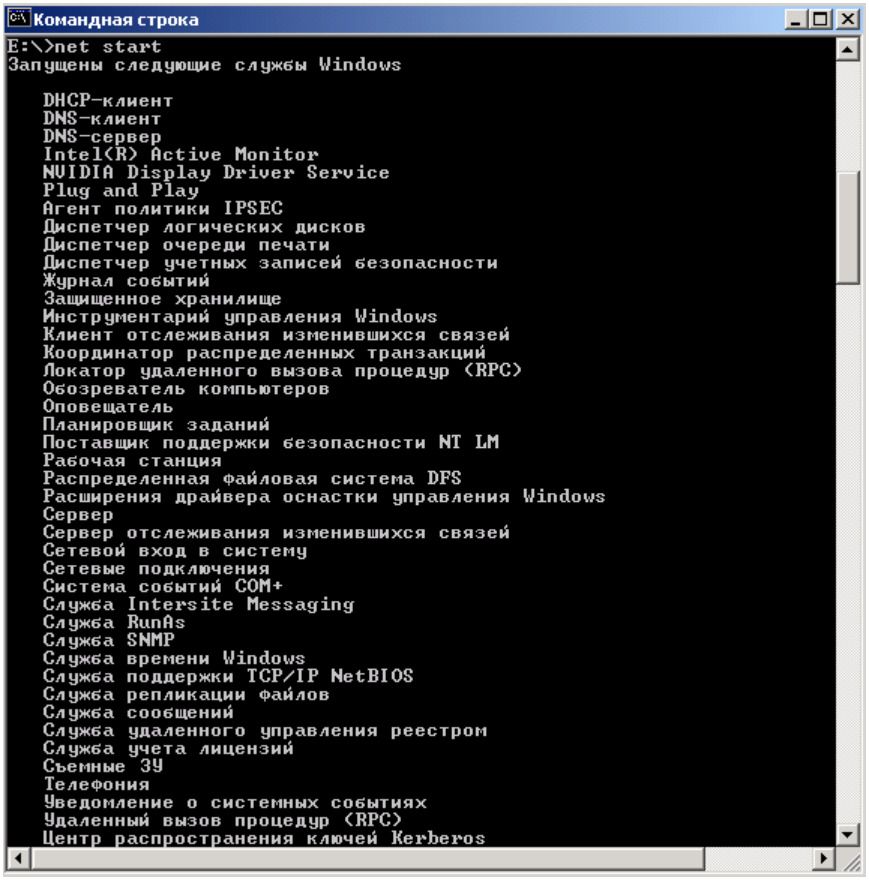
Существует несколько способов, посредством которых администратор может убедиться в том, что некоторый сервер на базе Windows Server по окончании операции повышения роли сервера (promotion) выполняет функции контроллера домена.

Раздел реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services должен содержать подраздел NTDS.

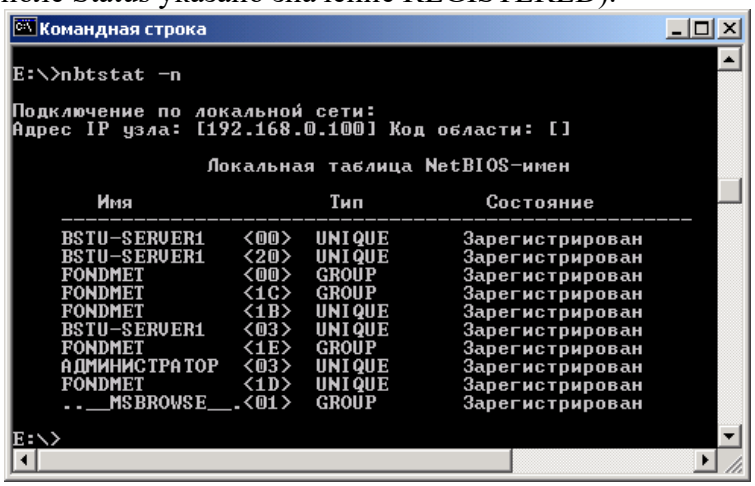
Введите в командной строке net accounts. Поле Computer role (Роль компьютера) должна содержать значение PRIMARY или BACKUP для контроллера домена. Для обычных серверов это поле имеет значение SERVERS.



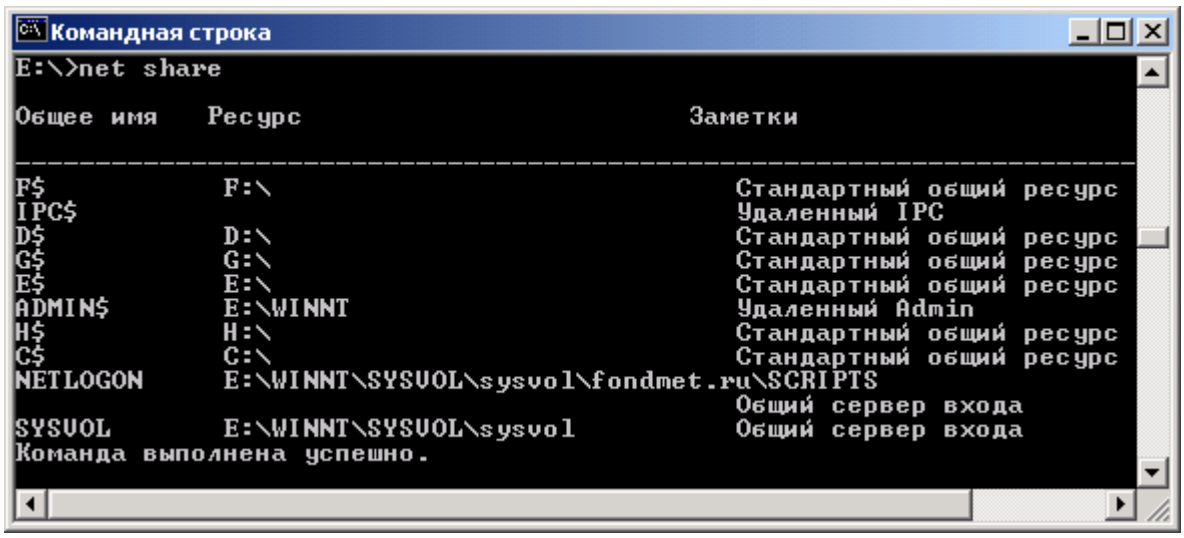
Введите в командной строке net start. В списке запущенных сервисов должна присутствовать служба Kerberos Key Distribution Center (Центр распределения ключей Kerberos). Если эта служба не запущена на контроллере домена, механизм аутентификации может не работать.



Введите в командной строке nbtstat -n. Имя домена, имеющее тип <ic>, должно быть зарегистрировано (в поле Status указано значение REGISTERED).



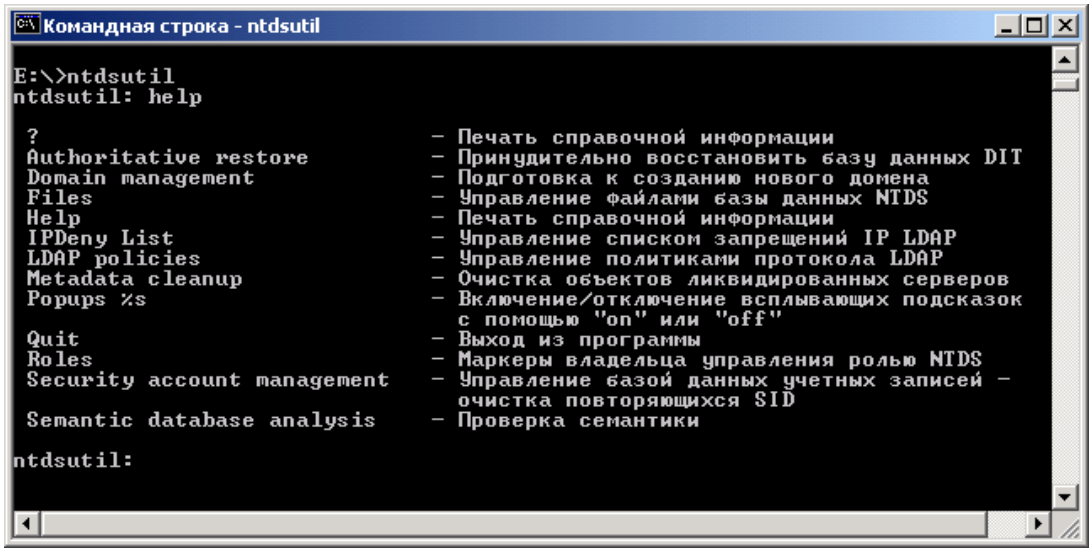
Введите в командной строке net share. На сервере должны присутствовать общие папки SYSVOL (%SystemRoot%\SYSVOL\sysvol) и NETLOGON (%SystemRoot%\SYSVOL\sysvol\<DomainDNSName>\SCRIPTS).



С помощью утилиты Ldp.exe проверьте значение атрибута isSynchronized объекта RootosE. По окончании процесса повышения роли сервера система должна полностью синхронизировать все разделы каталога. Когда синхронизация закончена, атрибут isSynchronized принимает значение TRUE.

Используйте утилиту командной строки NLtest.exe. Эта утилита поставляется в составе пакета вспомогательных утилит Windows Server 2003 Support Tools.

С помощью утилиты Ntdsutil.exe можно подключиться к только что установленному контроллеру домена и проверить его способность отвечать на запросы LDAP. Утилита позволяет также проверить, знает ли контроллер о расположении ролей FSMO в своем домене.



ЗАДАНИЕ НА РАБОТУ

Задание 1. Изучите теоретический материал по теме работы, основные принципы организации Active Directory (AD).

Задание 2. С помощью утилиты Ipconfig определить имя домена, в котором будет работать контроллер домена.

Задание 3. Проверить, удовлетворяет ли компьютер, повышаемый до роли контроллера домена, описанным выше требованиям.

Задание 4. Проверить доступность первичного контроллера домена по сети (имя первичного контроллера домена можно определить с помощью утилиты Ipconfig).

Задание 5. Запустить мастер установки Active Directory и следуя инструкциям установить AD, повысив тем самым роль сервера до контроллера домена.

Задание 6. Убедиться, что контроллер домена функционирует корректно описанными выше способами.

Задание 7. Сделать резервную копию AD.

Задание 8. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что Вы понимаете под «развертыванием контроллеров домена»?
2. Какие инструменты применяются для развертывания контроллеров домена?
3. Какие инструменты применяются для проверки работоспособности контроллеров домена?

Лабораторная работа №15
«МОНИТОРИНГ СЕТЕВОГО ТРАФИКА»

Цель работы: получить теоретические знания и практические навыки мониторинга сетевого трафика.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Под мониторингом сети понимают процесс сбора и анализа сетевого трафика, по результатам которого можно судить о качественных и количественных характеристиках работоспособности сети или ее отдельных компонентов. Программы мониторинга сети позволяют выполнять захват пакетов и их реассемблирование для дальнейшего анализа.

Для мониторинга используют специальные программы - анализаторы сети. Таких программ много, например Windows Network Monitor, tcpdump, Ethereal Network Analyzer (ENA), Wireshark и т.п. Они схожи по функциям, а отличаются в основном пользовательским интерфейсом и возможностями генерации статистических отчетов.

Онлайн-анализ трафика

В глобальной сети все большее распространение получают онлайн-сервисы, выполняющие мониторинг серверов. Основное назначение таких сервисов - контроль за работоспособностью узлов и оповещение администратора о нештатных ситуациях по эл.почте, через IM и по SMS. Основные проверки выполняются для сервисов прикладного уровня (HTTP, FTP, SMTP, POP3 и т.п.) с возможностью указания интервала проверок. Дополнительными возможностями являются, например, uptime-информеры, средства контроля за появлением вредоносного кода, подключение нескольких ресурсов на аккаунт и т.п. Детальное изучение онлайн-сервисов мониторинга выходит за рамки этой лабораторной работы.

Для выполнения этой работы рекомендуется использовать программы Ethereal Network Analyzer или Wireshark (версии для UNIX/Linux, Windows-версия работает не стабильно). Эти программы практически идентичны как по возможностям, так и по использованию.

ЗАДАНИЕ НА РАБОТУ

1. Изучите теоретический материал по теме работы.
2. В соответствии с рекомендациями выполните мониторинга сетевого трафика:

2.1. Запустить ENA в режиме захвата трафика (или другую программу для анализа трафика), проходящего через интерфейс, подключенный к локальной сети (обычно это eth0). Перейти к следующему заданию.

2.2 Эмулировать сетевую активность в течении 15-20 минут. Для этого можно выполнить, например, некоторые из указанных действий.

- Открыть любой сайт.
- Подключиться к серверу ftp://...
- Выполнить ping любых узлов.
- Подключиться к одному из доступных сетевых дисков Windows (если такие ресурсы представлены в сети).
- Выполнить прочие действия, требующие сетевого подключения.

2.2. Остановить захват. Исходные данные для заполнения таблицы получить у преподавателя.

2.3. Заполнить таблицу:

Параметр	Значение
Время захвата, мин	

К-во захваченных пакетов	
Объем, Мб	
Средн.размер пакета, Кб	
Средняя скорость, пакетов/сек	
Средняя скорость, Мбит/сек	

По данным таблицы определить *относительную загрузку* сети (в %) за контрольный период времени по формуле:

$$Загрузка = \frac{(Трафик, Мбит/Время, сек)\cdot 100}{(Пропускная способность, Мбит/сек)}$$

2.4. Составить таблицу 2 распределения трафика по протоколам:

Протокол	Трафик, Мб	Трафик, %
HTTP		
FTP		
...		
ИТОГО		100

По данным таблицы сделать выводы о качественном составе трафика, т.е. о соотношении *прикладных* и *служебных* протоколов.

2.5. Составить таблицу распределения Ethernet-трафика по узлам сети:

MAC-адрес	IP-адрес	Трафик					
		входящий		*исходящий*		*общий*	
		Мб	%	Мб	%	Мб	%
ИТОГО			100		100		100

3. Оформите подробный отчет.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие ВЫ знаете программы для анализа сетевого трафика?
2. Какие параметры оцениваются при анализе сетевого трафика?

Приложение А

Индивидуальные варианты для практических и лабораторных работ

Вариант 1. «РЕСТОРАН».

Постоянным клиентам предоставляется возможность заказать столик заранее. Официант указывает столик, открывает гостевой счет и вводит заказы в соответствии с меню. Далее заказ автоматически обрабатывается, формируются марки на приготовление выбранных блюд и направляют их на производство, в соответствующие цеха кухни, в бар. Расчеты с посетителем сводятся к простой операции: на бланке печатается итоговый счет. Если клиент – постоянный посетитель, то соответствующие привилегии рассчитываются автоматически, затем указываются способ оплаты и полученная от клиента сумма.

Вариант 2. «КИНОТЕАТР».

Продажа и бронирование билетов, а также резервирование мест для постоянных посетителей — основные технологические процессы работы кинотеатра. Важную роль здесь играет качество предоставления информации и контроль выполнения операций. Клиент в момент покупки билета должен видеть план зала и свободные места. Постоянные клиенты имеют возможность зарезервировать билеты по телефону или через Интернет. Формирование билета и его печать. Выводить анонс сеансов с указанием времени и кратким описанием.

Вариант 3. «ГОСТИНИЦА».

Номера в гостинице имеют разный уровень обслуживания и соответственно разную стоимость, (предоставление информации о свободных номерах и их стоимости). Клиенты могут бронировать номера по телефон или Интернету. За номерами прикреплен обслуживающий персонал. Необходимо вести учет обслуживания и оплаты номеров, (заказы в номер, телефонные звонки и т. д .) . Клиент может несколько раз останавливаться в гостинице в разных номерах.

Вариант 4. «ФИТНЕС – КЛУБА».

Они предлагают пакеты услуг – абонементы. Подразумевая предоплату определенного набора услуг. Абонемент позволяет пользоваться ими в течение определенного времени. Для идентификации владельца абонемента используются клубные карты. Комплекс позволяет быстро и просто осуществлять резервирование ресурсов по просьбе постоянного клиента предприятия: как тренера, так и места — спортзала, солярия, бассейна для персональных тренировок или занятий.

Вариант 5. «ОПТОВЫЙ СКЛАД».

Создаваемая информационная система предназначена для учета деятельности оптового склада. Оптовый склад состоит из нескольких складских помещений, каждое помещение имеет наименование, адрес и кладовщика. Склад принимает партии товаров от поставщиков и отпускает его клиентам мелкими партиями. Требуется вести (количественный и стоимостной) учет поступающих и отпускаемых товаров, поставщиков и клиентов, формировать приходные и расходные накладные. Сведения о товаре: Артикул,

Наименование полное, Наименование сокращенное, Производитель, Поставщик, Количество, Цена. Сведения о поставщике и клиенте: Наименование, Адрес, Телефон. Накладная включает: Номер, Дата, Клиент, Список товаров, Общая сумма, Кладовщик. В системе формируются отчеты о поступлении и отпуске товаров на складе за произвольный период.

Вариант 6. «РЕКЛАМНОЕ АГЕНТСТВО».

Создаваемая информационная система должна вести учет деятельности рекламного агентства. Рекламное агентство регистрирует заявки от рекламодателей и публикует рекламы в печатных изданиях. О рекламодателе регистрируются следующие данные: Наименование, Адрес, Руководитель, Телефон, Заявка, Оплата, Издание, Место размещения рекламы. Заявка включает: Вид рекламы, Объем, Желаемые издания, Количество выходов рекламы, Дополнительная информация. Заявка от рекламодателя может содержать публикацию в несколько печатных изданиях и на различные даты выхода. Справочник печатных изданий включает: Наименование, Виды реклам, Стоимость рекламы. Требуется вести списки печатных изданий с их расценками на рекламу, списки рекламодателей, заявок. Система должна обеспечить оперативный просмотр списка заявок (печатные издания, рекламодатель, стоимость) на любую вводимую дату, а также формирование отчета о заявленных и выполненных рекламах.

Вариант 7. «АЭРОПОРТ».

Создаваемая информационная система предназначена для учета движения самолетов и пассажиропотока. В аэровокзале имеется расписание движения самолетов, которое включает: Номер рейса, Тип самолета, Маршрут, Пункты промежуточной посадки, Время отправления, Дни полета. В системе ведется учет: Количество свободных мест на каждом рейсе, Общий вес пассажиров, Вес ручной клади, Вес багажа. Система формирует посадочную ведомость с учетом веса багажа и ручной клади. В системе имеется справочник типов самолетов, в котором учитываются: Количество мест, Суммарная грузоподъемность.

Вариант 8. «МАГАЗИН “ЦВЕТЫ”».

Создаваемая информационная система предназначена для учета деятельности магазина по продаже цветов. В системе формируется база данных отдельных цветов и готовых букетов: Наименование цветка или букета, Поставщик цветов, Состав букета, Стоимость, Срок поступления, Срок и место хранения (выставочный зал, склад), Дата продажи. В системе ведется учет бракованных и увядших цветов. Формируется отчет о движении товара за заданный период времени.

Вариант 9. «АДМИНИСТРАТОР ГОСТИНИЦЫ».

Создаваемая информационная система предназначена для учета деятельности гостиницы. В гостинице имеется список номеров: Место нахождения номера, Класс, Число мест, Признак занятости места, Дата освобождения номера. Каждый гость проходит регистрацию: Паспортные данные, Даты приезда и отъезда, Номер, Место, Цель приезда, Организация, в которую прибыл (в случае командировки). Администратор гостиницы осуществляется поселение гостя: выбор подходящего номера (при наличии свободных

мест), регистрация, оформление квитанции. В системе автоматически формируется квитанция об оплате услуг гостиницы. Система должна предусмотреть оформление дополнительной квитанции в случае продления гостем срока проживания в гостинице. В системе имеется возможность поиска гостя по произвольному признаку и формируется отчет о текущем состоянии номеров гостиницы (номер, место, не занят/ занят и кем, дата отъезда).

Вариант 10. «МУЗЫКАЛЬНЫЙ МАГАЗИН».

Создаваемая информационная система предназначена для учета музыкальных произведений в магазине. В системе формируются: База групп и исполнителей, База песен, База дисков с перечнем песен (в виде ссылок). База групп и исполнителей содержит: Наименование группы или исполнителя, Страна, Год образование группы или год начала творческого пути, Краткое содержание творческого пути. База песен содержит: Название, Автор текста, Автор музыки, Время звучания. База дисков содержит: Название диска, Перечень песен (название, исполнитель, время звучания, номер трека). Система имеет возможность поиска всех песен заданной группы (исполнителя). Имеется возможность выбора всех дисков, где встречается заданная песня.

Вариант 11. «АВТОЗАПРАВКА (АЗС)».

Создаваемая информационная система предназначена для учета деятельности автозаправки. На автозаправке имеются несколько колонок для заправки топливом: АИ-98, АИ-95, АИ-92, АИ-80, АИ-76, Дизельное топливо. В базе данных должна быть информация: О колонках, О видах бензина, О ценах и остатках. Необходимо учитывать отпуск топлива по чеку: Номер колонки, Тип топлива, Количество, Цена за литр, Стоимость. Предусмотреть отпуск топлива по дисконтной карточке со скидкой, при этом необходимо учитывать: Номер карточки, Общее количество отпущенного топлива, Скидка в %. Размер скидок зависит от общего количества заправленного топлива. В 20 часов – “пересменка” операторов АЗС, печатается отчет об отпуске топлива за время от 20 часов предыдущего дня до 20 часов текущего дня.

Вариант 12. «САЛОН КРАСОТЫ».

Создаваемая база данных предназначена для учета деятельности салона красоты. База данных салона красоты включает данные об оказываемых услугах, мастерах и оказанных услугах. В системе имеется график работы мастеров, и расписание на день с разделением на интервалы времени (например, по полчаса). О мастере учитываются следующие данные: Время работы, Обслуживаемые клиенты, Оказанные услуги, Сумма оказанных услуг. О клиенте учитываются следующие данные: Фамилия, Инициалы, Телефон. Осуществляется предварительная запись клиентов к мастерам. Случайный клиент может обслуживаться свободным мастером. В системе формируются отчеты по деятельности мастеров (процент их занятости, оказанные услуги).

Вариант 13. «КИНОТЕАТР».

Создаваемая информационная система предназначена для учета проданных билетов в кинотеатре. Кинотеатр имеет несколько залов. Сеансы планируются для каждого зала отдельно. Система формирует базу данных, включающую следующую информацию: Место и сеанс, Справочник кинозалов, Справочник сеансов и стоимость, Справочник фильмов, Справочник жанров. Система формирует отчеты: Отчет о посещаемости по месяцам, Отчет о популярности фильмов, Отчет о популярности жанров. Необходимо предусмотреть возврат билетов и денег.

Вариант 14. «ТУРИСТИЧЕСКАЯ ФИРМА».

Создаваемая информационная система предназначена для учета результатов деятельности туристической фирмы. В туристической фирме ведется учет путевок: Страна, Место пребывания, Сроки, Цена и скидки, Вид транспорта, Маршрут (дата, место пребывания), Количество мест, Гид, Данные о туристах (ФИО, дата рождения, паспорт, адрес, страховой полис, оплата), Менеджер. В системе формируются отчеты о реализации путевок. Необходимо предусмотреть возможность возврата путевки.

Вариант 15. «МАГАЗИН ИГРУШЕК».

Создаваемая база данных предназначена для учета товара в магазине игрушек. В системе формируется база данных магазина игрушек: Наименование товара, Изготовитель, Страна, По какой накладной поступил товар, Для какого возраста, Цена, Место нахождения (склад, торговый зал, бракованная группа), Факт продажи (дата, номер кассового чека), Факт возврата брака. Товары в магазин поступают по накладной, в которой отражены: Номер и дата накладной, Поставщик, Перечень товаров (наименование, количество, цена), Общая сумма товаров по накладной. В системе формируется отчет о движении товаров за заданный период.

Вариант 16. «КОНДИТЕРСКОЕ ПРЕДПРИЯТИЕ».

Создаваемая информационная система предназначена для учета изделий кондитерского предприятия. В системе ведется учет выпускаемых изделий: Конфеты, Печенье, Вафли, Пирожные, Торты, Напитки. Каждое изделие имеет соответствующий: Вид упаковки, Вес, Единицы измерения, Количество, Цену, Дату изготовления, Срок реализации, Дату отгрузки оптовым покупателям (магазины, фирмы и т.п.). В системе формируются отчеты о финансово-хозяйственной деятельности кондитерского предприятия, в котором отражаются данные о произведенных и реализованных кондитерских изделиях.

Вариант 17. «БИБЛИОТЕКА».

Создаваемая информационная система предназначена для учета книг публичной библиотеки. В библиотеке ведется картотека книг. По каждой книге учитываются данные: Авторы, Название, Экземпляр, Издательство, Год издания, Количество экземпляров, Раздел библиотеки (специальная литература, хобби, домашнее хозяйство, беллетристика и так далее), Происхождение книги (приобретена, подарена и т.п.), Наличие книги в данный момент (книгу может взять читатель), Оценка книги читателями. Учет читателей ведется по следующим данным: Читательский билет, Фамилия, Имя, Отчество, Адрес, Телефон,

Книги, Дата возврата (планируемая и фактическая) Система обрабатывает данные и выдает результат: Перечень выданных книг на текущую дату, Статистику (кто чаще берет книги, кто возвращает в срок, кто нарушает сроки возврата?)

Вариант 18. «ТРАНСПОРТНОЕ ПРЕДПРИЯТИЕ».

Создаваемая информационная система предназначена для учета путевых листов. Транспортное предприятие ведет учет путевых листов, выданных водителям (от одного дня до 14 дней). Данные путевого листа: номер путевого листа, дата выдачи, автомобиль (марка, госномер), водитель (может быть два водителя), маршрут, объем бака, заправка топливом (тип топлива, остаток в баке, количество заправленного топлива), показания спидометра на моменты выдачи путевого листа и сдачи путевого листа на обработку. В системе имеется база данных норм расхода топлива по каждой марке автомобиля. При обработке путевого листа производится расчет расхода топлива по спидометру (плановый расход) и сравнение с фактическим (по остатку топлива). Ежедневно выводятся результаты обработки путевых листов по расходу топлива.

Вариант 19. «КОММУНАЛЬНЫЕ УСЛУГИ».

Создаваемая информационная система предназначена для учета оплат коммунальных услуг жителями микрорайона. В ЖЭКе ведется учет оплаты жильцами коммунальных услуг: дом, квартира, основной квартиросъемщик, квартплата, электроэнергия, газ, вода (горячая и холодная), вывоз мусора, лифт. Учет электроэнергии ведется по соответствующим показаниям счетчиков. Учет газа и воды у части жильцов осуществляется по показаниям счетчиков, у части по тарифам. В системе имеются справочники по тарифам для расчета суммы оплаты, а так же льгот по оплате коммунальных услуг. Формируются отчеты об оплате коммунальных услуг.

Вариант 20. «ПОСТАВКА ТОВАРОВ».

Предположим, вы получили заказ на создание БД от поставщика продовольствия в бары кинотеатров и театров. В БД должны быть следующие реквизиты: название, адрес и телефон покупателя, дата заказа, номер заказа, условия продажи, дата выполнения заказа, перечень продуктов по каждому заказу, цена и количество каждого из продуктов в заказе.

Вариант 21. «РЕАЛИЗАЦИЯ ТОВАРОВ».

Предположим, вы предприниматель и занимаетесь реализацией продукции, вы ведете записи о реализованной продукции, которые должны содержать следующие данные, номер продукции, наименование продукции, ФИО заказчика, адрес заказчика, количество заказанной продукции, дата заказа.

Вариант 22. «МЕБЕЛЬНЫЙ САЛОН».

Разрабатываемая информационная система предназначена для учета деятельности мебельного салона. Мебельный салон имеет несколько магазинов. При получении мебели в магазин в базу данных заносятся данные: Название мебели, Тип мебели, Производитель мебели, Дата добавления мебели, Стоимость мебели, Количество мебели. При покупке мебели в базу данных заносятся данные: Код магазина, Номер мебели, Код типа мебели,

Количество выбранной мебели, Код покупателя. По требованию покупателя кассир распечатывает чек об оплате. При добавлении нового магазина в базу данных заносятся данные: Код магазина, Название магазина, ФИО заведующего магазином, Номер телефона, Местонахождение магазина. При возврате мебели в базу данных заносятся данные: Код магазина, Номер типа мебели, Номер мебели, Количество мебели, Причина возврата мебели. По требованию кассира покупатель показывает чек и причину возврата мебели. Данные по названию и виду мебели заносятся с чека.

Вариант 23. «СПРАВОЧНАЯ АПТЕК».

Создаваемая информационная система предназначена для учета медицинских препаратов в сети аптек города с возможностью поиска препаратов через сеть Интернет. В системе формируется база данных, содержащая данные: Название аптеки, Адрес аптеки, Телефон, Проезд к аптеке. Перечень медицинских препаратов: Группа товара, Наименование, Единица измерения, Область применения, Производитель, Страна, Поставщик, Дата поставки, Дата изготовления, Срок годности, Место нахождения (торговый зал, склад и т.п.), Количество и цена, Наличие препарата на данный момент. В системе формируются отчеты о движении медицинских препаратов.

Вариант 24. «АПТЕКА».

Предположим вы владелец аптеки. В аптеке имеются различные препараты различного назначения (от головной боли, сердечной, желудочной и т.д.) и различного применения (внутреннего, наружного и т.д.) соответственно по различным оптовым ценам. Вы работаете, в основном, с постоянными поставщиками и для этого вам необходимо знать наименование их фирм, фамилию, имя и отчество руководителя, адрес, номер телефона. Каждую операцию по поставке товаров вы заносите в книгу учета, где регистрируете дату поставки, вид оплаты (наличный, безналичный, по кредитной карточке), наименование поступающего медикамента, номер сопровождающего документа, количество.

Вариант 25.«МЕЖДУГОРОДНИМИ ПОСТАВКАМИ ГРУЗОВ».

Предположим, вы занимаетесь междугородними поставками грузов. У вас в наличии несколько грузовых машин. При оформлении рейса ведется составление регистрационной записи, содержащей следующую информацию: регистрационный номер, дату отправки, место назначения, ФИО водителя, бортовой номер машины, вид груза и его количества.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Технический колледж им. С.И. Мосина

Методические указания по выполнению самостоятельных работ по профессиональному модулю ПМ.11 РАЗРАБОТКА, АДМИНИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ

основной профессиональной образовательной программы среднего профессионального образования – программы подготовки специалистов среднего звена

по специальности
09.02.07 Информационные системы и программирование

квалификация
Программист

Тула 2021

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «14» октября 2021 г. № 3
Председатель цикловой комиссии _____ И.В. Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью и задачами изучения профессионального модуля (дисциплины) является формирование у студентов общих и профессиональных компетенций, знаний, умений и практического опыта, указанных в разделах 1, 4 рабочей программы профессионального модуля (дисциплины). В рамках достижения указанной цели учебным планом и рабочей программой предусмотрена самостоятельная работа студентов.

Самостоятельная работа студентов является одной из важнейших составляющих образовательного процесса. Независимо от полученной специальности и характера работы любой начинающий специалист должен обладать фундаментальными знаниями, профессиональными умениями и навыками деятельности своей квалификации, опытом творческой и исследовательской деятельности по решению новых проблем, опытом социально-оценочной деятельности.

Все эти составляющие образования формируются именно в процессе самостоятельной работы студентов, так как предполагает максимальную индивидуализацию деятельности каждого студента и может рассматриваться одновременно и как средство совершенствования творческой индивидуальности.

Основным принципом организации самостоятельной работы студентов является комплексный подход, направленный на формирование навыков репродуктивной и творческой деятельности студента в аудитории, при внеаудиторных контактах с преподавателем на консультациях и домашней подготовке.

Самостоятельная работа студента предполагает углубленное изучение тем, входящих в содержание профессионального модуля (дисциплины) (пункт 2.2 рабочей программы) и выполнение дополнительных заданий теоретического и практического характера.

Среди основных видов самостоятельной работы студентов традиционно выделяют: подготовка к лекциям, лабораторным работам и практическим занятиям, зачетам и экзаменам, презентациям и докладам; написание рефератов, выполнение лабораторных и контрольных работ.

1 Виды и формы организации самостоятельной работы студентов

Любой вид занятий, создающий условия для зарождения самостоятельной мысли, познавательной и творческой активности студента связан с самостоятельной работой. В широком смысле под самостоятельной работой понимают совокупность всей самостоятельной деятельности студентов как в учебной аудитории, так и вне ее, в контакте с преподавателем и в его отсутствие.

Самостоятельная работа может реализовываться:

- непосредственно в процессе аудиторных занятий – на лекциях, лабораторных работах, практических занятиях, при выполнении контрольных и лабораторных работ и др.;
- в контакте с преподавателем вне рамок аудиторных занятий – на консультациях по учебным вопросам, в ходе творческих контактов, при ликвидации задолженностей, при выполнении индивидуальных заданий и т.д.;
- в библиотеке, дома, в общежитии и других местах при выполнении студентом учебных и творческих заданий.

Цель самостоятельной работы студента – осмысленно и самостоятельно работать сначала с учебным материалом, заложить основы самоорганизации и самовоспитания с тем, чтобы привить умение в дальнейшем непрерывно повышать свою профессиональную квалификацию.

В учебном процессе выделяют два вида самостоятельной работы:

- аудиторная – самостоятельная работа выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию;
- внеаудиторная – самостоятельная работа выполняется студентом по заданию преподавателя, но без его непосредственного участия.

Содержание аудиторной и внеаудиторной самостоятельной работы студентов определяется в соответствии с рекомендуемыми видами учебных заданий, представленными в рабочей программе профессионального модуля (дисциплины).

Самостоятельная работа помогает студентам:

1) овладеть знаниями:

- чтение текста (учебника, первоисточника, дополнительной литературы и т.д.);
- составление плана текста, графическое изображение структуры текста, конспектирование текста, выписки из текста и т.д.;
- работа со справочниками и др. справочной литературой;
- ознакомление с нормативными и правовыми документами;
- учебно-методическая и научно-исследовательская работа;
- использование компьютерной техники и Интернета и др.;

2) закреплять и систематизировать знания:

- работа с конспектом лекции;
- обработка текста, повторная работа над учебным материалом учебника, первоисточника, дополнительной литературы, аудио и видеозаписей;

- подготовка плана;
- составление таблиц для систематизации учебного материала;
- подготовка ответов на контрольные вопросы;
- заполнение рабочей тетради;
- аналитическая обработка текста;
- подготовка мультимедиа презентации и докладов к выступлению на семинаре (конференции, круглом столе и т.п.);
- подготовка реферата;
- составление библиографии использованных литературных источников;
- разработка тематических кроссвордов и ребусов;
- тестирование и др.;

3) формировать умения:

- решение ситуационных задач и упражнений по образцу;
- выполнение расчетов (графические и расчетные работы);
- решение профессиональных кейсов и вариативных задач;
- подготовка к контрольным работам;
- подготовка к тестированию;
- подготовка к деловым играм;
- проектирование и моделирование разных видов и компонентов профессиональной деятельности;
- опытно-экспериментальная работа;
- анализ профессиональных умений с использованием аудио- и видеотехники и др.

Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности и уровня умений студентов.

Контроль результатов самостоятельной работы студентов должен осуществляться в пределах времени, отведенного на обязательные учебные занятия и внеаудиторную самостоятельную работу студентов по профессиональному модулю (дисциплине), может проходить в письменной, устной или смешанной форме.

Формы самостоятельной работы студента могут различаться в зависимости от цели, характера, профессионального модуля (дисциплины), объема часов, определенных учебным планом: подготовка к лекциям, семинарским, практическим и лабораторным занятиям; изучение учебных пособий; изучение и конспектирование хрестоматий и сборников документов; изучение в рамках программы курса тем и проблем, не выносимых на лекции и семинарские занятия; написание тематических докладов, рефератов и эссе на проблемные темы; аннотирование монографий или их отдельных глав, статей; выполнение исследовательских и творческих заданий; написание контрольных и лабораторных работ; составление библиографии и реферирование по заданной теме.

2 Требования к организации самостоятельной работы студентов при подготовке к аудиторным занятиям

2.1. Подготовка к лекциям

Главное в период подготовки к лекционным занятиям – научиться методам самостоятельного умственного труда, сознательно развивать свои творческие способности и овладевать навыками творческой работы. Для этого необходимо строго соблюдать дисциплину учебы и поведения. Четкое планирование своего рабочего времени и отдыха является необходимым условием для успешной самостоятельной работы.

В основу его нужно положить рабочие программы изучаемых в семестре дисциплин.

Каждому студенту следует составлять еженедельный и семестровый планы работы, а также план на каждый рабочий день. С вечера всегда надо распределять работу на завтрашний день. В конце каждого дня целесообразно подводить итог работы: тщательно проверить, все ли выполнено по намеченному плану, не было ли каких-либо отступлений, а если были, по какой причине это произошло. Нужно осуществлять самоконтроль, который является необходимым условием успешной учебы. Если что-то осталось невыполненным, необходимо изыскать время для завершения этой части работы, не уменьшая объема недельного плана.

Самостоятельная работа на лекции

Слушание и запись лекций – сложный вид аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность студента. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим студентом.

Не надо стремиться записать дословно всю лекцию. Такое «конспектирование» приносит больше вреда, чем пользы. Запись лекций рекомендуется вести по возможности собственными формулировками. Желательно запись осуществлять на одной странице, а следующую оставлять для проработки учебного материала самостоятельно в домашних условиях.

Конспект лекции лучше подразделять на пункты, параграфы, соблюдая красную строку. Этому в большой степени будут способствовать пункты плана лекции, предложенные преподавателям. Принципиальные места, определения, формулы и другое следует сопровождать замечаниями «важно», «особо важно», «хорошо запомнить» и т.п. Можно делать это и с помощью разноцветных маркеров или ручек. Лучше если они будут собственными, чтобы не приходилось просить их у однокурсников и тем самым не отвлекать их во время лекции.

Целесообразно разработать собственную «маркографию» (значки, символы), сокращения слов. Не лишним будет и изучение основ стенографии. Работая над конспектом лекций, всегда необходимо использовать не только учебник, но и ту литературу, которую дополнительно рекомендовал лектор.

Именно такая серьезная, кропотливая работа с лекционным материалом позволит глубоко овладеть знаниями.

2.2 Подготовка отчета по лабораторно-практической работе

Отчёт по лабораторно-практической работедолжен иметь структуру:

- тема;
- цель;
- ход работы;
- результаты работы;
- ответы на контрольные вопросы.

Критерии оценки работы:

- использование рабочего времени;
- уровень знания возможностей конкретной программы;
- последовательность выполнения задания;
- самостоятельность в работе;
- форма фиксации результатов работы;
- достаточная полнота и формулировка ответов на контрольные вопросы.

Работа, сделанная не по своему варианту или копирующая работу другого студента, засчитывается не будет вне зависимости от качества её выполнения.

2.3 Подготовка презентации и доклада

Презентация, согласно толковому словарю русского языка Д.Н. Ушакова: «… способ подачи информации, в котором присутствуют рисунки, фотографии, анимация и звук».

Для подготовки презентации рекомендуется использовать: Libre Office Impress, MS Power Point, текстовый процессор, Acrobat Reader.

Для подготовки презентации необходимо собрать и обработать начальную информацию. Последовательность подготовки презентации:

1. Четко сформулировать цель презентации: вы хотите свою аудиторию мотивировать, убедить, заразить какой-то идеей или просто формально отчитаться.

2. Определить каков будет формат презентации: живое выступление (тогда, сколько будет его продолжительность) или электронная рассылка (каков будет контекст презентации).

3. Отобрать всю содержательную часть для презентации и выстроить логическую цепочку представления.

4. Определить ключевые моменты в содержании текста и выделить их.

5. Определить виды визуализации (картинки) для отображения их на слайдах в соответствии с логикой, целью и спецификой материала.

6. Подобрать дизайн и форматировать слайды (количество картинок и текста, их расположение, цвет и размер).

7. Проверить визуальное восприятие презентации.

К видам визуализации относятся иллюстрации, образы, диаграммы, таблицы. Иллюстрация – представление реально существующего зрительного ряда. Образы – в отличие от иллюстраций – метафора. Их назначение – вызвать эмоцию и создать отношение к ней, воздействовать на аудиторию. С помощью хорошо продуманных и представляемых образов, информация может надолго остаться в памяти человека. Диаграмма – визуализация количественных и качественных связей. Их используют для убедительной демонстрации данных, для пространственного мышления в дополнение к логическому. Таблица – конкретный, наглядный и точный показ данных. Ее основное назначение – структурировать информацию, что порой облегчает восприятие данных аудиторией.

Практические советы по подготовке презентации

- готовьте отдельно: печатный текст + слайды + раздаточный материал;
- слайды – визуальная подача информации, которая должна содержать минимум текста, максимум изображений, несущих смысловую нагрузку, выглядеть наглядно и просто;
- текстовое содержание презентации – устная речь или чтение, которая должна включать аргументы, факты, доказательства и эмоции;
- рекомендуемое число слайдов 10-12;
- обязательная информация для презентации: тема, фамилия и инициалы выступающего; план сообщения; краткие выводы из всего сказанного; список использованных источников;
- раздаточный материал – должен обеспечивать ту же глубину и охват, что и живое выступление: люди больше доверяют тому, что они могут унести с собой, чем исчезающим изображениям, слова и слайды забываются, а раздаточный материал остается постоянным осязаемым напоминанием; раздаточный материал важно раздавать в конце презентации; раздаточный материалы должны отличаться от слайдов, должны быть более информативными.

Доклад, согласно толковому словарю русского языка Д.Н. Ушакова: «… сообщение по заданной теме, с целью внести знания из дополнительной литературы, систематизировать материл, проиллюстрировать примерами, развивать навыки самостоятельной работы с научной литературой, познавательный интерес к научному познанию».

Тема доклада должна быть согласованна с преподавателем и соответствовать теме учебного занятия. Материалы при его подготовке, должны соответствовать научно-методическим требованиям и быть указаны в докладе. Необходимо соблюдать регламент, оговоренный при получении задания. Иллюстрации должны быть достаточными, но не чрезмерными.

Работа студента над докладом-презентацией включает отработку умения самостоятельно обобщать материал и делать выводы в заключении, умения ориентироваться в материале и отвечать на дополнительные вопросы слушателей, отработку навыков ораторства, умения проводить диспут.

Докладчики должны знать и уметь: сообщать новую информацию; использовать технические средства; хорошо ориентироваться в теме всего

семинарского занятия; дискутировать и быстро отвечать на заданные вопросы; четко выполнять установленный регламент (не более 10 минут); иметь представление о композиционной структуре доклада и др.

Структура выступления

Вступление помогает обеспечить успех выступления по любой тематике. Вступление должно содержать: название, сообщение основной идеи, современную оценку предмета изложения, краткое перечисление рассматриваемых вопросов, живую интересную форму изложения, акцентирование внимания на важных моментах, оригинальность подхода.

Основная часть, в которой выступающий должен глубоко раскрыть

суть затронутой темы, обычно строится по принципу отчета. Задача основной части – представить достаточно данных для того, чтобы слушатели заинтересовались темой и захотели ознакомиться с материалами. При этом логическая структура теоретического блока не должны даваться без наглядных пособий, аудио-визуальных и визуальных материалов.

Заключение – ясное, четкое обобщение и краткие выводы, которых всегда ждут слушатели.

2.4. Подготовка к зачету и экзамену

Каждый учебный семестр заканчивается сессией. Подготовка к сессии, выполнение контрольной работы, сдача зачетов и экзаменов является также самостоятельной работой студента. Основное в подготовке к сессии – повторение всего учебного материала междисциплинарного курса, профессионального модуля (дисциплины).

Только тот студент успевает, кто хорошо усвоил учебный материал. Если студент плохо работал в семестре, пропускал лекции, слушал их невнимательно, не конспектировал, не изучал рекомендованную литературу, то в процессе подготовки к сессии ему придется не повторять уже знакомое, а заново в короткий срок изучать весь учебный материал. Все это зачастую невозможно сделать из-за нехватки времени.

Для такого студента подготовка к зачету или экзамену будет трудным, а иногда и непосильным делом, а конечный результат – возможное отчисление из учебного заведения.

3 Требования к студентам при подготовке письменных работ

3.1. Подготовка реферата

Реферат – письменный доклад по определенной теме, в котором собрана информация из одного или нескольких источников. Рефераты пишутся обычно стандартным языком, с использованием типологизированных речевых оборотов вроде: «важное значение имеет», «уделяется особое внимание», «поднимается вопрос», «делаем следующие выводы», «исследуемая проблема», «освещаемый вопрос» и т.п.

К языковым и стилистическим особенностям рефератов относятся слова и обороты речи, носящие обобщающий характер, словесные клише. У

рефератов особая логичность подачи материала и изъяснения мысли, определенная объективность изложения материала.

Признаки реферата

Реферат не копирует дословно содержание первоисточника, а представляет собой новый вторичный текст, создаваемый в результате систематизации и обобщения материала первоисточника, его аналитико-синтетической переработки.

Будучи вторичным текстом, реферат составляется в соответствии со всеми требованиями, предъявляемыми к связанному высказыванию: так ему присущи следующие категории: оптимальное соотношение и завершенность (смысловая и жанрово-композиционная). Для реферата отбирается информация, объективно-ценная для всех читающих, а не только для одного автора. Автор реферата не может пользоваться только ему понятными значками, пометами, сокращениями.

Работа, проводимая автором для подготовки реферата должна обязательно включать самостоятельное мини-исследование, осуществляемое студентом на материале или художественных текстов по литературе, или архивных первоисточников по истории и т.п.

Организация и описание исследования представляет собой очень сложный вид интеллектуальной деятельности, требующий культуры научного мышления, знания методики проведения исследования, навыков оформления научного труда и т.д. Мини-исследование раскрывается в реферате после глубокого, полного обзора научной литературы по проблеме исследования.

В зависимости от количества реферируемых источников выделяют следующие виды рефератов:

- монографические – рефераты, написанные на основе одного источника;
- обзорные – рефераты, созданные на основе нескольких исходных текстов, объединенных общей темой и сходными проблемами исследования.

Структура реферата

1. Титульный лист
2. Содержание
3.Введение
4. Основная часть
5. Заключение
6. Список использованных источников
7. Приложения

Титульный лист является первой страницей и заполняется по строго определенным правилам (Приложение А).

После титульного листа помещают содержание, в котором приводятся все заголовки работы и указываются страницы, с которых они начинаются. Заголовки оглавления должны точно повторять заголовки в тексте. Сокращать их или давать в другой формулировке и последовательности нельзя. Все заголовки начинаются с прописной буквы без точки на конце. Последнее слово каждого заголовка соединяют отточием с соответствующим ему номером

страницы в правом столбце оглавления. Заголовки одинаковых ступеней рубрикации необходимо располагать друг под другом.

Введение к реферату – важнейшая его часть. Здесь обычно обосновывается актуальность выбранной темы, цель и задачи, краткое содержание, указывается объект рассмотрения, приводится характеристика источников для написания работы и краткий обзор имеющейся по данной теме литературы. Актуальность предполагает оценку своевременности и социальной значимости выбранной темы, обзор литературы по теме отражает знакомство автора с имеющимися источниками, умение их систематизировать, критически рассматривать, выделять существенное, определять главное.

Основная часть. Основная часть реферата структурируется по главам и параграфам (пунктам и подпунктам), количество и название которых определяются автором. Содержание глав основной части должно точно соответствовать теме работы и полностью ее раскрывать. Данные главы должны показать умение студента сжато, логично и аргументировано излагать материал, обобщать, анализировать и делать логические выводы. Основная часть реферата, помимо почерпнутого из разных источников содержания, должна включать в себя собственное мнение студента и сформулированные выводы, опирающиеся на приведенные факты.

В основной части реферата обязательными являются ссылки на авторов, чьи позиции, мнения, информация использованы в реферате. Ссылки на источники могут быть выполнены по тексту работы постранично в нижней части страницы (фамилия автора, его инициалы, полное название работы, год издания и страницы, откуда взята ссылка) или в конце цитирования - тогда достаточно указать в квадратных скобках номер литературного источника из списка использованной литературы с указанием конкретных страниц, откуда взята ссылка, (например, [7, с. 67–89]). Номер литературного источника должен указываться после каждого нового отрывка текста из другого литературного источника.

Цитирование и ссылки не должны подменять позиции автора реферата. Излишняя высокопарность, злоупотребления терминологией, объемные отступления от темы, несоразмерная растянутость отдельных глав, разделов, параграфов рассматриваются в качестве недостатков основной части реферата.

Заключительная часть предполагает последовательное, логически стройное изложение обобщенных выводов по рассматриваемой теме. Заключение не должно превышать объем 2 страниц и не должно слово в слово повторять уже имеющийся текст, но должно отражать собственные выводы о проделанной работе, а может быть, и о перспективах дальнейшего исследования темы. В заключении целесообразно сформулировать итоги выполненной работы, краткого и четкого изложить выводы, представить анализ степени выполнения поставленных во введении задач и указать то новое, что лично для себя студент вынес из работы над рефератом.

Список использованных источников составляет одну из частей работы, отражающую самостоятельную творческую работу автора, и позволяет судить о степени фундаментальности данного реферата. В список использованной литературы необходимо внести все источники, которые были изучены студентами в процессе написания реферата.

В работах используются следующие способы построения библиографических списков: по алфавиту фамилий авторов или заглавий; по тематике; по видам изданий; по характеру содержания; списки смешанного построения. Литература в списке указывается в алфавитном порядке (более распространенный вариант – фамилии авторов в алфавитном порядке), после указания фамилии и инициалов автора указывается название литературного источника без кавычек, место издания и название издательства – при города Москва и Санкт-Петербург как место издания обозначаются сокращенно – М.; СПб., название других городов пишется полностью. (М.: Академия), год издания, страницы – общее количество или конкретные.

Список использованных источников, приводится в следующей последовательности: 1) законодательные акты (в хронологическом порядке);

2) статистические материалы и нормативные документы (в хронологическом порядке); 3) литературные источники (в алфавитном порядке) – книги, монографии, учебники и учебные пособия, периодические издания, зарубежные источники, Интернет-источники. Например:

1. Указ Президента РФ “О защите потребителей от недобросовестной рекламы” от 10.06.94 г. № 1183// Российская газета. 1994. 16 июня. № 112.

2. Блинова М.С. Социология миграции: история становления и перспективы развития: учебное пособие/ М.С. Блинова. – М.: КДУ, 2009. – 192 с.

Для работ из журналов и газетных статей необходимо указать фамилию и инициалы автора, название статьи, а затем наименование источника со всеми элементами титульного листа, после чего указать номер страницы начала и конца статьи. Например:

1. Петренко К.В. Демографические характеристики трудового потенциала нефтегазодобывающих регионов Севера России// Научное обозрение. Серия 2. Гуманитарные науки. – М., 2012. – № 5. – С. 85 – 89.

2. Артемьев З. Мигрантам дадут на работу три года// Вечерняя Москва. 2013. № 184

21

(26509).

Для Интернет-источников необходимо указать название работы, источник работы и сайт. Например:

1. О мерах по созданию и развитию малых предприятий [Электронный ресурс]: постановление Совета министров СССР от 8 авг. 1990 г. № 790. – Режим доступа:

[14.05.2012]// http://www.consultant.ru. – Загл. с экрана.

2. Информационные ресурсы справочно-поисковой системы Рамблер -//http:// www.rambler.ru

После списка использованных источников могут быть помещены различные приложения (таблицы, графики, диаграммы, иллюстрации и пр.). В приложение рекомендуется выносить информацию, которая загромождает текст реферата и мешает его логическому восприятию. В содержательной части работы эта часть материала должна быть обобщена и представлена в сжатом виде. На все приложения в тексте реферата должны быть ссылки. Каждое приложение нумеруется и оформляется с новой страницы.

Требования к оформлению реферата

Работа выполняется на компьютере (гарнитура Times New Roman, шрифт 14) через 1,5 интервала с полями: верхнее, нижнее – 2; левое – 3; правое – 1,5. Отступ первой строки абзаца – 1,25. Сноски – постраничные (шрифт 12), их нумерация должна быть сквозной по всему тексту реферата.

Нумерация страниц должна быть сквозной (номер не ставится на титульном листе, но в общем количестве страниц учитывается).

Таблицы и рисунки встраиваются в текст работы, их нумерация должна быть сквозной по всему реферату. Они все должны иметь название и в самом тексте реферата на них должна быть ссылка. (Например: Как следует из таблицы 1 общая численность безработных в первое десятилетие XXI века в разрезе ряда европейских стран резко увеличивалась). После названия таблицы и рисунка точка не ставится.

Общее количество страниц в реферате, без учета приложений, не должно превышать 15 страниц. Значительное превышение установленного объема является недостатком работы и указывает на то, что студент не сумел отобрать и переработать необходимый материал.

В приложении помещают вспомогательные или дополнительные материалы, которые загромождают текст основной части работы (таблицы, рисунки, карты, графики, неопубликованные документы, переписка и т.д.).

Каждое приложение должно начинаться с новой страницы с указанием в центре верхней строки слова «ПРИЛОЖЕНИЕ», иметь номер и тематический заголовок. При наличии в работе более одного приложения они нумеруются русскими буквами (без знака «№»). Нумерация страниц, на которых даются приложения, должна быть сквозной и продолжать общую нумерацию страниц основного текста. Связь основного текста с приложениями осуществляется через ссылки, которые помещаются в круглые скобки – например, (ПРИЛОЖЕНИЕ А).

ПРИЛОЖЕНИЕ А

Пример оформления титульного листа реферата

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

РЕФЕРАТ

по МДК «……»

на тему: « »

Автор работы,
студент гр._____ А.А.Петров

Руководитель,
преподаватель П.П.Иванов

Тула 20__