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 02.02 Инструментальные средства разработки программного обеспечения

профессионального модуля
ПМ.2 Осуществление интеграции программных модулей

специальности СПО
09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «13» января 2023 г. № 6
Председатель цикловой комиссии И.В.Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

Методические указания определяют задачи, цель и организацию курсовой работы по междисциплинарному курсу МДК 02.02 Инструментальные средства разработки программного обеспечения, 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 02.02 Инструментальные средства разработки программного обеспечения является составной часть реализации профессионального модуля ПМ.2 Осуществление интеграции программных модулей ОПОП по специальности СПО 09.02.07 Информационные системы и программирование.

Цель курсовой работы – систематизация, закрепление, углубление и обобщение знаний, полученных при изучении МДК 02.02 Инструментальные средства разработки программного обеспечения.

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 – проектирования, разработка и тестирование программных продуктов по различным предметным областям с использованием инструментальных средств.

Примерные темы курсовой работы:

1. Разработка приложения для малого предприятия связи.
2. Разработка приложения для автотранспортного предприятия.
3. Разработка приложения для учета сдельной оплаты труда
4. Разработка приложения для учета материальных ресурсов предприятия
5. Разработка приложения для автоматизации складского учета

6. Разработка приложения для автоматизации учета платежей по договорам.

7. Разработка приложения для учета поступления и реализации товаров в розничной торговле.

8. Разработка приложения для учета реализации товаров в оптовой торговле.

9. Разработка приложения для автоматизации кассовых операций торгового предприятия.

10. Разработка приложения для учета обмена валют.

11. Разработка приложения для учета запасов предприятия.

12. Разработка приложения для учета риэлтерских операций.

13. Разработка приложения для сотрудника кредитного отдела банка.

14. Разработка приложения для учета ценных бумаг на предприятии.

15. Разработка приложения для учета внутреннего перемещения материалов

16. Разработка приложения для учета операций по импорту товаров.

17. Разработка приложения для автоматизации учета расчетов за проживание в общежитии

18. Разработка приложения для автоматизации учета реализации и затрат на доставку мебели

19. Разработка приложения для специалиста службы технической поддержки пользователей

20. Разработка приложения для инженера-тестировщика ПО.

21. Разработка приложения для работника библиотеки.

22. Разработка приложения для учета комплектующих компьютерной техники.

23. Разработка приложения для учета посещаемости колледжа.

24. Разработка приложения для учета успеваемости обучающихся.

25. Разработка приложения для проверки знаний студентов по предмету…

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 содержать обоснование основных проектных решений, принятых обучающимся на каждом этапе разработки. Решения должны приниматься исходя из особенностей проектируемого продукта и специфики области его применения. Не должно быть обоснований типа «удобнее», «целесообразнее» и т. п. Необходимо пояснить, чем удобнее, почему целесообразно. По возможности необходимо четко формулировать основания для принятия того или иного решения.

Пояснительная записка должна брошюроваться в следующем порядке:
Титульный лист.
Задание на курсовую работу (на бланке).
Аннотация
Содержание
Введение
Анализ задания и выбор технологии, языка разработки.
Разработка диаграммы вариантов использования.
Определение структуры программного продукта.
Описание реализации программного продукта.
Выбор стратегии тестирования и отладка программного средства.

Заключение
Список использованных источников
Приложение А. Техническое задание.

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 и разработки приложения, отражено его назначение. Объем введения не более 5 страниц.

Анализ задания и выбор технологии, языка разработки. В этом разделе записки обосновывается и осуществляется выбор одной из современных технологий программирования. А затем поясняется выбор языка и среды разработки.

Разработка диаграммы вариантов использования. Определяются способы и формы взаимодействия пользователей с системой и разрабатывается диаграмма вариантов использования с применением языка моделирования UML.

Определение структуры программного продукта. В данном разделе проводится анализ предметной области задачи и ее разбиение (декомпозиция) в соответствии с выбранной технологией.

Описание реализации программного продукта. Для программы, при разработке которой использовалась объектно-ориентированная технология, обязательно должна быть разработана диаграмма классов. Для каждого класса нужно указать необходимые атрибуты и операции, соответственно обосновывая их назначение и функции.

В этом же разделе при необходимости можно привести алгоритмы некоторых методов.

Каждый алгоритм должен быть представлен:

– таблицей и (или) списком используемых в нем глобальных переменных;

– схемой алгоритма, использующей имена переменных, приведенных в таблице или списке;

– описанием процесса обработки данных в соответствии с приведенной схемой алгоритма.

Описание каждого алгоритма должно включать:

– функциональное назначение алгоритма;

– входные и выходные данные (результаты выполнения);

– список формальных параметров и их назначение;

– пример вызова модуля или подпрограммы;

– используемые технические средства;

– ссылку на таблицу переменных алгоритма;
– ссылку на рисунок со схемой алгоритма;
– описание процесса обработки данных в соответствии со схемой;
– если имеется приложение с полным текстом программы, то ссылку на соответствующую страницу приложения.

При описании процесса обработки данных в соответствии со схемой алгоритма необходимо пояснить все циклы, каждую альтернативу ветвления, принятое решение по результатам анализа альтернатив и последующие действия. Тексты описания алгоритмов должны быть структурными, предложения короткими. Описание алгоритма должно отражать суть процесса обработки. В зависимости от темы работы по согласованию с руководителем курсовой работы разрабатываются другие виды диаграмм с использованием языка UML – диаграммы взаимодействия, деятельностей, состояний, реализации.

Выбор стратегии тестирования и отладка программного средства.

Данный раздел должен содержать обоснование выбора той или иной стратегии тестирования программного средства, тестовые наборы данных (тесты) по всем частям программного продукта как с использованием правильных входных данных, так и входных данных, не соответствующих принятым ограничениям, а также иллюстрироваться экранными распечатками и комментариями процесса отладки.

Отладка включает в себя поиск ошибки в тексте программного модуля (локализация ошибки) и исправление обнаруженной ошибки. Описывается проведённый анализ ошибок, выявленных в ходе написания, трансляции, тестирования и отладки программного средства. Приводятся распечатки экранных форм, отражающие полученные результаты решения поставленной задачи. Делается вывод о соответствии числовых значений результатов, их точности, форм выдачи и т.д. требованиям поставленной задачи. Можно привести данные статистической отчётности - количество допущенных ошибок (по видам), трудозатраты на разных этапах разработки и отладки модулей программного средства, расход вычислительных ресурсов на отдельных этапах выполнения задания. Описываются обнаруженные некорректные или нерациональные приёмы программирования и программные конструкции, ошибки в программе, ошибки в алгоритме и постановке задачи.

В **заключении** приводятся выводы по разработанному продукту, рекомендации по его использованию и возможные направления дальнейшего усовершенствования.

Техническое задание должно включать следующие разделы:

1. Введение.

Во введении указывается наименование продукта, кратко обосновывается актуальность разработки, дается краткая характеристика области применения программы.

2. Назначение разработки.

В данном разделе указывается, для чего предназначена данная разработка (более подробно, чем во введении).

3. Требования к программе.

Требования к программному продукту подразделяются на группы и указываются в соответствующих разделах:

3.1.Требования к функциональным характеристикам (составу выполняемых программой функций, характеристикам и форме представления входных и выходных данных).

3.2.Требования к надежности (контроль входной и выходной информации, создание резервных копий промежуточных результатов и т. п.).

3.3.Требования к составу и параметрам технических средств (необходимые параметры, используемых ЭВМ - тип микропроцессора, объем памяти, наличие внешних устройств, например, мыши).

3.4.Требования к информационной и программной совместимости (при необходимости здесь задаются методы решения, используемые языки программирования, а также используемая операционная система и другие системные и пользовательские программные средства).

4. Требования к программной документации.

В данном разделе указывается необходимость наличия руководства программиста, руководства пользователя и руководства системного программиста.

Техническое задание утверждается руководителем курсовой работы.

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 мотивы 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 Федорова, Г. Н. Разработка модулей программного обеспечения для компьютерных систем : учебник для среднего профессионального образования / Г. Н. Федорова. 2-е изд., стер. Москва : Академия, 2018. 384 с. : ил. (Профессиональное образование) . ISBN 978-5-4468-6992-3

2 Зубкова, Т. М. Технология разработки программного обеспечения : учебное пособие / Т. М. Зубкова. — Санкт-Петербург : Лань, 2019. — 324 с. — ISBN 978-5-8114-3842-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/122176

3 Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2020. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/453640

4 Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем : учебник для среднего профессионального образования / Е. А. Черткова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 147 с. — (Профессиональное образование). — ISBN 978-5-534-09823-5. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/454414.

Дополнительные источники

1 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 .

2 Лауферман, О. В. Разработка программного продукта: профессиональные стандарты, жизненный цикл, командная работа : учебное пособие / О. В. Лауферман, Н. И. Лыгина. — Новосибирск : Новосибирский государственный технический университет, 2019. — 75 с. — ISBN 978-5-7782-3893-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/99215.html. — Режим доступа: для авторизир. пользователей

3 Долженко, А. И. Технологии командной разработки программного обеспечения информационных систем : курс лекций / А. И. Долженко. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 300 с. — ISBN 978-5-4486-0525-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/79723.html

4 Вичугова, А. А. Инструментальные средства разработки компьютерных систем и комплексов : учебное пособие для СПО / А. А. Вичугова. — Саратов : Профобразование, 2017. — 135 с. — ISBN 978-5-4488-0015-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/66387.html

5 Синицын, С. В. Верификация программного обеспечения : учебное пособие / С. В. Синицын, Н. Ю. Налютин. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 367 с. — ISBN 978-5-4497-0653-9. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/97540.html

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/

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 2023

УТВЕРЖДЕНЫ
Цикловой комиссий информационных технологий
Протокол от «13» января 2023 г. №6
Председатель цикловой комиссии И.В. Миляева

Авторы: Воронцова Н.В., канд. техн. наук

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:

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

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

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

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 α_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

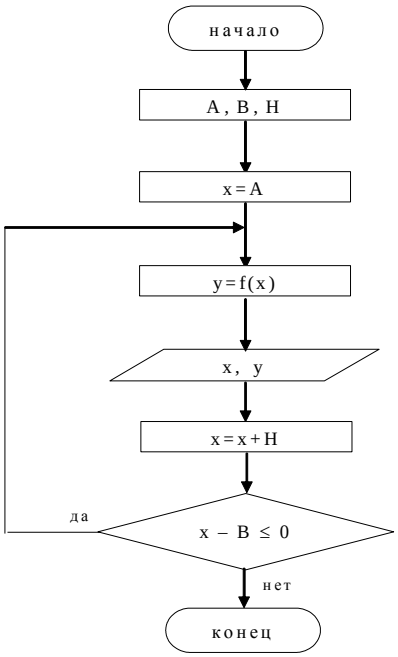


Рис. 1 Схема алгоритма метода табуляции функции y=f(x) на отрезке [A, B] с шагом H.

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.

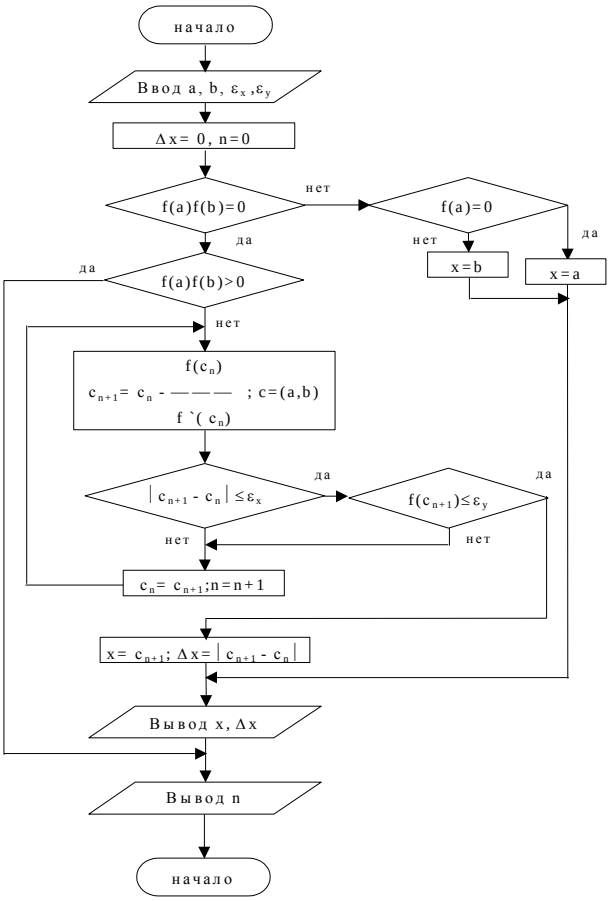


Рис.2. Схема алгоритма уточнения корня уравнения методом касательных с предварительным определением подвижного конца отрезка

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)

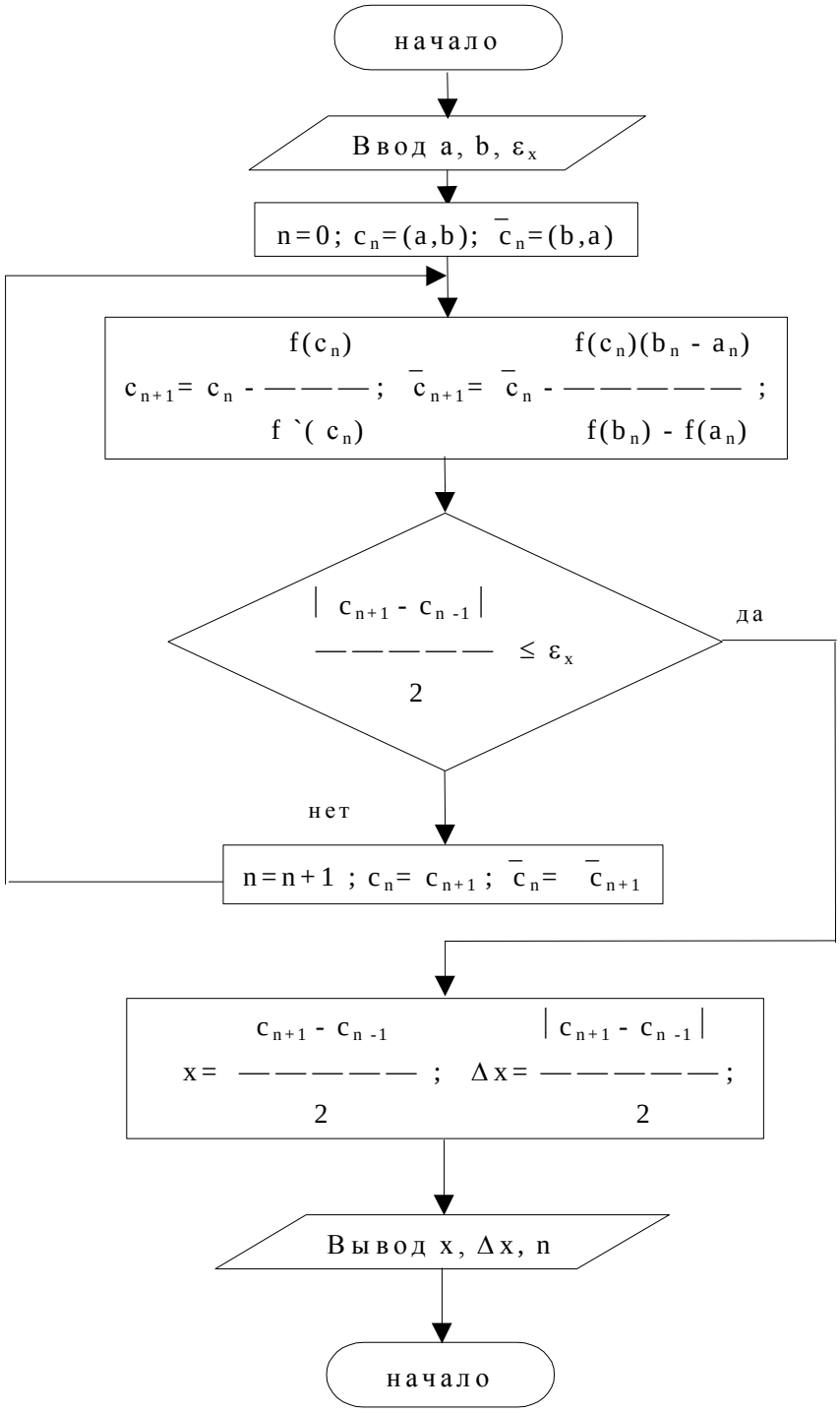


Рис. 3 Упрощённая схема алгоритма уточнения корня уравнения на отрезке отделения совмещённым методом хорд и касательных

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

$$x_1 + b_{12}x_2 + b_{13}x_3 + b_{14}x_4 = b_{15}, \quad (2)$$

$$x_1 = b_{15} - b_{12}x_2 - b_{13}x_3 - b_{14}x_1.$$

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 (3). Разделив коэффициенты первого уравнения системы (1'') на ведущий коэффициент $a_{ij}^{(2)}$

$$x_3 + b_{34}^{(2)} x_4 = b_{35}^{(2)}, \qquad (2'')$$

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 $a_{i1}b_{1j}$ 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:

$$c_{i6}=\sum_{j=1}^{5} a_{ij} \quad (i=1,2,3,4) \qquad b_{i6}=\sum_{j=1}^{5} b_{ij}+1 \quad (i=1,2,3,4)$$

Над контрольными суммами в каждой строке проделываются те же операции, что и над остальными элементами этой строки. При отсутствии

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	$\bar{x}$	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	$x_4 = -1$	$\bar{x}_4 = 0$	V
		1		$x_3 = -1$	$\bar{x}_3 = 0$	
	1			$x_2 = 1$	$\bar{x}_2 = 2$	
1				$x_1 = 1$	$\bar{x}_1 = 2$	

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:

$$a_{11}^{(2)} = a_{11}^{(1)} - a_{12}^{(1)} b_{21}^{(1)} = 2 - 3 \cdot 0{,}5 = 0{,}5,$$
$$a_{45}^{(3)} = a_{45}^{(2)} - a_{43}^{(2)} b_{35}^{(2)} = 0 - (-0{,}5)(-1) = -0{,}5.$$

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.

производятся те же действия, что и над остальными столбцами и в итоге сумма

j -

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:*

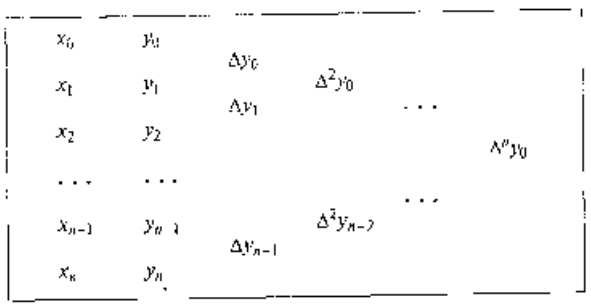
$$\Delta y_i = y_{i+1} - y_i \quad (i = 0,\ 1,\ \ldots,\ n-1).$$

Из конечных разностей первого порядка образуются *конечные разности второго порядка,*

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i \quad (i = 0,\ 1,\ \ldots,\ n-2).$$

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 $a_3 \quad {}^3y_o /3h^3$. 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

$$P_4(x) = 1,715 + \frac{0,855}{0,5}(x-0,5) + \frac{0,146}{2!\cdot 0,5^2}(x-0,5)(x-1) +$$

$$+\frac{1,237}{3!\cdot 0,5^3}(x-0,5)(x-1)(x-1,5) - \frac{1,157}{4!\cdot 0,5^4}(x-0,5)(x-1)(x-1,5)(x-2) =$$

$$= 1,715 + 1,266(x-0,5) + 0,292(x-0,5)(x-1) +$$

$$+1,649(x-0,5)(x-1)(x-1,5) - 0,771(x-0,5)(x-1)(x-1,5)(x-2).$$

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_3	y_3	y_3/P_3
x_2	1,50	−1,14	−2,26	3,40	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 $\int_a^b y\,dx$.

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

$$\int_{x_0=a}^{x_n=b} y\,dx = \int_{x_0=a}^{x_n=b} L_n(x)\,dx = \sum_{i=0}^{n} A_i y_i, \qquad (2)$$

где A_i- некоторые постоянные коэффициенты. Получим их явные выражения. Полином Лагранжа получен ранее.

$$L_n(x) = \sum_{i=0}^{n} (-1)^{n-i} \cdot \frac{q(q-1)(q-2)\ldots(q-n)}{i!(n-i)!(q-i)} \cdot y_i = \sum_{i=0}^{n} (-1)^{n-i} \cdot \frac{q^{[n+1]}}{i!(n-i)!(q-i)} \cdot y_i. \qquad (3)$$

Подставляя (3) в (2), получим:

$$\int_{x_0}^{x_n} y\,dx = \int_{x_0}^{x_n} \sum_{i=0}^{n} \frac{(-1)^{n-i}}{i!(n-i)!} \cdot \frac{q^{[n+1]}}{q-i} \cdot y_i\,dx, \qquad (4)$$

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.

$$x_0 = 0 \quad y_0 = \frac{\cos 0}{1+0} = 1 \quad x_1 = 0.4 \quad y_1 = \frac{\cos 22.5^\circ}{1+0.4} = 0.659 \quad x_2 = 0.8 \quad y_2 = \frac{\cos 45^\circ}{1+0.8} = 0.393$$

$$x_3 = 1.2 \quad y_3 = \frac{\cos 67.5^\circ}{1+1.2} = 0.174 \quad x_4 = 1.6 \quad y_4 = \frac{\cos 90^\circ}{1+1.6} = 0$$

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) $\int_{1.3}^{2.3} \frac{\sqrt{x^2+0.6}dx}{1.4+\sqrt{0.8x^2+1.}}$

№ 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) $\int_{0.8}^{2.4} \frac{\sqrt{1.1x^2+0.9}dx}{1.6+\sqrt{0.8x^2+1.4}}$;

№ 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) $\int_{1.2}^{2.7} \frac{\sqrt{1.3x^2+0.8}dx}{1.7x+\sqrt{2x+0.5}}$;

№ 22. 1) $\int_{0.6}^{1.8} \frac{\sqrt{x^2+0.5}dx}{2x+\sqrt{x^2+2.5}}$;

№ 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) $\int_{1.3}^{2.3} \frac{\sqrt{0.8x^2+1.3}dx}{1.4+\sqrt{x^2+0.6}}$;

№ 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.

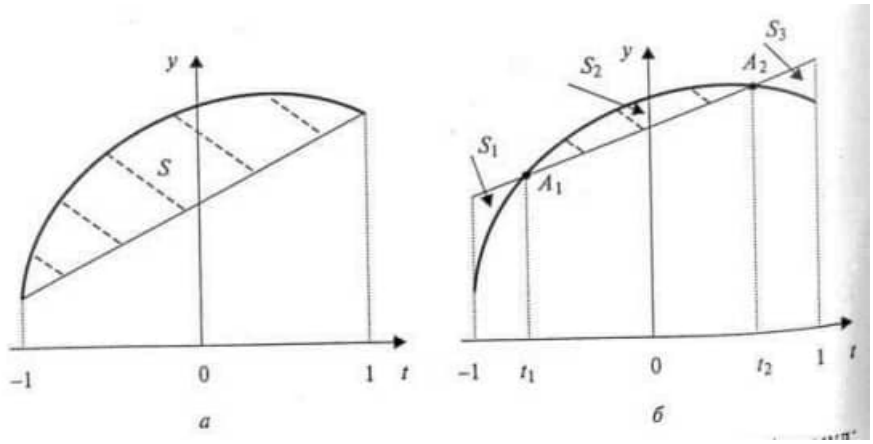


Рисунок 1 - Иллюстрация подходов к построению квадратурных формул:
а — фиксированные узлы линейной интерполяции подынтегральной функции (метод трапеций); погрешность интегрирования характеризуется величиной площади S,
б — подвижные узлы интерполяции (метод Гаусса); величина погрешности зависит от степени 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 $f(x) = x^2 \sin x$

$f^{IV}(x) = (x^2 - 12)\sin x - 8x\cos x.$

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) $\int_{1.2}^{2.0} \frac{\sqrt{2x^2+1.6}\,dx}{2x+\sqrt{0.5x^3+3}}$;

№ 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) $\int_{0.6}^{2.6} \frac{\sqrt{1.1x^2+0.9}\,dx}{1.6+\sqrt{0.8x^2+1.4}}$;

№ 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) $\int_{1.2}^{2} \frac{\sqrt{0.7x^2+1}\,dx}{2.1x+\sqrt{0.6x+1.7}}$;

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 $M_0(xo, yo)$- Двигаясь вдоль этой касательной (рисунок 4), получим приближенное значение решения в точке x_1:

$$y_1 = y_0 + hf(x_0, y_0). \qquad (8)$$

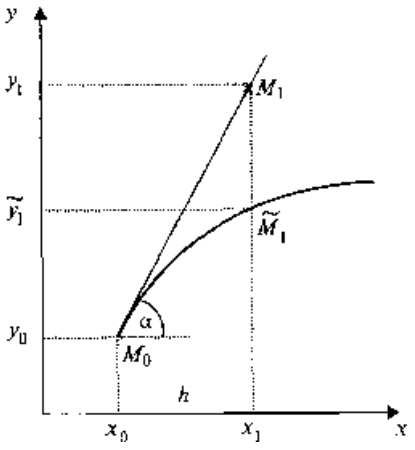


Рисунок 4 - Иллюстрация первого шага метода Эйлера

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 $tg\beta = f(x_1, y_1)$, 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:

$$\alpha_2,\ldots,\ \alpha_q,\ldots,\ p_1,\ldots,\ p_q, b_{ij}, \qquad 0 < j < i \le q .$$

Последовательно полагаем:

$$\begin{aligned} &k_1(h) = hf(x, y),\\ &k_2(h) = hf\left(x + \alpha_2 h, y + p_{2,1}k_1(h)\right),\\ &\ldots\ldots\ldots\ \ldots\ldots\ldots\ \ldots\ldots\\ &k_q(h) = hf\left(x + \alpha_q h, y + p_{q,1}k_1(h) + \ldots + \beta_{q,q-1}k_{q-1}(h)\right), \end{aligned} \tag{1}$$

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 $k_1(h),\ldots, k_q(h)$ и φ(h) – гладкие функции параметра h. Предположим также, что

$$\begin{aligned} &R(0) = R'(0) = \ldots = R^{(s)}(0) = 0,\\ &R^{(s+1)}(0) \ne 0. \end{aligned} \tag{3а}$$

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:

$$\begin{aligned} &R(h) = y(x + h) - y - p_1 hf(x, y);\\ &R(0) = 0;\\ &R'(0) = \left[y'(x + h) - p_1 f(x, y)\right]\Big|_{h=0} = f(x, y)(1 - p_1);\\ &R''(h) = y''(x + h). \end{aligned} \tag{4}$$

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:

$$y'=f,\quad y''=f_x+f_yf,\quad y'''=f_{xx}+2f_{xy}f+f_{yy}f^2+f_yy''. \qquad (10)$$

Подставим в формулы (7 – 9) значение h=0 и воспользуемся для упрощения формулы (10). В результате получим:

$$R(0)=y-y=0,$$

$$R'(0)=(1-p_1-p_2)f(x,y), \qquad (11)$$

$$R''(0)=(1-2p_2\alpha_2)f_x(x,y)+(1-2p_2\beta_{21})f_y(x,y)f(x,y), \qquad (12)$$

$$R'''(0)=\left(1-3p_2\alpha_2^2\right)f_{xx}(x,y)+(2-6p_2\alpha_2\beta_{21})f_{xy}(x,y)f(x,y)+$$
$$+\left(1-3p_2\beta_{21}^2\right)\cdot f_{yy}(x,y)(f(x,y))^2+f_y(x,y)y''(x). \qquad (13)$$

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

$$\left|\hat{y}_i - y(x_i)\right| \le \frac{1}{15}\left|\hat{y}_i - y_i\right|, \qquad (25)$$

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

$$h^4 < E. \qquad (26)$$

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:

$$\left.\begin{aligned} \frac{dy_1}{dx} &= \varphi_1(x, y_1, y_2, \ldots, y_n) \\ \frac{dy_2}{dx} &= \varphi_2(x, y_1, y_2, \ldots, y_n) \\ \ldots\ldots\ldots\ &\ldots\ldots\ldots\ \ldots\ldots\ldots\ \ldots\ldots\ldots \\ \frac{dy_n}{dx} &= \varphi_n(x, y_1, y_2, \ldots, y_n) \end{aligned}\right\} \text{(A)}$$

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.

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

ПО ДИСЦИПЛИНЕ

ОП.11 КОМПЬЮТЕРНЫЕ СЕТИ

специальности СПО
09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЕНЫ

Цикловой комиссией информационных технологий

Протокол от «13» января 2023 г. № 6

Председатель цикловой комиссии _____И.В. Миляева

Авторы: Баранова Е.М., преподаватель, канд. техн. наук

Лабораторная работа №1
Работа с эмулятором сетей Cisco Packet Tracer.
Добавление компьютеров в существующую сеть

Задачи

- Настройка компьютеров для использования DHCP.
- Настройка статической адресации.
- Использование команды ipconfig для получения сведений о параметрах IP узла.
- Использование команды ping для проверки связи.

Совет. *Чтобы во время выполнения интерактивного задания инструкции оставались видимыми, установите флажок* ***"Top"*** *(поверх) в нижнем левом углу окна с указаниями.*

Исходные данные

В этом упражнении вам необходимо добавить два компьютера к сети филиала. В компании для динамической адресации всех ПК используется DHCP.

Этап 1. Изучение топологии

В топологии отображаются два ПК, коммутатор, сервер, маршрутизатор и облако. Рассмотрим ПК.

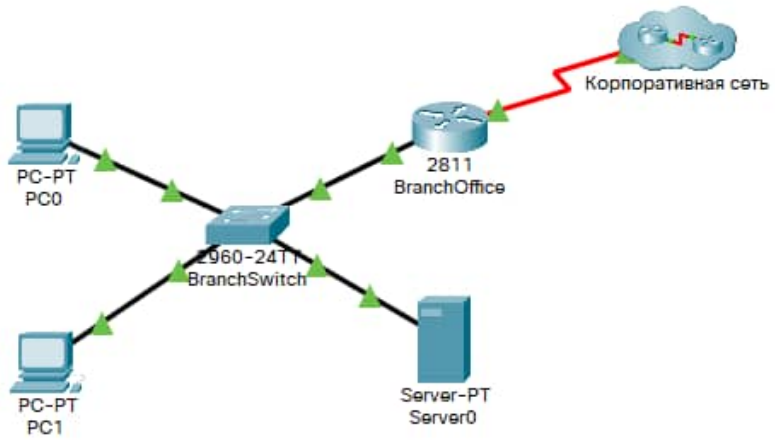


Рисунок 1 – Схема сети

a. Обратите внимание, что ПК подключены к коммутатору **BranchSwitch** (Коммутатор филиала) с помощью прямых кабелей. В Packet Tracer прямые соединительные кабели Ethernet обозначаются сплошной линией.

b. Обратите внимание на зеленые точки на концах прямых соединительных кабелей (рядом с каждым ПК и рядом с коммутатором **BranchSwitch**). Зеленые точки на обоих концах кабеля

обозначают, что для соединения этих устройств был выбран правильный тип кабеля.

Примечание. Зеленые точки должны появиться на обоих концах каждого кабельного подключения. Если зеленые точки отсутствуют, перейдите по вкладкам "**Options** > **Preferences**" (Сервис > Настройки) в меню Packet Tracer и установите флажок "**Show Link Lights**" (Показывать индикаторы линка).

Этап 2. Настройка DHCP на ПК

a. Щёлкните узел **PC0** (ПК0). Появится окно **PC0**.

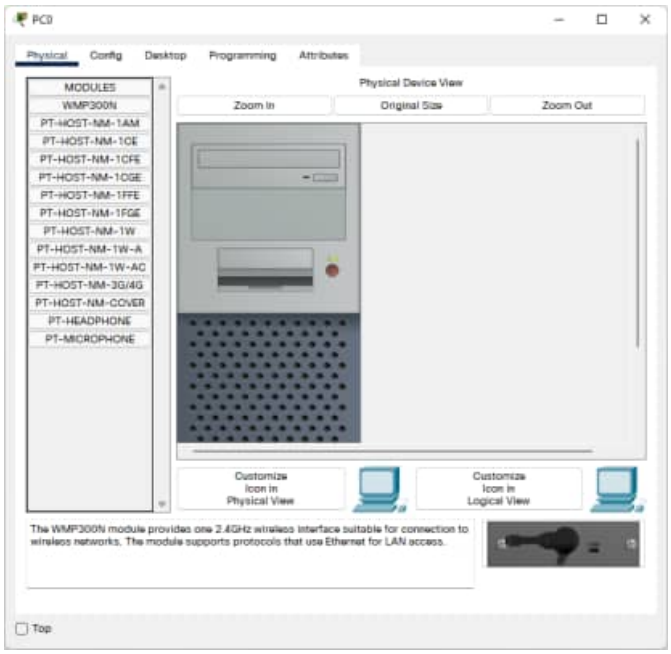


Рисунок 2 – Окно настройки ПК

b. В окне **PC0** выберите вкладку **Desktop** (Рабочий стол).

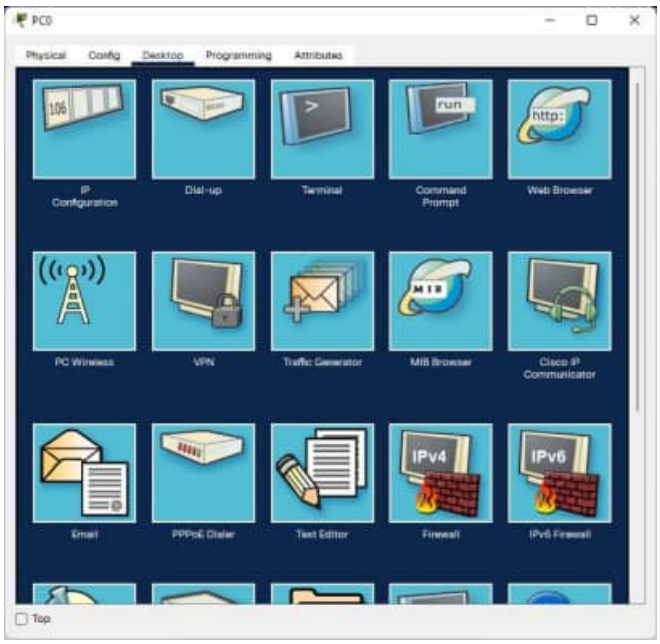


Рисунок 3 – Вкладка Desktop

c. Щёлкните пункт **IP Configuration** (Настройка IP) и выберите кнопку **DHCP**, чтобы узел мог выступать в качестве клиента DHCP. Клиент DHCP динамически получит сведения о настройке IP-адреса с сервера DHCP. (После нажатия кнопки **DHCP** появится следующее сообщение: *DHCP request successful (Запрос DHCP выполнен успешно)*.)

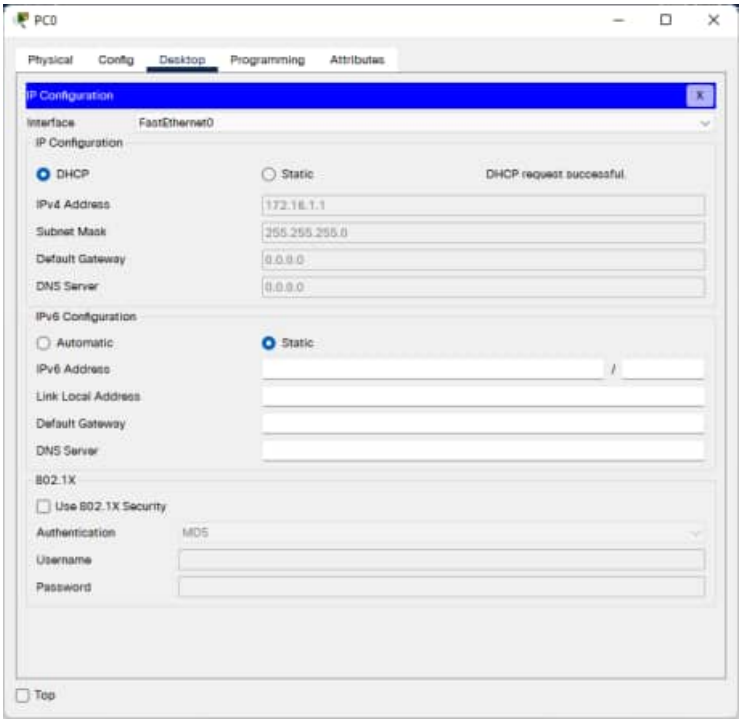


Рисунок 4 – Окно IP Configuration

d. Закройте окно настройки **PC0**, нажав кнопку **x** в правом верхнем углу окна.

e. Щёлкните узел **PC1** (ПК1). Появится окно **PC1**.

f. В окне **PC1** выберите вкладку **Desktop** (Рабочий стол).

g. Щёлкните пункт **IP Configuration** (Настройка IP) и выберите кнопку **DHCP**, чтобы узел мог выступать в качестве клиента DHCP.

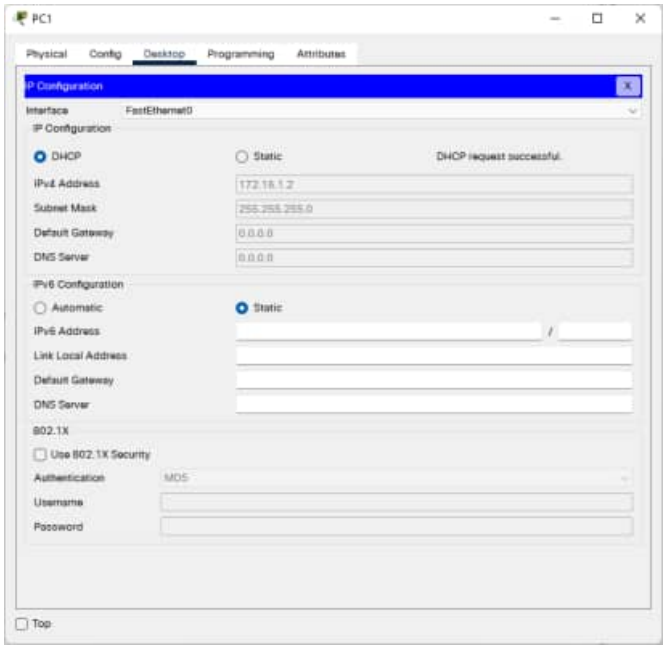


Рисунок 5 – Окно настройки IP Configuration для PC1

h. Закройте окно настройки **PC1**.

Этап 3. Ознакомление со сведениями о настройке IP для каждого ПК

a. Щёлкните **PC0**.

b. Выберите вкладку **Desktop** (Рабочий стол).

c. Щёлкните **Command Prompt** (Командная строка).

d. В командной строке **PC>** введите команду **ipconfig /all** .

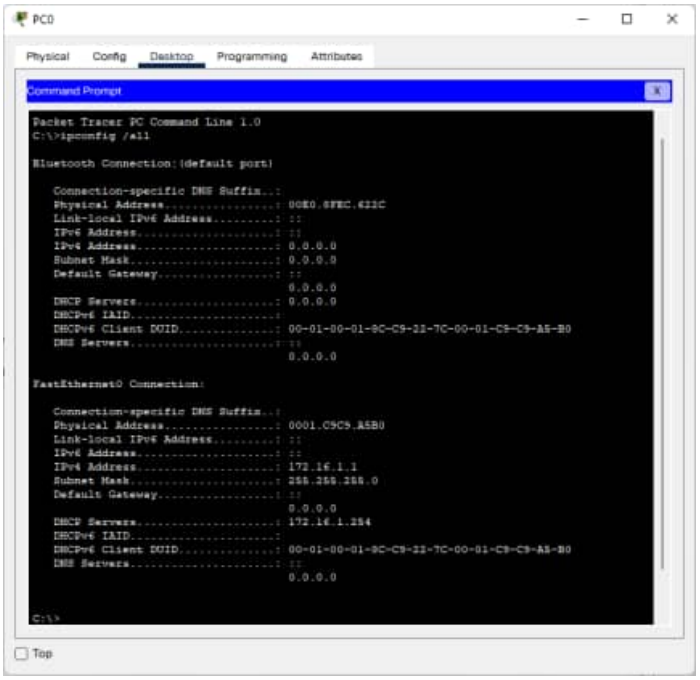


Рисунок 6 – Результат ввода команды ipconfig /all в окно Command Prompt

e. Запишите IP-адрес, маску подсети, шлюз по умолчанию и адрес сервера DNS, которые были динамически назначены по DHCP для **PC0**.

f. Запишите IP-адрес, маску подсети, шлюз по умолчанию и адрес сервера DNS, которые были динамически назначены по DHCP для **PC1**.

g. При помощи команды **ping** проверьте связь на уровне 3 между компьютерами и используемым по умолчанию маршрутизатором.
В командной строке **PC0>** введите команду **ping <IP-адрес PC1>**

В командной строке **PC0>** введите команду **ping <IP-адрес маршрутизатора>**

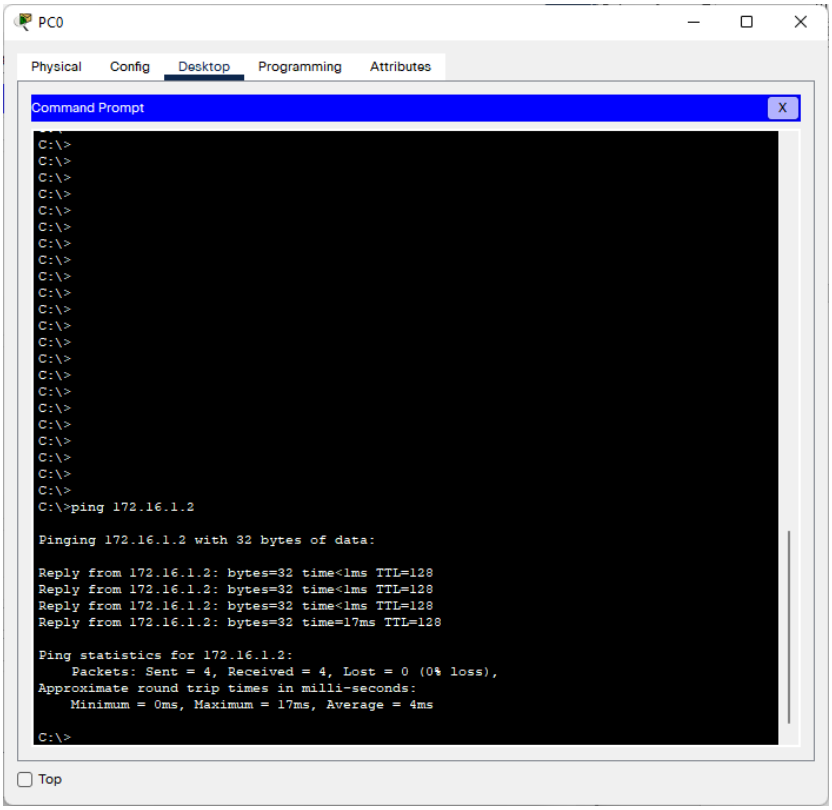


Рисунок 7 – Результат ввода команды ping в окне Command Prompt

В командной строке **PC1>** введите команду **ping <IP-адрес PC0>**

В командной строке **PC1>** введите команду **ping <IP-адрес маршрутизатора>**

Этап 4. Вопросы

Согласно топологии, изначально для подключения **PC0** и **PC1** к **BranchSwitch** использовались прямые кабели. Предположив, что вместо этого для подключения **PC1** к **BranchSwitch** был использован перекрестный кабель, ответьте на следующие вопросы:

a. Может ли в таком случае **PC1** получить IP-адрес через DHCP?
b. Почему?
c. Как будет обстоять дело с **PC0**? Получит ли он IP-адрес через DHCP, если **PC1** подключен к **BranchSwitch** перекрестным кабелем?

Этап 5. Переход на статическую адресацию

Несмотря на все преимущества таких динамических схем адресации, как DHCP, иногда необходимо использовать статическую схему. Измените настройку **PC1** с адресации через DHCP на статическую адресацию.

a. Щёлкните **PC1**, чтобы открыть его окно настройки.
b. Щёлкните вкладку **Desktop** (Рабочий стол).
c. Выберите пункт **IP Configuration** (Настройка IP).
d. Выберите пункт **Static** (Статическая).
e. Введите данные IP, как указано ниже.

IP-адрес: 172.16.1.20

Маска подсети: 255.255.255.0
Шлюз по умолчанию: 172.16.1.254
DNS: 200.75.100.10

f. **PC1** теперь настроен на статический адрес.

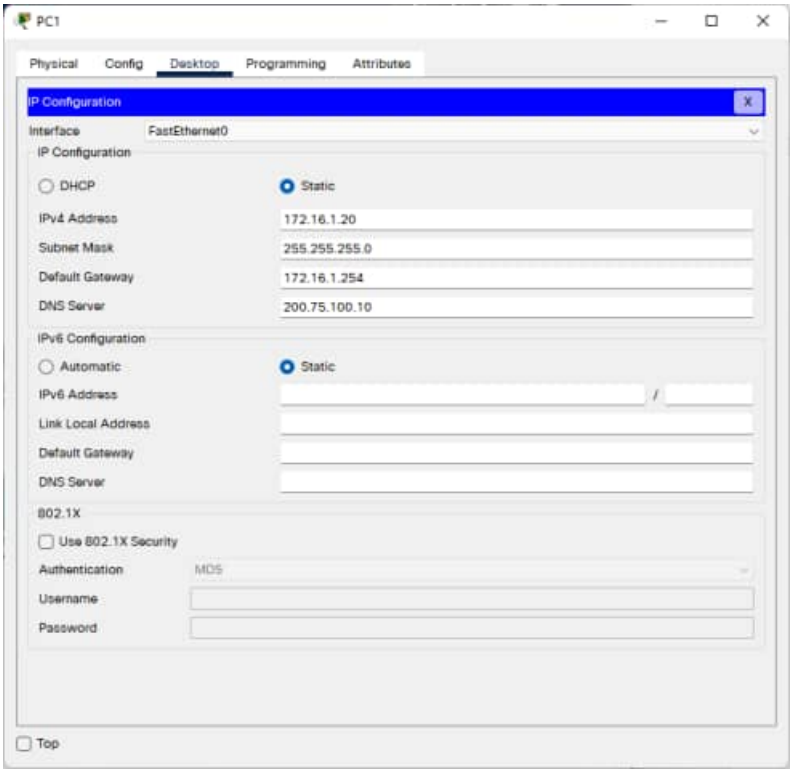


Рисунок 8 – Окно IP Configuration для PC1

g. Закройте вкладку **IP Configuration** (Настройка IP).

Этап 6. Проверка связи

Проверьте связь, отправив эхо-запросы по сети.

a. Щёлкните **PC1**, чтобы открыть его окно настройки.
b. Щёлкните **Desktop** (Рабочий стол).
c. Щёлкните вкладку **Command Prompt** (Командная строка).
d. Отправьте эхо-запрос на основной шлюз с помощью команды **ping 172.16.1.254**. Эхо-тестирование должно пройти успешно.

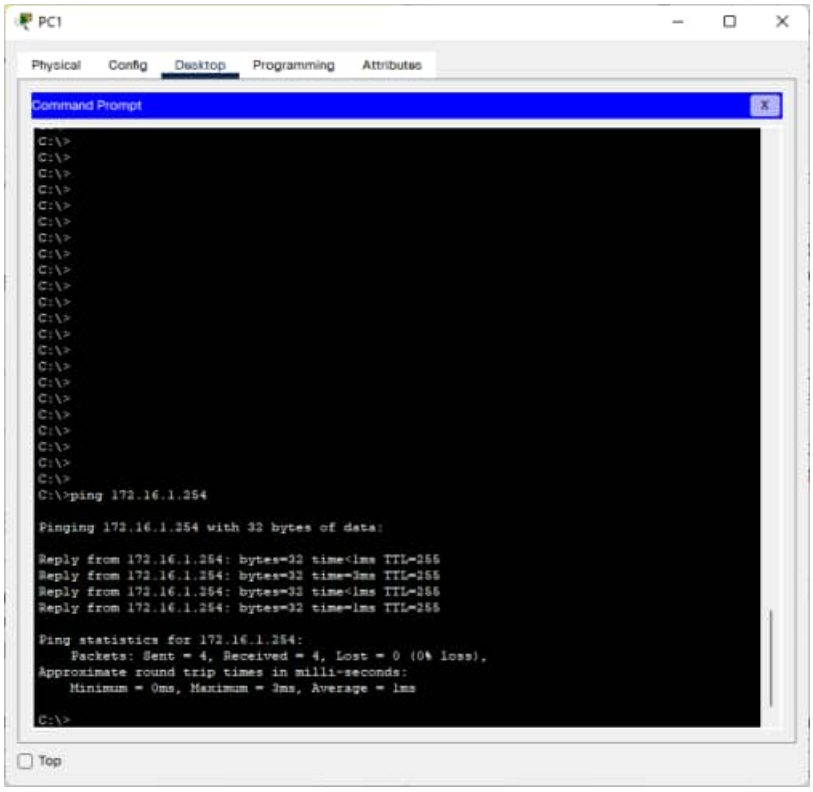


Рисунок 9 – Результат ввода команды ping в окне Command Prompt для PC1

e. Отправьте эхо-запрос на **Server0** с помощью команды **ping 172.16.1.100**. Эхо-тестирование должно пройти успешно.
f. Отправьте эхо-запрос на маршрутизатор, используемый в качестве точки входа в облако **Corporate** (Корпоративное), с помощью команды **ping 172.16.200.1**. Эхо-тестирование должно пройти успешно.
g. Отправьте эхо-запрос на сервер, размещенный внутри облака **Corporate** (корпоративное), с помощью команды **ping 200.75.100.10**. Эхо-тестирование должно пройти успешно.
h. Полная связь в пределах сети достигнута.

Проверьте свой результат. Он должен составлять 100%.

Лабораторная работа №2
Работа с эмулятором сетей Cisco Packet Tracer.
Подключение к беспроводному маршрутизатору и настройка основных параметров

Цели обучения

- Настройка компьютера для подключения к беспроводной сети.
- Проверка беспроводного подключения.

Введение

В этом интерактивном задании необходимо настроить беспроводной маршрутизатор Linksys WRT300N для подключения к нему беспроводного клиента **CompanyLaptop** (Портативный компьютер компании) и осуществления маршрутизации его IP пакетов.

Задача 0. Изменение отображаемого имени WRT300N

Этап 1. Изменение имени WRT300N

a. Щёлкните устройство **WRT300N** и перейдите на вкладку **Config** (Настройка).
b. Измените **Display Name** (Отображаемое имя) на **WRS1**.

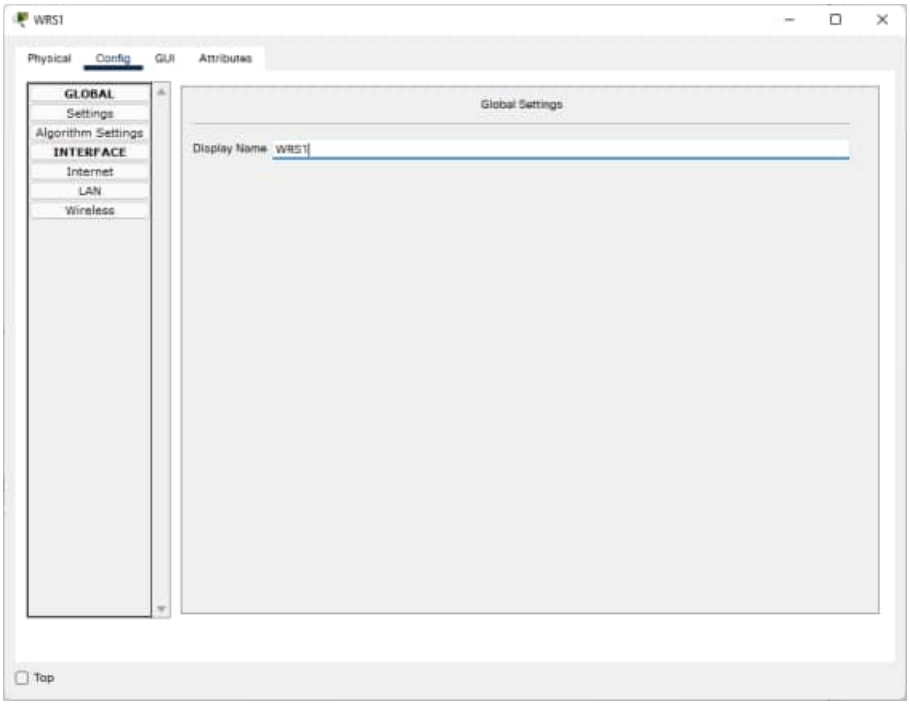


Рисунок 10 – Изменение отображаемого имени во вкладе Config устройства WRT300N

c. Закройте окно устройства **WRS1**.
d. Убедитесь, что в топологии отображается имя **WRS1**.

Задача 1. Подготовка сети

Этап 1. Подготовка сети

Примечание. В этом упражнении с целью приближения к реальной ситуации не будем обращать внимания на наличие вкладки **WRT300N GUI** (Графический интерфейс пользователя маршрутизатора WRT300N). Управляющий компьютер (**PC0**) готовится к получению доступа к беспроводному маршрутизатору Linksys через проводное подключение. Для выполнения всей настройки будет использован веб-браузер, работающий на компьютере **PC0**.

a. Выберите пункт **Connections** (Подключения) в левом нижнем углу окна **Packet Tracer**.

b. Выберите тип кабеля **Copper Straight-Through** (Медный прямой) (сплошная черная линия).

c. Когда курсор перейдет в режим подключения, щёлкните **PC0** и выберите **FastEthernet**.

d. Выберите устройство Linksys Wireless Router (Беспроводной маршрутизатор Linksys) , а затем порт **Ethernet 1**.

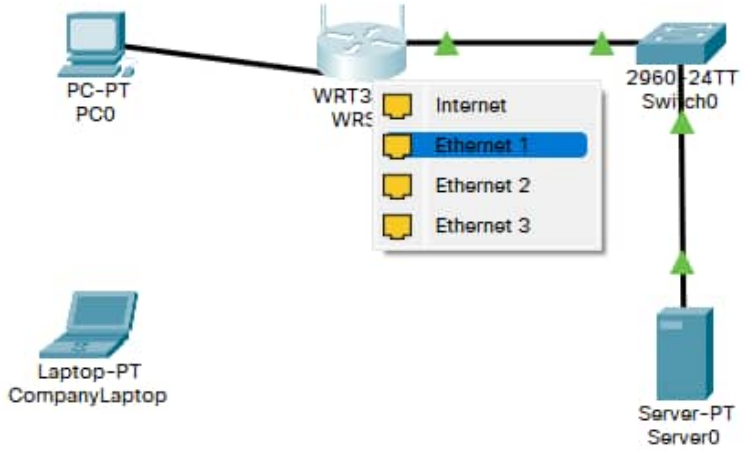


Рисунок 11 – Подключение ПК к маршрутизатору Linksys

Обратите внимание, что маршрутизатор WRT300N имеет 2 сетевых сегмента: **внутренний** и **межсетевой**. Порты **Ethernet 1-4** и **Wireless** (Беспроводной) принадлежат к **внутреннему** сегменту, тогда как порт **internet** (Межсетевой) входит в сегмент **Интернет**. **WRS1** будет использоваться как коммутатор уровня 2 для устройств, подключенных к его внутреннему сегменту, и как коммутатор уровня 3 между двумя сегментами. **PC0** теперь

подключен к внутреннему сегменту (**Ethernet 1**). После того, как в окне **Packet Tracer** появятся зеленые точки на обоих концах подключения между устройствами **PC0** и **WRS1**, перейдите к задаче 2.

Примечание. Если зеленые точки не появились, убедитесь в том, что включен режим **Show Link Lights** (показывать индикаторы связи) в меню **Options > Preferences** (Сервис > Параметры).

Задача 2. Подготовка PC0

Чтобы можно было перейти к странице управления **WRS1,** устройство **PC0** должно успешно осуществлять обмен данными по сети. В состав заводской настройки маршрутизатора Linksys входит сервер DHCP. Этот сервер работает по умолчанию на участке внутренней локальной сети маршрутизатора. Чтобы убедиться, что устройство **PC0** получает IP-адрес с коммутатора **WRS1**, выполните настройку **PC0** для получения данных IP через DHCP.

Этап 1. Настройка PC0 для использования DHCP

a. Щёлкните устройство **PC0**.

b. Выберите вкладку **Desktop** (Рабочий стол).

c. Выберите пункт **IP Configuration** (Настройка IP).

d. Выберите **DHCP**.

e. Укажите IP-адрес данного компьютера.

192.168.0.100

f. Укажите маску подсети данного компьютера.

255.255.255.0

g. Укажите основной шлюз данного компьютера.

192.168.0.1

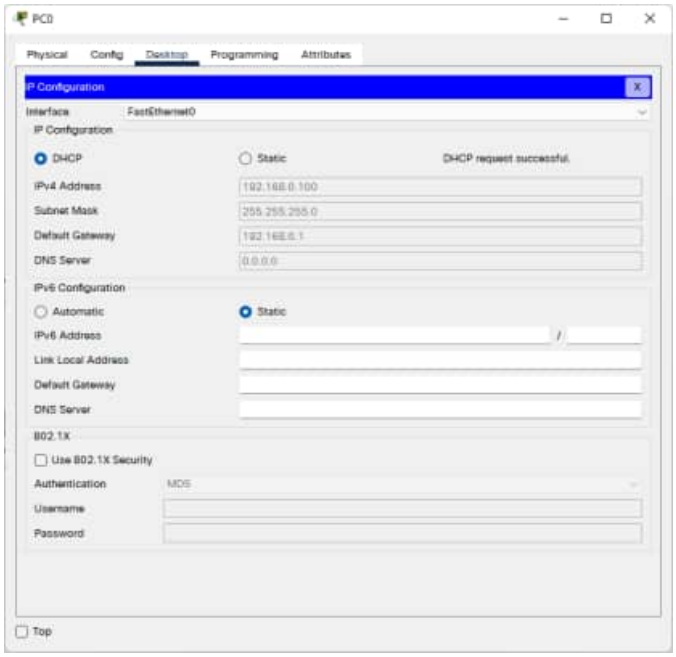


Рисунок 12 – Полученные данные подключения на PC0

Примечание. Значения могут меняться в допустимых для сети пределах при нормальном режиме работы DHCP.

Задача 3. Подключение к беспроводному маршрутизатору

Этап 1. Вход на беспроводной маршрутизатор

a. Закройте окно **IP Configuration** (Настройка IP).

b. На вкладке **Desktop** (Рабочий стол) в меню **PC0** щёлкните пункт **Web Browser** (Веб-браузер).

c. Введите IP-адрес беспроводного маршрутизатора **192.168.0.1**.

d. При запросе имени пользователя и пароля введите **admin** в оба поля. Эти имя пользователя и пароль являются заводскими параметрами по умолчанию для всех продуктов Linksys.

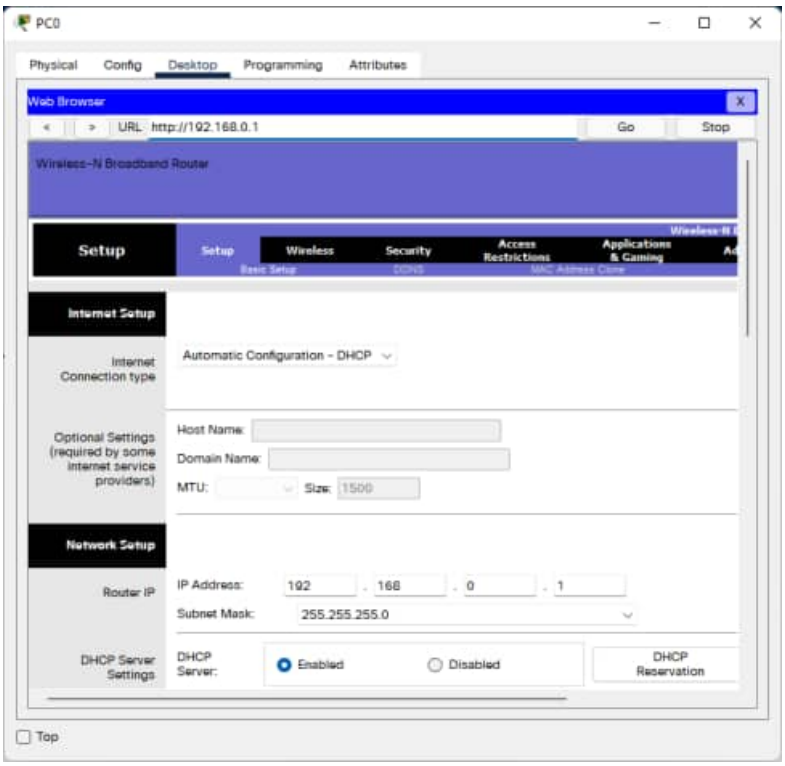


Рисунок 13 –\ Страница веб-настройки WRS1

e. После загрузки страницы веб-настройки **WRS1** перейдите к этапу 2.

Этап 2. Анализ главной страницы настройки WRS1

На главной странице отображаются сетевые параметры маршрутизатора. Прокрутите страницу вниз, обращая внимание на то, что сервер DHCP уже включен (Заводская настройка), и диапазон IP-адресов уже обслуживается маршрутизатором через сервер DHCP.

a. Диапазон IP-адресов, обслуживаемых маршрутизатором **WRS1** через сервер DHCP: **192.168.0.100/24**. Входит ли в этот диапазон IP-адрес устройства **PC0**? Ожидается ли это?

   Да. Устройство PC0 имеет IP-адрес 192.168.0.101/24, который входит в диапазон 192.168.0.100/24. Следовательно, устройство PC0 получило данные IP от маршрутизатора WRS1 посредством сервера DHCP.

Этап 3. Настройка порта Интернета маршрутизатора WSR1

Поскольку маршрутизатор **WRS1** будет направлять пакеты данных беспроводных клиентов к удалённым сетям, необходимо выполнить настройку его порта **Интернет**. Для продуктов Linksys этот интерфейс называется интерфейсом Интернета, поскольку он обычно подключен к внешней сети. В этом упражнении необходимо подключить этот интерфейс к сегменту сети, содержащему **Server0**.

a. Измените способ настройки IP-адреса в сети Интернет с **Automatic Configuration – DHCP** (Автоматическая настройка – DHCP) на **Static IP** (Статический IP-адрес).

b. Введите IP-адрес, чтобы назначить его интерфейсу Интернета:

   IP-адрес интерфейса Интернета: 192.168.20.1

   Маска подсети: 255.255.255.0

   Основной шлюз: 192.168.20.10

   Остальное оставьте без изменений.

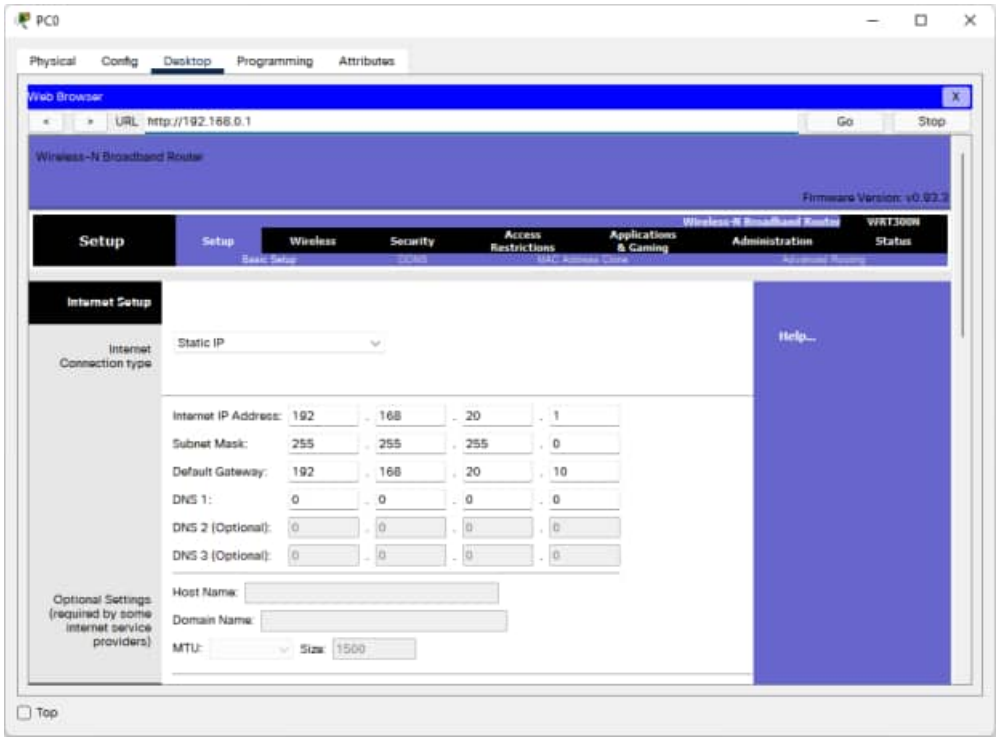


Рисунок 14 – Данные подключения для интерфейса Интернет

c. Прокрутите вниз страницу и нажмите кнопку **Save Settings** (Сохранить параметры).

d. Нажмите **Continue** (Продолжить) и перейдите к этапу 4.

Этап 4. Настройка имени сети (SSID) маршрутизатора WSR1

a. Из веб-обозревателя компьютера **PC0** подключитесь к маршрутизатору **WRS1** (дополнительные сведения приведены в **задаче 3, этап 1**).

b. Перейдите по вкладкам **Wireless** > **Basic Wireless Settings** (Беспроводная сеть > Основные настройки беспроводной сети).

c. Смените **имя сети (SSID)** с **linksys** на **aCompany**. Имена сети (SSID) вводятся с учетом регистра.

d. Оставьте остальные настройки на этой странице по умолчанию.

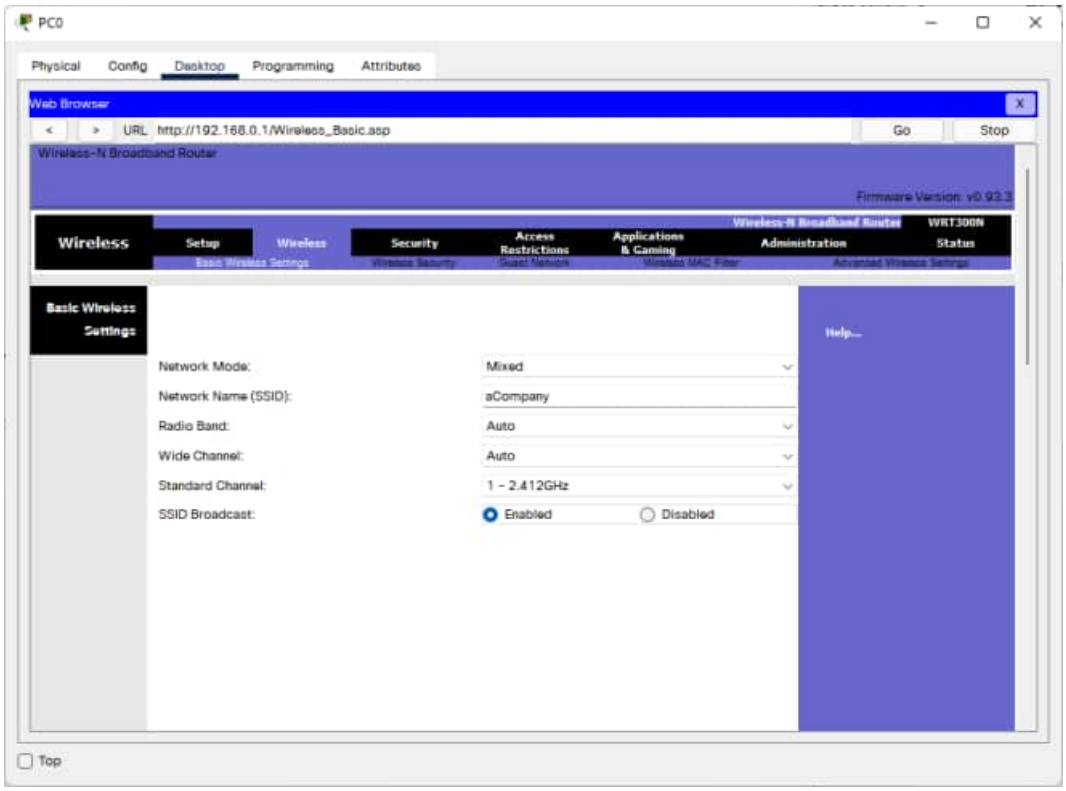


Рисунок 15 – Вкладка настроек беспроводной сети

e. Прокрутите страницу вниз и нажмите кнопку **Save Settings** (Сохранить параметры).

f. **Laptop0** теперь должен быть связан с **WRS1**.

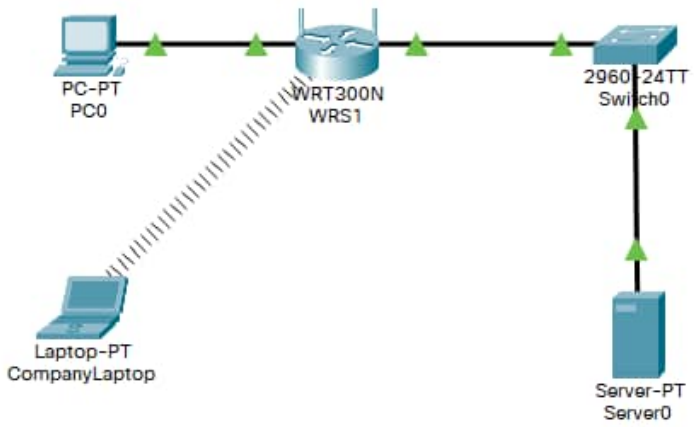


Рисунок 16 – Схема сети после настройки маршрутизатора Linksys

Задача 4. Расширенная настройка

Этап 1. Изменение пароля доступа к WRS1

a. Из веб-обозревателя компьютера **PC0** подключитесь к маршрутизатору **WRS1** (дополнительные сведения приведены в **задаче 3, этап 1**).

b. Перейдите по вкладкам **Administration** > **Management** (Администрирование > Управление) и измените текущий пароль **WRS1** на **cisco**.

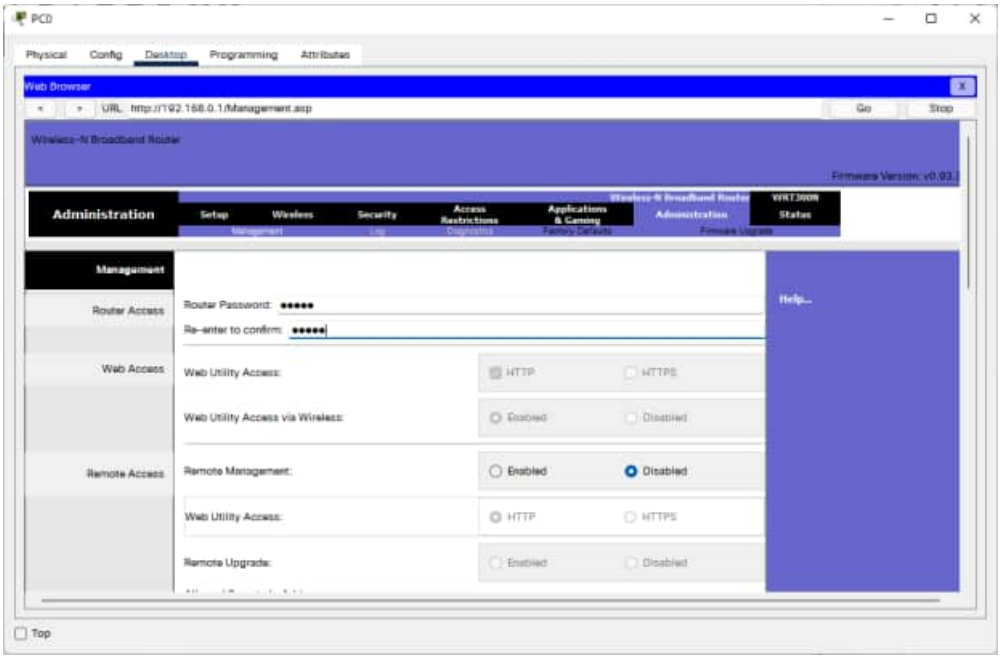


Рисунок 17 – Настройка доступа к маршрутизатору

c. Прокрутите окно вниз и нажмите кнопку **Save Settings** (Сохранить параметры).

d. Появится страница с сообщением **Settings are successful** (Параметры успешно сохранены). Нажмите **Continue** (Продолжить).

e. Появится окно входа. Выполните вход повторно, используя новый пароль **cisco**.

Этап 2. Изменение диапазона адресов DHCP в WRS1

Поскольку IP-адрес, назначенный интерфейсу внутренней локальной сети маршрутизатора **WRS1**, должен входить в диапазон IP-адресов, используемых DHCP, следовательно, IP-адрес этого интерфейса должен быть изменен.

a. Из веб-обозревателя компьютера **PC0** подключитесь к маршрутизатору **WRS1** (дополнительные сведения приведены в **задаче 3, этап 1**).

b. Перейдите к пункту **Setup > Basic Setup** (Настройка > Основная настройка).

c. Прокрутите вниз страницу до раздела **Network Setup** (Настройка сети).

d. Для внутренней локальной сети **WRS1** назначен IP-адрес 192.168.0.1/24. Измените его на 192.168.50.1/24.

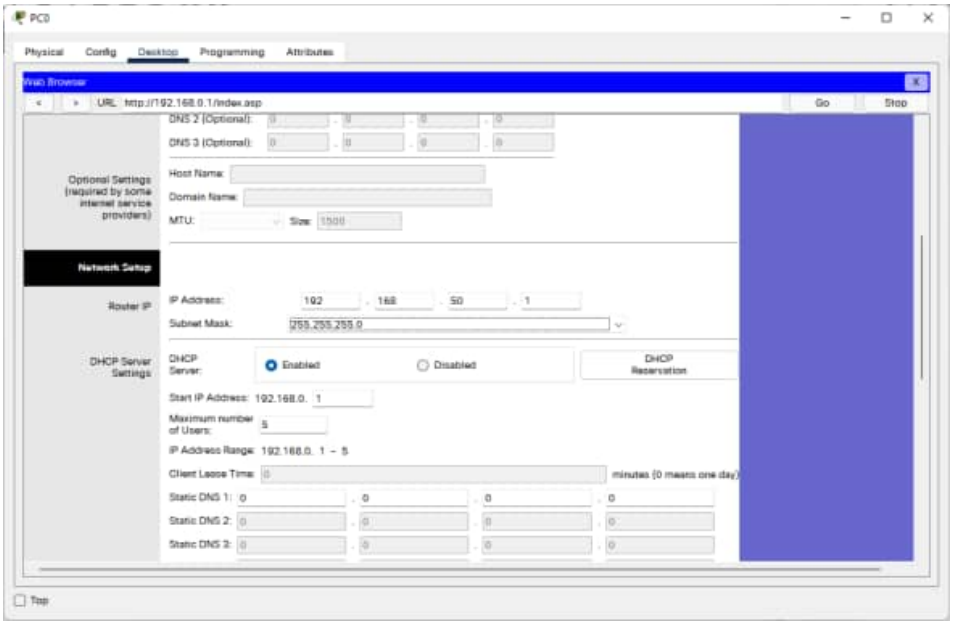


Рисунок 18 – Настройка локальной сети и DHCP-сервера маршрутизатора

e. Прокрутите страницу вниз и нажмите кнопку **Save Settings** (Сохранить параметры).

f. Обратите внимание, что диапазон адресов DHCP автоматически обновился, отображая изменения IP-адреса интерфейса.

g. Закройте веб-браузер **PC0**.

h. На вкладке **PC0 Desktop** (Рабочий стол ПК0) щёлкните **Command Prompt** (Командная строка).

i. Введите команду **ipconfig /renew** , чтобы компьютер **PC0** получил новые сведения об IP с сервера DHCP.

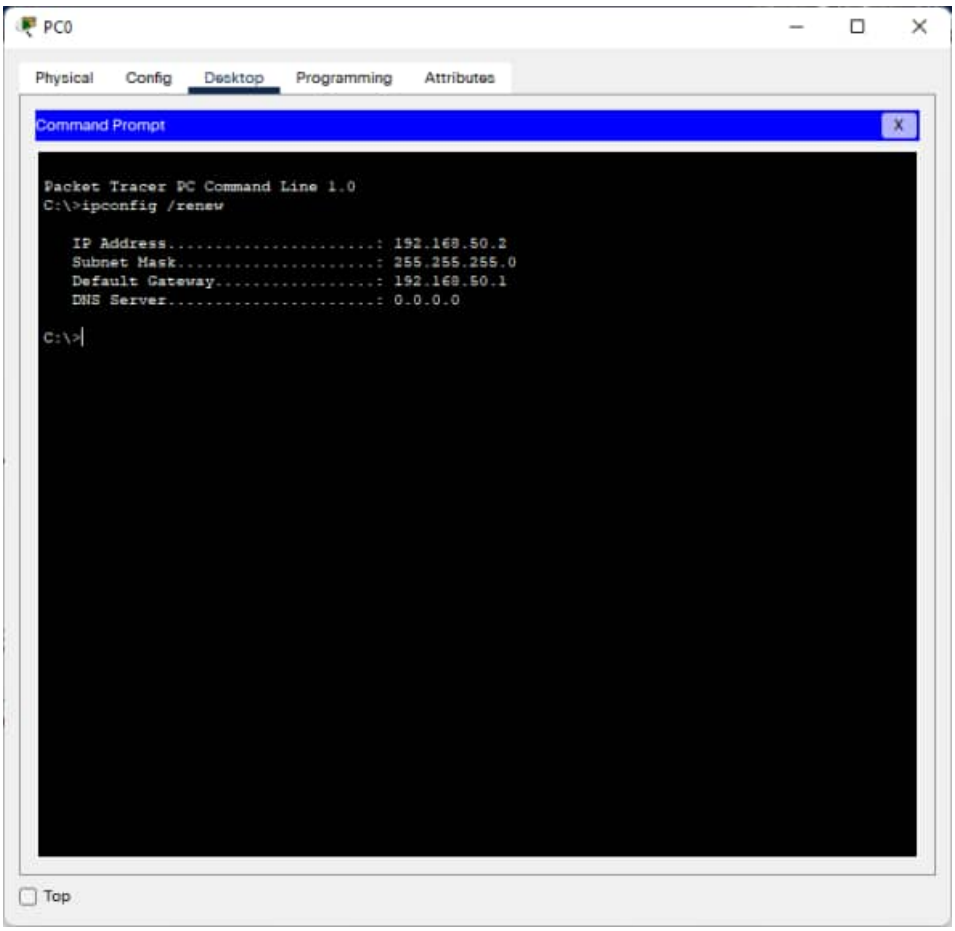


Рисунок 19 – Результат выполнения команды в Command Prompt на PC0

j. **PC0** теперь имеет IP-адрес, входящий в диапазон IP 192.168.50.0/24. Проверьте свой результат, он должен составить 1

Лабораторная работа №3
Работа с эмулятором сетей Cisco Packet Tracer.
Подключение беспроводных ПК к маршрутизатору Linksys WRT300N
Цели обучения

- Настройка основных беспроводных параметров на ПК.
- Настройка основных функций безопасности на Linksys-WRT300N.
- Проверка полноты связи.

Введение

В этом интерактивном задании предстоит выполнить настройку беспроводных ПК для подключения к сети через маршрутизатор Linksys WRT300N. Для этого необходимо выполнить настройку основных функций безопасности на Linksys WRT300N, изменив выбранное по умолчанию имя сети (SSID), пароль по умолчанию и добавив шифрование WEP.

Задача 1. Подключение к маршрутизатору Linksys WRT300N
Этап 1. Подключение к беспроводному маршрутизатору.

В окне **WirelessPC1** выберите вкладку **Desktop** (Рабочий стол), а затем **PC Wireless** (Беспроводной ПК). Выберите вкладку **Connect** (Подключение) и подключитесь к сети по умолчанию.

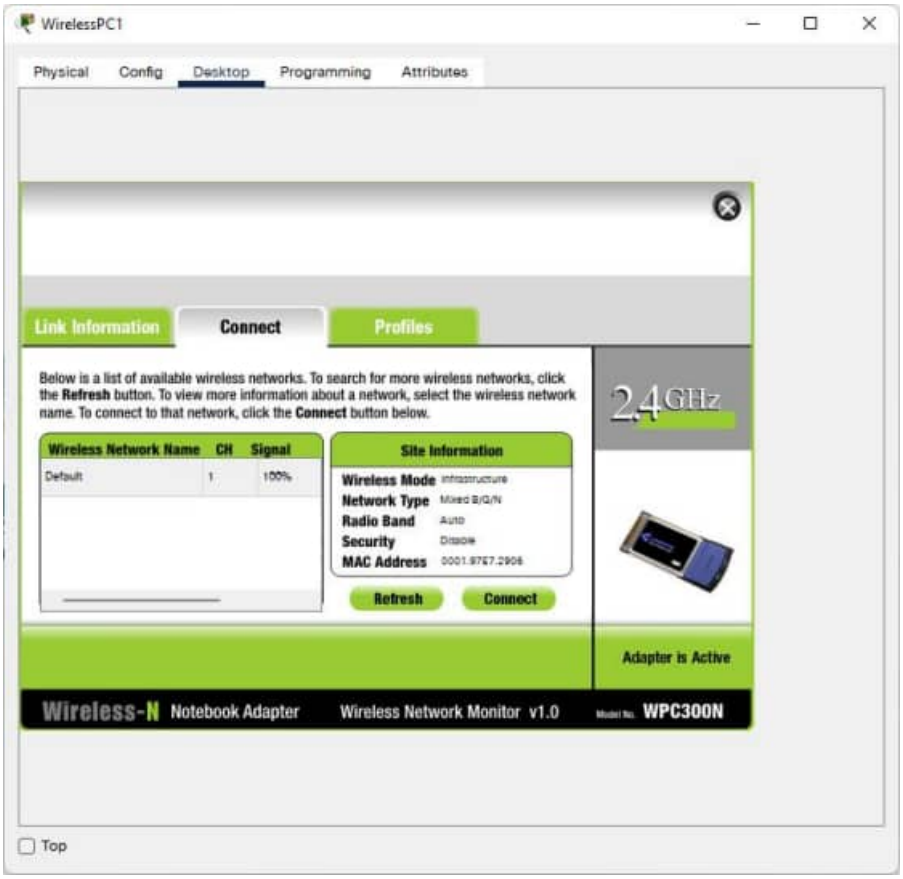


Рисунок 20 – Окно подключения к беспроводной сети

Этап 2. Проверка параметров связи.

С рабочего стола компьютера проверьте параметры связи с помощью **Command Prompt** (Командной строки), введя команду **ipconfig**.

PC>**ipconfig**

IP-адрес.....: 192.168.1.100

Маска подсети.....: 255.255.255.0

Шлюз по умолчанию.....: 192.168.1.1

PC>

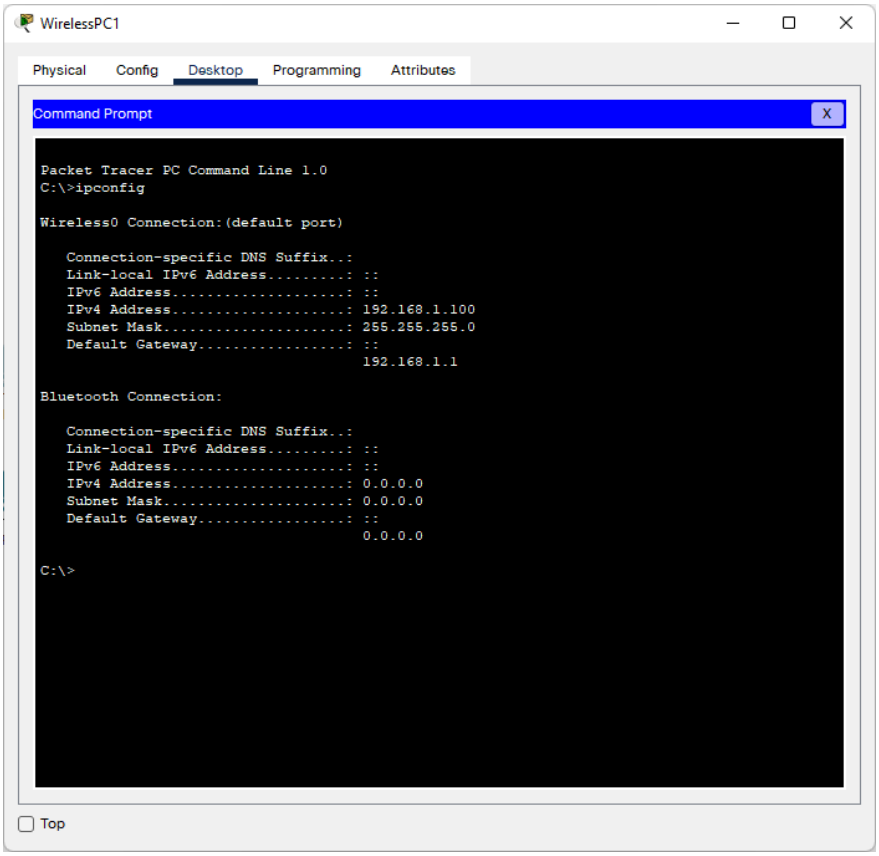


Рисунок 21 – Результат ввода команды ipconfig в Command Prompt на WirelessPC1

Примечание. Динамически полученный IP-адрес может быть отличаться от указанного.

Задача 2. Получение доступа к маршрутизатору Linksys WRT300N

Этап 1. Получение доступа к Linksys WRT300N, WRS1 с помощью веб-браузера.

В окне **WirelessPC1** закройте вкладку **Command Prompt** (Командная строка) и нажмите вкладку **Web Browser** (Веб-браузер). Введите URL-адрес **192.168.1.1** шлюза по умолчанию для компьютера.

Этап 2. Ввод параметров аутентификации.

Система запросит имя пользователя и пароль. Имя пользователя и пароль по умолчанию **admin**. После ввода данных входа появится страница по умолчанию служебного веб-приложения Linksys WRT300N.

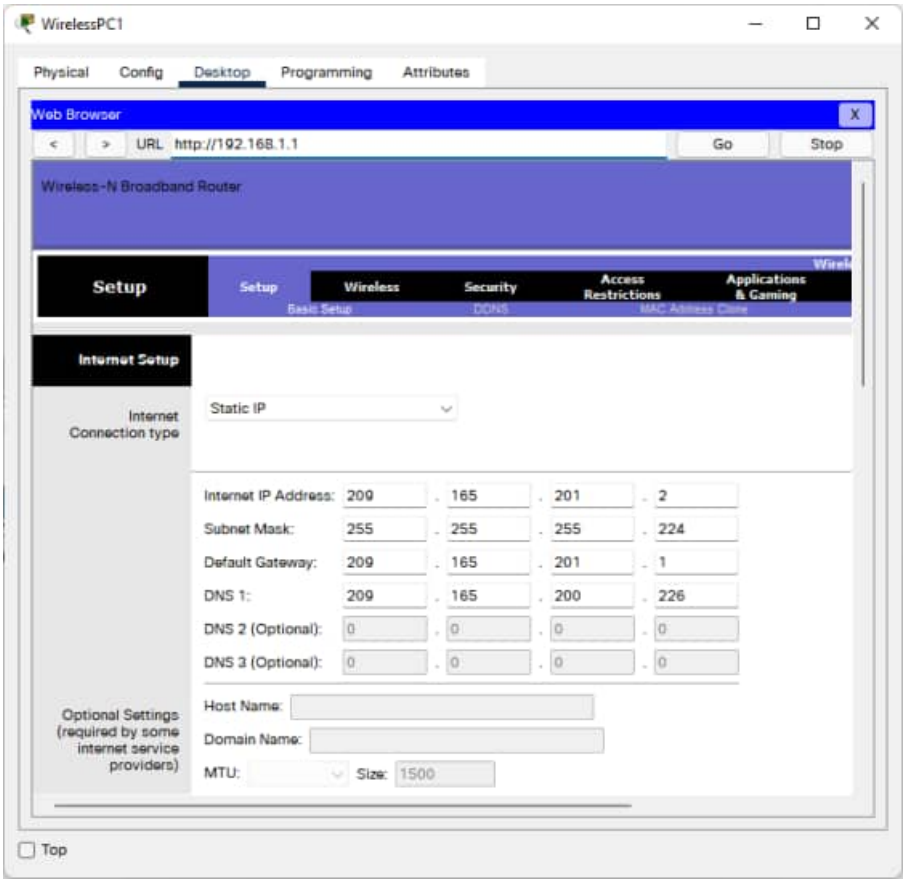


Рисунок 22 – Окно настройки маршрутизатора Linksys

Задача 3. Основные параметры беспроводной сети

Этап 1. Настройка имени сети (SSID).

Чтобы защитить беспроводную сеть, зайдите на страницу **Wireless** (Беспроводной доступ) и в поле **Network Name (SSID)** (Имя сети (SSID)) измените имя с **Default** (По умолчанию) на **WRS1**.

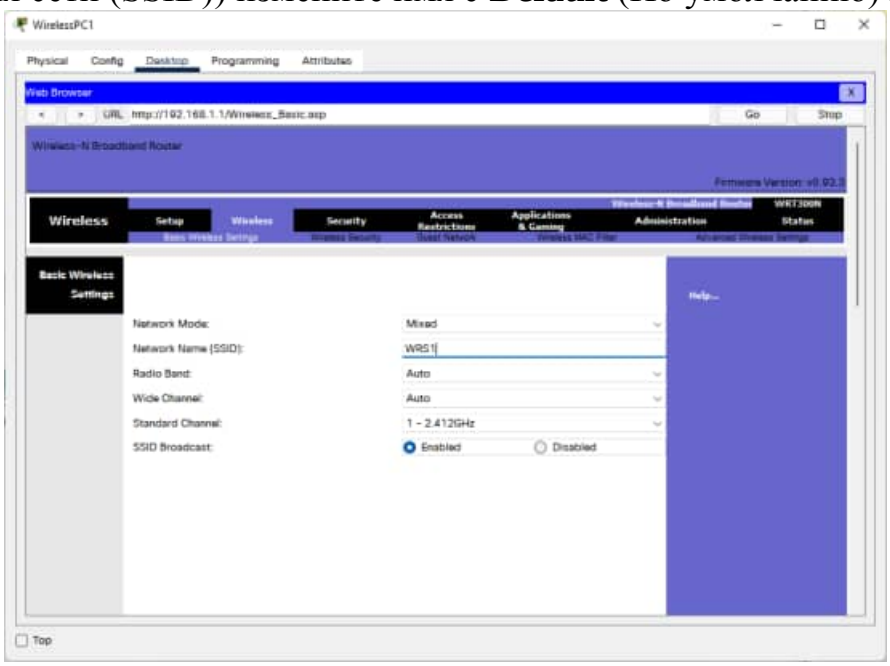


Рисунок 23 – Окно настройки беспроводной сети маршрутизатора Linksys

Этап 2. Сохранение параметров.

Прокрутите страницу вниз и нажмите **Save Settings** (Сохранить параметры). После принятия параметров маршрутизатором Linksys WRT300N в окне веб-браузера появится сообщение **Request Timeout** (Время ожидания запроса истекло). После появления этого сообщения перейдите к этапу 3.

Этап 3. Повторное подключение к беспроводной сети.

После изменения имени сети (SSID) устройство **WirelessPC1** пока не может получить доступ к сети. В окне **Desktop** (Рабочий стол) вернитесь к пункту **PC Wireless** (Компьютер с беспроводным доступом) и выберите вкладку **Connect** (Подключение). Выполните подключение к сети WRS1.

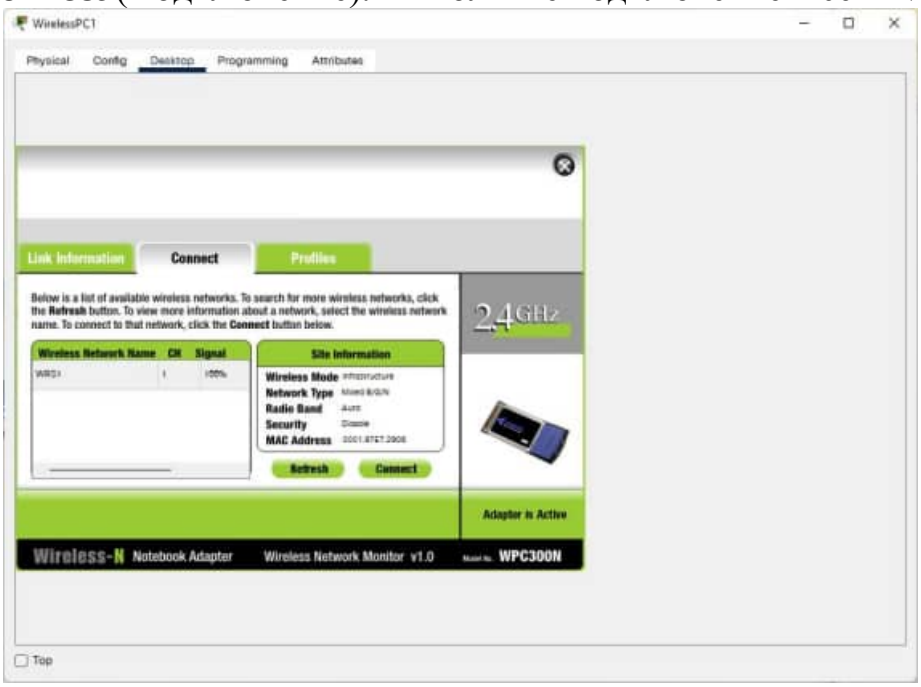


Рисунок 24 – Окно подключения к беспроводной сети

Задача 4. Включение защиты беспроводной сети

Этап 1. Из веб-браузера на компьютере WirelessPC1 выполните повторный вход на страницу настройки маршрутизатора (http://192.168.1.1).

Этап 2. Перейдите на страницу беспроводного доступа и выберите вкладку "Wireless Security" (Безопасность беспроводной сети).

Этап 3. В списке Security Mode (Режим безопасности) выберите WEP.

Этап 4. Введите ключ WEP.

Уровень безопасности сети определяется по уровню безопасности в ее наиболее уязвимой точке, а беспроводной маршрутизатор, обеспечивающий подключение пользователей к сети, подвергается сетевым угрозам в первую очередь. Если для подключения к сети запрашивается ключ WEP, уровень безопасности в сети повышается.

К сожалению, существуют средства, способные взломать шифр ключа WEP. Более надежными способами обеспечения безопасности в беспроводной сети

являются технологии WPA и WPA-2, которые в данный момент не поддерживаются программой Packet Tracer.
Введите ключ WEP **1234567890** в поле **Key1** (Ключ 1).

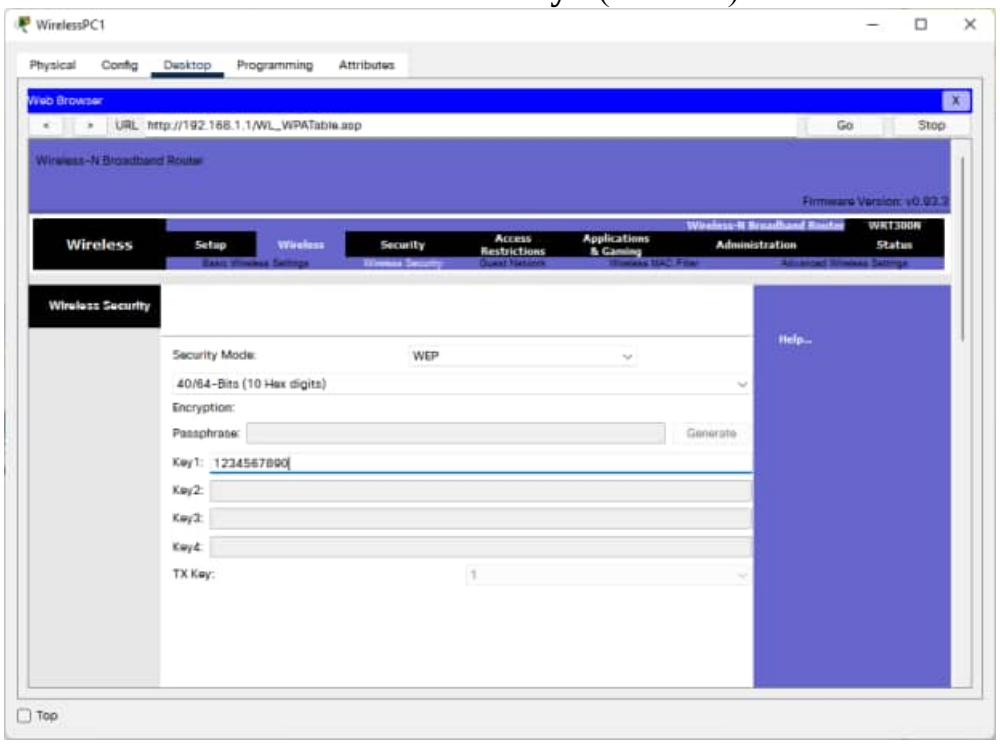


Рисунок 25 – Вкладка настройки WEP ключа беспроводной сети

Этап 5. Сохранение параметров.

Прокрутите страницу вниз и нажмите **Save Settings** (Сохранить параметры). После сохранения параметров вы будете снова отключены от сети. После принятия параметров маршрутизатором Linksys WRT300N в окне веб-обозревателя появится сообщение **Request Timeout** (Время ожидания запроса истекло). После появления этого сообщения перейдите к этапу 6.

Этап 6. Настройка WirelessPC1 для использования аутентификации WEP.

- Вернитесь на вкладку **Desktop** (Рабочий стол) и щёлкните **PC Wireless** (Беспроводной ПК).
- Щёлкните вкладку **Connect** (Подключение).
- Выберите **WRS1** в списке доступных беспроводных сетей и щёлкните **Connect** (Подключение).
- Появится окно запроса ключа WEP. В поле **WEP Key 1** введите ключ WEP: **1234567890** и нажмите **Connect** (Подключение).

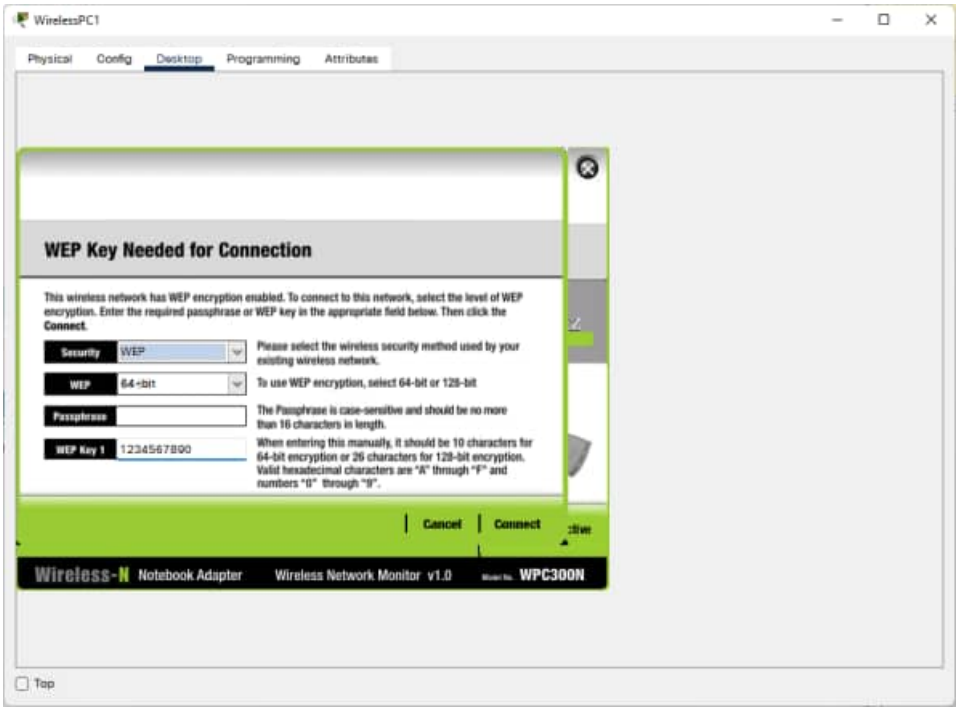


Рисунок 26 – Окно ввода пароля WEP.

- Щёлкните вкладку **Link Information** (Информация о каналах), чтобы проверить связь с точкой доступа.

Ваш процент выполненных работ должен составить 57%. Если это не так, щёлкните **Check Results** (Проверить результаты), чтобы посмотреть, выполнение каких компонентов еще не завершено.

Задача 5. Управление служебным веб-приложением маршрутизатора и обеспечение его безопасности

Этап 1. Настройка пароля веб-доступа.

С вкладки **Web Browser** (Веб-браузер) в меню WirelessPC1 вернитесь на страницу служебного веб-приложения маршрутизатора (http://192.168.1.1) и перейдите в раздел **Administration** (Администрирование). Измените пароль маршрутизатора по умолчанию на **cisco** для безопасного доступа к устройству Linksys WRT300N. Обратите внимание, что режим **HTTP Web Utility Access** (Доступ к служебному веб-приложению по протоколу HTTP) уже выбран по умолчанию.

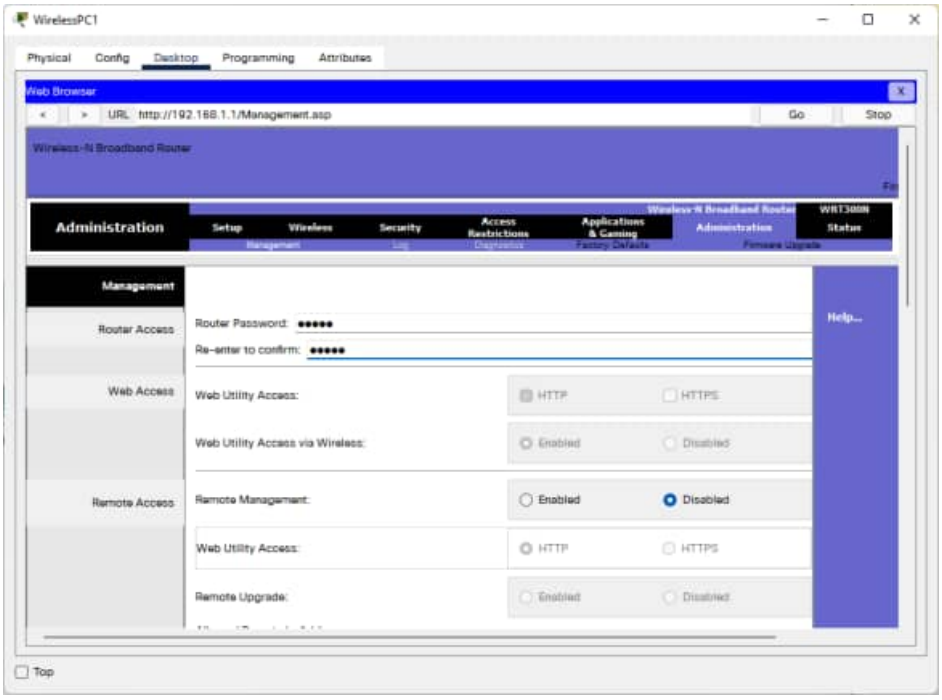


Рисунок 27 – Настройка доступа к управлению маршрутизатора.

Этап 2. Прокрутите страницу вниз и нажмите кнопку Save Settings (Сохранить параметры).

После того, как смена пароля будет зарегистрирована системой, на странице появится сообщение: *Settings are successful. (Параметры успешно изменены.)* Под этим сообщением находится ссылка для последующего входа в служебное веб-приложение для маршрутизатора Linksys WRT300N. Щёлкните ссылку **Continue** (Продолжить) для появления окна входа. Введите имя пользователя **admin** и используйте пароль **cisco** для повторного подключения к страницам веб-настройки Linksys WRT300N.

Задача 6. Изменение используемого беспроводного канала

Этап 1. Получение доступа к Linksys WRT300N, WRS1 с помощью веб-браузера.

В окне **WirelessPC1** щёлкните **Web Browser** (Веб-браузер). Введите URL-адрес 192.168.1.1.

Этап 2. Ввод параметров аутентификации.

Для аутентификации введите имя пользователя **admin** и пароль **cisco**.

Этап 3. Указание беспроводного канала.

Многие точки доступа работают на основе принципа автоматического использования ближайших каналов. В некоторых устройствах предусмотрен постоянное наблюдение за радиопространством для динамической корректировки параметров канала в ответ на изменения среды. В этом задании вам предстоит принудительно обеспечить работу точки через канал 6.

Так как клиенты используют только стандарт 802.11b/g, в меню **Radio Band** (Радиодиапазон) будет выбран пункт **Standard-20MHz Channel** (Стандартный канал 20 МГц).
Когда страница будет загружена, перейдите по вкладкам **Wireless > Basic Wireless Setup** (Беспроводная сеть > Основная настройка беспроводной сети) и найдите область **Standard Channel** (Стандартный канал). Измените его значение на **6 – 2.437 ГГц**.

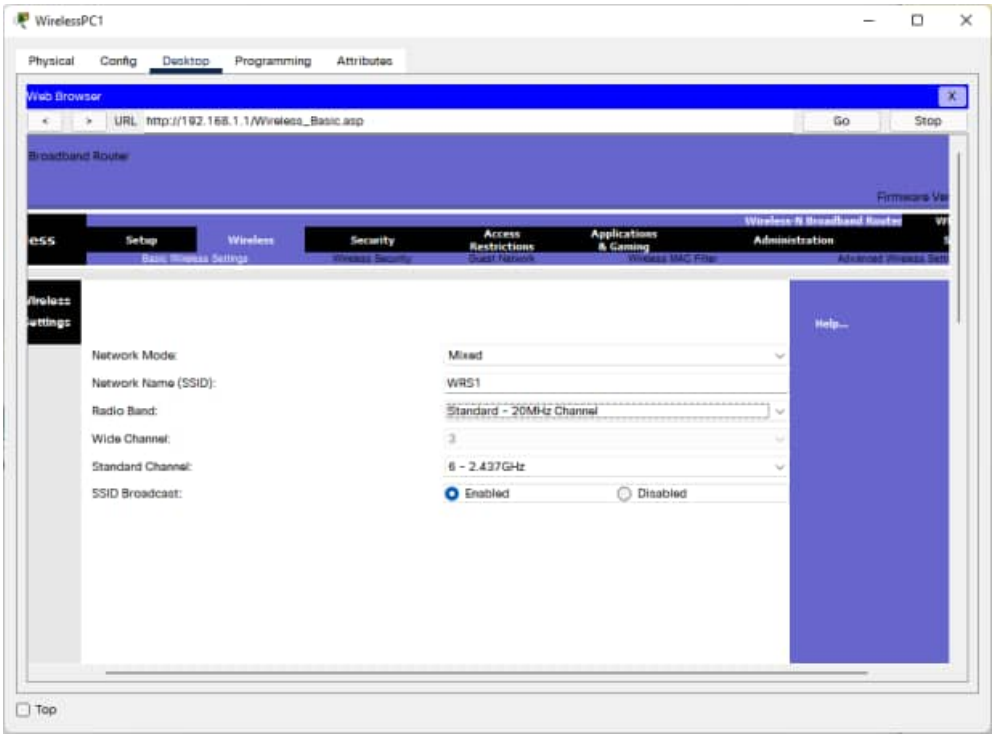


Рисунок 28 – Окно смены частоты работы беспроводной сети

Этап 4. Сохранение параметров.

Прокрутите страницу вниз и нажмите **Save Settings** (Сохранить параметры). После того, как параметры будут приняты маршрутизатором Linksys WRT300N, в окне веб-обозревателя появится сообщение **Settings are successful** (Параметры успешно сохранены). После отображения этого сообщения нажмите **Continue** (Продолжить), чтобы вернуться на страницу беспроводного доступа и закрыть ее.
WirelessPC1 теперь связан с маршрутизатором **WRS1**. Обратите внимание, что при смене каналов настройка устройства WirelessPC1 не требуется. Устройство **WRS1** само контролирует используемый канал.

Задача 7. Подключение к сети компьютера WirelessPC2

Этап 1. Настройка WirelessPC2 для использования аутентификации WEP.

- Со вкладки **WirelessPC2** перейдите на вкладку **Desktop** (Рабочий стол) и щёлкните **PC Wireless** (Беспроводной ПК).
- Щёлкните вкладку **Connect** (Подключение).

- Выберите **WRS1** в списке доступных беспроводных сетей и щёлкните **Connect** (Подключение).
- Появится окно запроса ключа WEP. В поле **WEP Key 1** введите ключ WEP: **1234567890** и нажмите **Connect** (Подключение).
- Щёлкните вкладку **Link Information** (Информация о каналах), чтобы проверить связь с точкой доступа.

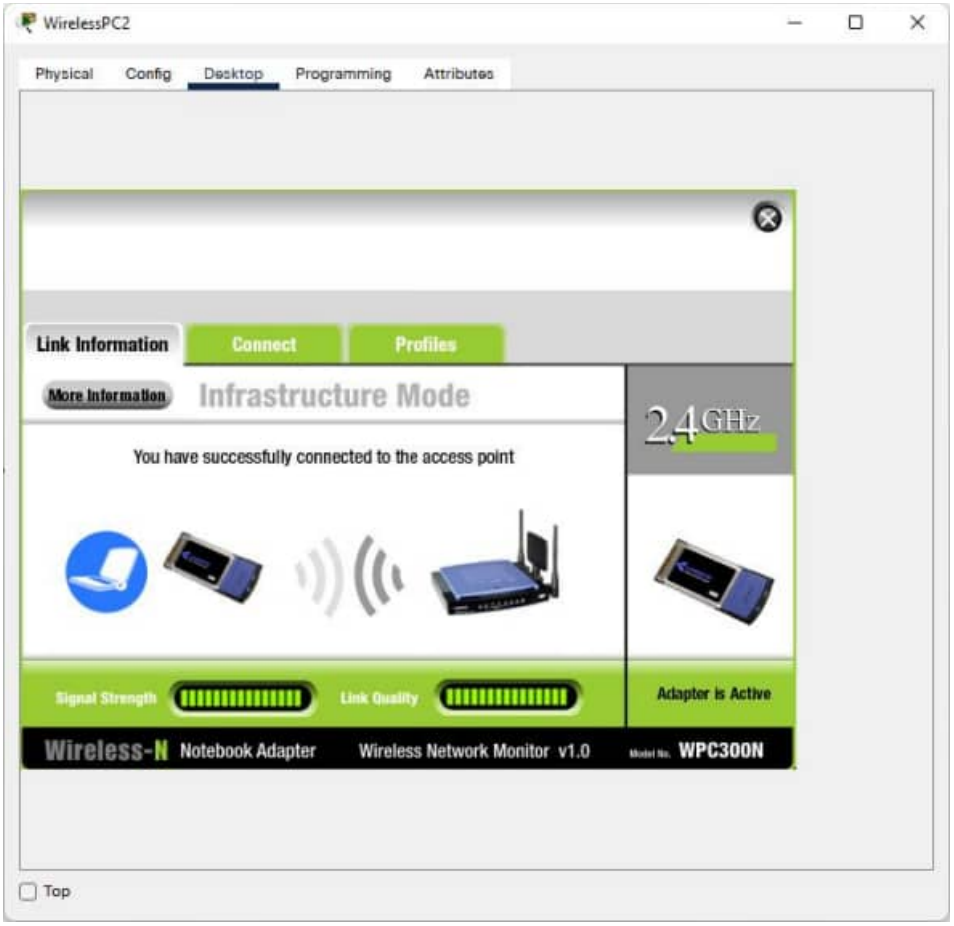


Рисунок 29 – Окно статуса подключения к сети

Ваш процент выполненных работ должен быть 100%. Если это не так, щёлкните **Check Results** (Проверить результаты), чтобы посмотреть, выполнение каких компонентов еще не завершено.

Лабораторная работа №4
Работа с эмулятором сетей Cisco Packet Tracer.
Приёмы обеспечения безопасности беспроводных сетей
Цели обучения

- Настройка WPA2 в Linksys WRT300N
- Настройка фильтрации по MAC-адресам в Linksys WRT300N
- Настройка переадресации одного порта в Linksys WRT300N

Введение

В этом интерактивном задании вам предстоит выполнить следующие настройки беспроводного маршрутизатора Linksys WRT300N:

- использование WPA2 Personal в качестве метода обеспечения безопасности;
- использование фильтрации по MAC-адресам для повышения безопасности;
- поддержку переадресации одного порта.

Задача 1. Подключение к беспроводному маршрутизатору

Этап 1. Вход на беспроводной маршрутизатор

a. На вкладке **Desktop** (Рабочий стол) в меню **PC0** (ПК0) выберите пункт **Web Browser** (Веб-браузер).

b. Введите IP-адрес беспроводного маршрутизатора **192.168.0.1**.

c. При запросе имени пользователя и пароля введите **admin** в оба поля. Эти имя пользователя и пароль являются заводскими параметрами по умолчанию для всех продуктов Linksys.

d. После загрузки веб-страницы настройки **WRS1** перейдите к задаче 2.

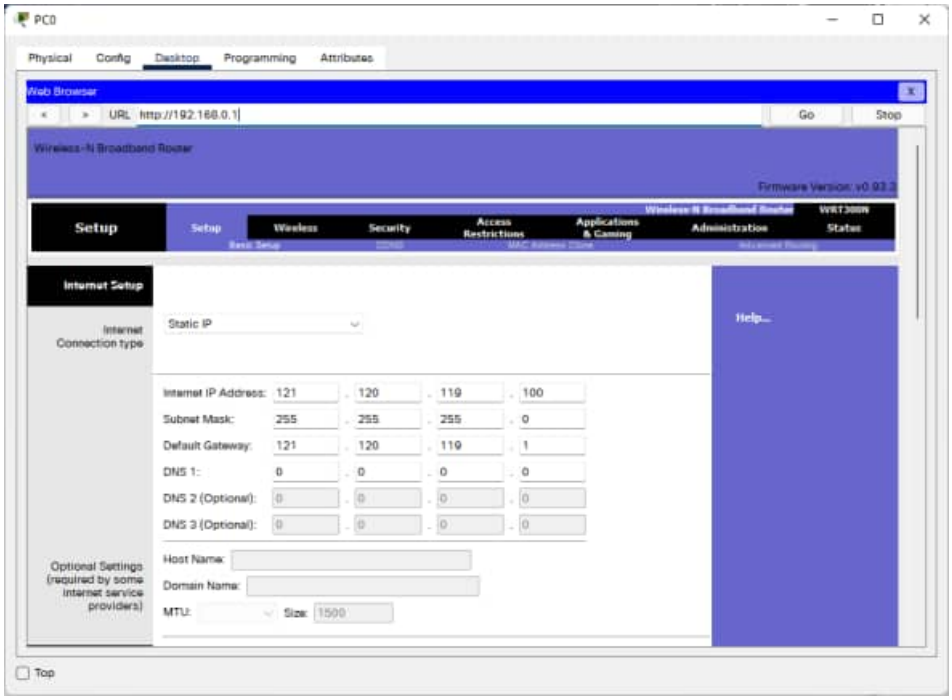


Рисунок 30 – Вкладка настройки подключения интерфейса Интернет маршрутизатора Linksys

Задача 2. Добавление функций безопасности в WRS1

Этап 1. Настройка WPA2+AES на маршрутизаторе WSR1

a. С той же веб-страницы настройки **WRS1** (открытой с помощью веб-браузера на **PC0)** перейдите на страницу **Wireless > Wireless Security** (Беспроводная сеть > Безопасность беспроводной сети).

b. Измените для параметра **Security Mode** (Режим безопасности) значение **Disabled** (Отключен) на **WPA2 Personal** (Персональная WPA2).

c. **AES** считается одним из наиболее стойких протоколов шифрования. Рекомендуется использовать именно его.

d. Пароль для этой беспроводной сети настраивается в поле **Passphrase** (Кодовая фраза). Введите пароль **aCompWiFi.** Обратите внимание, что пароли вводятся с учетом регистра.

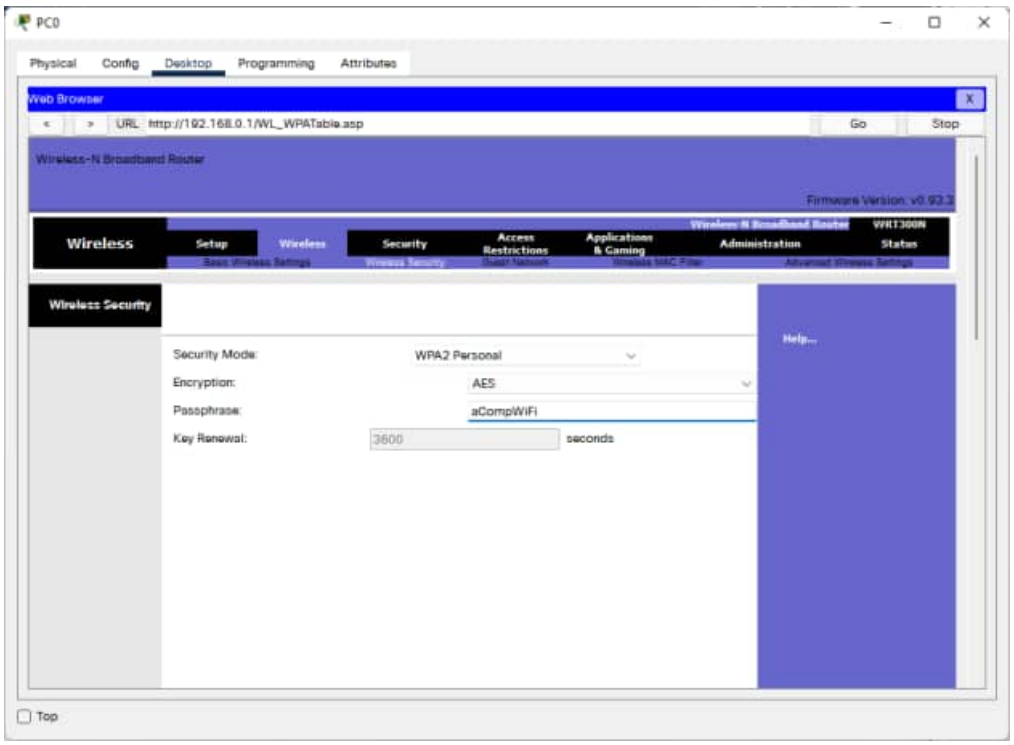


Рисунок 31 – Вкладка настройки беспроводной сети

e. Чтобы сохранить изменения, прокрутите страницу вниз и нажмите кнопку **Save Settings** (Сохранить параметры).

f. **WRS1** теперь настроен на использование режима **WPA2**.

Примечание. Устройство Laptop0 (Портативный компьютер 0) не может быть связано с маршрутизатором **WSPR1**, поскольку его настройка еще не выполнена. Настройка устройства **Laptop0** будет выполнена на этапе 2.

Этап 2. Настройки беспроводного клиента Laptop0

a. Нажмите **Laptop0**, чтобы открыть его окно настройки.

b. Щёлкните вкладку **Desktop** (Рабочий стол).

c. Нажмите кнопку **PC Wireless** (Беспроводной ПК).

d. Появится сообщение **No association with access point** (Нет связи с точкой доступа).

e. Щёлкните вкладку **Connect** (Подключение).

f. Подождите несколько секунд до появления широковещательной рассылки имени сети (SSID) через **WRS1**. После этого появится имя сети (SSID) **aCompany** в столбце **Wireless Network Name** (Имя беспроводной сети).

g. Щёлкните имя сети **aCompany**. Затем щёлкните **Connect** (Подключение).

h. В поле **Security** (Безопасность) выберите режим **WPA2-Personal** (Персональная WPA2).

i. Введите пароль для беспроводной сети **aCompWiFi** (тот же пароль, который настроен в **WRS1**) и нажмите **Connect** (Подключение).

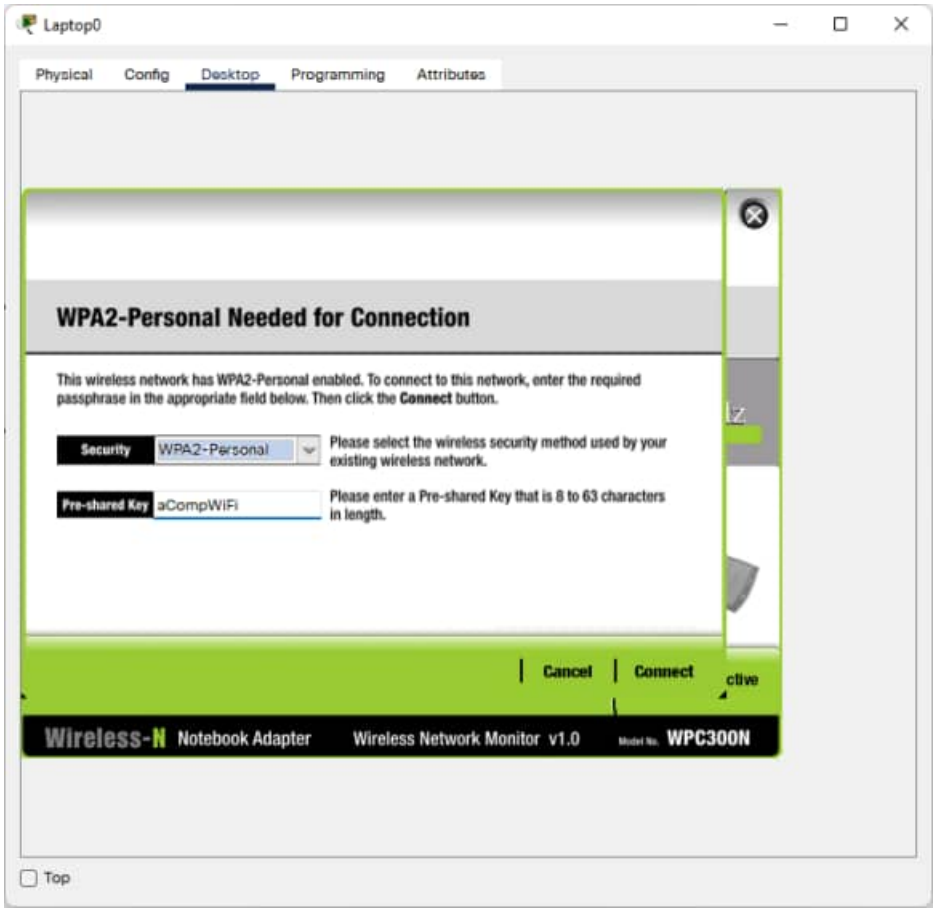


Рисунок 32 – Окно ввода пароля WPA2 беспроводной сети

j. **Laptop0** теперь связан с **WRS1**.

k. Запишите IP-адрес устройства **Laptop0** (полученный с помощью DHCP с маршрутизатора **WRS1)**, а также его MAC-адрес. Закройте окно **PC Wireless** (Беспроводной ПК) и на вкладке **Desktop** (Рабочий стол) выберите пункт **Command Prompt** (Командная строка).

l. Введите команду **ipconfig /all** и запишите IP и MAC-адреса устройства **Laptop0**.

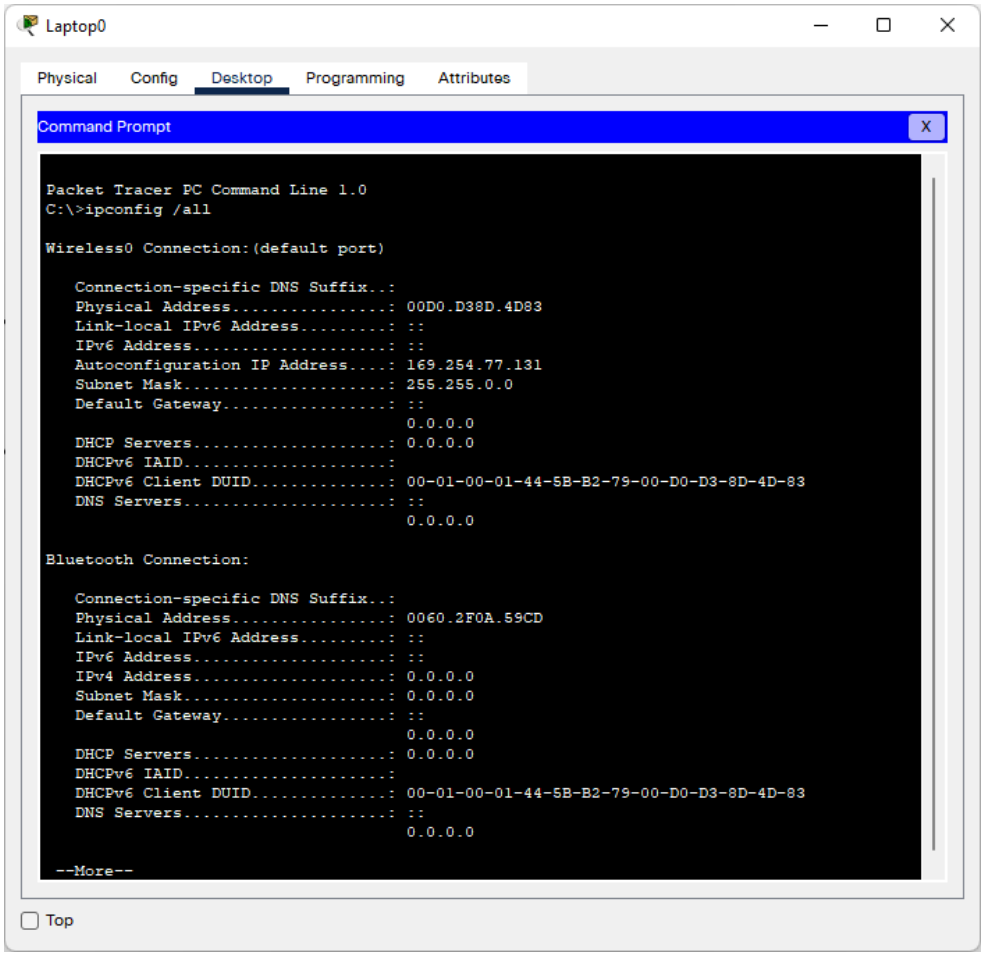


Рисунок 33 – Результат ввода команды ipconfig /all в Command Prompt на Laptop0

Этап 3. Настройка WRS1 для поддержки фильтрации по MAC-адресам

a. Закройте страницу настроек **Laptop0**.

b. Щёлкните **PC0** и откройте веб-обозреватель, перейдя по вкладкам **PC0 > Desktop > Web Browser** (ПК0 > Рабочий стол > Веб-браузер).

c. Введите IP-адрес **WRS1**, чтобы открыть его веб-страницу настройки. При появлении запроса введите **admin** в качестве имени пользователя и пароля.

d. Перейдите в пункт **Wireless > Wireless MAC Filter** (Беспроводная сеть > Фильтр беспроводных MAC-адресов).

e. Выберите пункты **Enabled** (Включено) и **Permit PCs listed below to access wireless network** (Разрешить указанным ниже ПК доступ к беспроводной сети).

f. Введите MAC-адрес устройства **Laptop0** (см. задачу 2, этап 2) в поле **MAC 01:** . Обратите внимание на формат MAC-адреса,

требуемый **WRS1**. Он должен быть следующим: **XX:XX:XX:XX:XX:XX**.

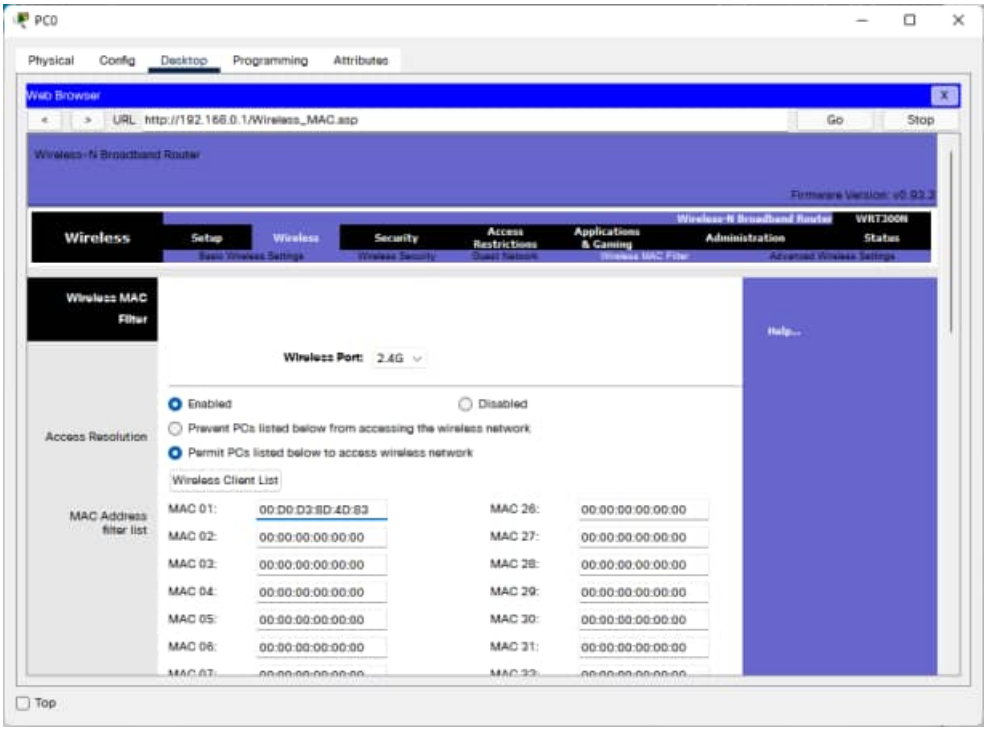


Рисунок 34 – Настройка фильтрации на маршрутизаторе Linksys

g. Прокрутите вниз страницу и нажмите кнопку **Save Settings** (Сохранить параметры).

h. Поскольку MAC-адрес был указан только для устройства **Laptop0**, **Laptop0** является единственным беспроводным устройством, связанным с **WRS1**.

Этап 4. Проверка связи

a. Щёлкните **PC0** и перейдите в пункт **Desktop > Command Prompt** (Рабочий стол > Командная строка).

b. Отправьте эхо-запрос на **RemotePC** (Удалённый ПК), введя команду **ping 200.100.50.10**. Эхо-тестирование должно пройти успешно.

c. Закройте окно настройки **PC0** и щёлкните **Laptop0**.

d. В окне настройки **Laptop0** перейдите в пункт **Desktop > Command Prompt** (Рабочий стол > Командная строка).

e. Отправьте повторный эхо-запрос на **RemotePC**, введя команду **ping 200.100.50.10**. Эхо-тестирование также должно пройти успешно.

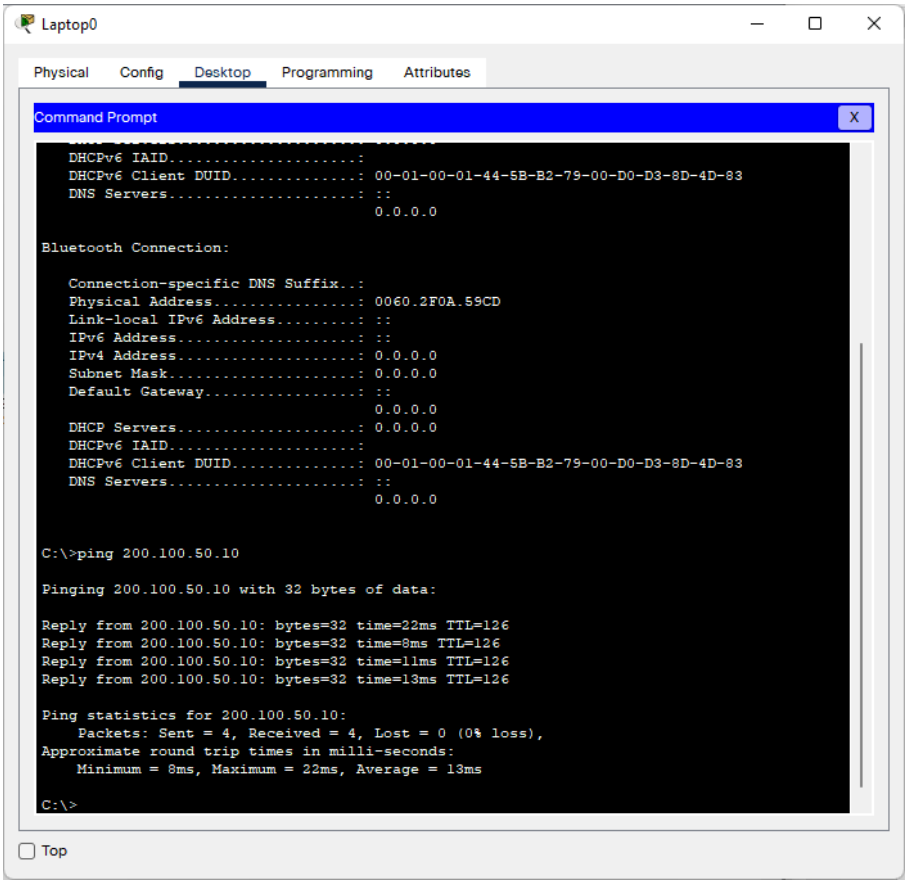


Рисунок 35 – Результат ввода команды ping на Laptop0

f. Закройте окно настройки **Laptop0**.

g. Нажмите **RemotePC** для вызова его окна настройки.

h. В окне настройки **RemotePC** перейдите по вкладкам **Desktop** > **Command Prompt** (Рабочий стол > Командная строка).

i. Отправьте эхо-запрос на **Server0** с **RemotePC** с помощью команды **ping 192.168.0.20**. Эхо-тестирование **не** должно быть успешным.

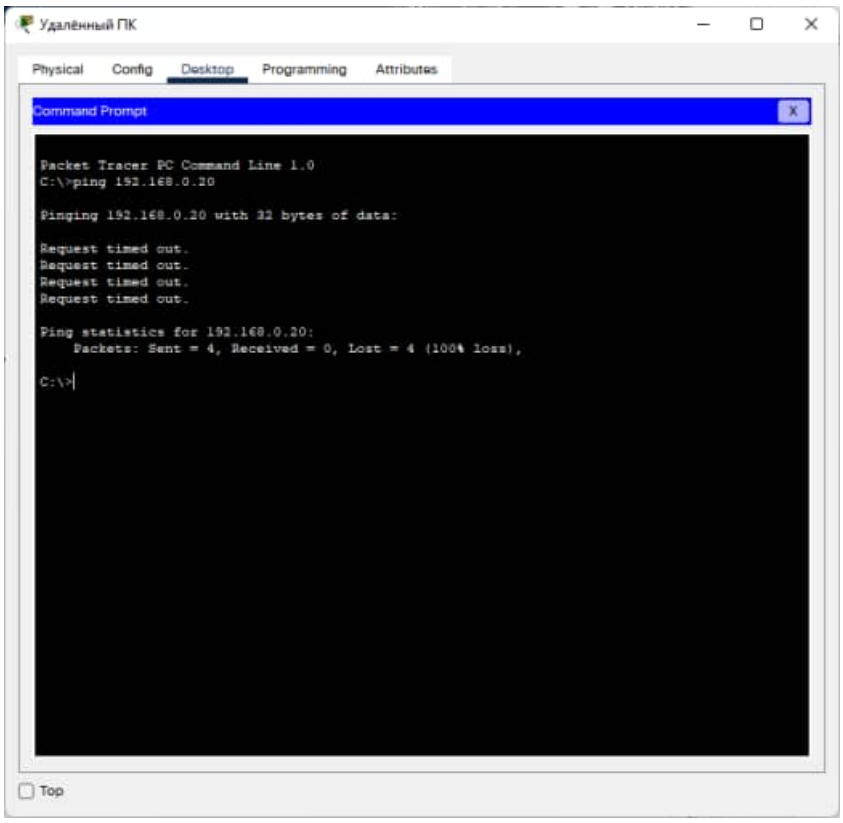


Рисунок 36 – Результат ввода команды ping на удаленном компьютере

j. Оставаясь в меню **RemotePC**, откройте **веб-браузер** (**Desktop > Web Browser**) (Рабочий стол > Веб-браузер) и введите адрес внутренней веб-страницы, доступной через **Server0** (Сервер 0), www.acompany.com. Страница **не** должна открыться.

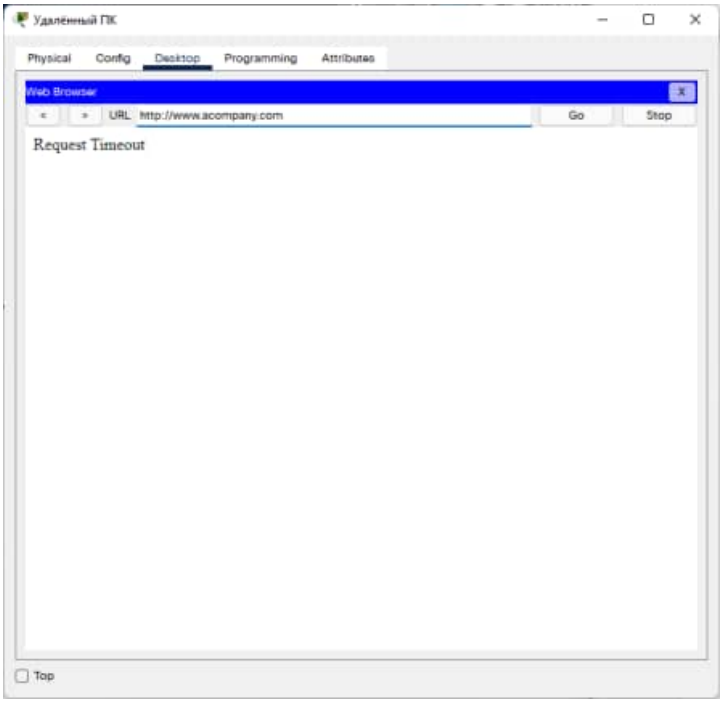


Рисунок 37 – Сообщение об ошибке при попытке открыть веб-страницу

Эхо-запрос и (или) HTTP запрос с компьютера **RemotePC** на **Server0** (или любое другое внутреннее устройство) не удался, поскольку маршрутизатору **WRS1** неизвестно, какое внутреннее устройство должно получить запрос. Для этого используется функция переадресации портов, которую необходимо настроить в **WRS1.**

k. Закройте все открытые окна настройки и перейдите к следующему этапу.

Этап 5. Настройка переадресации одного порта в WRS1.

a. Щёлкните **PC0** для вызова окна настройки.

b. Перейдите по вкладкам **Destop > Web Browser** (Рабочий стол > Веб-браузер) и подключитесь к устройству **WRS1**.

c. С веб-страницы настройки **WRS1** перейдите по вкладкам **Application & Gaming > Single Port Forwarding** (Приложения и игры > Переадресация одного порта).

d. В меню слева в первом поле со списком выберите **HTTP**.

e. В средней части экрана найдите первую строку. Измените IP-адрес в первой строке, чтобы он соответствовал IP-адресу устройства **Server0**: 192.168.0.20. Установите также флажок **Enabled** (Включено) в конце той же строки.

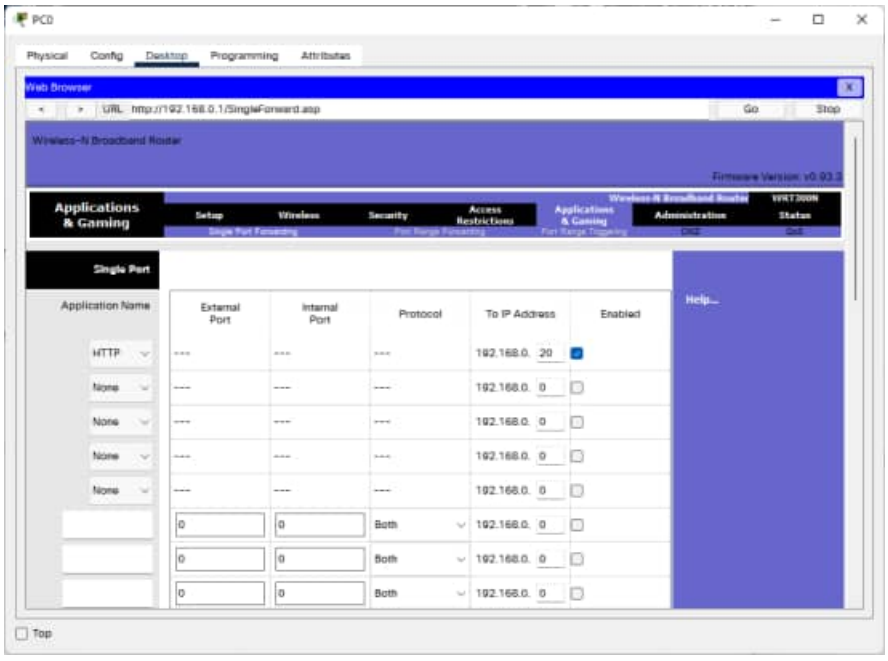


Рисунок 38 – Настройка переадресации

f. Прокрутите страницу вниз и нажмите кнопку **Save Settings** (Сохранить параметры).

g. Теперь у вас есть доступ к веб-странице, размещенной на устройстве **Server0**. Откройте веб-браузер на **RemotePC**.

h. В адресной строке введите **121.120.119.100**. Это IP-адрес, присвоенный порту Интернета маршрутизатора **WRS1**.

Рисунок 39 – Открытая веб-страница с удаленного компьютера

i. Теперь вы можете просмотреть веб-страницу, размещенную на сервере **Server0**.

Проверьте свой результат. Вы должны набрать 100%.

Лабораторная работа №5
Работа с эмулятором сетей Cisco Packet Tracer.
Проверка беспроводного подключения
Цели обучения

- Настройка компьютера для подключения к беспроводной сети.
- Проверка беспроводного подключения.

Введение

В этом интерактивном задании предстоит выполнить настройку **PC3** для подключения к сети через маршрутизатор Linksys WRT300N.

Задача 1. Настройка беспроводного подключения
Этап 1. Настройка PC3 для подключения к WRS1.

a. Щёлкните **PC3**, чтобы открыть окно **Physical Device View** (Просмотр физического устройства).

b. Щёлкните вкладку **Desktop** (Рабочий стол) для устройства **PC3.**

c. Нажмите кнопку **PC Wireless** (Беспроводной ПК). Откроется окно **Link Information** (Информация о каналах) с уведомлением: **No association with access point** (Отсутствует привязка к точке доступа).

d. В открывшемся окне щёлкните вкладку **Connect** (Подключение).

e. При появлении доступной беспроводной сети **WRS_LAN** нажмите кнопку **Connect** (Подключение).

f. Введите команду **ABCDE12345** в качестве ключа WEP и нажмите кнопку **Connect** (Подключение).

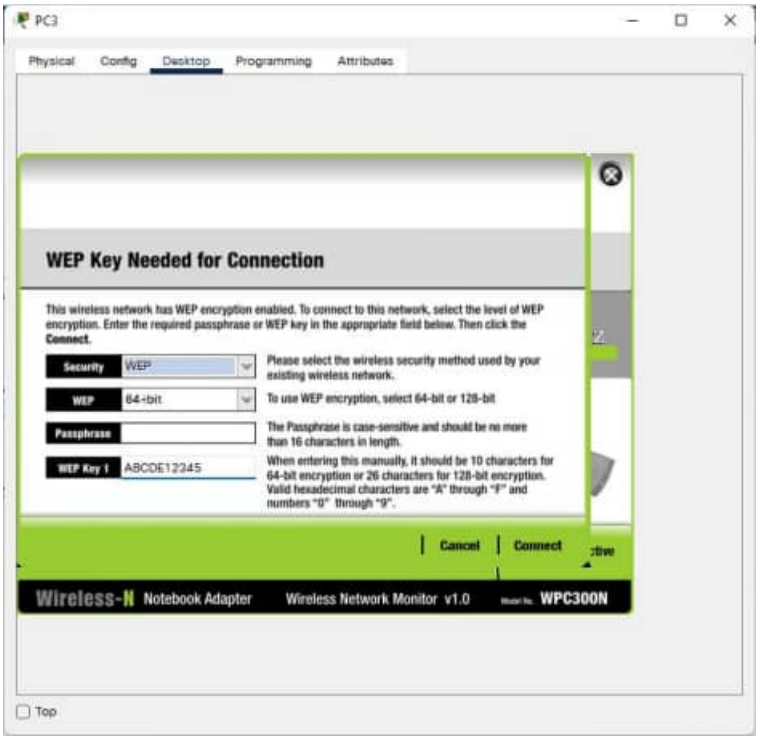


Рисунок 40 – Окно ввода пароля для подключения к беспроводной сети

g. Выберите вкладку **Link Information** (Информация о каналах). Появится сообщение: **You have successfully connected to the access point. (Подключение к точке доступа выполнено успешно.)** Если это сообщение не появилось, устраните ошибки, допущенные в ходе выполнения упражнения.

Задача 2. Проверка настройки адреса PC3

Этап 1. Вывод настройки IP-адреса компьютера PC3.

a. Закройте окно **PC Wireless** (Беспроводной ПК).

b. Нажмите кнопку **Command Prompt** (Командная строка).

c. Введите команду **ipconfig /all** и нажмите кнопку **Ввод** .

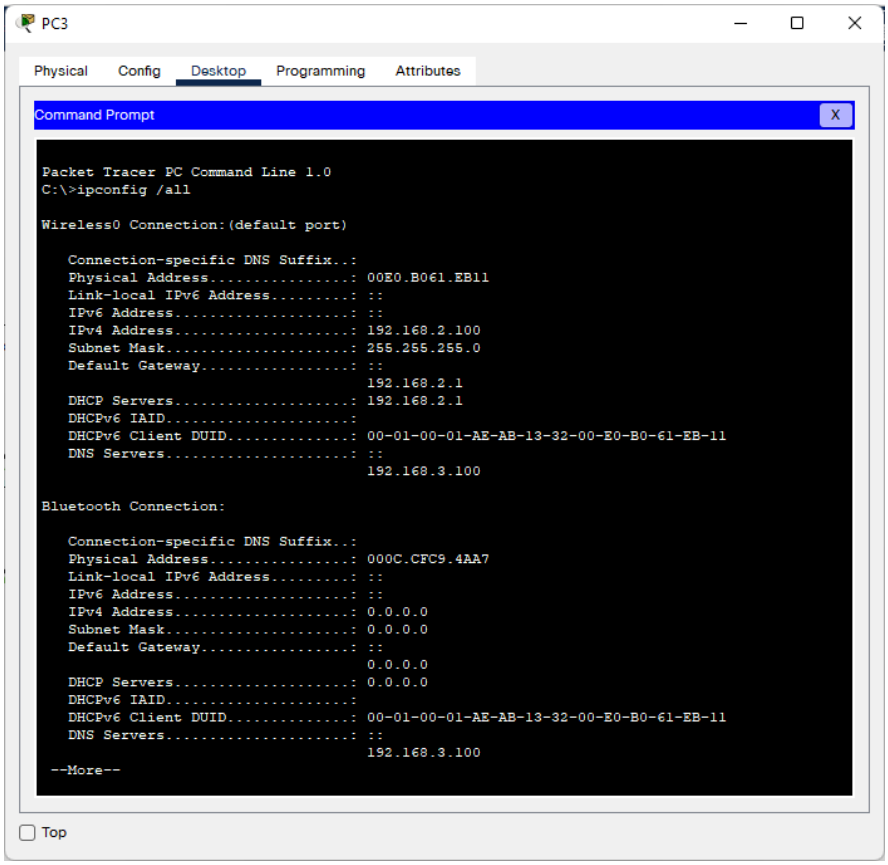


Рисунок 41 – Результат ввода команды ipconfig на PC3

d. Укажите физический адрес компьютера. Укажите другое имя для физического адреса компьютера

   00E0.B061.EB11 или 00:E0:B0:61:EB:11

   MAC-адрес

e. Укажите IP-адрес данного компьютера.

   192.168.2.100

f. Укажите маску подсети данного компьютера.

   255.255.255.0

g. Укажите основной шлюз данного компьютера.

   192.168.2.1

h. Укажите адрес сервера DNS. Какую службу поддерживает сервер DHCP в сети?
   192.168.3.100

   DNS-сервер осуществляет перевод имен узлов в IP-адреса.

Задача 3. Проверка сетевого подключения между PC3 и другими устройствами сети

Этап 1. Проверка связи между PC3 и другими устройствами сети с помощью команды ping.

a. В окне **Command Prompt** (Командная строка) отправьте эхо-запрос на шлюз, используемый по умолчанию устройством **PC3**. Эхо-тестирование должно пройти успешно. При успешном эхо-тестировании выводятся следующие результаты:

PC>ping 192.168.2.1
Pinging 192.168.2.1 with 32 bytes of data:
Reply from 192.168.2.1: bytes=32 time=203ms TTL=255
Reply from 192.168.2.1: bytes=32 time=94ms TTL=255
Reply from 192.168.2.1: bytes=32 time=94ms TTL=255
Reply from 192.168.2.1: bytes=32 time=78ms TTL=255

Ping statistics for 192.168.2.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
Approximate round trip times in milli-seconds:
Minimum = 78ms, Maximum = 203ms, Average = 117ms

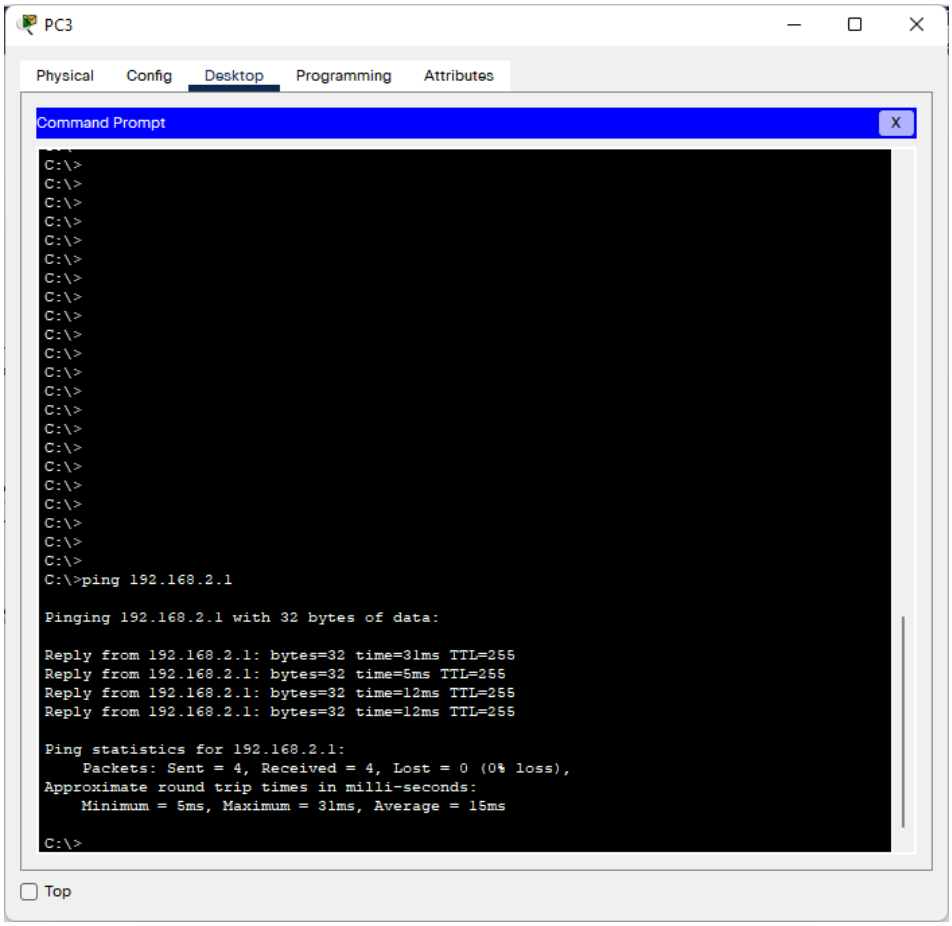


Рисунок 42 – Результат ввода команды ping на PC3

b. В окне **Command Prompt** (Командная строка) отправьте эхо-запрос на **PC1** , используя его IP-адрес 192.168.1.11. (Отправка эхо-запросов должна пройти успешно.)

Этап 2. Проверка связи и пути между PC3 и другими устройствами с помощью команды tracert.

a. Команда **tracert** используется для определения пути между локальным узлом, в данном случае **PC3**, и удалённым узлом. В окне **Command Prompt** (Командная строка) проверьте путь между **PC3** и **PC2** с помощью приведенной ниже команды из окна **Command Prompt** (Командная строка) на компьютере **PC3**.

Введите команду **tracert 192.168.1.12** и нажмите клавишу **Enter** .

b. При выполнении команды должны появиться сведения примерно такого вида:

PC>tracert 192.168.1.12

Tracing route to 192.168.1.12 over a maximum of 30 hops:

1 187 ms 94 ms 93 ms 192.168.2.1
2 * 125 ms 125 ms 192.168.1.12

Trace complete.

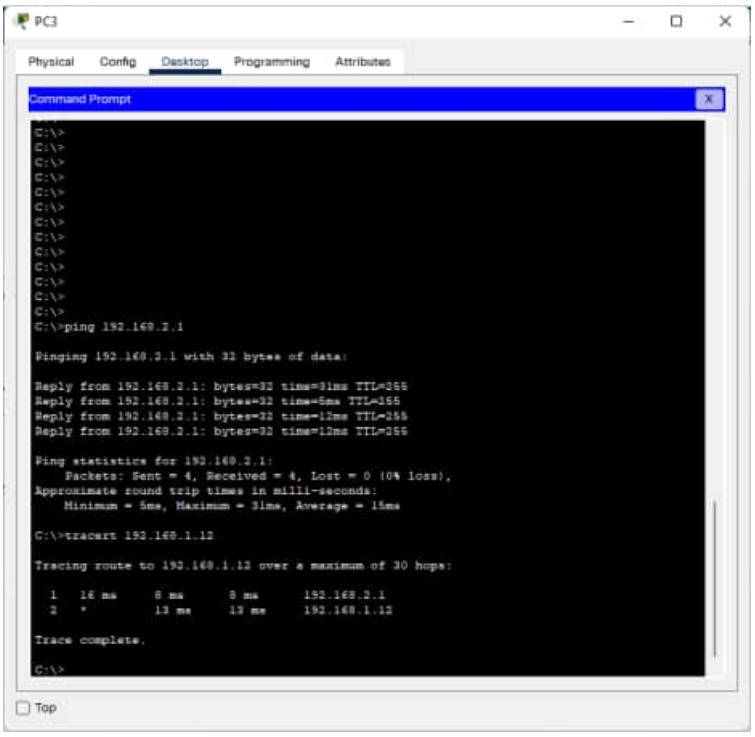


Рисунок 43 – Результат ввода команды tracert на PC3

c. Согласно выходным данным команды, пакеты **ICMP**, созданные с помощью команды **tracert** , отображают путь пакетов через интерфейс локальной сети **WRS1** к узлу **PC2**.

Задача 4. Использование DNS

Этап 1. Проверка связи с веб-сервером с помощью DNS.

a. Закройте окно командной строки на устройстве **PC3**.

b. Нажмите кнопку **Web Browser** (Веб-браузер).

c. Введите адрес **http://www.example.com** в окно адреса **URL** и нажмите кнопку **GO** (Переход). Отобразится веб-страница сервера.

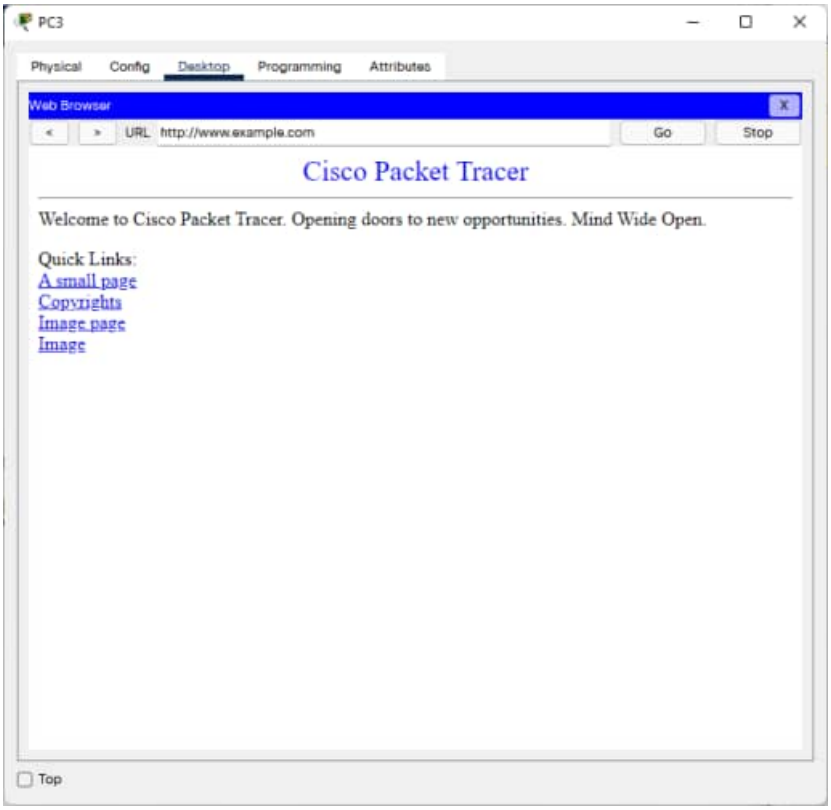


Рисунок 44 – Открывшаяся веб-страница

d. DNS используется для разрешения доменных имен в IP-адреса. Для проверки этого разрешения закройте окно **Web Browser** (Веб-браузер) на устройстве **PC3**.

e. Нажмите кнопку **Command Prompt** (Командная строка) для открытия окна **Command Prompt** (Командная строка) на **PC3**.

f. В окне **Command Prompt** (Командная строка) отправьте команду **ping** на веб-сервер, используя доменное

имя **www.example.com**. На экране должна появиться информация такого вида:

PC>ping www.example.com

Pinging 192.168.3.100 with 32 bytes of data:

Reply from 192.168.3.100: bytes=32 time=138ms TTL=126
Reply from 192.168.3.100: bytes=32 time=156ms TTL=126
Reply from 192.168.3.100: bytes=32 time=172ms TTL=126
Reply from 192.168.3.100: bytes=32 time=140ms TTL=126
Ping statistics for 192.168.3.100:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 138ms, Maximum = 172ms, Average = 151ms

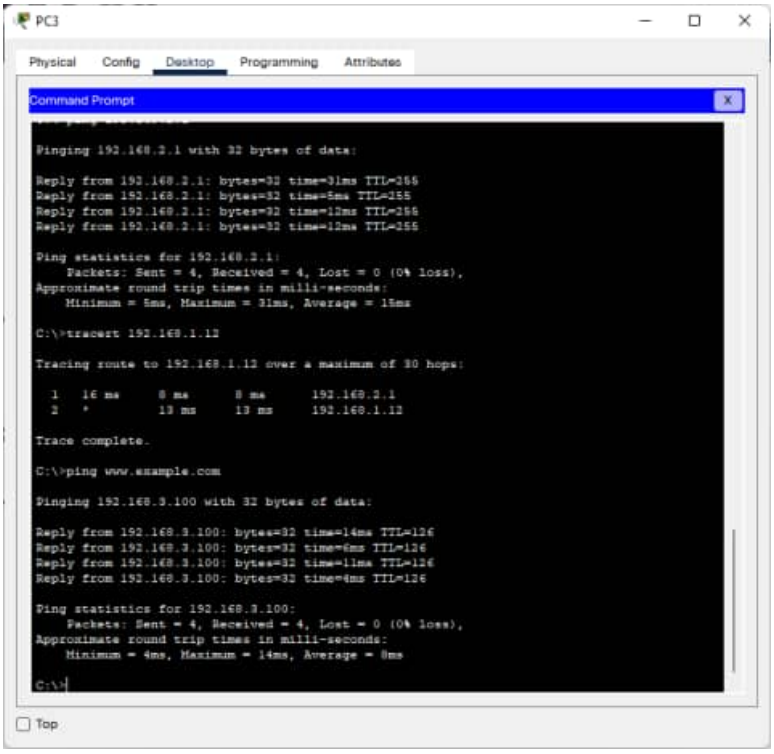


Рисунок 45 – Результат ввода команды ping на PC3

Обратите внимание, что доменное имя **www.example.com** было преобразовано сервером DNS в IP-адрес веб-сервера, 192.168.3.100. Это свидетельствует о правильной работе сервера DNS.

На данный момент все запросы DNS были автоматически выполнены другими приложениями. Для элемента (c) запрос был выполнен веб-браузером, а для элемента (f) — командой ping. Для создания запросов DNS прямо на сервере используйте команду **nslookup**.

g. В командной строке **PC3 Command Prompt** введите команду **nslookup www.example.com**. Команда и ее выходные данные должны иметь следующий вид:

PC>nslookup www.example.com
Server: [192.168.3.100]
Address: 192.168.3.100
Non-authoritative answer:
Name: www.example.com
Address: 192.168.3.100
PC>

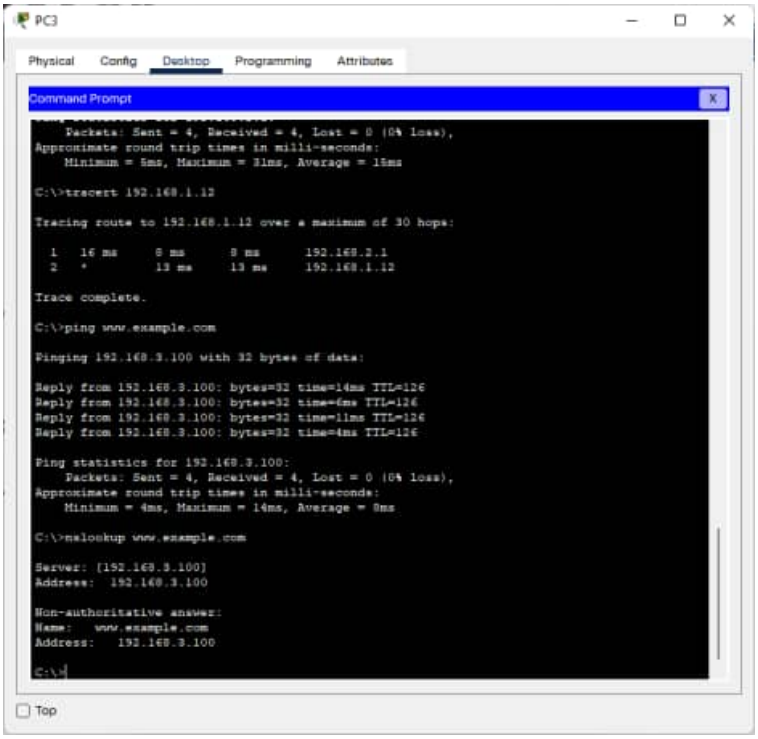


Рисунок 46 – Результат ввода команды nslookup на PC3

После того, как команда **nslookup** будет введена в указанном выше формате, на сервер DNS будет отправлен запрос «**Укажите IP-адрес, связанный с именем www.example.com**».

В первой строке выходных данных команды содержится имя сервера DNS, получившего запрос DNS. Компьютер PC3 отправил запрос на адрес 192.168.3.100, поскольку он получил сведения от WRS1 через DHCP, что адрес 192.168.3.100 использовался для преобразования имен. Поскольку для адреса 192.168.3.100 не было определено имени, на экране был показан IP-адрес.

Во второй строке содержится IP-адрес сервера DNS, использованного для запроса.

В третьей, четвертой и пятой строках содержится ответ на запрос: имя **www.example.com** связано с IP-адресом 192.168.3.100.

Ваш процент выполненных заданий должен быть 100%. Если это не так, щёлкните **Check Results** (Проверить результаты), чтобы посмотреть, выполнение каких компонентов еще не завершено.

Лабораторная работа №6
Работа с эмулятором сетей Cisco Packet Tracer.
Прокладка кабелей в простой сети

Задачи

- Получить представление об основных функциях Packet Tracer.
- Создать простую сеть из двух узлов.
- Узнать, насколько важно использовать правильный тип кабеля при подключении компьютеров.

Совет. Чтобы во время выполнения интерактивного задания инструкции оставались видимыми, установите флажок ***"Top"*** *(поверх) в нижнем левом углу окна с указаниями.*

Примечание. Интерактивное задание начинается пустым рабочим пространство. В этом пространстве вам предстоит размещать и подключать сетевые устройства.

Этап 1. Создание схемы сети с двумя компьютерами

В нижнем левом углу окна Packet Tracer отображены девять значков, соответствующие категориям или группам устройств, например, Routers (Маршрутизаторы), Switches (Коммутаторы) или End Devices (Оконечные устройства).

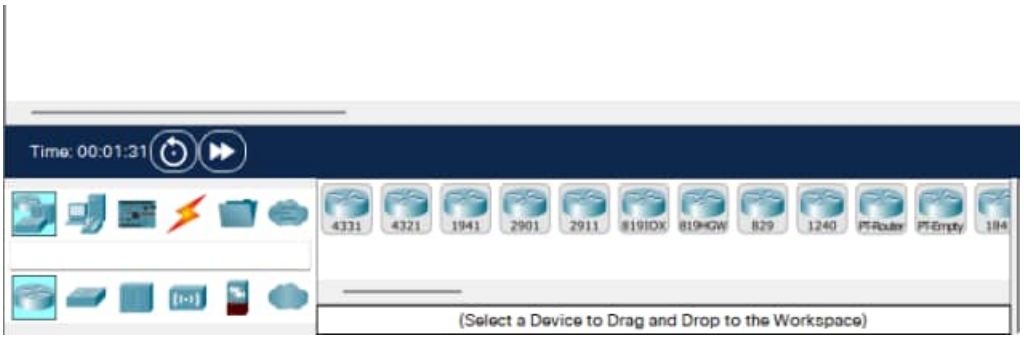


Рисунок 47 – Панель с доступными устройствами Cisco Packet Tracer

При перемещении курсора по категориям устройств в окне, находящемся между рядами устройств, отображается имя категории. Чтобы выбрать устройство, выберите сначала его категорию. После выбора категории устройств в окне рядом со списком категорий появятся варианты, доступные в выбранной категории. Выберите нужный вариант.

a. Выберите **End Devices** (Оконечные устройства) из списка в левом нижнем углу.

b. Перетащите два стандартных ПК (PC-PT) в зону **Logical Workspace** (Логическое рабочее пространство).

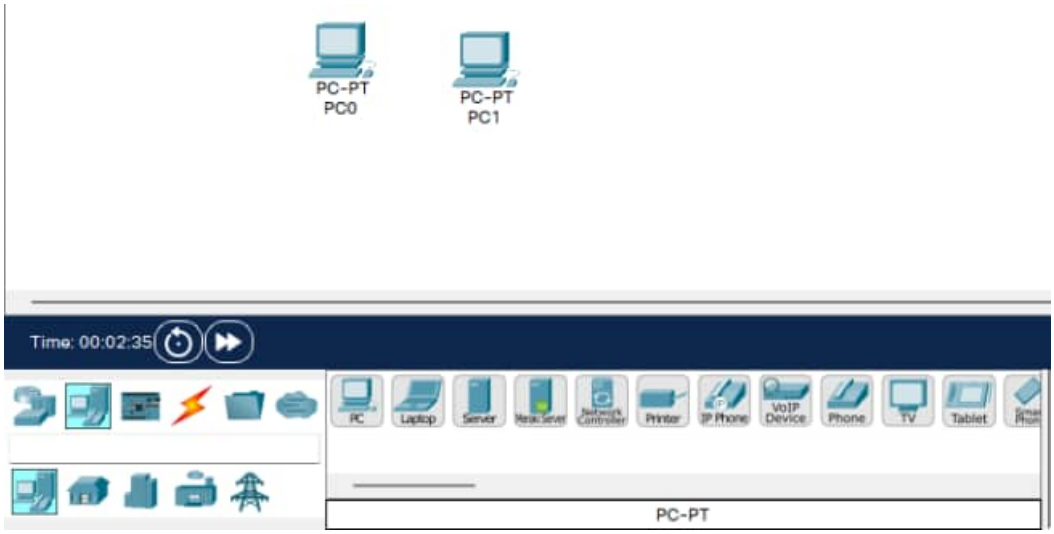


Рисунок 48 – Добавление компьютеров на рабочее пространство

c. Выберите пункт **Connections** (Подключения) из списка в левом нижнем углу.

d. Выберите тип кабеля **Copper Straight-Through** (Медный прямой).

e. Щёлкните первый узел **PC0** и назначьте выбранный кабель разъёму **FastEthernet** .

f. Щёлкните второй узел **PC1** и назначьте выбранный кабель разъёму **FastEthernet** .

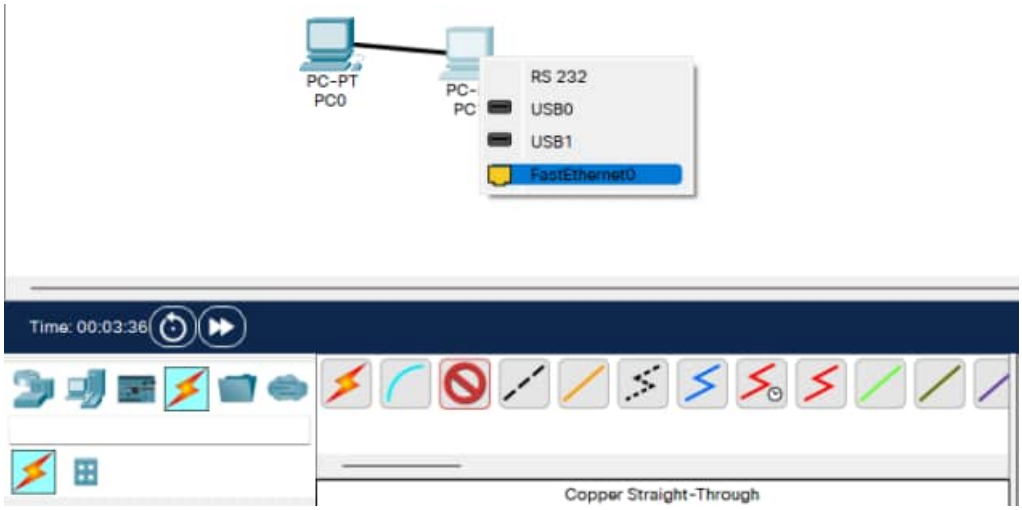


Рисунок 49 – Подключение кабеля между 2 компьютерами.

g. Красные точки указывают, что выбран неверный тип кабеля. Щёлкните красный значок **X** в правой части окна Packet Tracer. Это позволит удалить кабель **Copper Straight-Through** (Медный прямой).

h. Подведите курсор к названию кабеля и щёлкните кабель, чтобы удалить его.

i. Выберите тип кабеля **Copper Cross-Over** (Медный перекрестный).

j. Щёлкните первый узел **PC0** и назначьте выбранный кабель разъёму **FastEthernet** .

k. Щёлкните второй узел **PC1** и назначьте выбранный кабель разъёму **FastEthernet** . Зеленые точки на обоих концах кабеля означают, что выбран правильный тип кабеля.

Рисунок 50 – Отображение 2 правильно подключенных компьютеров.

Этап 2. Настройка имен узлов и IP-адресов на компьютерах

a. Щёлкните значок **PC0**. Появится окно **PC0**.

b. В окне **PC0** выберите вкладку **Config**.

c. Измените **Display Name** (Отображаемое имя) ПК на **PC-A**.

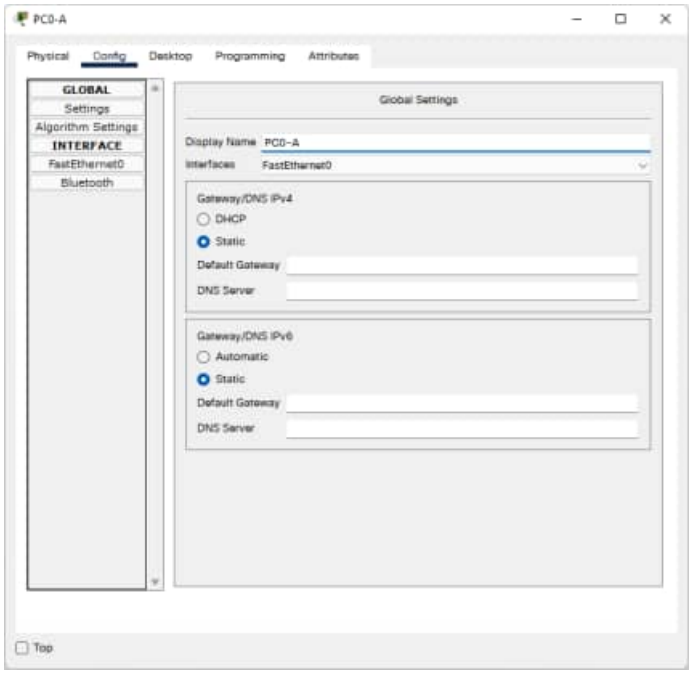


Рисунок 51 – Вкладка Config окна настройки комьютера

d. Выберите вкладку **FastEthernet** в левой части экрана.

e. В разделе "IP Configuration" (Настройка IP) введите IP-адрес **192.168.1.1** и маску подсети **255.255.255.0** .

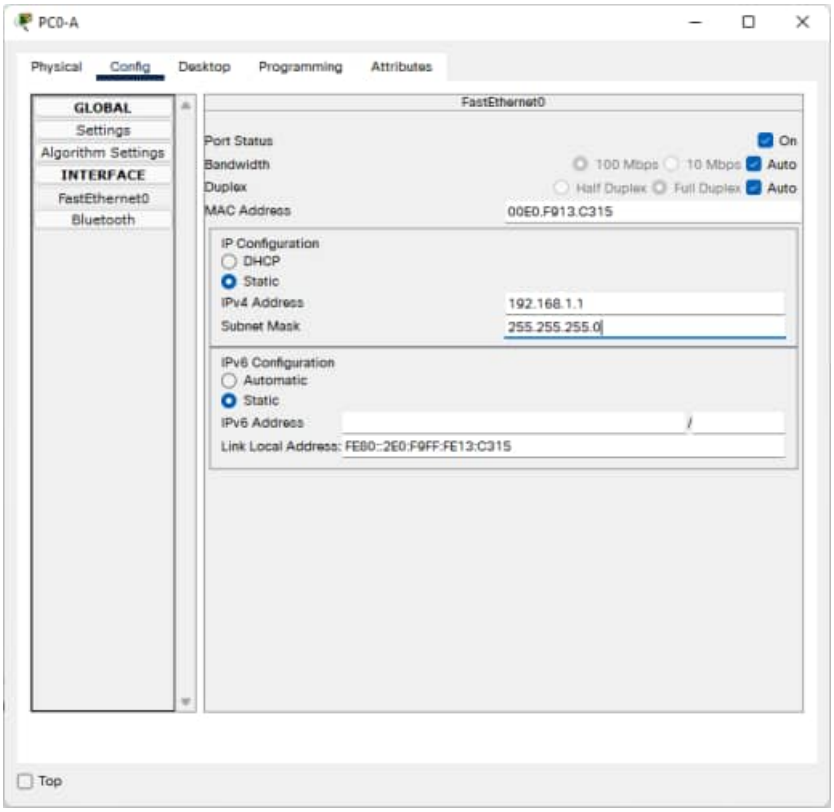


Рисунок 52 – Раздел настройки IP компьютера

f. Закройте окно настройки **PC-A**, щёлкнув значок **x** в правом верхнем углу окна.

g. Щёлкните **PC1**. Появится окно **PC1** .

h. В окне **PC1** выберите вкладку **Config** (Настройка).

i. Измените **Display Name** (Отображаемое имя) ПК на **PC-B**.

j. Выберите вкладку **FastEthernet** в левой части экрана.

k. В разделе "IP Configuration" (Настройка IP) укажите IP-адрес **192.168.1.2** и маску подсети **255.255.255.0**.

l. Закройте окно настройки **PC-B**, щёлкнув значок **x** в правом верхнем углу окна.

Этап 3. Создание схемы сети с двумя компьютерами и концентратором

На этапе 1 два компьютера были подключены друг к другу кабелем **Copper Cross-Over** (Медный перекрестный). Этот тип подключения является простым способом подключения двух компьютеров друг к другу. Для подключения двух или более компьютеров используется концентратор.

a. Начните создание новой настройки с нажатия красного значка **X** в правой части окна Packet Tracer, чтобы удалить тип кабеля **Copper Cross-Over** (Медный перекрестный), подключающий **PC-A** к **PC-B**.

b. Выберите пункт **Hubs** (Концентраторы) из списка в левом нижнем углу.

c. Перетащите типовой концентратор (Hub-PT) в зону **Logical Workspace** (Логическое рабочее пространство).

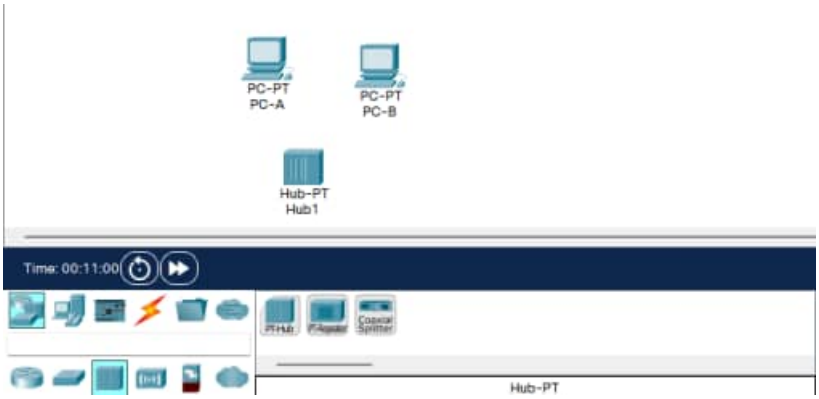


Рисунок 53 – Добавление концентратора на рабочее пространство

d. Выберите пункт **Connections** (Подключения) из списка в левом нижнем углу.

e. Выберите тип кабеля **Copper Cross-Over** (Медный перекрестный).

f. Щёлкните первый узел **PC-A** и назначьте выбранный кабель разъёму **FastEthernet** .

g. Щёлкните концентратор **Hub0** и выберите порт подключения **Port 0** для подключения к **PC-A**.

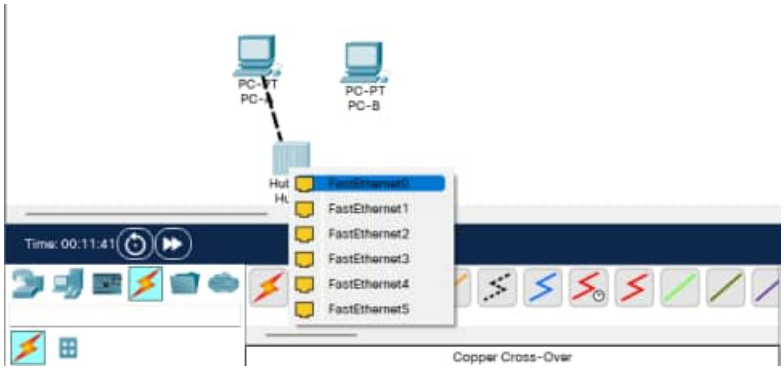


Рисунок 54 – Подключение компьютера к концентратору

h. Красные точки указывают, что выбран неверный тип кабеля. Щёлкните красный значок **X** в правой части окна Packet Tracer и удалите кабель **Copper Cross-Over** (Медный перекрестный).

i. Выберите тип кабеля **Copper Straight-Through** (Медный прямой).

j. Щёлкните первый узел **PC-A** и назначьте выбранный кабель разъёму **FastEthernet** .

k. Щёлкните концентратор **Hub0** и нажмите **Port 0** для подключения к **PC-A**.

l. Снова выберите тип кабеля **Copper Straight-Through** (Медный прямой).

m. Щёлкните второй узел **PC-B** и назначьте выбранный кабель разъёму **FastEthernet**.

n. Щёлкните концентратор **Hub0** и нажмите **Port 1** для подключения к **PC-B**.

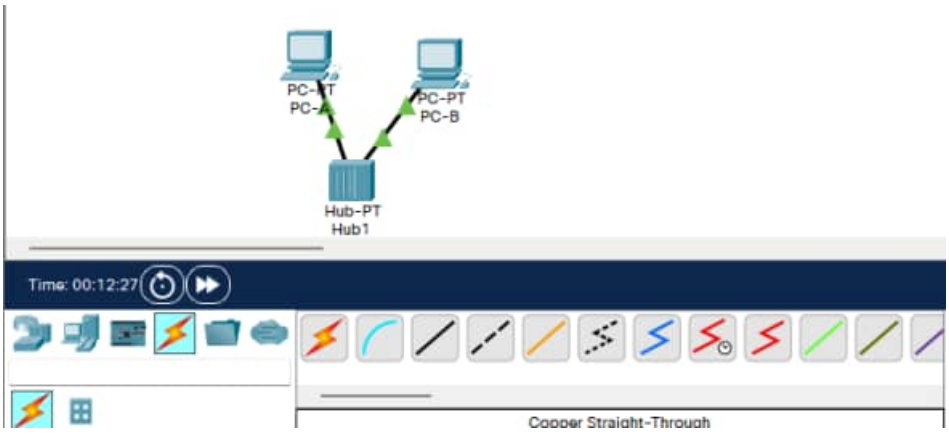


Рисунок 55 – Отображение правильно подключенных устройств к концентратору

Этап 4. Замена концентратора на коммутатор

На этапе 3 была создана сеть с концентратором. Эта сеть является действующей, однако ее производительность можно улучшить, используя вместо концентратора коммутатор. Замените концентратор на коммутатор.

a. Выберите концентратор и щёлкните красный значок **X** в правой части окна Packet Tracer. При выполнении этого действия будет удален концентратор и подключенные к нему кабели.
b. Выберите **Switches** (Коммутаторы) из списка в левом нижнем углу.
c. Перетащите коммутатор 2950-24 в зону **Logical Workspace** (Логическое рабочее пространство).

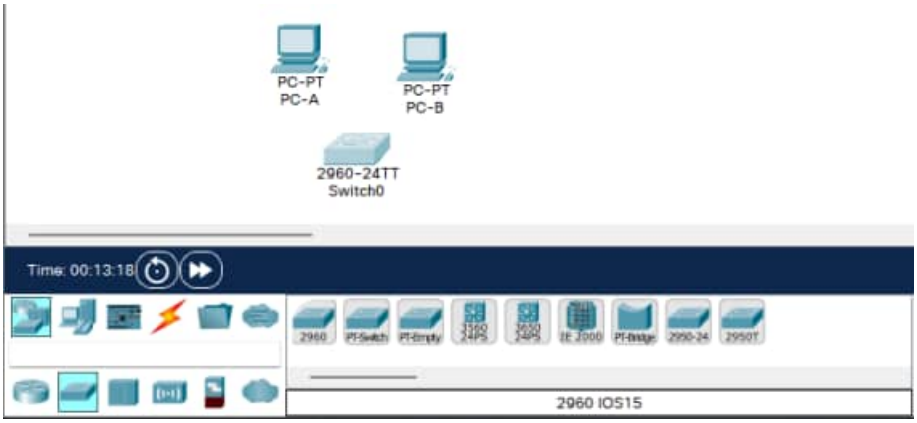


Рисунок 56 – Добавление коммутатора на рабочую область

d. Выберите пункт **Connections** (Подключения) из списка в левом нижнем углу.
e. Выберите тип кабеля **Copper Straight-Through** (Медный прямой).
f. Щёлкните первый узел **PC-A** и назначьте выбранный кабель разъёму **FastEthernet** .

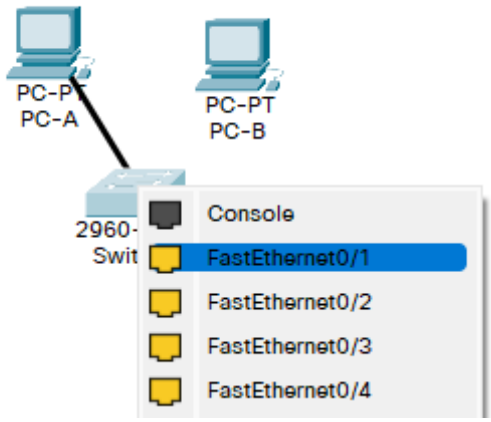


Рисунок 57 – Подключение компьютера к коммутатору

g. Щёлкните коммутатор **Switch0** и выберите порт подключения **FastEthernet0/1** для подключения к **PC-A**. Примерно через одну минуту на концах кабеля **Copper Straight-Through** (Медный прямой) должны появиться две зеленые точки. Они обозначают, что был использован правильный тип кабеля.

h. Снова выберите тип кабеля **Copper Straight-Through** (Медный прямой).

i. Щёлкните второй узел **PC-B** и назначьте выбранный кабель разъёму **FastEthernet**.

j. Щёлкните коммутатор **Switch0** и выберите порт подключения **FastEthernet0/2** для подключения к **PC-B**.

k. Нажмите кнопку **Check Results** (Проверить результаты) в нижней части окна с инструкцией, чтобы проверить правильность топологии.

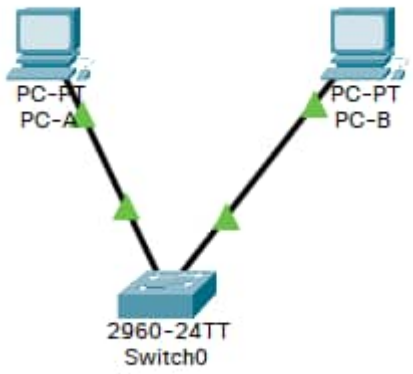


Рисунок 58 – Отображение правильно подключенных устройств к коммутатору

Ваш процент выполненных работ должен составлять 100%. На вкладке **Assessment Items** (Оцениваемые пункты) отображаются оценки за каждый элемент упражнения.

Лабораторная работа №7
Работа с эмулятором сетей Cisco Packet Tracer.
Физические топологии
Цели обучения

Ознакомление с физическими топологиями следующих типов: «звезда», «расширенная звезда» и «полносвязная»

Введение

При выполнении этого интерактивного задания вам предстоит построить несколько различных физических топологий, используя указанные устройства. Изучаются следующие физические топологии:

- звезда;
- расширенная звезда (или «иерархическая»);
- полносвязная.

После соединения устройств в указанные физические топологии вам предстоит соединить различные топологии между собой.

Задача 1. Соединение устройств в физическую топологию «звезда»
Этап 1. Соединение устройств первой «звезды»

a. Найдите следующие устройства: **PC00**, **PC01**, **PC02**, **PC03** и **SW0**. Они находятся в левом верхнем углу рабочего пространства Packet Tracer. Эти устройства будут соединены в топологию «звезда».
b. В меню **Connections** (Подключения) выберите тип кабеля **Copper Straight-Through** (Медный прямой).

   Совет. Удерживайте клавишу **<Control>** при нажатии значка **Copper Straight-Through** (Медный прямой) для добавления нескольких подключений.
c. Соедините указанные ПК с коммутатором **SW0**. Подключите устройства к портам коммутатора SW0: **PC00** к порту **Fast-Ethernet0/1**, **PC01 к порту Fast-Ethernet0/2**, **PC02 к порту Fast-Ethernet0/3** и **PC03 к порту Fast-Ethernet0/4**.
d. Теперь устройства соединены в топологию «звезда», где центральным является устройство **SW0**.

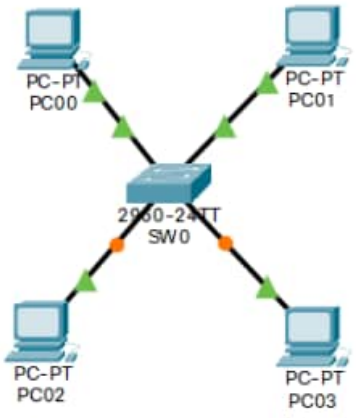


Рисунок 59 – Подключение компьютеров к коммутатору SW0

Задача 2. Создание других звезд

Этап 1. Соединение устройств второй «звезды»

a. Найдите устройства **SW1**, **PC10**, **PC11**, **PC12** и **PC13**. Эти устройства должны быть в левом нижнем углу рабочего пространства Packet Tracer.

b. По аналогии с задачей 1 соедините устройства во вторую «звезду». В меню **Connections** (Подключения) выберите тип кабеля **Copper Straight-Through** (Медный прямой).

   Совет. Удерживайте клавишу **<Control>** при нажатии значка **Copper Straight-Through** (Медный прямой) для добавления нескольких подключений.

c. Подключите устройства к портам коммутатора SW1: **PC10 к порту Fast-Ethernet0/1**, **PC11 к порту Fast-Ethernet0/2**, **PC12 к порту Fast-Ethernet0/3** и **PC13 к порту Fast-Ethernet0/4**.

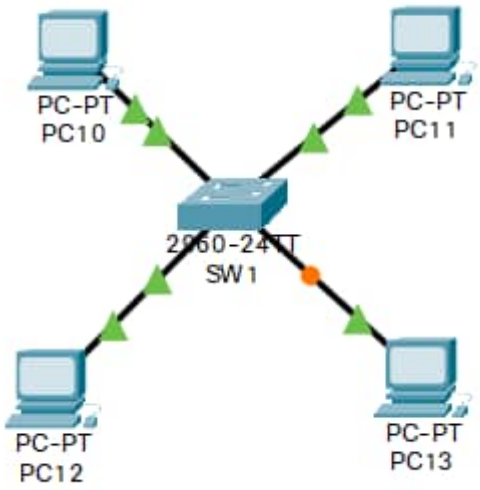


Рисунок 60 – Подключение компьютеров к коммутатору SW1

d. Второй набор устройств соединен в топологию «звезда».

Этап 2. Соединение устройств третьей «звезды»

a. Найдите устройства **SW2**,**PC20**, **PC21**, **PC22** и **PC23.** Эти устройства находятся в верхней центральной части рабочего пространства Packet Tracer.

b. Соедините устройства в третью «звезду». В меню **Connections** (Подключения) выберите тип кабеля **Copper Straight-Through** (Медный прямой).

Совет. Удерживайте клавишу **<Control>** при нажатии значка **Copper Straight-Through** (Медный прямой) для добавления нескольких подключений.

c. Подключите устройства к портам коммутатора SW2: **PC20** к порту **Fast-Ethernet0/1**, **PC21 к порту Fast-Ethernet0/2**, **PC22 к порту Fast-Ethernet0/3** и **PC23 к порту Fast-Ethernet0/4**.

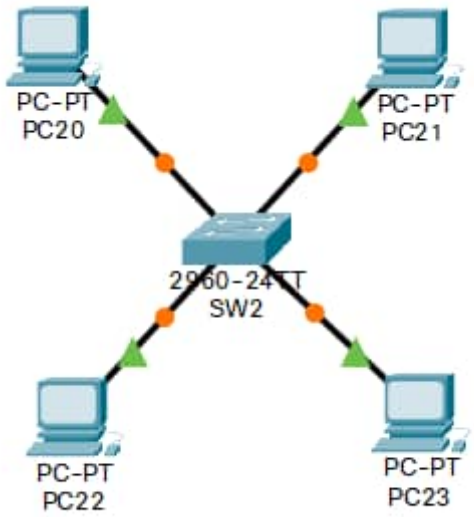


Рисунок 61 – Подключение компьютеров к коммутатору SW2

d. Таким образом, получается третья «звезда».

Этап 3. Соединение устройств четвертой «звезды»

a. Найдите устройства **SW3**, **PC30**, **PC31**, **PC32** и **PC33.** Эти устройства находятся в нижней центральной части рабочего пространства Packet Tracer.

b. Соедините устройства в четвертую «звезду». В меню **Connections** (Подключения) выберите тип кабеля **Copper Straight-Through** (Медный прямой).

Совет. Удерживайте клавишу **<Control>** при нажатии значка **Copper Straight-Through** (Медный прямой) для добавления нескольких подключений.

c. Подключите устройства к портам коммутатора SW2: **PC20** к порту **Fast-Ethernet0/1**, **PC21 к порту Fast-Ethernet0/2**, **PC22 к порту Fast-Ethernet0/3** и **PC23 к порту Fast-Ethernet0/4**.

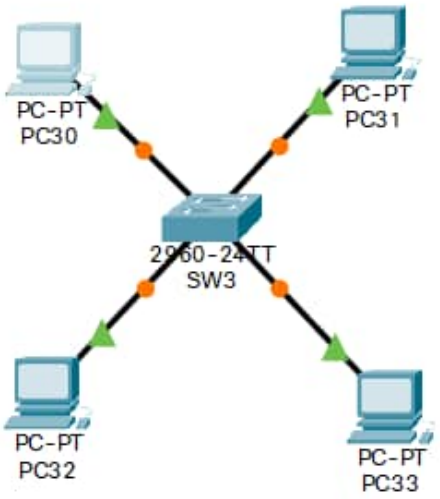


Рисунок 62 – Подключение компьютеров к коммутатору SW3

d. Таким образом, получается четвертая «звезда».

Топология «звезда» очень полезна в случае отказов. Например, если на коммутаторе **SW1** происходит отказ порта **Fast-Ethernet0/1**, это отразится только на работе узла **PC10** . Очень распространенным способом является помещение «звезды» на «спицу», при котором создается топология **расширенная звезда.**

Задание 3. Создание «расширенной звезды»

a. Найдите устройства **SW0**,**SW1**, **SW2**, **SW3** и **Dist_SW.**

b. В меню **Connections** (Подключения) выберите тип кабеля **Copper Cross-Over** (Медный перекрестный).

c. Подключите устройства **SW0**,**SW1**, **SW2** и **SW3** к **Dist_SW** в соответствии с приведенной ниже таблицей:

Устройства	**Порт коммутатора**	**Порт устройства Dist_SW**
SW0	Fast-Ethernet0/24	Fast-Ethernet0/10
SW1	Fast-Ethernet0/24	Fast-Ethernet0/11
SW2	Fast-Ethernet0/24	Fast-Ethernet0/12
SW3	Fast-Ethernet0/24	Fast-Ethernet0/13

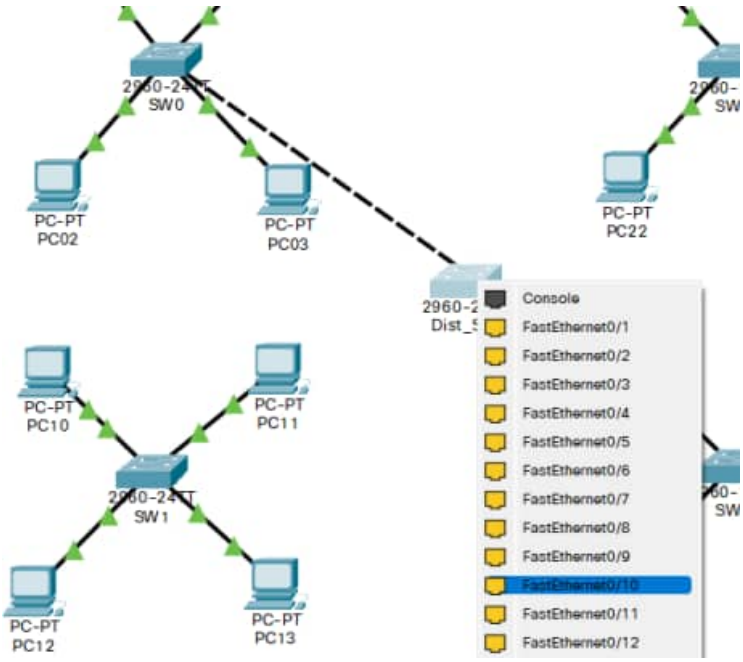


Рисунок 63 – Подключение коммутатора SW0 к Dist_SW

d. Таким образом, мы получаем «расширенную звезду», где четыре маленькие звезды являются «спицами».

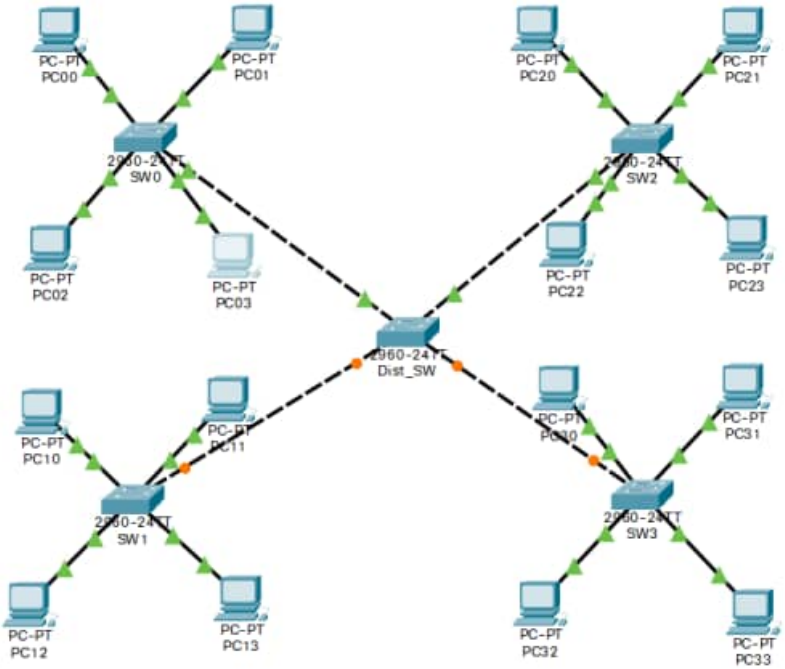


Рисунок 64 – Схема сети

Задача 4. Создание полносвязной топологии

Самым большим недостатком топологии «звезда» является наличие основной точки отказа. Отказ устройства, выполняющего функцию концентратора «звезды», ведет к отказу всех остальных устройств, составляющих

топологию. В случаях, когда существование единой точки отказа недопустимо, подключение устройств обычно осуществляется в виде **полносвязной** топологии. При подключении каждого устройства ко всем остальным устройствам создается избыточная топология.
В этом задании вам предстоит подключить магистральные устройства **MainCulster_SW1**, **MainCulster_SW2** и **MainCulster_SW3** и создать полносвязную топологию, подключая каждое из них ко всем остальным. Поскольку мы имеем три устройства, каждое из них должно иметь два исходящих канала.

Этап 1. Соединение магистральных коммутаторов.

a. Найдите устройства **MainCluster_SW1**, **MainCluster_SW2** и **MainCluster_SW3**. Эти устройства находятся в правой части рабочей области Packet Tracer.

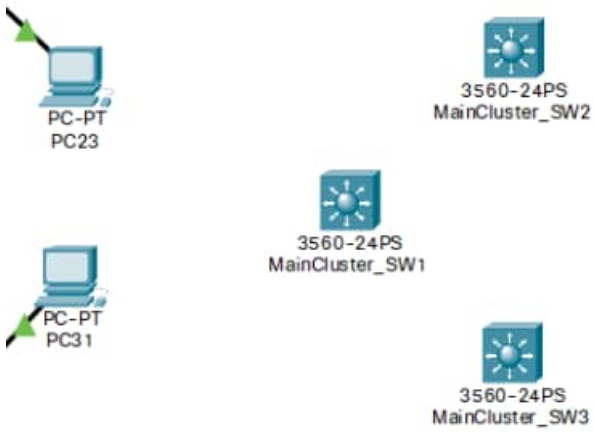


Рисунок 65 – Магистральные коммутаторы

b. В меню **Connections** (Подключения) выберите тип кабеля **Copper Cross-Over** (Медный перекрестный).

   Совет. Удерживайте клавишу **<Control>** при нажатии значка **Copper Straight-Through** (Медный прямой) для добавления нескольких подключений.

c. Подключите магистральные устройства по указанной ниже схеме:

Исходное устройство	**Исходный порт**	**Устройство назначения**	**Порт назначения**
MainCluster_SW1	GigabitEthernet0/1	MainCluster_SW2	GigabitEthernet0/1
MainCluster_SW1	GigabitEthernet0/2	MainCluster_SW3	GigabitEthernet0/1
MainCluster_SW2	GigabitEthernet0/2	MainCluster_SW3	GigabitEthernet0/2

d. Теперь, когда все коммутаторы **MainCluster** подключены друг к другу, между ними создана полносвязная топология.

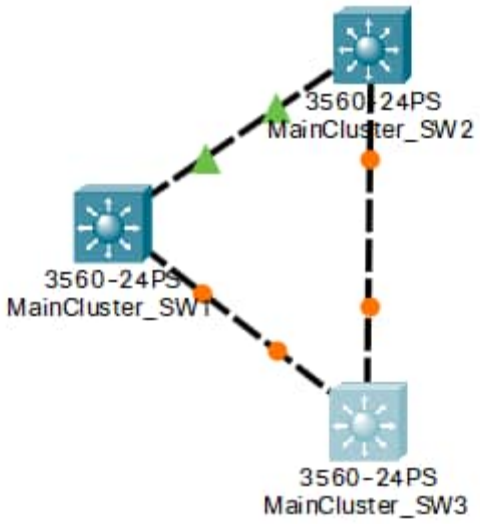


Рисунок 66 – Подключение магистральных маршрутизаторов между собою

Этап 2. Создание гибридной топологии

a. В меню **Connections** (Подключения) выберите тип кабеля **Copper Cross-Over (Медный перекрестный).**

b. Подключите порт **Fast-Ethernet0/24** устройства **MainCluster_SW1** к порту **Fast-Ethernet0/24** устройства **Dist_SW**. В результате подключения полносвязной топологии к топологии «расширенная звезда» получается гибридная топология.

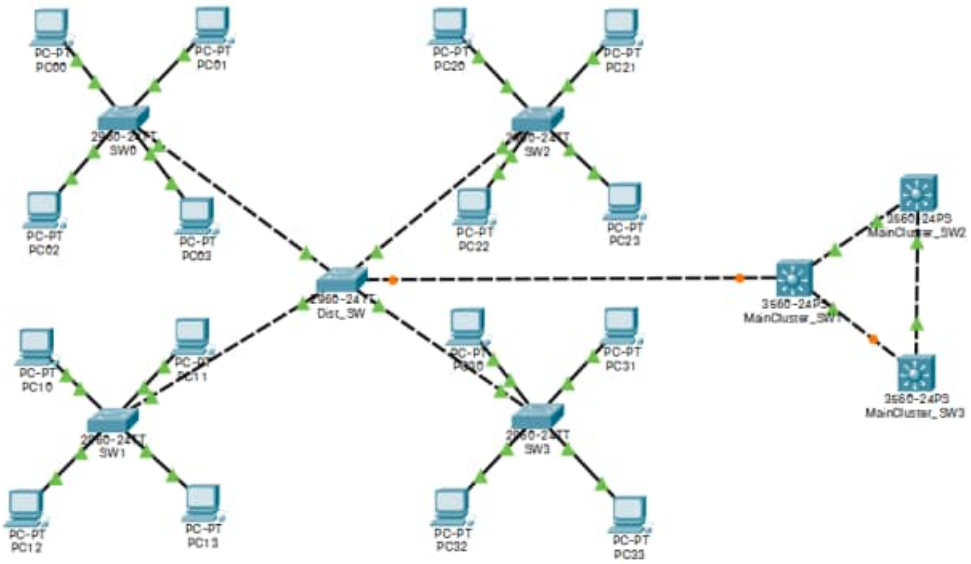


Рисунок 67 – Схема сети.

c. Packet Tracer должен сообщить о том, что задание выполнено на 100%. **Примечание.** Если ваш результат не составил 100%, еще раз проверьте используемые порты. В этом упражнении Packet Tracer также оценивает выбор порта.

Задача 5. Проверочное задание

Этап 1. Анализ точек отказа (и повышение избыточности)

a. Сколько точек отказа можно обнаружить?

   Поскольку в этом упражнении не предполагалась избыточность, существует несколько точек отказа. Наиболее очевидными из них являются каналы и концентраторы.

b. Как можно уменьшить количество точек отказа?

   Для магистральных устройств простым решением было бы дублирование каналов связи между магистральными устройствами. В случае отказа одного из каналов (вызванного проблемой с портом или кабелем), связь может осуществляться при помощи второго канала.

   Dist_SW также может быть дублирован. Благодаря наличию еще одного устройства, работающего в качестве концентратора «расширенной звезды», вся топология будет в меньшей степени подвержена отказам.

Лабораторная работа №8
Работа с эмулятором сетей Cisco Packet Tracer.
Установка адаптера беспроводной сети
Цели обучения

- Установка адаптера беспроводной сети.
- Настройка компьютера для подключения к беспроводной сети.

Введение

В этом интерактивном задании необходимо выполнить настройку компьютера **PC2** для беспроводного подключения к существующей беспроводной сети с помощью маршрутизатора Linksys WRT300N (**WRS1**).

Задача 1. Настройка проводных ПК для работы в беспроводной сети

Этап 1. Замена в компьютере PC2 адаптера проводной сети на адаптер беспроводной сети

а. Уберите кабель Ethernet, соединяющий компьютер **PC2** с **WRS1**, нажав кнопку **X** на правой панели и щёлкнув кабель Ethernet.

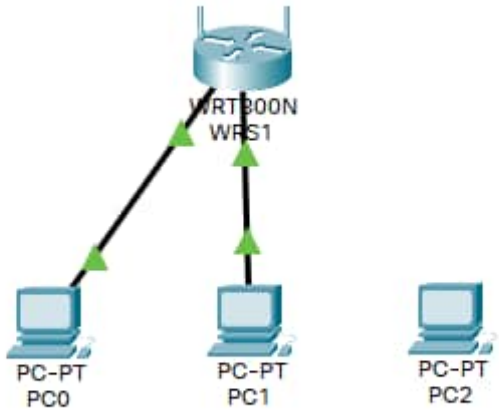


Рисунок 68 – Схема сети после удаления подключения

б. Щёлкните **PC2**, чтобы открыть его окно настройки.

в. Щёлкните вкладку **Physical** (Физическое устройство).

г. Отключите питание устройства **PC2**, нажав кнопку питания. (Чтобы увидеть кнопку питания, прокрутите страницу вверх или вниз).

д. Удалите проводную плату, перетащив ее с корпуса компьютера обратно в список модулей. (Чтобы увидеть установленную на компьютер плату, прокрутите страницу вверх или вниз).

е. Из списка **Modules** (Модули), который находится слева, перетащите модуль **Linksys-WMP300N** в пустое гнездо **PC2** для установки.

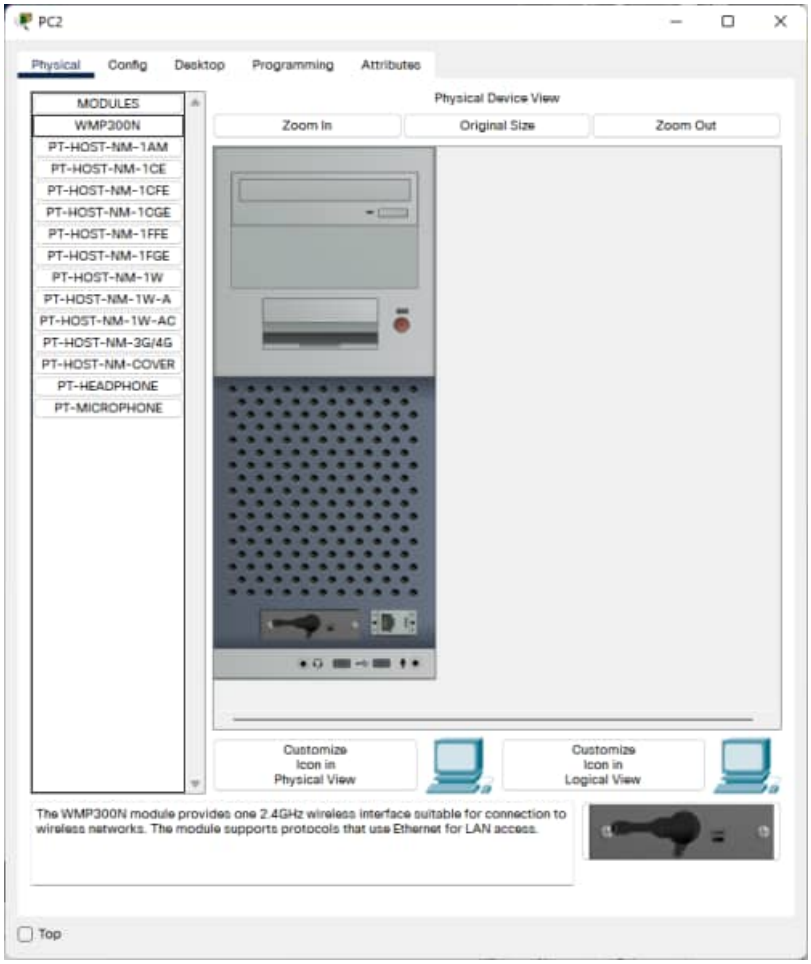


Рисунок 69 – Вкладка установки модулей

ж. Нажмите кнопку питания, чтобы снова включить **PC2.**

з. Поскольку **PC2** не был настроен для подсоединения к существующей сети, на данном этапе не удается выполнить подключение. Перейдите к этапу 2.

Этап 2. Настройка PC2 для подключения к существующей беспроводной сети.

а. Щёлкните **PC2**, чтобы открыть окно настройки устройства.

б. Щёлкните вкладку **Desktop** (Рабочий стол) и выберите пункт **PC Wireless** (Беспроводной компьютер).

в. Обратите внимание, что компьютер **PC2** еще не связан ни с одной беспроводной сетью.

г. В пункте **PC Wireless** (Беспроводной компьютер) щёлкните вкладку **Connect** (Подключение).

д. Подождите несколько секунд, чтобы **PC2** определил сигналы-маяки, отправляемые маршрутизатором **WRS1** по беспроводной сети. После этого в столбце **Wireless Network Name** (Имя беспроводной сети) под именем **aCompany** отобразится существующая беспроводная сеть.

е. Щёлкните сеть под именем **aCompany**, выбирая ее как сеть, которую вы будете использовать, и нажмите кнопку **Connect** (Подключение).

Рисунок 70 – Окно подключения к беспроводной сети

ж. Убедитесь, что устройство **PC2** связано с маршрутизатором **WRS1**, щёлкнув вкладку **Link Information** (Информация о каналах).

з. Перейдите по вкладкам **Desktop > IP Configuration** (Рабочий стол > Настройка IP) и выберите пункт **DHCP**.

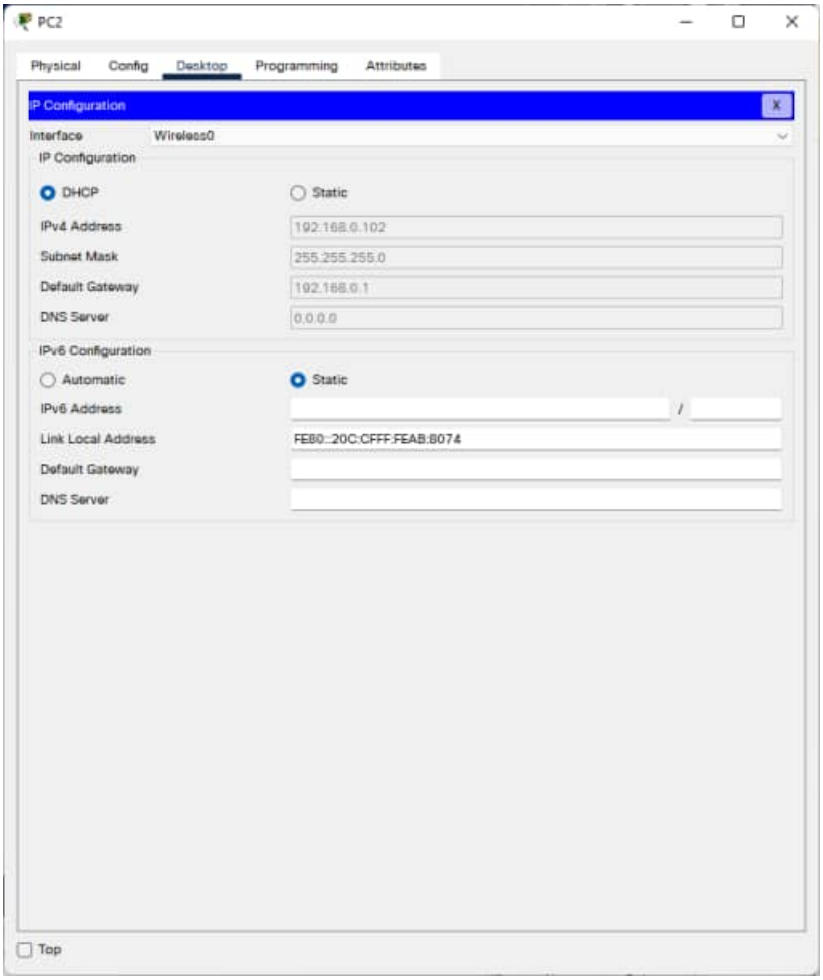


Рисунок 71 – Окно настройки IP компьютера PC2

и. **PC2** теперь готов к установлению связи с другими устройствами, подключенными к сети.

Задача 2. Проверка настройки адресов PC2

Этап 1. Просмотр настройки IP-адресов PC2.

а. Щёлкните **PC2**, чтобы открыть окно настройки устройства.

б. На вкладке Desktop (Рабочий стол) нажмите кнопку **Command Prompt** (Командная строка).

в. Введите команду **ipconfig /all** и нажмите клавишу **Ввод**.

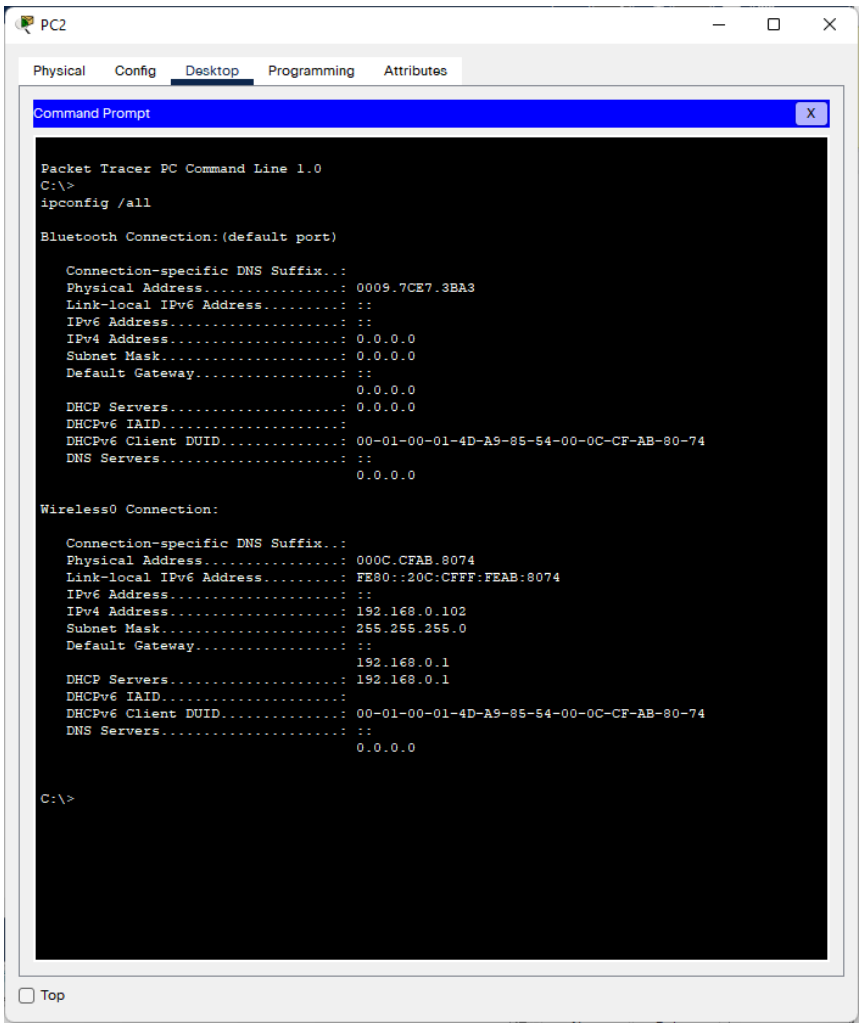


Рисунок 72 – Результат выполнения команды ipconfig /all на PC2

г. Проверьте IP-адрес данного компьютера.

192.168.1.103

д. Проверьте маску подсети данного компьютера.

255.255.255.0

е. Проверьте основной шлюз данного компьютера.

192.168.0.1

ж. В том же окне **Command Prompt** (Командная строка) устройства **PC2** отправьте эхо-запрос двум другим устройствам в сети (**PC0 и PC1**). Эхо-тестирование должно пройти успешно.

Примечание: Поскольку сведения об IP были получены через DHCP, IP-адреса могут отличаться от приведенных выше.

Проверьте свой результат. Он должен составлять 100%.

Лабораторная работа №9
Работа с эмулятором сетей Cisco Packet Tracer.
Изучение веб-запросов

Задача

Просмотрите трафик клиента или сервера, отправленного с ПК на веб-сервер при запросе к веб-службам.

Упражнение начнется со 100% выполнения. Это связано с тем, что данное упражнение предназначено для демонстрации потока пакетов между ПК и веб-сервером. Оно не подразумевает постепенного выполнения.

Шаг 1. Проверка соединения с веб-сервером

а. Нажмите кнопку «External Client» (внешний клиент) и перейдите к **Command Prompt** (командная строка) из вкладки **Desktop** (рабочий стол).

б. С помощью команды **ping** проверьте достижимость URL-адреса **ciscolearn.web.com**.

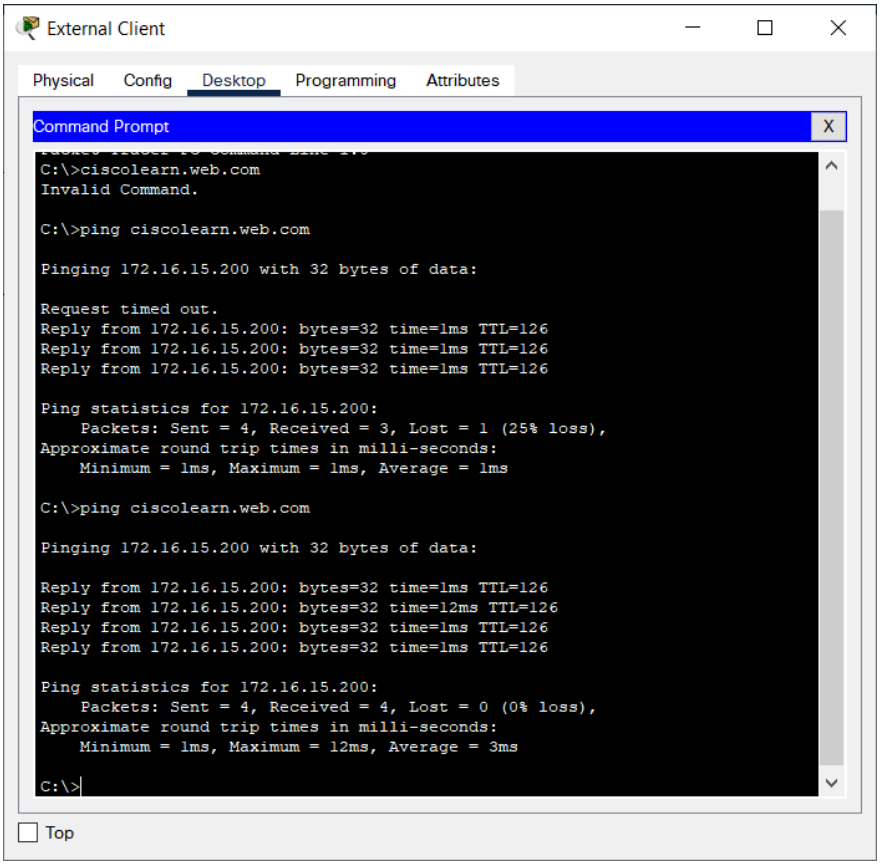


Рисунок 73 – Результат выполнения команды ping на удаленном компьютере

*Обратите внимание, что IP-адрес включен в выходные данные команды **ping**. Такой адрес предоставляется DNS-сервером. При пересылке любого трафика через сеть используются сведения об IP-адресе.

в. Закройте окно Command Prompt (командная строка), однако окно рабочего стола "External Client" (внешний клиент) оставьте открытым.

Шаг 2. Подключение к веб-серверу

а. Откройте **Веб-обозреватель** с рабочего стола.

б. В строке URL-адреса введите **ciscolearn.web.com**.

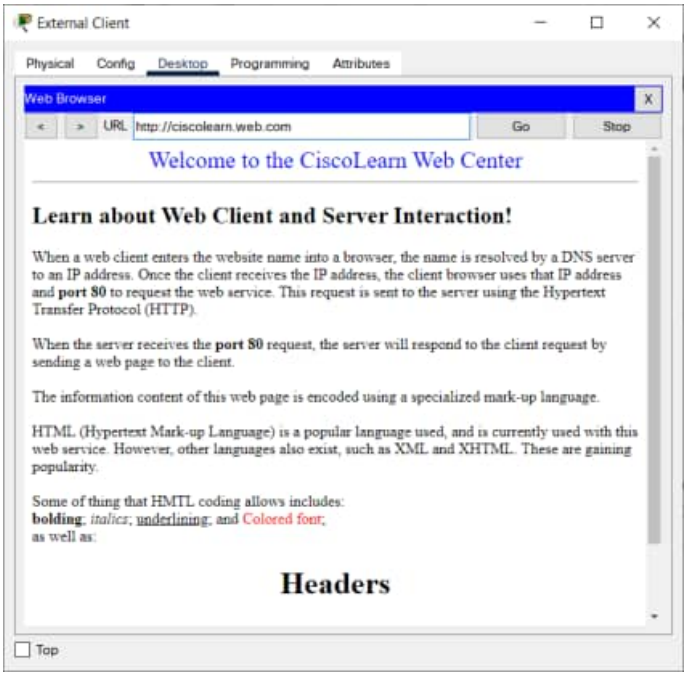


Рисунок 74 – Отображаемая веб-страница

*Убедитесь в возможности чтения отображаемой веб-страницы. Оставьте данную страницу открытой.

Шаг 3. Просмотр HTML-кода

а. Щелкните ссылку сервера **ciscolearn.web.com**.

б. Перейдите на вкладку **Config** (настройка) > **HTTP**.

в. Сравните текст с HTML-кодами, записанный на сервере, со страницей, отображаемой в веб-обозревателе внешнего клиента. Может потребоваться повторно развернуть окно внешнего клиента, если оно свернулось при открытии окна сервера.

г. Закройте окно внешнего клиента и веб-сервера.

Шаг 4. Обзор трафика между клиентом и веб-сервером.

а. Войдите в режим моделирования, для чего нажмите вкладку **Simulation** (моделирование) в правом нижнем углу. Вкладка **Simulation** (моделирование) расположена за вкладкой **Realtime** (в реальном времени) и обозначается символом секундомера.

б. Дважды щелкните панель «Simulation» (панель моделирование), чтобы заблокировать ее из PT-окна. Это позволяет перемещать панель «Simulation» (панель моделирование) для просмотра всей топологии сети.

в. Просмотрите трафик путем создания сложного PDU в режиме моделирования.

1. В **Simulation Panel** (панели моделирования) выберите команду **Edit Filters** (изменить фильтры) и поставьте флажки в полях TCP и HTTP.

2. Добавьте **Complex PDU** (сложный PDU), щелкнув значок открытого конверта, расположенный над значком режима моделирования.

3. Щелкните External Client (внешний клиент) и сделайте его источником. Откроется окно Complex PDU (сложный PDU).

4. Щелкните имя сервера **ciscolearn.web.com** и сделайте его адресатом. Обратите внимание, что IP-адрес веб-сервера появился в поле адресата окна сложного PDU.

г. Укажите параметры **Complex PDU** (сложный PDU), изменив следующие настройки в окне Complex PDU (сложный PDU):

1. Раздел **PDU Settings** (параметры PDU) > **Select Application** (выбор приложения) должно быть установлено **HTTP.**

2. В разделе **Source Port** (порт источника) введите: **1000.**

3. В разделе **Simulation Settings** (настройки моделирования) выберите **Periodic Interval** (периодический интервал) и введите **120** секунд.

4. Создайте PDU, щелкнув поле **Create PDU** (создать PDU) в окне **Create Complex PDU** (создание сложного PDU).

д. Просмотрите поток трафика, щелкнув кнопку **Auto Capture / Play** (автозахват / воспроизведение) на панели «Simulation» (панель моделирования). Ускорьте анимацию с помощью ползунка управления воспроизведением.
*При появлении окна Buffer Full (буфер заполнен) закройте его, щелкнув значок **x.**

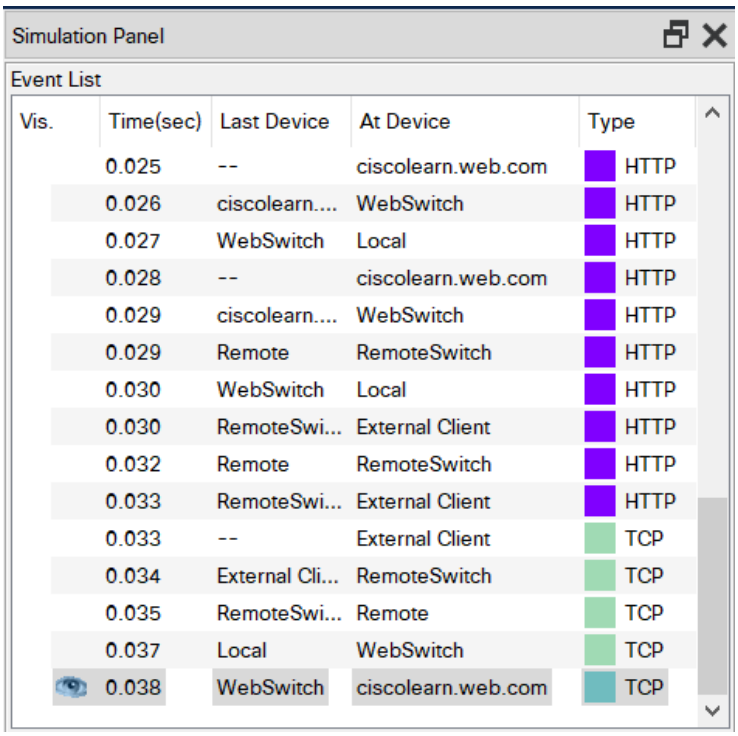


Рисунок 75 – Окно симуляции

e. Прокрутите список событий. Обратите внимание на количество пакетов, переданных от источника к адресату. HTTP — это протокол TCP, требующий установки соединения и подтверждения получения пакетов, что значительно повышает объем трафика.

Лабораторная работа №10
Работа с эмулятором сетей Cisco Packet Tracer.
Использование команды Ipconfig

Задача

С помощью команды **ipconfig** определите неверно настроенный ПК.

Исходные данные

Один из четырех ПК в офисе небольшой компании не подключается к Интернету. На всех ПК настроена статическая IP-адресация. С помощью команды **ipconfig /all** определите неверно настроенный ПК.

Шаг 1. Проверка конфигураций

а. На каждом ПК откройте **Command Prompt** (командная строка) и введите команду **ipconfig /all**.
б. Проверьте настройки IP-адреса, маски подсети и основного шлюза для каждого ПК.

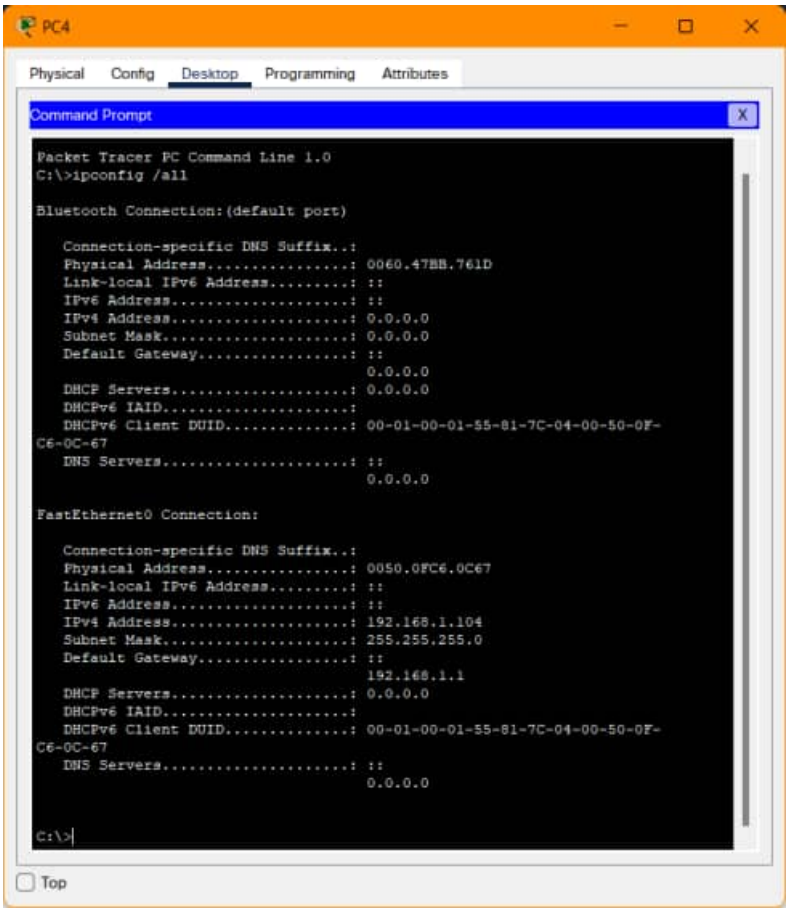


Рисунок 76 – Результат выполнения ipconfig /all на PC4

*Обязательно запишите эти IP-настройки для каждого ПК, чтобы выявить неправильно настроенный ПК.

Шаг 2. Исправление всех неправильно выполненных настроек

а. Выберите неправильно настроенный ПК и откройте вкладку **Config** (настройка).
б. Щелкните вкладку **Desktop** (рабочий стол) > **IP Configuration** (IP-конфигурация) и устраните ошибки.
в. Для проверки сделанной работы нажмите кнопку **Check Results** (проверить результаты) в нижней части окна инструкций.

Лабораторная работа №11
Работа с эмулятором сетей Cisco Packet Tracer.
Использование команды Ping

Задача

С помощью команды **ping** определите неверно настроенный ПК.

Исходные данные

Владелец небольшой компании узнал, что пользователь на ПК2 не может получить доступ к веб-сайту. На всех ПК настроена статическая IP-адресация. С помощью команды **ping** определите неполадку.

Шаг 1. Проверка подключения

а. На каждом ПК откройте вкладку **Desktop** (рабочий стол) > **Web Browser** (веб-обозреватель) и введите URL **ciscolearn.more.com**. Определите все ПК, которые не могут соединиться с веб-сервером.

б. Какой ПК не может подключиться к веб-серверу?

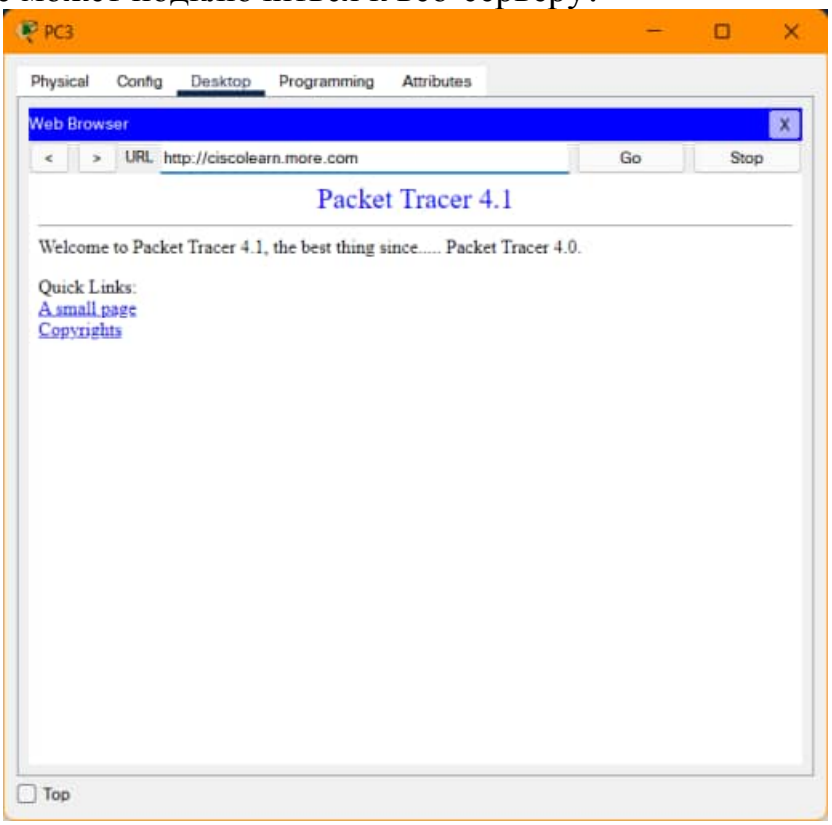


Рисунок 77 – Открывшаяся веб-страница

*Примечание. Всем устройствам требуется время для завершения процесса загрузки. Подождите ответа на запрос к веб-серверу в течение одной минуты.

Шаг 2. Отправка эхо-запроса на веб-сервер с ПК2

а. Откройте **Command Prompt** (командная строка) ПК2 из вкладки **Desktop** (рабочий стол).
б. Введите **ping ciscolearn.more.com**.

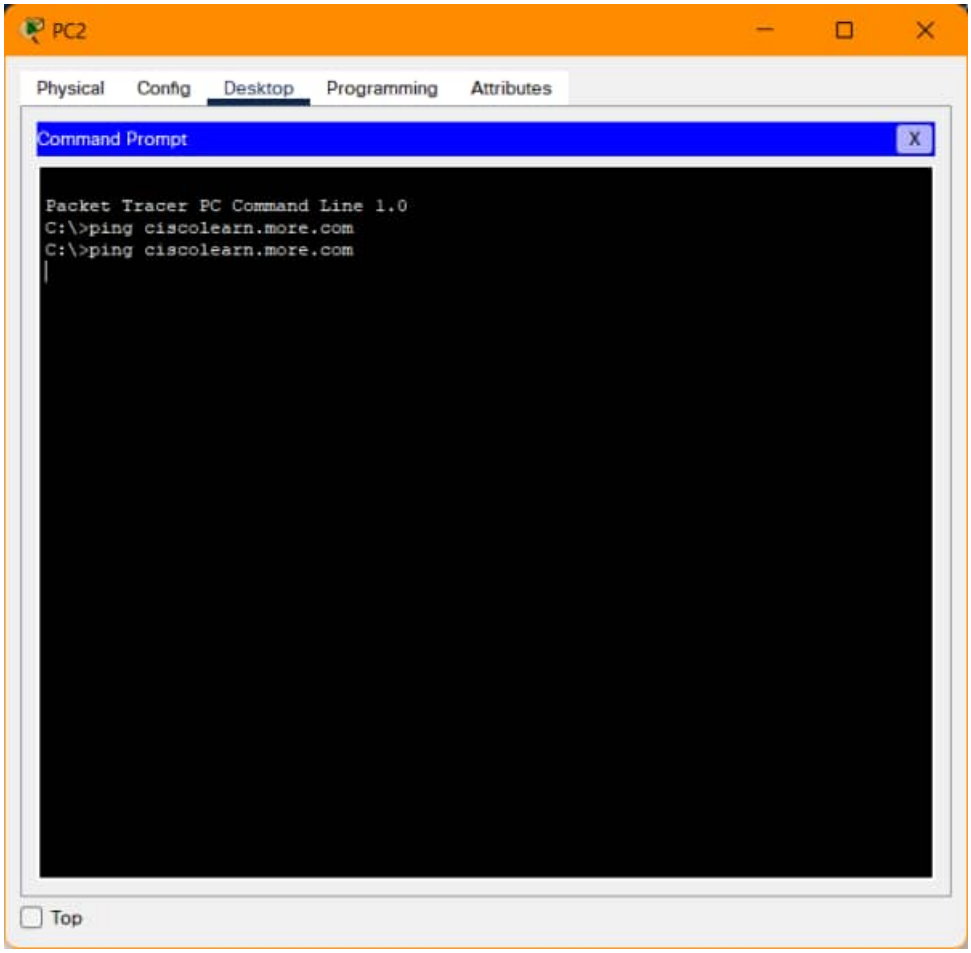


Рисунок 78 – Результат ввода команды ping на PC2

в. Ответ на **ping** получен? Если ответ получен, то какой IP-адрес в нем отображается?

Шаг 3. Отправка эхо-запроса на веб-сервер с ПК1

а. Откройте **Command Prompt** (командная строка) ПК1 из вкладки **Desktop** (рабочий стол).
б. Введите **ping ciscolearn.more.com**.

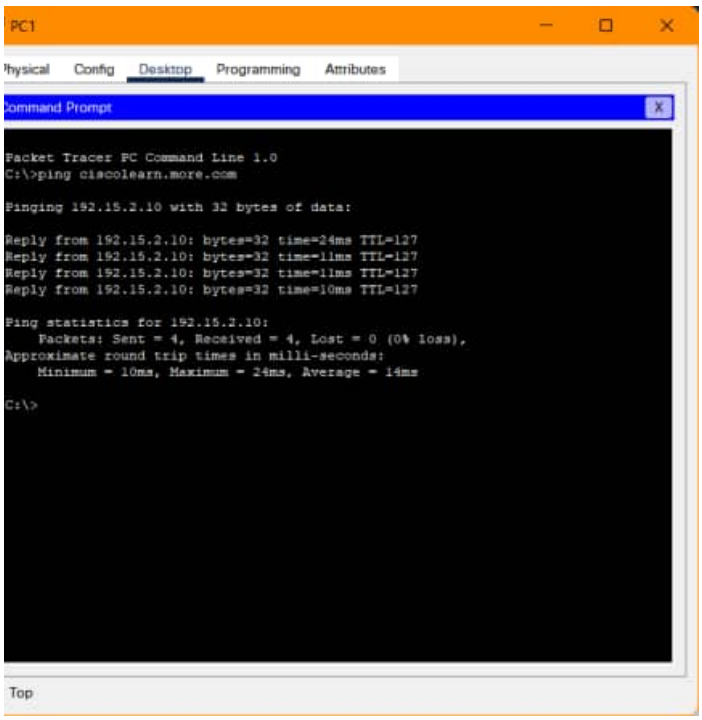


Рисунок 79 – Результат ввода команды ping на PC1

в. Ответ на **ping** получен? Какой IP-адрес возвращен (если возвращен)?

Шаг 4. Проверка IP-адреса веб-сервера с помощью эхо-запроса с ПК2

а. Откройте **Command Prompt** (командная строка) ПК2 из вкладки **Desktop** (рабочий стол).

б. Попытайтесь достигнуть IP-адреса веб-сервера с помощью команды **ping 192.15.2.10**.

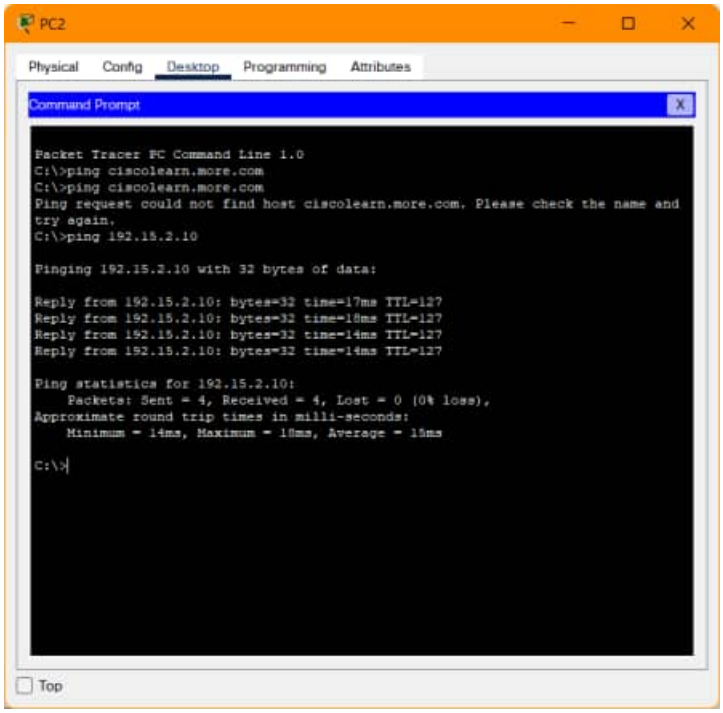


Рисунок 80 – Результат выполнения команды ping на PC2

в. Ответ на **ping** получен? Если да, то тогда ПК2 может соединиться с веб-

сервером по IP-адресу, но не может это сделать по имени домена. Возможно, проблема в настройке сервера DNS на ПК2.

Шаг 5. Сравнение настройки сервера DNS на ПК2 с настройками на других ПК в локальной сети.

а. Откройте **Command Prompt** (командная строка) ПК1.
б. Проверьте настройки сервера DNS на ПК1 с помощью команды **ipconfig /all**.

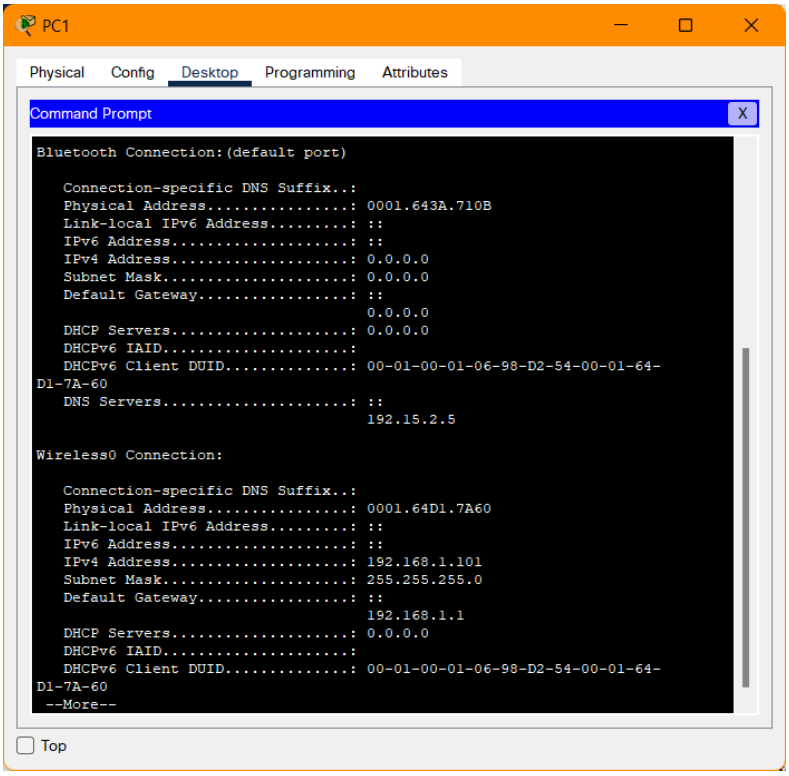


Рисунок 81 – Результат ввода команды ipconfig /all на PC1

в. Откройте **Command Prompt** (командная строка) ПК2.
г. С помощью команды **ipconfig /all**, проверьте настройки сервера DNS на ПК2.
Одинаковы ли обе проверенные настройки?

Шаг 6. Внесение необходимых изменений в конфигурацию ПК2

а. На ПК2 откройте вкладку **Config** (настройка) и внесите все необходимые изменения настроек.

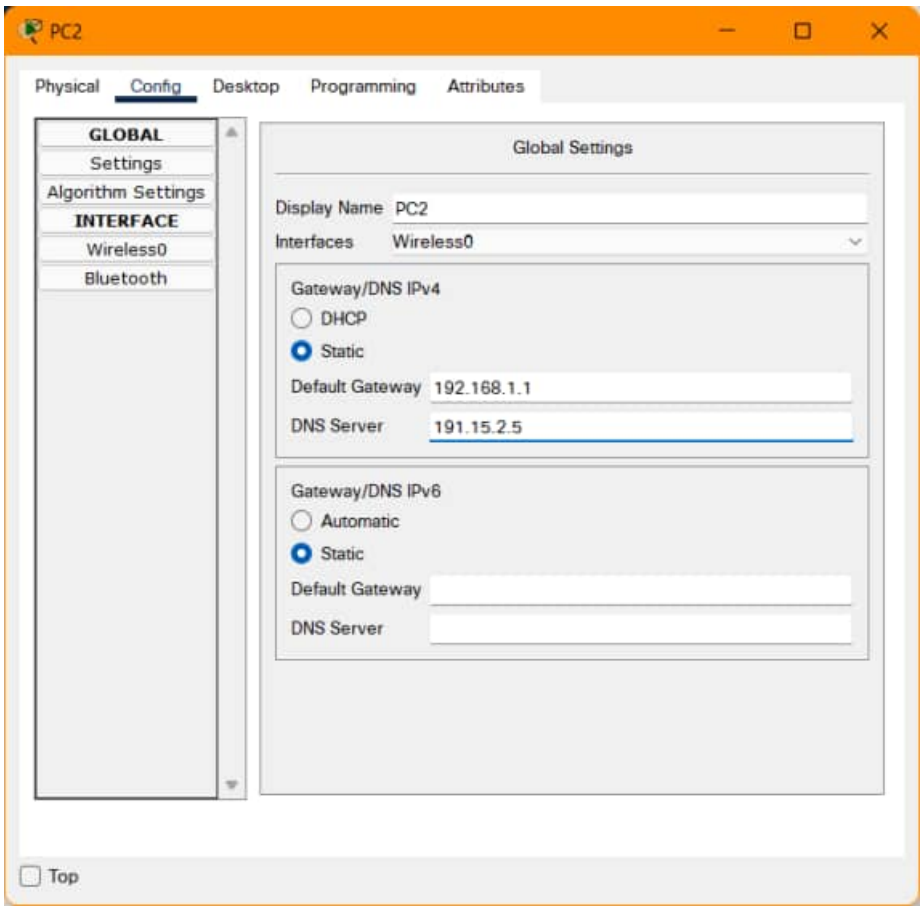


Рисунок 82 – Настройки IP во вкладке Config на PC2

б. С вкладки **Desktop** (рабочий стол) откройте окно **Web Browser** (веб-обозреватель), подключитесь к **ciscolearn.more.com** и проверьте, помогло ли изменение решить проблему.

в. Для проверки сделанной работы нажмите кнопку **Check Results** (проверить результаты) в нижней части окна инструкций.

Лабораторная работа №12
Работа с эмулятором сетей Cisco Packet Tracer.
Настройка DHCP многофункционального устройства

Задачи

- Подключить три ПК к многофункциональному устройству Linksys-WRT300N.
- Изменить настройку DHCP на определенный сетевой диапазон.
- Настроить клиенты на получение IP-адреса через DHCP

Исходные данные

Пользователь хочет соединить между собой 3 ПК с помощью устройства Linksys-WRT300N. Все три ПК должны получать IP-адреса автоматически от устройства Linksys.

Шаг 1. Настройка топологии сети

а. Добавьте в рабочую область три ПК.
б. Добавьте в рабочую область устройство Linksys-WRT300N.
в. Подключите каждый из ПК к портам Ethernet устройства Linksys прямым кабелем.

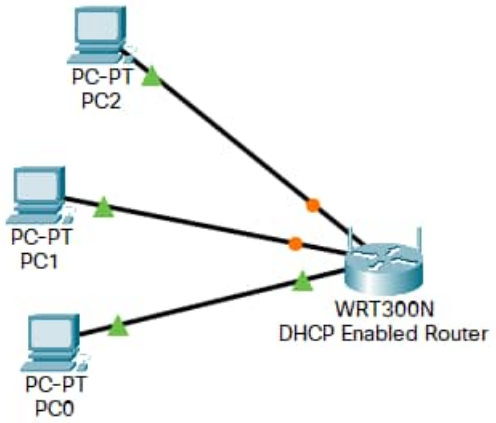


Рисунок 83 – Схема подключения компьютеров к маршрутизатору Linksys

Шаг 2. Проверка выбранных по умолчанию настроек DHCP

а. Щелкните маршрутизатор Linksys-WRT300N для вызова окна конфигурации.
Щелкните вкладку **Config** (настройка) и измените **Display**

Name (отображаемое имя) на **DHCP Enabled Router** (маршрутизатор с включенным DHCP).

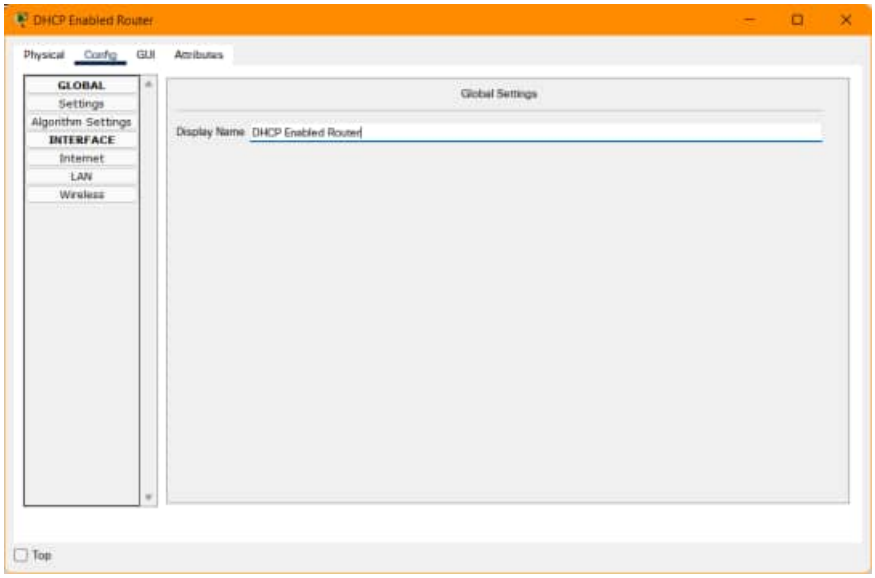


Рисунок 84 – Изменение названия маршрутизатора во вкладке Config

*Примечание. Откроется всплывающее окно, предупреждающее, что изменение отображаемого имени может негативно сказаться на оценке. Продолжите процесс изменения отображаемого имени, учитывая, что имя должно точно соответствовать заданию упражнения.

в. Выберите вкладку **GUI** (графический интерфейс).

*Откроется страница **Setup / Basic Setup** (настройка / базовая настройка) графического интерфейса Linksys.

г. Прокрутите страницу основных настроек и просмотрите настройки по умолчанию, включая IP-адрес устройства Linksys.

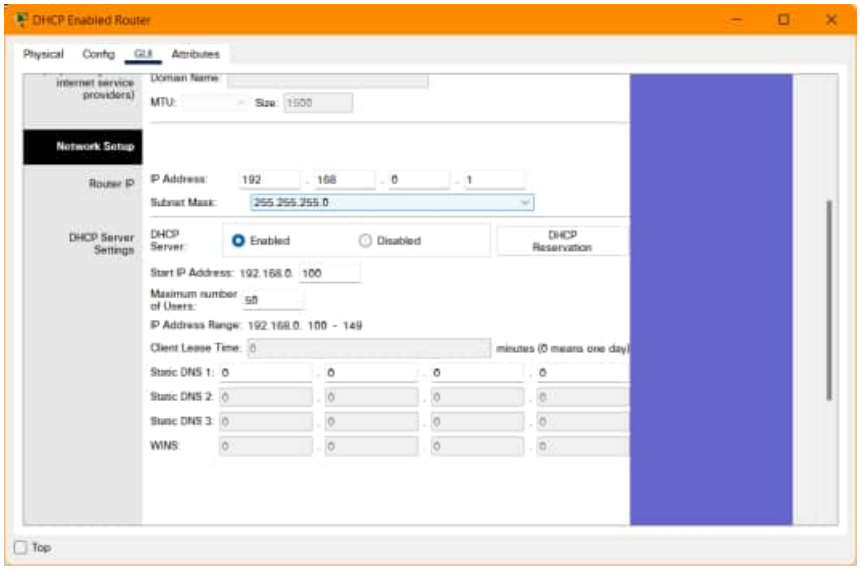


Рисунок 85 – Настройка IP адреса локальной сети в маршрутизаторе Linksys

*Обратите внимание, что функция DHCP включена, клиентам доступен

начальный адрес диапазона DHCP и диапазон адресов.

Шаг 3. Изменение IP-адреса устройства Linksys по умолчанию

а. Откройте раздел **Router IP** (IP маршрутизатора) и измените IP-адрес устройства Linksys на **192.168.5.1**.

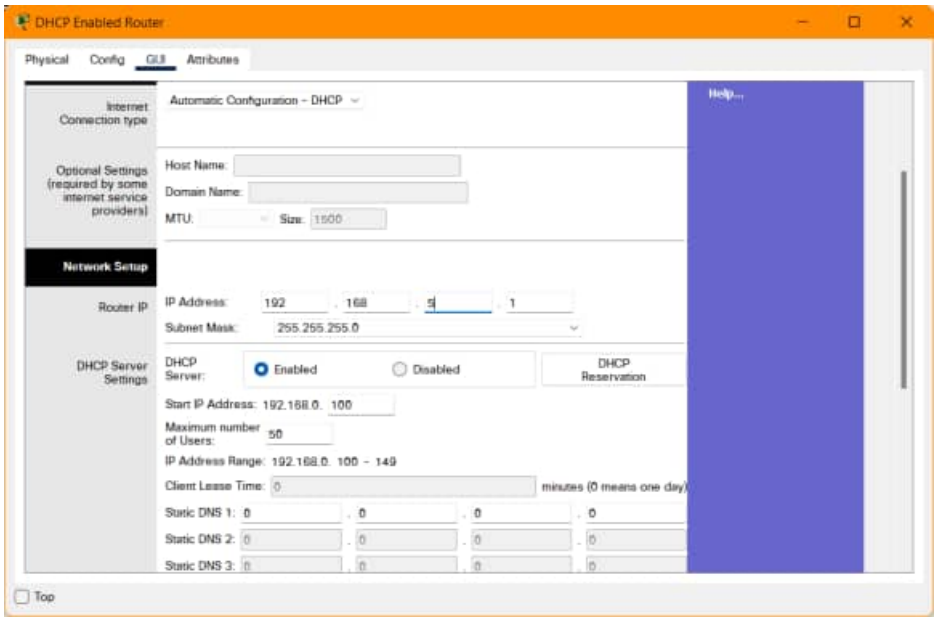


Рисунок 86 – Изменение IP адреса локальной сети в маршрутизаторе Linksys

б. Прокрутите страницу графического интерфейса до конца и нажмите кнопку **Save Settings** (сохранить параметры).
в. Вернитесь в раздел IP маршрутизатора и убедитесь, что изменение внесено.

Шаг 4. Изменение диапазона адресов DHCP по умолчанию

а. Обратите внимание, что начальный IP-адрес в настройках сервера DHCP изменился и находится в той же сети, что и IP-адрес устройства Linksys: **192.168.5.100**.
б. Измените **Starting IP Address** (начальный IP-адрес) со значения **192.168.5.100** на значение **192.168.5.26**.
в. Измените значение **Maximum number of Users** (максимальное количество пользователей) на: **75**

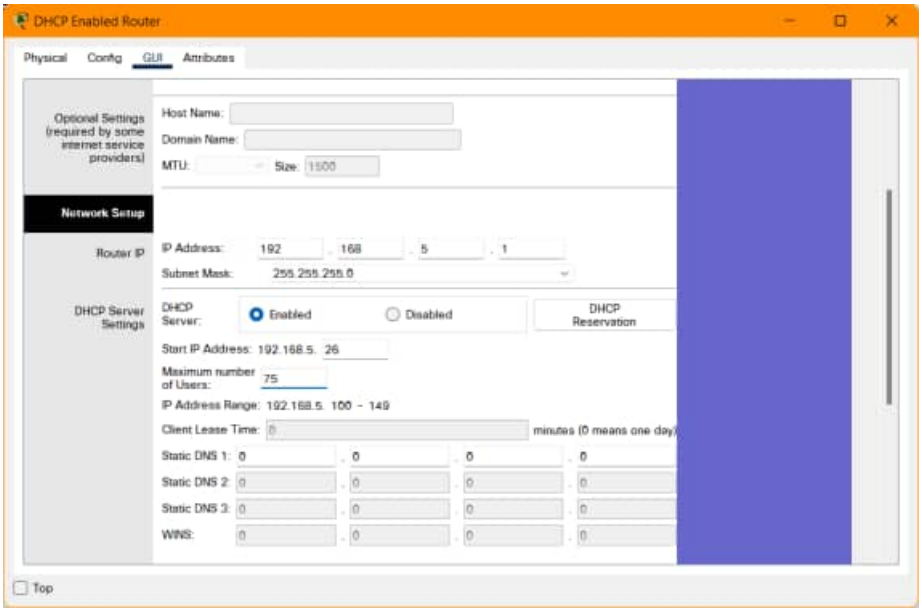


Рисунок 87 – Настройка максимального количества пользователей в локальной сети на маршрутизаторе Linksys

г. Прокрутите страницу графического интерфейса до конца и нажмите кнопку **Save Settings** (сохранить параметры).

д. Вернитесь в раздел "Настройка сервера DHCP" и убедитесь, что изменение внесено.

*Убедитесь, что диапазон доступных клиентам адресов изменился соответствующим образом.

е. Закройте окно настройки Linksys.

Шаг 5. Настройка DHCP на рабочих станциях клиента

а. Включите DHCP на рабочей станции **PC0**.

1. Щелкните рабочую станцию PC0.
2. Щелкните вкладку **Config** (настройка). Откройте подменю интерфейса **FastEthernet**.
3. Включите DHCP, нажав кнопку **DHCP** на панели IP Configuration (настройка IP).

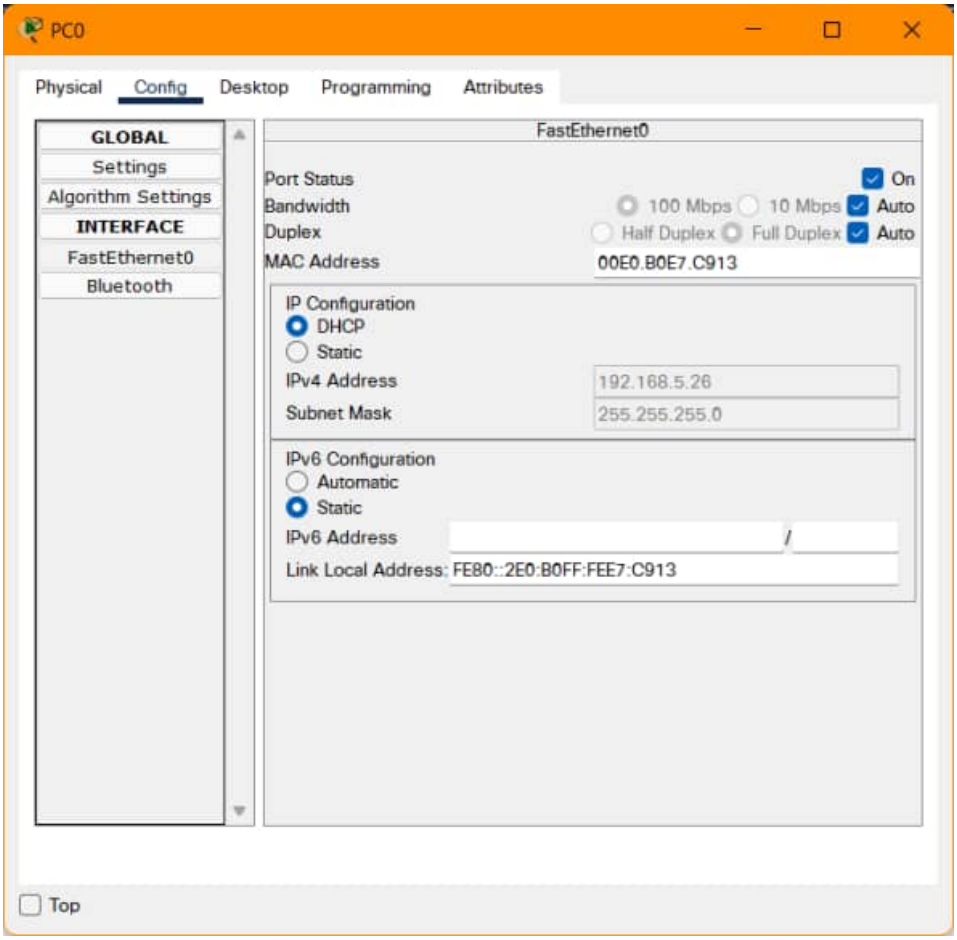


Рисунок 88 – Настройка IP компьютера PC0

*Обратите внимание, что IP-адрес и маска подсети присваиваются автоматически.

4. Закройте окно настройки.

б.Проверьте настройки IP клиента, где не включен DHCP.

1. Щелкните рабочую станцию PC1.
2. Щелкните вкладку **Desktop** (рабочий стол) > **Command Prompt** (командная строка).
3. Введите **ipconfig** и нажмите клавишу *ВВОД*.

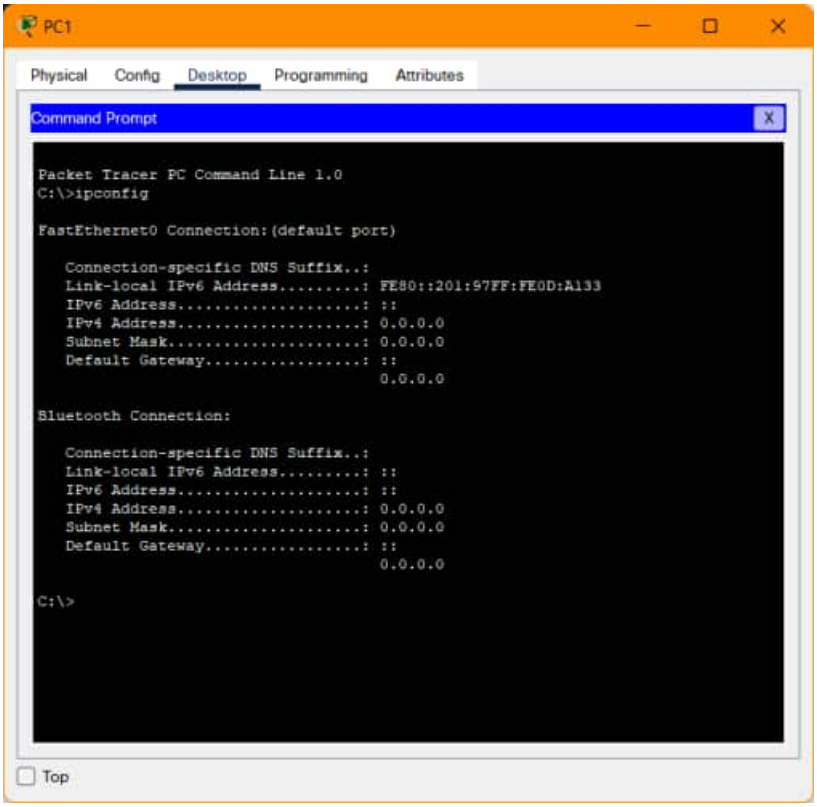


Рисунок 89 – Результат ввода команды ipconfig на PC1

*Обратите внимание, что значение всех настроек - 0.0.0.0. Статически IP-адрес не присвоен, автоматически компьютер его от DHCP не получил.

в. Включите DHCP на PC1 и PC2 с помощью вкладки **Config** (настройка), как указано в шаге 5а.
*Обратите внимание, что PC1 и PC2 автоматически получили разные IP-адреса, отличающиеся от адреса PC0, и одинаковую маску подсети.

г. Закройте окно настройки.

Шаг 6. Проверка подключения

а. Щелкните рабочую станцию PC1 и выберите вкладку **Desktop** (рабочий стол) > **Command Prompt** (командная строка).

б. Введите **ipconfig** для просмотра IP-конфигурации PC1.

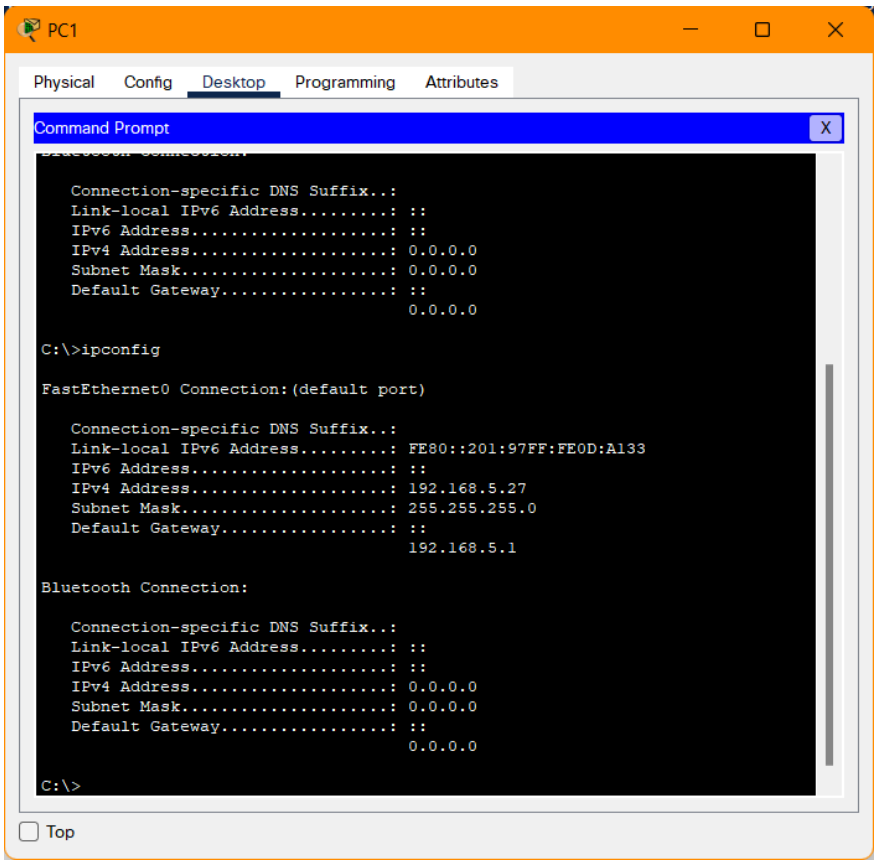


Рисунок 90 – Результат ввода ipconfig на PC1

в. Введите **ping 192.168.5.1**, чтобы отправить эхо-запрос устройству Linksys

.

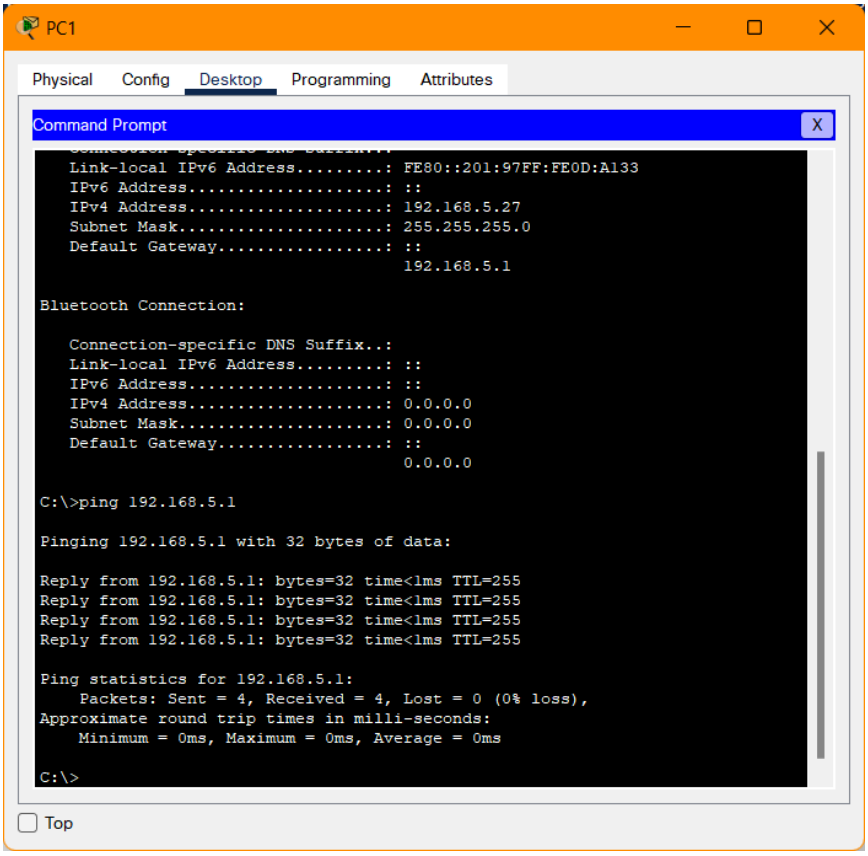


Рисунок 91 – Результат ввода команды ping на PC1

г. Введите **ping 192.168.5.26** для отправки эхо-запроса рабочей станции PC0.

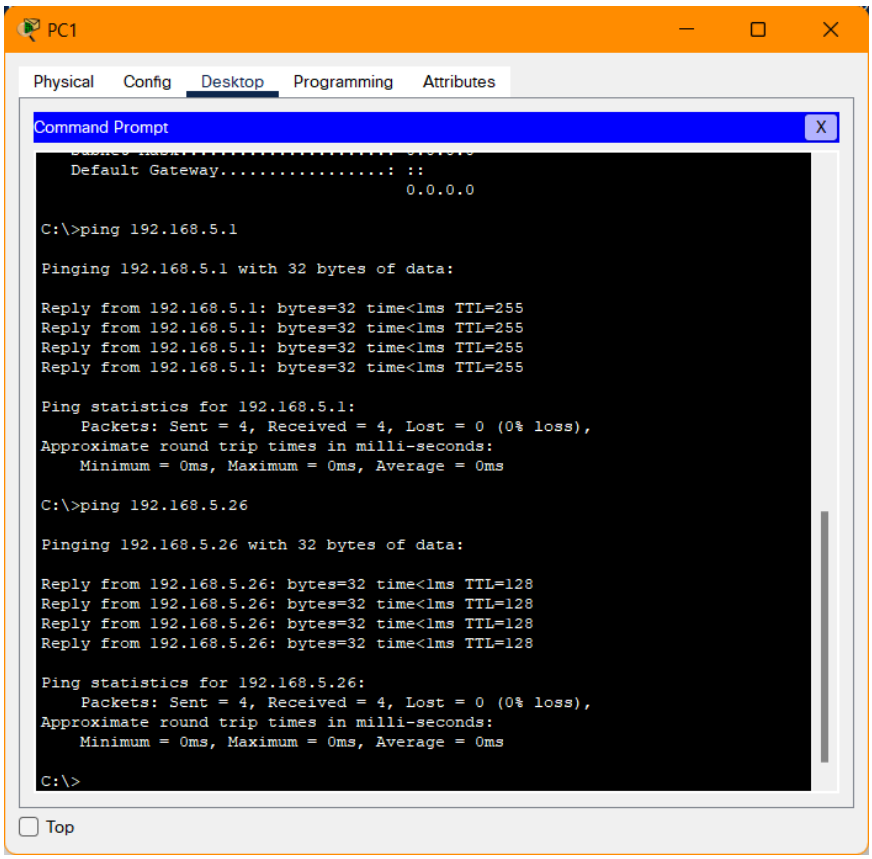


Рисунок 92 – Результат ввода команды ping на PC1

*Оба устройства должны дать ответ.

д. Закройте окно настройки и нажмите кнопку **Check Results** (проверить результаты) в нижней части окна для проверки работы.

е. Выберите вкладку **Assessment Items** (оцениваемые пункты) для просмотра ошибок конфигурации.

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ

по дисциплине

ЭКОНОМИКА ОТРАСЛИ

специальности СПО

09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЕНЫ
на заседании цикловой комиссии информационных технологий

Протокол от «13» января 2023 г. № 6

Председатель цикловой комиссии И.В. Миляева

Автор: Амеличкина С.Г., преподаватель колледжа

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В результате выполнения практических работ по учебной дисциплине Экономика отрасли специальности 09.02.07 Информационные системы и программирование обучающийся должен:

иметь практический опыт:

-оценки эффективности использования материально-технических, трудовых и финансовых ресурсов организации

уметь:

-находить и использовать необходимую экономическую информацию;

-рассчитывать по принятой методологии основные технико-экономические показатели деятельности организации.

знать:

-общие положения экономической теории.

-организацию производственного и технологического процессов.

-механизмы ценообразования на продукцию (услуги), формы оплаты труда в современных условиях.

-материально-технические, трудовые и финансовые ресурсы отрасли и организации, показатели их эффективного использования.

-методику разработки бизнес-плана.

Выполнение практических работ влияет на формирование общих и профессиональных компетенций.

Код	**Наименование результата обучения**
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности
ОК 04	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.
ОК 09	Использовать информационные технологии в профессиональной деятельности
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языках.
ОК 11	Использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере.
ПК 11.1	Осуществлять сбор, обработку и анализ информации для проектирования баз данных.

Практическая работа №1
Определение состава и структуры основного капитала предприятия, отрасли

Цель:
уметь:
-рассчитывать стоимостную оценку основного капитала
-определять состав и структуру основного капитала

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Проведение стоимостной оценки основных фондов предполагает определение первоначальной, восстановительной и остаточной стоимости и среднегодовой стоимости основных производственных фондов.

Задача 1

Стоимость приобретения оборудования составляет 900 тыс. руб., транспортные и монтажные затраты – 100 тыс. руб. Работы по пуску и наладке нового оборудования предприятию обойдутся в 50 тыс. руб. Определить первоначальную стоимость основных производственных фондов предприятия.

Решение

Первоначальная стоимость основных фондов $Ф_п$ включает в себя стоимость их приобретения $Ц_о$ с учетом затрат, связанных с вводом нового объекта основных фондов $З_{вв}$. в состав этих затрат входят транспортные, монтажные и, если имеют место, пуско-наладочные затраты:

$$Ф_п = (Ц_о + З_{вв}).$$

В нашем случае первоначальная стоимость основных производственных фондов будет равна

$$Ф_п = (900 + 100 + 50) = 1050 \text{ тыс. руб.}$$

Ответ: первоначальная стоимость основных производственных фондов равна 1050 тыс. руб.

Задача 2

Первоначальная стоимость оборудования для предприятия составляет 100 тыс. руб. период эксплуатации оборудования – 8 лет. среднегодовые темпы роста производительности труда в отрасли составляют 3 %. Определить восстановительную стоимость основных производственных фондов.

Решение

Восстановительная стоимость основных фондов $Ф_{восст}$ рассчитывается с учетом их переоценки:

$$\Phi_{восст} = \frac{\Phi_n}{(1+П_{отр})^t},$$

где $П_{отр}$ – среднегодовые темпы роста производительности труда в отрасли;
t – время между годами выпуска и переоценки (например, год выпуска 2015, год переоценки – 2020, значит t= 5).
Восстановительная стоимость основных фондов с учетом их переоценки в нашей задаче равна:

$$\Phi_{восст} = \frac{100}{(1+0{,}03)^8} = 78940\,руб.$$

Ответ: восстановительная стоимость основных производственных фондов равна 78940 руб.

Задача 3
Первоначальная стоимость основных производственных фондов предприятия составляет 100 тыс. руб. период эксплуатации оборудования – 8 лет. Определить остаточную стоимость основных производственных фондов, если норма амортизационных отчислений для данного оборудования составляет 10 %.
Решение
Первоначальная стоимость, уменьшенная на величину перенесенной стоимости, представляет собой остаточную стоимость основных производственных фондов $Ф_{ост}$. Поэтому для решения данной задачи используем следующую формулу:

$$\Phi_{ост} = \Phi_n(1 - Н_А \cdot t_{экспл}),$$

где $Н_А$ – норма амортизационных отчислений;
$t_{экспл}$ – период эксплуатации основных фондов.
Подставив известные из условия задачи данные, получаем:

$$\Phi_{ост} = 100(1 - 0{,}1 \cdot 8) = 20\,тыс.руб.$$

Ответ: остаточная стоимость основных производственных фондов составляет 20 тыс. руб.

Среднегодовая стоимость основных фондов определяется по формуле:

Фср. г. =Фн. г+(Фвв. * n:12) - (Фвыб*m:12)

где Фн. г. - стоимость основных фондов на начало года, руб. ;
Фвв. - стоимость вводимых основных фондов, руб. ;
Фвыб. - стоимость выбывающих основных фондов, руб. ;
n - количество месяцев функционирования вводимых основных фондов (до конца года)
m - количество месяцев нефункционирования выводимых основных фондов (до конца года) (m =12- n).

Задача 4
На предприятии следующий состав основных фондов по группам на начало года (млн. руб.): здания – 165, сооружения – 51, передаточные устройства-

11,2, силовые машины 8,8, рабочие машины – 83,4, транспортные машины – 22,7,вычислительная техника-5,1,измерительные приборы -5,6,прочие фонды -11,2. 1марта введено основных фондов на 17,5млн.руб,а 1 августа выбыло на 3,8 млн.руб. Объем товарной продукции за год составил 312,9млн.руб.
Определить структуру основных фондов на данном предприятии (активную и пассивную часть), среднегодовую стоимость основных фондов.

Задача 5
Основные производственные фонды предприятия на начало 2020 года составляли 3000 тыс. руб. В течение года было введено основных фондов на сумму 125 тыс. руб., а ликвидировано – на сумму 25 тыс. руб. рассчитать стоимость основных фондов на конец года.
Решение
Стоимость основных производственных фондов на конец года есть стоимость основных фондов на начало года с учетом изменений, произошедших в их структуре за этот год:

$$Ф_к = Ф_н + (Ф_{вв} - Ф_{выб}),$$

где $Ф_к$ – стоимость основных фондов на конец года, руб.;
$Ф_{вв}$ – стоимость введенных основных фондов, руб.;
$Ф_к$ – стоимость основных фондов на конец года, руб.
Подставив известные из условия задачи значения, рассчитываем стоимость основных фондов на конец года

$$Ф_к = 3000 + (125 – 25) = 3100 \text{ тыс. руб.}$$

Ответ: стоимость основных фондов на конец года составляет 3100 тыс. руб.

В основе характеристики состава и структуры основных фондов предприятия лежит расчет коэффициентов обновления, выбытия и прироста основных фондов.

Задача 6
На предприятии в течение года было введено основных производственных фондов на сумму 150 тыс. руб. так что стоимость основных фондов на конец года составила 3000 тыс. руб. Рассчитать коэффициент обновления основных фондов.
Решение
Коэффициент обновления – один из показателей, которые используются для проведения анализа изменения структуры основных производственных фондов.
Зная стоимость основных фондов предприятия на конец года, а также сколько было введено основных фондов, коэффициент обновления основных фондов можно рассчитать по формуле:

$$К_{обн} = \frac{Ф_{вв}}{Ф_к},$$

где $Ф_{вв}$ – стоимость введенных основных фондов, руб.;
$Ф_к$ – стоимость основных фондов на конец года, руб.

Коэффициент обновления основных производственных фондов составит:

$$K_{обн} = \frac{150}{3000} = 0{,}05.$$

Таким образом, за год на нашем предприятии произошло пятипроцентное обновление основных производственных фондов.
Ответ: коэффициент обновления основных фондов равен 0,05.

Задача 7

Основные производственные фонды предприятия на начало 2020 года составляли 3000 тыс. руб. В течение года было ликвидировано основных фондов на сумму 300 тыс. руб. Рассчитать коэффициент выбытия основных фондов.
Решение
Коэффициент выбытия основных фондов рассчитывают по формуле:

$$K_{выб} = \frac{Ф_{выб}}{Ф_{н}},$$

где $Ф_{выб}$ – стоимость выбывающих (ликвидируемых) основных фондов, руб.;
$Ф_{н}$ – стоимость основных фондов на начало года, руб.
Рассчитаем коэффициент выбытия основных производственных фондов:

$$K_{выб} = \frac{300}{3000} = 0{,}1.$$

Таким образом, на предприятии было ликвидировано 10% основных производственных фондов.
Ответ: коэффициент выбытия основных фондов равен 0,1.

Задача 8

На предприятии в течение года прирост основных производственных фондов составил 80 тыс. руб. стоимость основных фондов на конец года – 4000 тыс. руб. Рассчитать коэффициент прироста основных фондов.
Решение
Коэффициент прироста – еще один показатель, который наряду с коэффициентами обновления и выбытия используется для проведения анализа изменения структуры основных производственных фондов.
Коэффициент прироста основных фондов рассчитывается как отношение:

$$K_{прир} = \frac{Ф_{прир}}{Ф_{к}},$$

где $Ф_{прир}$ – прирост основных фондов в денежном выражении, руб.;
$Ф_{к}$ – стоимость основных фондов на конец года, руб.
Соответственно, коэффициент прироста основных фондов:

$$Ф_{прир} = \frac{80}{4000} = 0{,}02.$$

Ответ: прирост основных фондов составил 2 %.

Практическая работа №2
Расчет амортизации основного капитала.
Определение показателей эффективности использования основного капитала

Цель:
уметь:
-рассчитывать по принятой методике показатели использования основных фондов,
-рассчитывать величину амортизационных отчислений.

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Для оценки эффективности использования основных средств на предприятиях используется система показателей, включающая общие и частные показатели. Общие характеризуют эффективность использования всей совокупности основных средств. При расчете этих показателей используется стоимостная оценка основных средств. Важнейшими показателями этой группы являются:

Фондоотдача (Фотд) – показатель выпуска продукции на 1 руб. стоимости основных фондов; определяется как отношение объема выпуска продукции (V) к стоимости основных производственных фондов (Ф) за сопоставимый период времени (месяц, год).

Фондоемкость (Фе) – величина, обратная фондоотдаче; показывает долю стоимости основных фондов, приходящуюся на 1 руб. выпускаемой продукции.

Фондовооруженность труда (Фв) рассчитывается как отношение стоимости основных фондов (Ф) к числу рабочих на предприятии, работавших в наибольшую смену (Ч).

1. Фондоотдача – выпуск продукции на 1 руб. стоимости основных производственных фондов. Она определяется:

$$Ф_о = ВП/Ф$$

где: ВП – объем выпущенной продукции;
Ф - среднегодовая стоимость ОПФ.
Среднегодовая стоимость ОПФ определяется:

$$Ф = Ф_{н.} +(Ф_в \bullet t_в)/12- (Ф_л \bullet t_л)/12$$

где: $Ф_{н.}$ – стоимость ОПФ на начало года в руб.;

$Ф_{в.}$ и $Ф_{л.}$ – стоимость соответственно вновь вводимых и ликвидируемых основных фондов.

$t_{в.}$ – число полных месяцев эксплуатации вновь введенных ОПФ

$t_{л.}$ – число месяцев, оставшихся со времени выбытия основных фондов до конца года.

2. Фондоемкость – показатель обратный фондоотдаче.

$$Ф_{е.} = Ф/ВП, \text{ или } Ф_{е.}=1/ Ф_{о}$$

3. Фондовооруженность, степень вооруженности рабочих ОПФ:

$$Ф_{в.}= Ф./ Ч_{ср.}$$

где $Ч_{ср.}$ – среднесписочная численность работающих.

4. Баланс основных производственных фондов определяется:

$$Ф_{к.}= Ф_{н.} + Ф_{в}- Ф_{л}$$

5. Коэффициент выбытия ОПФ:

$$К_{л.}= Ф_{л} /Ф_{н}$$

6. Коэффициент ввода ОПФ

$$К_{в.}= Ф_{в} /Ф_{к.}$$

Задача 1.

На основе данных для выполнения задачи определите коэффициенты выбытия, обновления ОПФ.

На предприятии имеются в наличии основные фонды (тыс. руб.): основные фонды на начало года – 17430, поступило в отчетном году – 1360, в т.ч. введено в действие – 1130, выбыло в отчетном году – 670, износ основных фондов на начало года – 1620, износ основных фондов на конец года – 1440.

ОПФ кг = ОПФ нг + ОПФ вв - ОПФ выб

Коэффициент выбытия: ОПФ выб / ОПФ нг

Коэффициент обновления: ОПФ вв / ОПФ кг

Задача 2. На основе данных для выполнения задачи определите фондоотдачу и фондоемкость и фондовооруженность ОПФ

Годовая выработка тепловой энергии в денежном выражении составляет 6828 тыс. руб., среднегодовая стоимость основных средств котельной 4425 тыс. руб., численность работников котельной 142 человека.

Задача 3

На начало года стоимость основных производственных фондов составляла 359 млн.руб. 1 февраля предприятие приобрело станков на сумму 52 млн.руб., а 1 июня было ликвидировано основных фондов на 25 млн.руб. В среднем норма амортизации 18%. За год предприятие выпустило продукции на сумму 826 млн.руб. Численность производственных рабочих 1825 человек.

Определить:

1.Среднегодовую стоимость основных производственных фондов;

2.Фондоотдачу, фондоемкость, фондовооруженность.

Амортизация — это процесс перенесения стоимости изношенной части основных производственных фондов на создаваемую продукцию или выполняемые работы, например погрузочно-разгрузочные. Изношенные основные фонды необходимо заменять. Возмещение изношенных основных фондов осуществляется за счет амортизационных отчислений. Месячная сумма амортизации рассчитывается как произведение первоначальной стоимости объекта и нормы амортизации:

$$A = Ф_п \Leftrightarrow Н_а/100\%,$$

где А — месячная (годовая) сумма амортизации (руб.);

$Ф_п$— первоначальная балансовая стоимость основных средств (руб.);

$Н_а$ — норма амортизации (%).

Норма амортизации для каждого объекта определяется по формуле:

$$Н_а = \frac{1 \times 100\ \%}{N}$$

где N — срок полезного использования (месяцев).

Задача 1. На начало года стоимость ОПФ составляла 30млн.руб. В марте предприятие приобрело станки на сумму 6 млн.руб., а в июне было ликвидировано на 4 млн.руб. в среднем норма амортизации 12%. За год предприятие выпустило продукции на сумму 26 млн.руб. Определите: среднегодовую стоимость ОПФ; сумму амортизационных отчислений за год; фондоотдачу.

Задача 2. Первоначальная стоимость станка – 30000.руб., Нормативный срок службы – 12 лет. Выручка от реализации отдельных деталей и узлов станка, стоимость лома после износа – 2500 руб. Определите норму амортизационных отчислений.

Задача 3

Определить остаточную стоимость шлифовального станка на 1.01.2020 г., купленного в декабре 2008 г. за 400000 руб. Затраты на доставку составляют 5% от цены, на монтаж и установку 3%, норма амортизации 5% в год.

Практическая работа №3
Определение показателей эффективности использования оборотного капитала

Цель:

-уметь: рассчитывать показатели использования оборотных средств и оценить их эффективность.

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Оборотные средства – это совокупность оборотных производственных фондов и фондов обращения в денежном выражении. Эти составные части оборотных средств по-разному обслуживают процесс воспроизводства: первые – в сфере производства, а вторые – в сфере обращения.

Важнейшими показателями использования оборотных средств предприятия являются коэффициент оборачиваемости оборотных средств и длительность одного оборота.

Эффективное использование оборотных средств характеризуют три показателя: длительность одного оборота, коэффициент оборачиваемости, оборачиваемости в днях и коэффициент загрузки.

Продолжительность одного оборота в днях (О) показывает, за какой срок к предприятию возвращаются его оборотные средства в виде выручки от реализации продукции.

$$О = \frac{С_о \times Д_{п.}}{В_{т.п.}}$$

где: $С_о$ - остатки оборотных средств среднегодовые или на конец периода;

$Д_{п.}$ – число дней в отчетном периоде;

$В_{т.п.}$ – объем товарной продукции в руб.

Коэффициент оборачиваемости показывает число оборотов, совершаемых оборотными средствами за определенный период.

$$К_о = \frac{В_{т.п.}}{С_о}$$

где: $В_{т.п.}$ – объем товарной продукции в руб.;

$С_о$ - остатки оборотных средств среднегодовые или на конец периода.

Коэффициент загрузки оборотных средств показатель обратный коэффициенту оборачиваемости. Он характеризует величину оборотных средств, приходящихся на единицу реализованной продукции.

$$К_з = \frac{С_о}{В_{т.п.}}$$

Задача 1

Определить коэффициент оборачиваемости оборотных средств и длительность одного оборота, если известно, что выпуск продукции за год составил 10000 шт.; себестоимость изделия – 80руб., цена изделия на 25% превышает его себестоимость; среднегодовой остаток оборотных средств – 50000руб.

Задача 2

Определите дополнительный объем продукции в планируемом году при тех же оборотных средствах, если число оборотов увеличится на 1. Исходные данные: выпуск продукции в базисном году – 30000руб., средний размер оборотных средств в базисном году – 10000руб.

Задача 3

Определите сокращение потребности в оборотных средствах за год, если продолжительность одного оборота за счет внедрения мероприятий НТП сократилась на 2 дня, плановый объем реализации 78 млн. руб. в год, плановый норматив оборотных средств 13млн.руб.

Задача 4. На основе данных для выполнения задачи найдите сумму оборотных средств, которая необходима предприятию при условии, что объем реализованной продукции останется прежним

Исходные данные. Средние остатки оборотных средств в 2019 г. составляли 15 885 тыс. руб., объем реализованной продукции за 2019 г. – 68 956 тыс. руб. В 2020 г. длительность оборота планируется сократить на 2 дня.

Задача 5. Средняя стоимость оборотных средств 1266 руб. Выручка от реализации продукции 2359 руб. Число календарных дней равно 90. Определите время оборота в днях.

Задача 6. Определите норматив оборотных средств в незавершенном производстве, если известно, что выпуск продукции за год составит12 тыс. ед.; себестоимость изделия - 1,5 тыс. руб.; длительность производственного цикла изготовления изделий - 5 дней; коэффициент нарастания затрат в незавершенном производстве - 0,4.

Задача 7. Предприятие имело средний остаток оборотных средств в сумме 60 млн. руб., при планируемом объеме выпуска продукции 720 млн. руб. Наметили сократить длительность оборота на 6 дней. Определить сумму высвобожденных средств и ускорение оборачиваемости оборотных средств.

Задача 8. В отчетном году при среднегодовом нормативе оборотных средств 5 млн. руб. было выпущено продукции на 15 млн. руб. Как должен измениться норматив оборотных средств, если на планируемый год предусматривается увеличение программы выпуска продукции на 10%, а коэффициент оборачиваемости оборотных средств возрастет на 15%?

Практическая работа №4
Планирование численности рабочих

Цель:

- определение трудовых ресурсов отрасли и организации, показатели их использования
-уметь рассчитывать количественные характеристики персонала, виды численности

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Для характеристики трудового потенциала предприятия используется система показателей.

Количественная характеристика персонала предприятия измеряется такими показателями как, списочная, явочная и среднесписочная численность работников.

Качественная характеристика трудовых ресурсов (персонала) определяется степенью профессиональной и квалификации пригодности его работников для выполнения целей предприятия и производимых ими работ. Для качественной характеристики кадров используют следующие параметры:
- характер участия в производственном процессе;
- особенности трудовой деятельности;
- отраслевая принадлежность;
- личностные характеристики и т.д.

Качественная характеристика персонала представлена следующими **группами показателей:**

- Экономические (квалификация работников, отраслевая принадлежность, трудовой стаж).
- Личностные (дисциплинированность, наличие навыков, творческая активность и т.д.).
- Организационно-технические (техническое оснащение и рациональная организация труда).
- Социально-культурные (коллективизм, социальная активность).

Структурная характеристика трудовых ресурсов (персонала) предприятия определяется составом и количественным соотношением отдельных категорий и групп работников предприятия (удельного веса категорий работников по классификации: руководителей, специалистов, служащих, рабочих (основных, вспомогательных, МОП)

К категории МОП (младший обслуживающий персонал) относятся работающие, занятые выполнением функций по уходу за

служебными и производственными помещениями, обслуживанию АУЛ, рабочих, ИТР и служащих. МОП может включаться в категорию рабочих. К рабочим относятся лица, непосредственно занятые в процессе создания материальных ценностей, а также занятые ремонтом, перемещением грузов, перевозкой пассажиров, оказанием материальных услуг и др.

К рабочим, в частности, относятся лица, занятые:

1. управлением, регулированием и наблюдением за работой автоматов, автоматических линий, автоматических приспособлений, а также непосредственно управлением или обслуживанием машин, механизмов, агрегатов и установок;
2. изготовлением материальных ценностей вручную, а также при помощи простейших механизмов, приспособлений, инструментов;
3. строительством и ремонтом зданий, сооружений, монтажом и ремонтом оборудования, ремонтом транспортных средств;
4. перемещением, погрузкой или выгрузкой сырья, материалов, готовой продукции;
5. на работах по приему, хранению и отправке грузов на складах, базах, в кладовых и других хранилищах;
6. уходом за машинами, оборудованием, обслуживанием производственных и непроизводственных помещений;
7. проходкой наземных и подземных горных выработок, бурением, испытанием, опробованием и освоением скважин, поисковыми и другими видами геолого-разведочных работ
8. машинисты, водители, кочегары, дежурные стрелочных постов, обходчики путей и искусственных сооружений, грузчики, проводники, рабочие по ремонту и уходу за транспортными линиями, линиями связи, по ремонту и уходу за оборудованием и средствами передвижения, трактористы, механики, рабочие растениеводства и животноводства;
9. радиооператоры, операторы связи, операторы вычислительных и электронно-вычислительных машин;
10. дворники, уборщики, курьеры, гардеробщики, сторожа (МОП)

Формула расчета удельного веса имеет вид:

$$\text{Удельный вес части (доля части)} = \frac{Часть\_целого(группа\_персонала)}{Целое(общая\_численность\_персонала)} \times 100\%$$

Задача 1

Определить структуру персонала ООО «Кварц» по категориям работников, результаты анализа представить в таблице 1

Решение

1.Расчет по каждому периоду одинаковый, поэтому приведем расчет за 2020 год.

Всего за год работает 96человек -100%

$$\text{Удельный вес части (доля части)} = \frac{Часть\_целого(группа\_персонала)}{Целое(общая\_численность\_персонала)} \times 100\%$$

Удельный вес руководителей = 17 / 96 * 100 = 18%
Удельный вес специалистов = 38 / 96 * 100 = 40%
Удельный вес служащих = 19 / 96 * 100 = 20%
Удельный вес рабочих = 14 / 96 * 100 = 14%
Удельный вес МОП = 8 / 96 * 100 = 8%

Результаты расчетов заносим в таблицу.
Таблица 1. Структура персонала ООО «Кварц»

Категория персонала	2018		2019		2020	
	числ., чел.	уд. вес, %	числ., чел.	уд. вес, %	числ., чел.	уд. вес, %
Руководители	17	18	17	17	17	18
Специалисты	38	40	39	40	35	38
Служащие	19	20	20	21	21	22
Рабочие	14	14	14	14	13	14
МОП	8	8	8	8	8	9
Итого:	96	100	98	100	94	100

Задача 2
На производственном участке работают:
Начальник- 1человек
Технолог-1человек
Учетчики-9человек
Рабочие-20человек
Уборщица-1человек
Определить структуру персонала.
Решение
Руководитель- 1человек
Специалист-1человек
Служащие-9человек
Рабочие+МОП=21человек
Всего работает 32человека -100%

$$\text{Удельный вес части (доля части)} = \frac{Часть\_целого(группа\_персонала)}{Целое(общая\_численность\_персонала)} \times 100\%$$

Удельный вес руководителей = 1 / 32 * 100 = 3,1%
Удельный вес специалистов = 1/ 32 * 100 = 3,1%
Удельный вес служащих = 9 / 32 * 100 = 28,1%
Удельный вес рабочих+МОП = 21 / 32 * 100 = 65,7%

Виды численности персонала:
1.Штатная численность-складывается только из количества штатных сотрудников, стоящих в расписании компании. Сезонные и временные работники не входят в это число.
2.Явочная численност- та часть персонала, которая в данный момент находится на рабочем месте.
3.Фактическая численность работников-количество людей, реально работающих в компании на какую-либо конкретную дату.
4.Среднесписочная численность -чтобы вычислить этот показатель, определяют отчетный период времени, за который высчитывается среднее число работников, задействованных в компании как на полный, так и на неполный рабочий день, относительно их реального рабочего времени.

Задача 3
Сезонное предприятие начало работать с 17 мая.
Число работников по списку составило(человек):
17 мая-300;
18 мая- 330;
19 мая - 350;
20 мая 360;
с 21 по 31 - 380;
с 1 июня по 31 декабря -400.
Определить среднесписочную численность работников в мае, 2 квартале, за 1 год.

Задача 4
На заводе списочное число сотрудников было:

- февраль-223
- март-218
- апрель-234
- май-228
- июнь-226

Рассчитать среднесписочное число подчиненных за первый квартал, второй квартал и за год.

Задача 5

Сезонное предприятие начало работать с 24 июля. Число работников по списку составило: 24 июля-570 человек; 27 июля устроилось на работу 6 человек, 28 июля уволился один человек, 29 июля устроилось на работу 5 человек, 31июля принято еще на работу 3 человека. Определить среднесписочную численность работников за июль.

Задача 6

На фабрике на 1 декабря работало 205 человек, 7 декабря к ним присоединились еще 15 новых сотрудников, а 15 декабря было уволено 5 подчиненных, а 29 декабря было снова принято на работу 10 новых сотрудников.

Рассчитать среднесписочное количество рабочих фабрики за декабрь месяц.

Задача 7

Среднесписочная численность по месяцам составляет:

- Январь – 345;
- Февраль – 342;
- Март – 345;
- Апрель – 344;
- Май – 345;
- Июнь – 342;
- Июль – 342;
- Август – 341;
- Сентябрь – 348;
- Октябрь – 350;
- Ноябрь – 351;
- Декабрь – 352.

Определить среднесписочную численность за год,1,2,3 квартал.

Задача 8

Средняя численность компании за январь-март составила 30 человек, апрель-август – по 25 человек, а с сентября по декабрь – по 31 человек. Определить среднесписочную численность за год, 1полугодие.

Практическая работа №5
Производительности труда. Расчет экономии труда от воздействия факторов роста производительности труда.
Расчет зарплаты различных категорий работников

Цель:
-умение рассчитывать эффективность использования трудовых ресурсов;
-приобретение навыков расчёта заработной платы различных категорий персонала.

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Эффективность использования трудовых ресурсов на предприятии характеризуется производительностью труда. Производительность труда определяется количеством продукции, произведенной в единицу рабочего времени, или затратами труда на производство единицы продукции.
Производительность труда измеряют показателем выработки продукции и трудоемкости продукции.
Выработка продукции определяется следующим образом:

$$B = \frac{Q}{T},$$

где
Q – количество произведенной продукции;
T – затраты рабочего времени на производство этой продукции.
В зависимости от того, в каких единицах выражены затраты рабочего времени (человеко-часы, человеко-дни, средняя численность промышленно-производственного персонала), различают показатели почасовой, дневной и выработки на одного работника.
Трудоемкость продукции представляет собой отношение трудоемкости продукции на произведенную продукцию на данном предприятии:

$$TE = \frac{T}{Q}.$$

В зависимости от характера и назначения затрат различают нормативную, плановую и фактическую трудоемкость.
По объему исчисления различают трудоемкость операции, детали, изделия, товарной и валовой продукции.
По месту приложения труда выделяют трудоемкость рабочего места, бригады, участка, цеха, предприятия.

Факторы роста производительности труда:
• материально-технические, связанные с техническим уровнем производства,

совершенствованием технологий, техники, применяемых материалов;
• организационные, характеризующие организацию труда, производства и управления;
• социально-экономические, относящиеся к человеческому компоненту производства - качеству работников, их мотивированности и удовлетворенности трудом.
Система показателей производительности труда определяется единицей измерения объема произведенной продукции. Эти единицы могут быть натуральными, условно-натуральными, трудовыми и стоимостными.
Применяют натуральный, условно-натуральный, трудовой и стоимостный методы измерения уровня и динамики производительности труда.
Типовые формулы расчета показателей производительности труда

$$ПТ = ОП / Р$$

(ОП) – объем произведенной продукции
(Р) – численность работников

$$ПТ = ОП / Т$$

(Т) – единица отработанного времени
Тпр = Тт + То.

Задача 1. На основе данных для выполнения задачи определите производительность труда.
Исходные данные.
Металлургический комбинат за год произвел 50 тыс.тонн проката, а среднесписочная численность работников комбината за год составила 2 тыс. чел.
Производительность труда: ПТ = ОП / Р
Задача 2. На основе данных для выполнения задачи определите производительность обслуживания.
Исходные данные.
За год затраты труда вспомогательных рабочих основных цехов составили 50 тыс. чел.-час., а затраты труда рабочих вспомогательных участков и служб, занятых обслуживанием производства, - 75 тыс. чел.-час.
То = Тпр (затраты труда всех рабочих основных + затраты труда вспомогательных цехов)= _____чел-час.
Заданча 3. На основе данных для выполнения задачи определите производительность.
Исходные данные.
Технологическая трудоемкость составила 200 тыс. чел.-час., трудоемкость обслуживания производства – 125 тыс. чел.-час.
Тпр = Тт + То.
Задача 4. На основе данных для выполнения задачи определите производительность труда.
Исходные данные.
Предприятие за год произвело изделий А - 5 тыс. ед., изделий Б – 10 тыс.ед.
Полная трудоемкость изделий А в отчетном году составила 10 нормо-час.,

изделия Б – 6 нормо-час. Среднесписочная численность работников предприятия за год была равна 500 чел.

ПТ = ОП / Р

Задача 5

Определить уровень производительности труда, и ее изменение, если в отчетном году при числе работающих 658 чел. Цех выпустил продукции на 900000 руб., а в плановом году при числе работников 690 чел. должен выпустить 1300000.

Задача 6

Норма времени на производства единицы изделия 78 минут. В результате мероприятий по НОТ она снизилась на 0,35 часа.

Определить рост производительности труда возможный годовой выпуск продукции, если предприятие работает в 2 смены, количество рабочих дней в году 256. Продолжительность смены 480 минут.

Задача 7

В первом полугодии отчетного года выпуск продукции составил 485,8 тыс.руб., а численность работающих – 128 человек.

Во втором полугодии выпуск продукции возрастает на 2,1 %, а численность работающих снизится на 0,8% по сравнению с соответствующими показателями первого полугодия. Определить производительность труда работающих в каждом полугодии

Задача 8

Запланировано на заводе увеличить выпуск продукции по сравнению с прошлым годом на 10% ,а численность работающих на 2%.

Определите планируемый рост производительности труда и увеличение выпуска продукции за счет повышения производительности труда.

Существуют две основные формы заработной платы- повременная и сдельная .При повременной заработной плате размер заработной платы устанавливается в зависимости от количества отработанного времени (в часах или днях) и тарифной ставки (часовой или дневной) или установленного оклада.

При сдельной заработной плате размер заработной платы определяется в зависимости от количества произведенной продукции (работ, услуг) и расценок за единицу продукции (работ, услуг).

Расценки рассчитываются в соответствии с тарифной ставкой, соответствующей разряду данного вида работ, и с установленной нормой времени (выработки):

$$Р_{сд} = Ч_{т.ст} \Leftrightarrow Н_{вр} \text{ или } Р_{сд} = Ч_{т.ст} / Н_{выр},$$

где, $Ч_{т.ст}$ - часовая тарифная ставка по разряду данного вида работ;

$Н_{вр}$ - норма времени на выполнение единицы работы, ч;

$Н_{выр}$ - норма выработки за единицу времени.

При сдельной премиальной системе рабочих сверх заработной платы по прямым сдельным расценкам дополнительно получает премию за определенные количественные и качественные показатели, предусмотренные действующими на предприятии условиями премирования.

Задача 9

Рабочий 4 разряда за месяц изготовил 1500 штук изделий А, норма штучно-калькуляционного времени 10 мин и 800 штук изделий Б, норма времени 15 мин. Часовая тарифная ставка 150 руб. Процент премии за перевыполнение плана 40% , в месяце 24 рабочих дня; продолжительность смены - 8 часов.
Определить:

1. сдельные расценки по изделиям
2. норму выработки в месяц
3. размер заработной платы, которую получит рабочий за месяц
4. тип оплаты труда

Задача 10

За месяц рабочий изготовил 1200 штук изделий, норма времени на изготовление одной детали 10 мин. Часовая тарифная ставка 120 руб. Процент премии за перевыполнение плана 40% , в месяце 24 рабочих дня; продолжительность смены - 8 часов.
Определить:

1. норму выработки в месяц
2. сдельную расценку
3. размер заработной платы, которую получит рабочий за месяц
4. тип оплаты труда

Практическая работа №6
Расчет себестоимости и процента снижения себестоимости единицы доходов. Калькуляция себестоимости единицы продукции. Составление калькуляции и сметы затрат

Цель:
-уметь рассчитывать по принятой методике показатели себестоимости
- научиться определять структуру сметной стоимости работ и проводить необходимые расчеты

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Для выявления экономической эффективности необходимо определить те затраты на обработку детали, которые зависят от характера технологического процесса, т.е. технологическую себестоимость детали. Технологическая себестоимость – это сумма всех прямых затрат, связанная с выполнением технологической операции, выраженная в денежной форме.

Расчёт технологической себестоимости включает в себя следующие статьи затрат:

- стоимость основного материала;
- основная и дополнительная зарплата производственных рабочих

с учётом социальных отчислений;

- стоимость электроэнергии;
- расходы на амортизацию оборудования;
- расходы на ремонт и содержание оборудования.

Результаты расчётов технологической себестоимости детали.

№ п/п	Наименование статей затрат	Сумма, руб./шт.
1	Стоимость основного материала	
2	З/п производственных рабочих	
3	Стоимость электроэнергии	
4	Амортизация оборудования	
5	РСЭО	
	Итого:	

Основные показатели эффективности проектируемого техпроцесса показывают снижение себестоимости изготовления.

Структурой сметной стоимости работ является распределение общей стоимости по группам затрат с указанием их удельного веса. Примерная структура сметной стоимости строительства в зависимости от отрасли строительства, %

Виды затрат	Виды строительства		
	Жилищно-гражданское	Промышленное	Гидротехническое
Строительно-монтажные работы, **Ссмр**	75-90	40-60	70-80
Приобретение основного и вспомогательного технологического оборудования, **Соб**	15-5	50-25	20-15
Прочие работы и затраты, **Спр**	10-5	10-15	10-15
Всего	**100**	**100**	**100**

Задача 1.

Предприятие имеет следующие показатели деятельности: стоимость основного материала – 640000руб., оплата труда – 52000руб., амортизационные отчисления – 28000руб., прочие цеховые расходы – 4%. Определите цеховую себестоимость.

Задача 2

Определить полную себестоимость изделия по следующим данным:

- затраты на основные материалы 4500 руб.;
- энергия на технологические 1100 руб.;
-основная заработная плата 3100 руб.;
- РСЭО - 205%;
-цеховые расходы 90%;
-общезаводские расходы 40%;
-прочие производственные и внепроизводственные расходы 1,5%;
-дополнительная заработная плата основных производственных рабочих 15%;
-социальные выплаты - 30%.

Задача 3

Рассчитать калькуляцию по следующим данным:

- затраты на сырье-220 руб.
- затраты на вспомогательные материалы принять равными 3-5% от стоимости основных материалов
-затраты на топливо и электроэнергию-30 руб.
-расходы на оплату труда - 27 руб.
- содержание и эксплуатация оборудования -70 руб.

Задача 4

Себестоимость одного станка составляет 242500 рублей, прибыль от реализации 44875 рублей, налог на добавленную стоимость 20%. Определить отпускную цену станка.

Задача 5

Себестоимость одного зеркала заднего вида для легковых автомобилей 850 рублей, прибыль от реализации зеркала запланирована 30% , НДС-20%

Определите оптовую цену. Менеджеру по продажам необходимо определить какое количество зеркал может закупить организация, если сумма финансовых средств составляет 7519500.

Задача 6

Определите розничную цену 1-го изделия (товара), если известно, что себестоимость составляет 26000 рублей, прибыль от реализации 30%, наценка посреднических организаций 15%, акцизы 4380 рублей за одно изделия, а торговая наценка 30% (НДС определяется в установленном порядке 20%).

Задача 7

Оптовая рыночная цена 1 единицы изделия 1580 рублей. Определите себестоимость 1 единицы изделия, если известно, что прибыль от их реализации составляет 420 рублей, НДС – 360 рублей, за 1 единицу изделия, наценка посреднических организаций 96 рублей. Менеджеру поп продажам необходимо определить, как изменится прибыль, если себестоимость снизится на 3%.

Задача 8

Отпускная цена одного изделия отечественного производства составляет 2800 рублей. Определите его оптовую рыночную и розничную цену, если известно, что наценка посреднических организаций 25%, торговая наценка 15%. Как изменится розничная цена одного изделия, если торговая наценка увеличится до 20%

Задача 9

Используя данные сметной документации определить структуру сметной стоимости строительно-монтажных работ в текущем уровне цен.

Для определения структуры сметной стоимости необходимо составить и заполнить следующую таблицу:

Статьи затрат	**Тыс. руб.**	**Удельный вес %**
Прямые затраты в том числе: материалы фонд оплаты труда эксплуатация машин		
Накладные расходы		
Итого сметная себестоимость		
Сметная прибыль		
Всего сметная стоимость СМР		

Для заполнения таблицы:1. Произвести заполнение таблицы на основе данных локальной сметы на устройство тепловых сетей.

2.Произвести расчет удельного веса каждой из указанных статей затрат в общем итоге. Статью «Всего сметная стоимость СМР» принять за 100%, путем составления и решения пропорции вычислить удельный вес каждой статьи в общем итоге, полученные данные занести в таблицу, записать ход решения под таблицей.

Практическая работа №7
Расчет прибыли и рентабельности

Цель:
-усвоить методику расчета показателей прибыли и рентабельности,
-уметь выявлять потребности организации для определения точки безубыточности.

Содержание: Студентам следует ответить на приведенные вопросы, выполнить предложенные задания.

Время выполнения 2часа

Порядок выполнения

Прибыль (убыток) от реализации продукции (работ, услуг) определяется как разница между выручкой от реализации продукции в действующих ценах без НДС и затратами на производство и реализацию продукции

$$П = ТП – Сполн,$$

где П – прибыль от реализации продукции, тыс. руб.;
ТП – выручка от реализации товарной продукции, тыс. руб.;
С – полная себестоимость товарной продукции, тыс. руб.
Валовая прибыль – это сумма прибылей (убытков) предприятия как от реализации продукции, так и дохода (расходов), не связанных с ее производством и реализацией. Определяется суммой:

- прибыли от реализации продукции работ и услуг;
- прибыли от прочей реализации (прибыль от реализации товарно-материальных ценностей + прибыль от реализации подсобных хозяйств + прибыль от реализации основных фондов и нематериальных активов);
- финансовых результатов от внереализационных операций (прибыль от долевого участия в деятельности совместных предприятий + прибыль от сдачи в аренду основных средств – пени и штрафы полученные и уплаченные – убытки от списания дебиторской задолженности – убытки от стихийных бедствий)

Валовая прибыль определяется по формуле:

$$П_В = Пр ± Пи ± Пвр$$

Налогооблагаемая прибыль может быть меньше или больше валовой прибыли на сумму корректировок по доходам (расходам), исключаемым при расчете основного налога на прибыль.
Налогооблагаемая прибыль определяется по формуле:

$$Пн.обл. = Пв – Скоррект. \text{ или } Пн.обл. = Пв + Скоррект.$$

*Балансовая (чистая, нераспределенная) прибыль предприятия***,** прибыль оставшаяся в распоряжении предприятия после уплаты налогов и других обязательных платежей.
Балансовая (чистая) прибыль определяется по формуле:

$$Пбал. = Пв – НП$$

Балансовая прибыль (чистая прибыль) может быть распределена:

- на развитие производства,
- на социальное развитие,
- на материальное поощрение

Наличие нераспределенной прибыли, использованной на развитие производства за последний год, а также нераспределенной прибыли прошлых лет свидетельствует о финансовой устойчивости предприятия, что является важной предпосылкой для последующего развития производства.

Соизмерение прибыли с затратами предприятия означает рентабельность, или норму рентабельности. Рентабельность продукции рассчитывается в виде процентного отношения прибыли от реализации продукции к ее полной себестоимости. Рентабельность производственных фондов Рп, % рассчитывается как процентное отношение балансовой прибыли к среднегодовой стоимости основных производственных фондов и оборотных средств

$$Рп= (Пбал/ Сср.год+ Соб.ф)*100\%$$

где Пбал – балансовая прибыль, тыс. руб.;

Сср.год – среднегодовая стоимость основных производственных фондов, тыс. руб.

Соб.ф. – среднегодовая стоимость нормируемых оборотных средств предприятия, тыс. руб.

Определяем рентабельность изделия по формуле

$$Рп= (П/ Сполн)*100\%$$

Задача 1

Типография располагает следующими данными:

– реализованная продукция 65034,6 тыс. руб.;

– полная себестоимость продукции 53481 тыс. руб.

Рассчитать прибыль от реализации продукции, рентабельность изделий.

Решение:

Определим прибыль от реализации продукции по формуле:

П = ТП – Сполн,

Если, ТП = 65034,6 тыс. руб

С = 53481 тыс. руб.

Тогда:

П = 65034, 6 – 53481 = 11553,6 тыс. руб

Определим рентабельность изделия по формуле:

Рп= (П/ Сполн)*100%

Рп=11553,6 тыс/53481 тыс)*100%=21,6%

Ответ***:*** прибыль от реализации продукции составит 11553, 6 тыс. руб; рентабельность изделия 21,6%

Задача 2

Определить прибыль и рентабельность от реализации 5000 шт. журнала при себестоимости 1 шт. 128 руб. и оптовой цене 140 руб.

Решение:

Определим себестоимость от реализации 5000 шт. журнала:

5000 шт. х 128 руб = 640000 руб.

Определим выручку реализации 5000 шт. журнала

5000 шт. х 140 руб = 700000 руб

Определим прибыль по формуле:

П = ТП – Сполн,

Если, ТП = 700000 руб

Сполн = 640000 руб.

Тогда:

П = 700000 – 640000 = 60000 руб

Определим рентабельность изделия по формуле:

Рп= (П/ Сполн)*100%

Ответ: прибыль от реализации журналов составит 60000 руб., рентабельность от их реализации 9,6%

Задача 3

Полиграфкомбинат реализовала за год продукции на сумму 95800 тыс. руб. Полная себестоимость составила 74350 тыс. руб. Определить прибыль от реализации продукции, рентабельность изделий.

Решение:

Определим прибыль от реализации продукции по формуле:

П = ТП – Сполн,

Если, ТП = 95800 тыс. руб

Сполн = 74350 тыс. руб.

Тогда:

П = 95800 – 74350 = 21450 тыс. руб

Определим рентабельность изделия по формуле:

Рп= (П/ Сполн)*100%

Ответ***:*** прибыль от реализации продукции составит 21450 тыс. руб; рентабельность изделия 28,9%

Задача 4

Экономические показатели поточной линии комбината:

– фактический выпуск продукции, учебной литературы в год 170000 шт;

– себестоимость 1 учебника 114,86 руб.;

– цена оптовая 1 учебника 129,89 руб.

Рассчитать сумму прибыли от производства продукции и уровень рентабельности изделия.

Решение:

Определим выручку реализации учебной литературы:

170000 шт. х 129,89 руб = 22081300 руб

Определим себестоимость учебной литературы:

170000 шт. х 114,86 руб = 19526200 руб

Определим прибыль от производства продукции по формуле:
П = ТП – Сполн,
Если, ТП = 22081300 руб
Сполн = 19526200 руб.
Тогда:
П = 22081300 – 19526200 = 2555100 руб
Определим рентабельность изделия по формуле:
Рп= (П/ Сполн)*100%
Ответ**:** прибыль от реализации журналов составит 2555100 руб., рентабельность от их реализации 13,1%

Задача 5

Определить прибыль, выручку товарной продукции, затраты на 1 рубль товарной продукции по следующим данным:
– полная себестоимость 36075,7 руб.;
– плановая рентабельность 20%.

Задача 6

Определить прибыль, рентабельность изделий, затраты на 1 рубль товарной продукции по следующим данным:
– товарная продукция 59451,4 руб.;
– полная себестоимость 48570,1 руб.

Задача 7

Определить прибыль, рентабельность изделий, затраты на 1 рубль товарной продукции по следующим данным:
– товарная продукция – 134678,8 руб.;
– полная себестоимость – 110840,9 руб.

Задача 8

Определить прибыль, рентабельность изделий, затраты на 1 рубль товарной продукции, если:
– товарная продукция 81330,9 руб.;
– полная себестоимость 66905,2 руб.

Задача 9

Определить валовую прибыль, налогооблагаемую прибыль и сумму налога (налог составляет 20%) по следующим данным:
– прибыль от реализации продукции 5345,0 руб
– доход от прочей реализации 546,5 руб;
– внереализационные расходы 234,7 руб;
– сумма необлагаемая налогом 200,0

Задача 10

Полиграфкомбинат в планируемом году должна выработать товарной продукции на сумму 39200 тыс. руб. при полной себестоимости, равной 36300 тыс. руб. В планируемом году предполагается получить внереализационных доходов на сумму 1480 тыс. руб. Планируемые операционные расходы 980 тыс. руб. Определить на плановый год балансовую прибыль.

Задача 11

По утвержденному плану типографии:
– прибыль от реализации 21350 тыс. руб.;
– внереализационные доходы 251 тыс. руб.;
– внереализационные расходы – 195 тыс. руб.

Определить балансовую (валовую) прибыль. Среднегодовая стоимость основных производственных фондов должна составить 32440 тыс. руб., нормируемых оборотных средств 27800 тыс. руб. Определить общую рентабельность типографии.

Задача 12.

Товарная продукция в оптовых ценах 7500 тыс.руб. Себестоимость товарной продукции 6800 тыс.руб. Прибыль от внереализованных операций – 150 тыс.руб. Определить прибыль от реализации продукции основной деятельности предприятия, общую балансовую прибыль предприятия.

Задача 13.

Рассчитать точку безубыточности на промышленном предприятии при следующих условиях: Средняя цена единицы продукции — 100 руб;
Переменные затраты на производство единицы продукции — 75 руб;
Постоянные расходы — 150 000 руб.

Задача 14.

Рассчитать точку безубыточности на промышленном предприятии при следующих условиях: Средняя цена единицы продукции — 100 руб;
Переменные затраты в точке безубыточности составляют — 450 000 руб;
Постоянные расходы — 150 000 руб

Задача 15.

Постоянные затраты по производству продукции за месяц составляют 600 тыс. руб., переменные затраты на 1 кг изделий – 30 руб. Предприятие реализует свои изделия магазинам по цене 45 руб. за 1 кг.

Рассчитайте:

Каким должен быть объем реализации продукции для получения прибыли в размере 210 тыс. руб.? Определите точку безубыточности в количественном и денежном выражении.

Задача 16.

Рассчитать точку безубыточности и порог безубыточности производства. Определить уровень отпускной цены. Определить цену продукции при заданном объеме прибыли. Определить выручку от реализации.

Исходные данные:

1. Постоянные затраты на единицу продукции = 150 руб.
2. Переменные затраты на единицу продукции = 250 руб.
3. Рыночная цена = 400 руб.
4. Спрос на продукцию = 1000 шт.
5. Заданная сумма прибыли = 50 тыс. руб.

Задача 17.
Постоянные расходы – 500 тыс. руб., себестоимость 1 т продукции в части переменных затрат – 20 тыс. руб., цена реализации продукции – 40 тыс. руб. за 1 т. Определите графически (построить график и определить) минимальный объем продукции, необходимый предприятию для вступления в зону прибыльности.

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ

по дисциплине

ЭКОНОМИКА ОТРАСЛИ

специальности СПО

09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЁН
на заседании цикловой комиссии информационных технологий
Протокол от «13» января 2023 г. № 6
Председатель цикловой комиссии И.В. Миляева

Автор: Амеличкина С.Г., преподаватель колледжа

ВВЕДЕНИЕ

При изучении по учебной дисциплины Экономика отрасли специальности 09.02.07 Информационные системы и программирование предусмотрена самостоятельная работа студента в объеме 6 часов.

Виды самостоятельной работы: самостоятельная работа студента по выполнению заданий, подготовке к выполнению практических работ и подготовка доклада.

Темы самостоятельных работ студентов и распределение времени

Тема и содержание работы	Количество часов
Тема 2. Ресурсы хозяйствующих субъектов и эффективность их использования	**2**
Материально-технические, трудовые ресурсы отрасли и организации, показатели их эффективного использования.	
Тема 3. Результаты коммерческой деятельности	**2**
Финансовые ресурсы отрасли и организации, показатели их эффективного использования.	
Тема 4. Планирование и развитие деятельности хозяйствующего субъекта	**2**
Основные технико-экономические показатели деятельности организации и их анализ.	
итого	**6**

Тема 2. Ресурсы хозяйствующих субъектов и эффективность их использования

Материально-технические, трудовые ресурсы отрасли и организации, показатели их эффективного использования.

Задание.

Ознакомиться в Интернете с возможностями программы «1С: Управление торговлей 8», которая позволяет отражать материальные, трудовые и финансовые затраты. Программа позволяет регистрировать и распределять расходы, формирующие стоимость оборотных активов — формирование полной стоимости приобретения и владения товарно-материальными ресурсами.

Тема 3. Результаты коммерческой деятельности

Финансовые ресурсы отрасли и организации, показатели их эффективного использования.

Задание.

Оценить финансовые ресурсы отрасли и организации, показатели их эффективного использования, виды прибыли их взаимосвязь с помощью подготовки доклада.

Примерная тематика докладов

1) Прибыль как показатель эффективности производства.
2) Виды прибыли. Методы распределения валовой и чистой прибыли.
3) Основные направления роста прибыли.
4) Рентабельность, ее виды. Основные направления роста рентабельности.
5) Определение финансовых результатов деятельности предприятия.
6) Роль прибыли в воспроизводственном процессе предприятия.
7) Основные пути повышения экономической эффективности деятельности предприятия.
8) Оптимизация структуры капитала организации.
9) Инфляция и ее влияние на принятие финансовых решений хозяйствующего субъекта.
10) Анализ и оценка финансового состояния хозяйствующего субъекта.
11) Кредитные взаимоотношения организаций.
12) Налогообложение юридического лица и направления его совершенствования.
13) Совершенствование расчетов организаций с поставщиками и покупателями.
14) Бюджетирование и его роль в деятельности хозяйствующего субъекта.
15) Финансовые риски субъекта хозяйствования: понятие, оценка и минимизация.
16) Маркетинговые исследования и ценовая политика предприятия.
17) Влияние учетной политики на финансовый результат деятельности юридического лица.

18) Ликвидность деятельности хозяйствующих субъектов и определяющие ее факторы.

19) Инвестиционная политика организации: анализ и оптимизация.

20) Финансовый менеджмент в организациях.

21) Особенности финансов организаций малого бизнеса.

Доклад – вид самостоятельной научно-исследовательской работы, где автор раскрывает суть исследуемой проблемы; приводит различные точки зрения, а также собственные взгляды на нее. Различают устный и письменный доклад (по содержанию близкий к реферату).

В докладе соединяются три качества исследователя: умение провести исследование, умение преподнести результаты слушателям и квалифицированно ответить на вопросы.

Отличительной чертой доклада является научный, академический стиль. Академический стиль - это совершенно особый способ подачи текстового материала, наиболее подходящий для написания учебных и научных работ. Данный стиль определяет следующие нормы:

- предложения могут быть длинными и сложными;
- часто употребляются слова иностранного происхождения, различные термины;
- употребляются вводные конструкции типа “по всей видимости”, “на наш взгляд”;
- авторская позиция должна быть как можно менее выражена, то есть должны отсутствовать местоимения “я”, “моя (точка зрения)”;
- в тексте могут встречаться штампы и общие слова.

Этапы работы над докладом

1) подбор и изучение основных источников по теме (как и при написании реферата, рекомендуется использовать не менее 4-5 источников);

2) составление библиографии;

3) обработка и систематизация материала. Подготовка выводов и обобщений;

4) разработка плана доклада;

5) написание;

6) публичное выступление с результатами исследования.

Требования к докладу

1) Доклад не копируется дословно из первоисточника, а представляет собой новый вторичный текст, создаваемый в результате осмысленного обобщения материала первоисточника;

2) При написании доклада следует использовать только тот материал, который отражает сущность темы;

3) Изложение должно быть последовательным и доступным для понимания докладчика и слушателей;

4) Доклад должен быть с иллюстрациями, таблицами, если это требуется для полноты раскрытия темы;

5) При подготовке доклада использовать не менее 4-5 первоисточников.

Требования к оформлению доклада

1) Наличие титульного листа (см. ПРИЛОЖЕНИЕ)

2) Основное содержание - 2-3страницы печатного текста (на одной стороне белой бумаги) следующего формата:

страница:

- ориентация: книжная;
- поля: верхнее и нижнее — 20 мм, левое — 30 мм, правое — 10 мм;
- размер бумаги: А4

шрифт:

- Times New Roman;
- размер: 14 пт;
- цвет: черный;

абзац:

- выравнивание заголовков - по центру,
- выравнивание основного текста - по ширине,
- отступ первой строки - 1,25 см.
- междустрочный интервал – полуторный (1,5 строки)

3) 3. Наличие списка используемых информационных источников (книги, журналы, сайты Интернет с указанием URL-адреса сайта)

Тема 4. Планирование и развитие деятельности хозяйствующего субъекта

Основные технико-экономические показатели деятельности организации и их анализ.

Задание.

Провести анализ производственных результатов деятельности организации по следующим направлениям:

- оценка динамики производства и реализации продукции;
- анализ производства продукции по номенклатуре и ассортименту;
- анализ влияния структурных сдвигов на объем продукции;
- анализ ритмичности работы предприятия;
- анализ качества и конкурентоспособности продукции.

Изучить в Интернете источники информации для анализа производственных результатов:

1) плановые и оперативные планы-графики;

2) данные текущей и годовой отчетности:

- форма 1-П «Отчет предприятия (объединения) по продукции»;
- форма «Баланс предприятия»,
- форма «Отчет о финансовых результатах»;

3) данные текущего бухгалтерского и статистического учета:

- ведомость № 16 «Движение готовых изделий, их отгрузка и реализация»;
- журнал-ордер № 1, карточки складского учета готовой продукции и др.

ПРИЛОЖЕНИЕ

Пример оформления титульного листа

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

ДОКЛАД

по дисциплине «Экономика отрасли»

на тему: «Прибыль как показатель эффективности производства»

Автор работы,
студент гр. **А. А. Петров**

Руководитель,
преподаватель **П. П. Иванова**

Тула 20__

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторно-практических работ

по междисциплинарному курсу
МДК 02.01 «Технология разработки программного обеспечения»

профессионального модуля
ПМ.02. ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕ

специальности СПО

09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «13» января 2023 г. № 6
Председатель цикловой комиссии _____ И.В. Миляева

Авторы: Сафронова М.А., преподаватель, канд. техн. наук

Практическая работа №1
«ОБЗОР И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Цель работы: получить практические навыки в получении сведений об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты предметной области, возможных воздействий окружающей среды на элементы и явления предметной области, а также обратные воздействия этих элементов и явлений на среду влияющих на проектирование программного обеспечения и эффективность его работы.

Теоретические сведения

1. Определение предметной области

Предметную область можно определить как сферу человеческой деятельности, выделенную и описанную согласно установленным критериям.

В описываемое понятие должны входить сведения об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты этой деятельности.

В описании предметной области должны присутствовать характеристики возможных воздействий окружающей среды на элементы и явления предметной области, а также обратные воздействия этих элементов и явлений на среду.

Работа по изучению и анализу предметной области: проектировании интеллектуальных систем оказывает решающее влияние на эффективность ее работы.

Специфика предметной области может оказывать существенное влияние на характер функционирования проектируемой интеллектуальной системы, выбор метода представления знаний, способов рассуждения о знаниях, и т. д.

Предметную область можно определить как объект или производственную систему со всем комплексом понятий и знаний о ее функционировании.

При исследовании проблемной области необходимы знания о задачах, решаемых в производственной системе, и стоящих перед ней целях. Определяются также возможные стратегии управления и эвристические знания, используемые в процессе эксплуатации системы.

2. Анализ предметной области (анализ осуществимости, бизнес - моделирование)

Одна из первых задач, с решением которых сталкивается разработчик программной системы - это изучение, осмысление и анализ предметной области. Дело в том, что *предметная область сильно влияет на все аспекты проекта: требования к системе, взаимодействие с пользователем, модель хранения данных, реализацию и т.д.*

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации. Для документирования могут быть использованы различные средства.

Для управления обсуждением области действия проекта можно использовать методику "будет - не будет". В простейшем случае - это список с двумя столбцами, в одном из которых записывается, что проект будет делать, а во втором - что не входит в проект. Такой список, формируется заинтересованными лицами при рассмотрении каждой бизнес-цели проекта, используя любую технику, например метод "мозгового штурма" (см. тему "Выявление требований"). Полученные характеристики позволяют четко определить

границы проекта и довольно просто преобразуются в предположения, которые фиксируются в документе.

Функциональная область действия определяет услуги, предоставляемые системой, и вначале до конца неизвестны. В определении услуг системы может помочь список "Действующее лицо/Цель", в котором перечислены все цели пользователя, поддерживаемые системой. При его разработке в первую графу вписываются имена основных действующих лиц, т.е. тех, кто имеет цели, во вторую графу - цель каждого действующего лица, а в третью - приоритет или предположение о том, в какую версию войдет эта услуга. Формы списков приведены на рисунке.

Для определения основных функций продукта можно использовать, например, краткое описание варианта использования. Описание каждой функции можно представить также в виде списка, состоящего из трех граф: действующее лицо, цель и краткое описание варианта использования.

Анализ предметной области является основой для анализа осуществимости проекта и определения образа (концепции) продукта и границ проекта.

Анализ осуществимости

Разработка новых программных систем должна начинаться с анализа осуществимости. На основании анализа предметной области, общего описания системы и ее назначения необходимо принять решение о продолжении или завершении проекта. Для этого необходимо ответить на следующие вопросы.

1. Отвечает ли система бизнес-целям заказчика и разработчика?
2. Можно ли реализовать систему, используя известные технологии и не выходя за пределы заданной стоимости и заданного времени?
3. Можно ли объединить систему с другими уже эксплуатируемыми системами?

Для ответа на первый (и главный) вопрос можно опросить заинтересованных лиц, например, менеджеров операторов и т.п. отделов/подразделений, в которых будет использоваться система, для выяснения того,

- что произойдет с организацией, если система не будет введена в эксплуатацию;
- как система будет способствовать целям бизнеса;
- какие текущие проблемы поможет решить система и т.д.

После получения и обработки информации готовится отчет, в котором должны быть даны рекомендации относительно продолжения разработки системы.

Постановку бизнес-задачи надо обсуждать с Заказчиком, или будущим Пользователем системы.

Вопросы, которые ему стоит задать, это:

1) Почему вообще пошла речь о создании системы?
2) В чём Вы видите её назначение?
3) Какие бизнес-возможности она должна реализовать?
4) Какие проблемы должна решить?

В качестве "Стандарта" на вопросы такого рода смотрите шаблон документа "Stakeholder Request", например, из RUP.

Бизнес-требования может выразить Заказчик или Эксперты предметной области. Они обычно фиксируются в виде списка из 10-30 ключевых свойств продукта - в качестве шаблона см. Vision из RUP.

Бизнес-моделирование надо проводить на основе информации от, а лучше совместно с экспертами предметной области. Вопросы по сути сводятся к "Что, почему, когда, как и кем происходит в предметной области и как оно взаимосвязано?":

1. Каковы основные понятия предметной области, их определения и взаимосвязи? Результат можно оформить в виде глоссария и/или концептуально-семантической модели предметной области.

2. На основании каких правил - международных, федеральных, муниципальных, районных и т.д. законов, указов, стандартов, спецификаций, регламентов и т.д. - происходит то, что происходит в предметной области? Результат оформляете в виде структурированного списка или прикрепляете к элементам концептуальной модели.

3. Что реально (какие процессы, события, факты) происходит и в какой последовательности, взаимосвязи? Результат оформляете в виде сценариев описания бизнес-процессов (что достаточно универсально) или диаграмм SADT (IDEF0, IDEF3, DFD) / ARIS (eEPC и т.д.) / UML (Business Use-case Diagram (BUC) + Activity Diagram + Sequence Diagram). Это один из сложнейших этапов.

4. Какими свойствами обладает каждое из выделенных понятий - структурными и поведенческими? Результат описывается в виде таблиц с атрибутами Концептуальных сущностей или Детальной концептуальной моделью - ER - IDEF1X / UML Class Diagram (BOM).

Существует российский стандарт функционального моделирования Р 50.1.028-2001, созданный на основе IDEF0.

Определение требований - частично Бизнес-требования и Требования, проистекающие из *предметной области* вы уже определили выше, теперь осталось исследовать *Пользовательские требования* и *Системные требования* и ограничения к отдельным аспектам качества системы. *Пользовательские требования*, как можно понять, нужно выявлять из общения с потенциальными пользователями системы. Вопросы:

1) На какую систему будет похожа создаваемая?
2) С какими системами и как давно вы работаете?
3) Какое у вас образование?
4) Каковы ваши ожидания от системы - что и как она должна делать, какие задачи помогать решать, как должна выглядеть?
5) Какие шаги необходимо предпринять для решения каждой задачи?
6) В каком случае вы будете считать, что система "Хороша"?

Результаты анкетирования/интервьюирования обычно представляют в виде пользовательских историй (User Story, Agile) или Пользовательских сценариев (Use-case), также возможно их диаграммное представление средствами диаграмм потока работ (IDEF3), ARIS, Activity/State UML Diagram.

Системные требования нужно выяснять у IT-специалистов Заказчика, если таковые имеются, из специфики контекста использования системы, опыта построения аналогичных систем и Специалистов по отдельным аспектам системы, значимым для данного проекта и Заказчика:

1) Будет ли система единичной или тиражируемой?
2) В каких странах она будет работать?
3) Насколько важна информация, хранящаяся, обрабатываемая и передающаяся системой?
4) Каков возможный ущерб от потери той или иной информации?
5) Сколько пользователей будет работать с системой сегодня, завтра, через год?

Переработанный результат оформляется в виде Системных требований (Software Requirement Specification, стандарт IEEE-STD-830-1998, или ТЗ ГОСТ 34-602-89 или неформально в виде Supplementary Specificaion из RUP).

Приложения для настольных компьютеров подобны широкоугольным объективам в том смысле, что в типичных случаях они отображают значительный объем информации, который позволяют предоставлять пользователю экраны большого размера. В отличие от этого мобильные приложения напоминают увеличительное стекло или объектив с переменным фокусным расстоянием. Они предоставляют пользователю возможность

быстро просматривать необходимые подробные данные, быстро переходить к ограниченным наборам данных и получать к ним доступ, а также принимать решения в реальном масштабе времени. Как правило, мобильные приложения пре доставляют более специализированный набор сценариев по сравнению с приложениями, ориентированными на настольные компьютеры. Очень важно точно определиться с тем, на каких сценариях должно специализироваться ваше приложение.

Прежде чем приступать к реальной разработке приложения, определите подмножество функциональных средств, к которым пользователь сможет получать быстрый доступ в манере, свойственной мобильным устройствам. В случае создания нового приложения, аналога которого для настольных компьютеров не существует, выпишите ключевые сценарии, которые пользователи смогут выполнять с помощью вашего приложения, а также порядок действий пользователя, обеспечивающий использование этих сценариев на мобильном устройстве. Во многих случаях вам будет легче придать этим сценариям реальные очертания, если вы подготовите соответствующие рисунки или создадите прототипы. Если подразумевается определенная группа конечных пользователей, пообщайтесь с ними и предоставьте им возможность поработать некоторое время с экспериментальными версиями своих приложений, чтобы они могли дать о них свои отзывы.

Оптимальный подбор предоставляемых средств определяет все остальное.

Если вы правильно выделите ключевые сценарии и возможности вашего приложения, то это окажет определяющее влияние на всю оставшуюся часть процесса разработки. Наличие явно сформулированного описания того, как конечные пользователи будут использовать ваше приложение, и детальное понимание их потребностей окажут вам неоценимую помощь при настройке производительности приложения, а также проектировании пользовательского интерфейса, коммуникационной системы и модели памяти.

Если вы не определите важнейшие с вашей точки зрения сценарии и возможности, то в результате вы получите бессистемную смесь средств, объединенных в одно приложение. Отсутствие явного списка основных функций приложения или разделения функций на группы в соответствии с их приоритетами приведет к тому, что пользовательский интерфейс не будет оптимизирован для эффективного решения ключевых задач. Например, если ожидается, что пользователь в основном будет заинтересован во вводе данных, то вы должны оптимизировать пользовательский интерфейс таким образом, чтобы обеспечить как можно более точное и надежное выполнение операций ввода. И наоборот, если ввод данных используется лишь изредка, то вариант пользовательского интерфейса ввода, оптимизированного не самым идеальным образом, может оказаться вполне допустимым, что позволит перебросить ресурсы проектирования и разработки на другие направления. Лишь только если группой разработчиков будут идентифицированы, перечислены и согласованы наиболее важные сценарии, эксплуатационные характеристики приложения могут быть настроены для их выполнения должным образом, а конечные пользователи не будут лишены важных для них средств из-за недосмотра.

Чтобы процесс разработки мог быть успешно завершен, составьте список ключевых требований, которым должно удовлетворять приложение, и возможностей, которые оно должно обеспечивать, и пусть этот список будет первым разделом вашего основного документа проекта.

3. Методы сбора и анализа информации (предметной области)

Анализ предметной области - деятельность, направленная на выявление реальных потребностей заказчика, а также на выяснения смысла высказанных требований, называется анализом предметной области (бизнес-моделированием, если речь идет о потребностях коммерческой организации).

Анализ предметной области – это первый шаг этапа системного анализа, с которого начинается разработка программной системы. Разработчики должны научиться:

- понимать язык, на котором говорят заказчики;
- выявить цели их деятельности;
- определить набор решаемых ими задач;
- определить набор сущностей, с которыми приходится иметь дело при решении этих задач.

Для этого разработчики с участием заказчика решают следующие *основные задачи*:

- изучают миссию предприятия, цели деятельности и стратегии их достижения;
- исследуют функции предприятия, их распределение между подразделениями предприятия;
- выполняют статическое описание предприятия (объектной, функциональной, организационной структуры, структуры управления);
- выполняют динамическое описание предприятия (бизнес-функций и потоков деятельности);
- моделируют отдельные бизнес-процессы на предприятии.

Результатом анализа предметной области является детальное описание данных информационных потребностей пользователей ПО.

Информационные потребности пользователей ПО представляют собой информационное описание задач, которые будут решаться пользователями с применением ПО. Такие задачи называют задачами автоматизации.

В описание каждой задачи автоматизации включают:

- общие характеристики задачи (наименование, пользователи, длительность и режим выполнения, связь с другими задачами и др.);
- входная информация (данные, документы) задачи;
- выходная информация (данные, документы) задачи;
- характеристики информационных процессов.

К информационным процесса и их характеристикам относятся:

- ввод данных (источники, способы сбора данных, методы ввода, формы, объемы данных, режимы);
- обработка данных (правила, критические сроки, объем операций),
- вывод данных (способы, форматы, средний и критический объемы, образцы, размещение адресатов, сроки);
- хранение данных (организация, структура данных, их объем, режим и частота запросов, правила и методы сжатия, обновления и удаление);
- управление данными (целостность и непротиворечивость, безопасность, конфиденциальность);
- передача данных (формат и объем передаваемых данных, временные характеристики, протоколы передачи).

Основными методами сбора информации для проведения анализа предметной области являются:

- изучение документов;
- анкетирование;
- наблюдение;

- интервьюирование;
- экспертные оценки;
- эксперименты;
- контент-анализ;
- вебометрический анализ;
- анализ ситуаций.

Изучение документов – метод сбора данных в ходе проведения исследований деятельности предприятия с использованием документов.

Анкетирование – письменный опрос сотрудников предприятия с использованием опросных листов (анкет), которые им предлагается заполнить.

Наблюдение – метод сбора информации, с помощью которого осуществляется систематическое, планомерное изучение поведения того или иного объекта или субъекта.

Интервью - это метод сбора первичной информации путем выяснения субъективных мнений, предпочтений, установок людей в отношении какого-либо объекта при непосредственном общении с ними.

Экспертная оценка - это метод исследования процессов квалифицированными специалистами - экспертами.

Эксперимент - это исследование влияния одного фактора на другой при одновременном контроле посторонних факторов.

Контент-анализ представляет собой формализованный метод качественно-количественного изучения информации, основанный на выделении в содержании материалов определенных смысловых категорий.

Вебометрический анализ - исследуются количественные аспекты конструирования и использования информационных ресурсов, структур и технологий применительно к Internet.

Задания на работу

Задание № 1. Ознакомиться с предложенным вариантом описания предметной области (согласно индивидуальному варианту).

№№ варианта	*Предметная область*
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

Задание № 2. Используя различные методы анализа предметной области (контент-анализ, вебометрический анализ, анализ ситуаций, моделирование и др.) проанализировать предметную область, уточнив и дополнив ее, руководствуясь собственным опытом, консультациями и любыми источниками (книгами, учебниками или интернет-источниками).

Задание № 3. Выполнить структурное разбиение предметной области на отдельные подразделения (подсистемы) согласно выполняемым ими функциям.

Задание № 4. Определить задачи и функции системы в целом и функции каждого подразделения (подсистемы).

Задание № 5. Продумать подробное описание работы каждого подразделения (подсистемы), алгоритмов и сценариев выполнения ими отдельных работ. Продумать виды входной и выходной информации для каждого подразделения (подсистемы).

Задание № 6. Описать схему работы будущей информационной системы, учитывая выделенные и описанные ранее подсистемы.

Задание № 7. Определить группу пользователей, для которой данная система будет более востребована. Описать перечень функций системы, которые будут доступны данной группе пользователей.

Задание № 8. Расписать основные функциональные возможности администратора системы, как одного из пользователей системы.

Задание № 9. Оформить отчет.

Контрольные вопросы

1. Дайте определение «Предметная область»
2. Что позволяет выявить анализ предметной области?
3. Какие существуют методы анализа предметной области?
4. Что включает в себя постановка задачи и предпроектные исследования?
5. Перечислите функциональные и эксплуатационные требования к программному продукту.

Практическая работа №2
«РАЗРАБОТКА И ОФОРМЛЕНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ»

Цель работы: ознакомиться с правилами написания технического задания.

Теоретические сведения

Техническое задание представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемо-сдаточных испытаний. В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

1. Порядок разработки технического задания

Разработка технического задания выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспечение собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

1.1 Общие положения

1. Техническое задание оформляют в соответствии с ГОСТ 34.602-89 на листах формата А4 и А3 по ГОСТ 2.105-2019 как правило, без заполнения полей листа.

2. Для внесения изменений и дополнений в техническое задние на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

3. Техническое задание должно содержать следующие разделы:

- введение;
- наименование 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 зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них. При необходимости допускается в техническое задание включать приложения.

1.2. Содержание разделов ТЗ

1. Введение должно включать краткую характеристику области применения программы или программного продукта, а также объекта (например, системы), в котором предполагается их использовать. Основное назначение введения — продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

2. В разделе «Наименование и область применения» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

3. В разделе «Основание для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка. Таким документом может служить план, приказ, договор и т. п.
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

4. В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

5. Раздел «Технические требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

1) В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

2) В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т. п.).

3) В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т. п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

4) В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их технических характеристик.

5) В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования. При необходимости должна обеспечиваться защита информации и программ.

6) В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

7) В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

6. В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

7. В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.

8. В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

9. В приложениях к техническому заданию при необходимости приводят:

- перечень научно-исследовательских и других работ, обосновывающих разработку;
- схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
- другие источники разработки.

В случаях, если какие-либо требования, предусмотренные техническим заданием, заказчик не предъявляет, следует в соответствующем месте указать «Требования не предъявляются».

Пример ТЗ представлен в приложении А.

Задания на работу

1. Разработать техническое задание в соответствии с ГОСТ 34.602-89 на программное средство согласно индивидуальному варианту.

№№ варианта	*Название программного средства (информационной системы)*
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»

2. Оформить ТЗ в соответствии с ГОСТ 2.105-2019.

Контрольные вопросы

1. Приведите этапы разработки программного обеспечения.
2. Перечислите правила разработки технического задания.
3. Назовите основные разделы технического задания.

Практическая работа №3
«ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРОГРАММНОГО СРЕДСТВА»

Цель работы: ознакомиться с вопросами построения архитектуры программного средства (ПС).

Теоретические сведения

Архитектура ПС - это его строение как оно видно (или должно быть видно) извне его, т.е. представление ПС как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы. Разработка архитектуры является первым этапом борьбы со сложностью ПС, на котором реализуется принцип выделения относительно независимых компонент.

Основные задачи разработки архитектуры ПС:

- Выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) ПС;
- определение способов взаимодействия между выделенными программными подсистемами.

С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

При возникновении потребностей в заказе, приобретении, разработке, эксплуатации и сопровождении программ перед всеми сторонами, вовлеченными в жизненный цикл программного средства (ПС), возникает целый ряд вопросов, связанных с определением и детальным структурированием жизненного цикла (ЖЦ) ПС, с организационными и техническими правами и обязанностями сторон, с управлением ЖЦ и контролем за его реализацией. Одним из действенных инструментов для решения данных вопросов является использование унифицированных подходов, закрепленных в современных международных и российских стандартах.

Понятия «жизненный цикл системы» или «жизненный цикл программного средства» часто появляются в статьях и звучат в разговорах разработчиков, по крайней мере руководителей проектов и подразделений. Всем понятно, что относятся они к тому, что и в какой последовательности должно делаться при создании и эксплуатации систем. Но прежде чем две организации или два специалиста договорятся о том, что конкретно входит или не входит в ЖЦ, проходит значительное время. А позже вполне может обнаружиться, что эти двое (две «стороны») все-таки по-разному понимают, какие работы будут входить в ЖЦ, а какие - нет, какие проверки будут планироваться, когда и т. д. Естественно, общие принципы организации работ описаны давно, но что делать сторонам в конкретном проекте — это каждый раз приходится решать заново.

В стандартах, регламентирующих жизненный цикл программных средств, обобщаются опыт и результаты исследований множества специалистов и рекомендуются наиболее эффективные современные методы и процессы создания и развития комплексов программ. В результате таких обобщений оттачиваются технологические процессы и приемы разработки, а также методическая база для их автоматизации.

ЖЦ ПС в стандартах представляет собой набор этапов, частных работ и операций в последовательности их выполнения и взаимосвязи, регламентирующих ведение работ от подготовки технического задания до завершения испытаний ряда версий и окончания эксплуатации ПС или информационной системы (ИС).

Стандарты включают правила описания исходной информации, способов и методов выполнения операций, устанавливают правила контроля технологических процессов, требования к оформлению их результатов, а также регламентируют содержание технологических и эксплуатационных документов на комплексы программ.

Они определяют организационную структуру коллектива, обеспечивают распределение и планирование заданий, а также контроль за ходом создания ПС.

Кроме вопросов выбора типа общего устройства ЖЦ есть проблемы с решением частных вопросов о включении или невключении в ЖЦ отдельных работ, очень важных для качества ПС и системы: что документировать при создании системы и ПС, какие работы должны будут гарантировать качество продукта, с какой степенью организационной независимости должны выполняться проверочные процедуры разных типов, чем будет обеспечиваться соответствие разрабатываемого ПС требованиям ко всей системе и соответствие ПС потребностям в системе.

Для того чтобы привнести порядок и понимание, общие для любых сторон, участвующих в ЖЦ систем и ПС, давно разрабатывались стандарты различных уровней утверждения - национальные и международные.

В стандарте ГОСТ Р ИСО/МЭК 12207 программное обеспечение (ПО) или программный продукт определяется как набор компьютерных программ, процедур и, возможно, связанной с ними документации и данных.

Процесс определяется как совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные. Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными от других процессов, и результатами.

В соответствии с ГОСТ Р ИСО/МЭК 12207 все процессы ЖЦ ПО разделены на три группы:

1) Основные процессы:
- приобретение;
- поставка;
- разработка;
- эксплуатация;
- сопровождение.

2) Вспомогательные процессы:
- документирование;
- управление конфигурацией;
- обеспечение качества;
- верификация;
- аттестация;
- совместная оценка;
- аудит;
- разрешение проблем.

3) Организационные процессы:
- управление;
- усовершенствование;
- создание инфраструктуры;
- обучение.

Процесс разработки предусматривает действия и задачи, выполняемые разработчиком, и включает следующие действия:

1) Подготовительная работа начинается с выбора модели ЖЦ ПО, соответствующей масштабу, значимости и сложности проекта. Действия и задачи процесса должны соответствовать выбранной модели. Разработчик должен выбрать, адаптировать к условиям проекта и использовать согласованные с заказчиком стандарты, методы и средства разработки, а также составить план выполнения работ.

2) Анализ требований к системе подразумевает определение ее функциональных возможностей, пользовательских требований, требований к надежности и безопасности, требований к внешним интерфейсам и т.д. Требования с системе оцениваются исходя из критериев реализуемости и возможности проверки при тестировании.

Анализ требований к ПО предполагает определение следующих характеристик для каждого компонента ПО:

- функциональных возможностей, включая характеристики производительности и среды функционирования компонента;
- внешних интерфейсов;
- спецификаций надежности и безопасности;
- эргономических требований;
- требований к используемым данным;
- требований к установке и приемке;
- требований к пользовательской документации;
- требований к эксплуатации и сопровождению.

Требования к ПО оцениваются исходя из критериев соответствия требованиям к системе, реализуемости и возможности проверки при тестировании.

3) Проектирование архитектуры системы на высоком уровне заключается в определении компонентов ее оборудования, ПО и операций, выполняемых эксплуатирующим систему персоналом. Архитектура системы должна соответствовать требованиям, предъявляемым к системе, а также принятым проектным стандартам и методам.

Проектирование архитектуры ПО включает задачи (для каждого компонента ПО):

- трансформацию требований к ПО в архитектуру, определяющую на высоком уровне структуру ПО и состав ее компонентов;
- разработку и документирование программных интерфейсов ПО и баз данных;
- разработку предварительной версии пользовательской документации;
- разработку и документирование предварительных требований к тестам и планам интеграции ПО.

Архитектура компонентов ПО должна соответствовать требованиям, предъявляемым к ним, а также принятым проектным стандартам и методам.

4) Детальное проектирование ПО включает следующие задачи:

- описание компонентов и интерфейсов между ними на более низком уровне, достаточном для их последующего самостоятельного кодирования и тестирования;
- разработку и документирование детального проекта базы данных;
- обновление (при необходимости) пользовательской документации;
- разработку и документирование требований к тестам и плана тестирования компонентов ПО;
- обновление плана интеграции ПО.

5) Кодирование и тестирование ПО охватывает задачи:

- разработку и документирование каждого компонента ПО и базы данных а также совокупности тестовых процедур и данных для их тестирования;
- тестирование каждого компонента ПО и базы данных на соответствие предъявляемых к ним требованиям. Результаты тестирования компонентов должны быть документированы;
- обновление (при необходимости) пользовательской документации;

- обновление плана интеграции ПО.

6) Интеграция ПО предусматривает сборку разработанных компонентов ПО в соответствии с планом интеграции и тестирование агрегированных компонентов. Для каждого из агрегированных компонентов разрабатываются наборы тестов и тестовые процедуры, предназначенные для проверки каждого из квалификационных требований при последующем квалификационном тестировании.

Интеграция системы заключается в сборке всех ее компонентов, включая ПО и оборудование. После интеграции система, в свою очередь, подвергается квалификационному тестированию на соответствие совокупности требований к ней. При этом также производится оформление и проверка полного комплекта документации на систему.

7) Квалификационное тестирование - это набор критериев и условий, которые необходимо выполнить, чтобы квалифицировать программный продукт как соответствующий своим спецификациям и готовый к использованию в условиях эксплуатации.

Квалификационное тестирование ПО проводится разработчиком в присутствии заказчика (по возможности) для демонстрации того, что ПО удовлетворяет своим спецификациям и готово к использованию в условиях эксплуатации. Квалификационное тестирование выполняется для каждого компонента ПО по всем разделам требований при широком варьировании тестов. При этом также проверяются полнота технической и пользовательской документации и ее адекватность самим компонентам ПО.

8) Установка ПО осуществляется разработчиком в соответствии с планом в той среде и на том оборудовании, которые предусмотрены договором. В процессе установки проверяется работоспособность ПО и баз данных. Если устанавливаемое программное обеспечение заменяет существующую систему, разработчик должен обеспечить их параллельное функционирование в соответствии с договором.

9) Приемка ПО предусматривает оценку результатов квалификационного тестирования ПО и системы и документирование результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу ПО заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

Задания на работу

Разработать проект архитектуры программного средства в соответствии с ГОСТ Р ИСО/МЭК 12207 на высоком уровне (ПС выбрать исходя из индивидуального задания практической работы №2).

Контрольные вопросы

1. В какой форме представлен ЖЦ ПО в ГОСТ Р ИСО/МЭК 12207?
2. Какая нормативная информация включена в современные стандарты, регламентирующие жизненный цикл программных средств?
3. Чем объясняется актуальность стандарта ГОСТ Р ИСО/МЭК 12207 в настоящее время?
4. Приведите определение программного обеспечения в соответствии с ГОСТ Р ИСО/МЭК 12207.

5. Как определяется процесс в ГОСТ Р ИСО/МЭК 12207?
6. Какие группы процессов ЖЦ ПО выделены в ГОСТ Р ИСО/МЭК 12207?
7. Какие действия и задачи, выполняемые разработчиком, предусмотрены ГОСТ Р ИСО/МЭК 12207 в процессе разработки ПС?

Практическая работа №4
«ИЗУЧЕНИЕ РАБОТЫ В СИСТЕМЕ КОНТРОЛЯ ВЕРСИЙ»

Цель работы: ознакомиться с системой управления версиями Git сервиса Github.

Теоретические сведения

1. Системы управления версиями.

Системы управления версиями (Version control system, VCS) – программное обеспечение, предназначенное для управления быстро меняющейся информацией. Системы управления версиями позволяют сохранять нужное количество предыдущих версий информации (обычно – текстового документа, в том числе с программным кодом , или набора таких документов – проекта), обращаться к этим версиям при необходимости, а также позволяют организовать редактирование одного и того же документа разными пользователями.

Существуют централизованные и распределенные системы управления версиями.

В централизованных VCS функции по хранению версий документа и предоставлению доступа к ним берет на себя сервер. Для внесения изменений в какой-либо документ пользователю в первую очередь обычно требуется скачать на свой компьютер «рабочую копию» нужной ему версии документа или обновить уже имеющуюся версию до последней (update) .После внесения изменений пользователь закачивает новую версию документа на сервер (commit), причем предыдущая версия не удаляется, а также сохраняется на сервере.

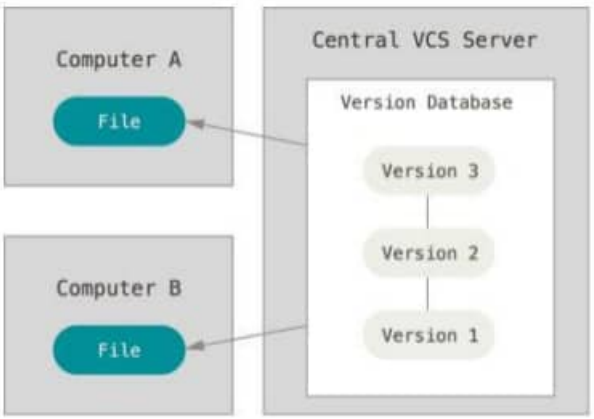


Рисунок 1 - Централизованная система управления версиями**.**

При работе в многопользовательском режиме сервер также осуществляет отслеживание и управление конфликтами изменений, сделанных разными пользователями. Если несколько пользователей одновременно редактируют один и тот же файл, то изменения, сделанные одним пользователем, будут отменены тем пользователем, который загрузит свою версию того же документа на сервер после первого пользователя. Чтобы избежать конфликтов, может применяться блокировка документа, который один из пользователей скачал на свой компьютер с целью изменения. При этом остальные пользователи не имеют возможности внести в него изменения(то есть загрузить на сервер свою новую версию этого документа), пока первый пользователь не «разблокирует» документ после внесения в него изменений.

Так как механизм блокировки приводит к замедлению совместной работы над документом, часто (например, при слиянии ветвей, см. ниже) вместо недопущения конфликтов применяется механизм их автоматического или ручного разрешения. Автоматически могут быть разрешены конфликты, при которых пользователи внесли

изменения в разные части документа. Если же изменения, сделанные разными пользователями, пересекаются, или в случае, если формат документа отличен от текстового (картинка, бинарный файл), система управления версиями предложит пользователям разрешить его конфликт вручную.

Централизованные VCS, как и любые централизованные системы имеют риски потери некоторой части информации при выходе из строя центрального сервера. Для устранения этих рисков могут использоваться децентрализованные системы управления версиями. В децентрализованных VCS пользователи скачивают на свои рабочие компьютеры не только нужные им версии нужных им файлов, а полностью весь репозиторий (хранилище файлов). В это м случае при выходе из строя сервера VCS вся информация может быть восстановлена из рабочих компьютеров пользователей.

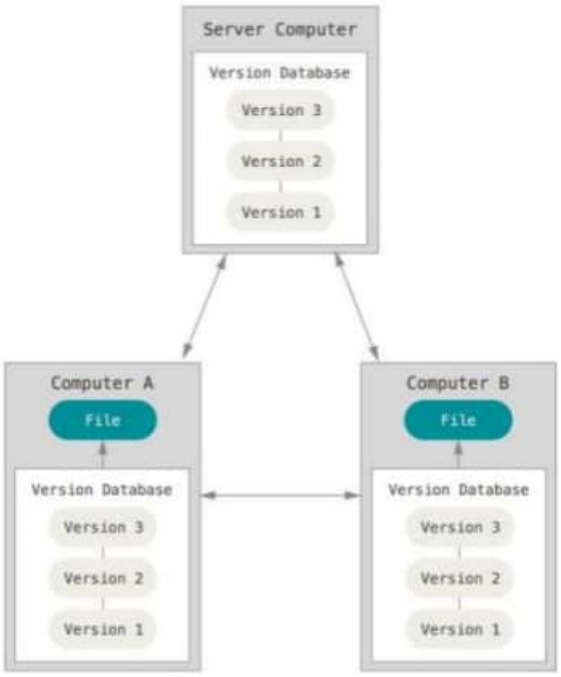


Рисунок 2 - Децентрализованная система управления версиями.

Помимо этого, системы управления версиями хранят информацию о том, кто, когда и какие изменения выполнил, позволяет сравнивать две версии документа, а также реализует функцию ветвления. Создание «ветви» (branch или fork) позволяет создать две копии одного и того же набора документов, развивать их параллельно и при необходимости снова слить в один проект (merge). Ветвление позволяет, например, поддерживать несколько версий одного и того же продукта и зачастую выполняется перед началом работы над достаточно объемным обновлением для продукта.

2. Система управления версиями Git.

Git – распределенная кроссплатформенная система управления версиями. Благодаря своему удобству и скорости работы является в настоящий момент одной из самых популярных VCS. С Git можно работать как посредством командной строки, так и при помощи любого из множества графических интерфейсов.

Установку Git на Linux-системы можно произвести стандартной командой, например, для Ubuntu:

```
sudo apt-get install git
```

Для Windows можно использовать Git for Windows (https://gitscm.com/download/win) или любой другой клиент на ваш выбор.

Если вы решили освоить работу с git через командную строку, то ниже приведены некоторые команды для работы с Git из командной строки.

Создание репозитория Git в той директории, в которой выполняется команда:

```
git init
```

Создание локальной рабочей копии локального или удаленного репозитория:

```
git clone [url]
```

После создания рабочей копии можно добавлять и изменять в ней нужные файлы. Файлы в Git могут находиться в трех различных состояниях:

1) *Committed или зафиксированном* – файл сохранен в локальном репозитории. Репозиторий Git хранит в сжатом виде все версии файлов и метаданные, это основное хранилище проекта.

2) *Modified* – файл был изменен в рабочей копии проекта (куда данны выгружаются из репозитория для их изменения), но новая версия еще не была загружена в локальный репозиторий.

3) *Staged или подготовленном* (другое наименование – добавлен в индекс, индексирован) – новая версия файла еще не добавлена в локальный репозиторий, но проиндексирована, чтобы быть добавленной при следующем выполнении команды commit.

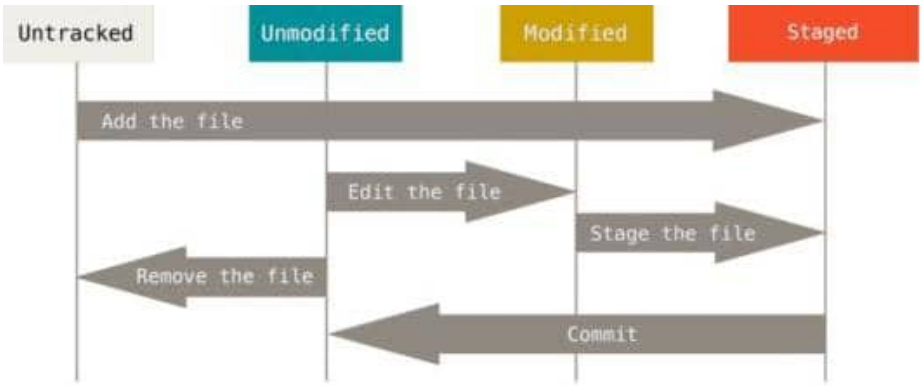


Рисунок 3 - Жизненный цикл состояний файлов в Git.

Таким образом, сохранение новой версии файла (а также добавление нового файла) в локальный репозиторий выполняется в два шага – сначала производится индексирование:

```
git add [имя_файла] или git add *
```

Затем непосредственно добавление в репозиторий:

```
git commit -m "Commit message"
```

Желательно при выполнении commit писать комментарий с кратким пояснением выполненных изменений.

Для создания веток используется команда:

```
git checkout -b [новая_ветка]
```

Для переключения на другую ветку:

```
git checkout [репозиторий]/[ветка]
```

Для работы с удаленными репозиторием необходимо его подключить:

```
git remote add origin [url]
```

Для отправки изменений из ветки master локального репозитория origin на удаленный сервер (в соответствующую ветку):

```
git push origin master
```

Для копирования изменений с удаленного сервера в свой локальный репозиторий:

```
git pull origin master
```

Для проверки состояния репозитория используется команда:

```
git status
```

3. Сервис хостинга проектов GitHub

GitHub – это популярный интернет-сервис для хостинга и совместной разработки проектов. GitHub позволяет размещать в открытом (бесплатно) или закрытом (платно) доступе свои проекты, осуществлять совместную разработку, легко создавать новые ветки проектов, а также скачивать код проектов не только через Git, но и просто в виде файлов.

Зарегистрироваться и создать свой открытый репозиторий можно по адресу https://github.com/.

Задания на работу

1. Установить на свой рабочий компьютер Git.
2. Выполнить команды создания локального репозитория, внести изменения в рабочую копию, сохранить их в репозиторий. После каждого из шагов проверять состояние репозитория командой git status
3. Зарегистрироваться на GitHub.
4. Создать новый репозиторий через GitHub.
5. Подключиться с рабочего компьютера к репозиторию GitHub и сохранить туда файлы своего локального репозитория или наоборот, создать копию удаленного репозитория у себя на рабочем компьютере.
6. Оформить отчет (В отчете необходимо указать URL вашего репозитория в GitHub, а также скриншоты выполнения основных действий в Git на локальном компьютере (add и commit, status, push и т.п.).

Если вы уже зарегистрированы на GitHub пункты 3-4 можно пропустить.

Контрольные вопросы

1. Что такое Git?
2. Что такое GitHub?
3. Какая команда используется для создания репозитория Git в той директории?
4. Какая команда используется для создания локальной рабочей копии локального или удаленного репозитория?
5. Какая команда используется для непосредственного добавление в репозиторий?
6. Какая команда используется для создания веток?
7. Какая команда используется для переключения на другую ветку?
8. Какая команда используется для работы с удаленными репозиторием?
9. Какая команда используется для отправки изменений из ветки master локального репозитория origin на удаленный сервер (в соответствующую ветку)?

10. Какая команда используется для копирования изменений с удаленного сервера в свой локальный репозиторий?

11. Какая команда используется для проверки состояния репозитория используется команда?

Лабораторная работа № 1
«ПОСТРОЕНИЕ ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ И ДИАГРАММЫ ПОСЛЕДОВАТЕЛЬНОСТИ»

Цель работы: изучение основ моделирования предметной области с использованием возможностей диаграмм (Вариантов использования и Последовательности) языка моделирования UML.

Теоретические сведения

1. Краткие сведения о диаграмме вариантов использования.

Диаграмма вариантов использования является самым общим представлением функциональных требований к системе. Для последующего проектирования системы требуются более конкретные детали, которые описываются в документе, называемом *сценарием варианта использования* или *потоком событий (flow of events).* Сценарий подробно документирует процесс взаимодействия действующего лица с системой, реализуемого в рамках варианта использования. Основной поток событий описывает нормальный ход событий (при отсутствии ошибок). Альтернативные потоки описывают отклонения от нормального хода событий (ошибочные ситуации) и их обработку.

Достоинства модели вариантов использования заключаются в том, что она:

- определяет пользователей и границы системы;
- определяет системный интерфейс;
- удобна для общения пользователей с разработчиками;
- используется для написания тестов;
- является основой для написания пользовательской документации;
- хорошо вписывается в любые методы проектирования (как объектно-ориентированные, так и структурные)

Основные элементы диаграмм вариантов использования

Активный субъект (actor) отождествляется с чем-то или с кем-то, взаимодействующим с системой, т.е. играет определённую роль по отношению к системе, это может быть не обязательно пользователь будущей системы, также это может быть внешняя система.

Варианты использования (use cases) позволяют моделировать диалог между активным субъектом и системой и отображают функции системы. С каждым вариантом использования связан определенный поток событий, происходящих по мере выполнения соответствующих функций системы. При описании потока событий определяется, что необходимо осуществить, и игнорируются аспекты того, как это делается.

Между активным субъектом и вариантом использования устанавливаются связь *ассоциация (association relationship*), которая выполняет коммуникативную функцию, сообщая о взаимодействии субъекта с системой в рамках определенного варианта использования. Направление связи указывает, кто (субъект или система) является инициатором взаимодействия.

Помимо связей между субъектом и вариантом использования, связи могут устанавливаться и между вариантами использования.

Связи бывают двух типов - *включающими (inclusive*) и *расширяющими (extensive).*

Порядок построения Usecase Diagram:

1. Создать usecase диаграмму с именем «Основная функциональность»
2. Проанализировать, какие активные субъекты должны взаимодействовать с будущей системой.
3. Создать actor'ов. (*Например, Менеджер, Бухгалтер и Кладовщик).*
4. Создать прецеденты. Например,
   - Оформление заказа.
   - Оформление счёта.
   - Оформление накладной.
   - Выдача товара.
5. Для пояснения можно использовать комментарии.
6. Расставить связи, обозначающие зависимость (необходимо продумать, какие прецеденты находятся в отношении зависимости).
7. Результатом является подобная диаграмма:

Рисунок 1 – Пример диаграммы *вариантов использования.*

3. Диаграмма последовательности

Диаграмма последовательности служит для выявления классов и определения наборов их методов, а так же описания их взаимодействия в процессе выполнения того или иного действия.

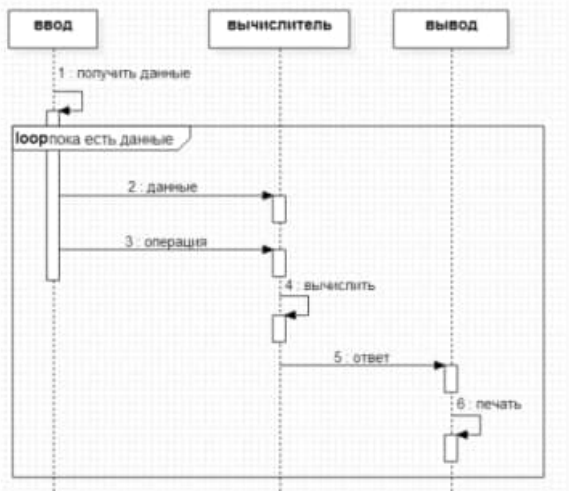


Рисунок 2 – Пример диаграммы последовательности

На рисунке приведён пример описания процесса работы простого вычислителя арифметических выражений при помощи диаграммы последовательностей.

В самом верху диаграммы изображены задействованные в процессе классы. От них, пунктиром, отмечены их линии жизни.

Прямоугольником на линии жизни объекта обозначается фокус управления. Фокус управления обозначает этап активной работы объекта.

Стрелками указаны сообщения посылаемые объектом. Стрелки, указывающие сами на себя, обозначают методы, меняющие состояние объекта. Стрелки, ведущие к линиям жизни других объектов, обозначают вызов методов этих объектов с некоторыми параметрами.

Рамка с подписью “loop” в левом верхнем углу, называется комбинированным фрагментом и обозначает, что действия в её границах выполняются циклично, до тех пор пока не кончатся данные поступившие на вход.

Существуют следующие виды фреймов взаимодействия:

Alt – условие. Содержимое фрагмента выполняется, только если условие истинно.

Loop – цикл. Содержимое фрагмента будет выполняться, пока не выполнено условие.

Neg – ошибка. Выполняется в случае возникновения ошибки в последовательности.

Opt – условие без альтернативного варианта. Аналогично Alt, но без варианта else.

Par – параллелизм. Фрагменты выполняются (могут выполняться) одновременно.

Ref – ссылка. Ссылка на последовательность, описанную в других диаграммах.

Region – критическая секция. Описывает ограничения действующие на время выполнения фрагмента.

Sd – описание. Может охватывать всю диаграмму и содержать некие комментарии.

Следует помнить, что диаграмма последовательности описывать набор классов и способы их взаимодействия, не делая акцент на деталях их реализации.

Задания на работу

- Предметная область для моделирования и создания UML диаграмм по варианту индивидуального задания из практических работ №№ 1-3.
- Для выполнения лабораторной работы может быть использован любой другой редактор UML диаграмм.

1. Создать диаграмму Вариантов использования.
2. Проанализировать, какие активные субъекты должны взаимодействовать с будущей системой.
3. Создать actor’ов. Создать прецеденты. (Для пояснения можно использовать комментарии).
4. Расставить связи, обозначающие зависимость (необходимо продумать, какие прецеденты находятся в отношении зависимости).
5. Создать Диаграмму последовательности (осуществить выявление классов и определения наборов их методов, а так же описания их взаимодействия в процессе выполнения того или иного действия).
6. Оформить отчет.

Контрольные вопросы

1. Основное назначение Диаграммы Вариантов использования?
2. Основное назначение Диаграммы последовательности?

Лабораторная работа № 2
«ПОСТРОЕНИЕ ДИАГРАММЫ КООПЕРАЦИИ И ДИАГРАММЫ РАЗВЕРТЫВАНИЯ»

Цель работы: изучение основ моделирования предметной области с использованием возможностей диаграмм (кооперации и развертывания) языка моделирования UML.

Теоретические сведения

1. Диаграмма кооперации (Collaboration diagram)

Диаграмма кооперации (Collaboration diagram) - диаграмма поведения, на которой показано взаимодействие и подчеркнута структурная организация объектов, посылающих и принимающих сообщения.

В качестве примера рассмотрим построение диаграммы кооперации для моделирования процесса телефонного разговора с использованием обычной телефонной сети (Объектами в этом примере являются два абонента а и Ь, два телефонных аппарата с и </, коммутатор и сам разговор как объект моделирования. При этом как коммутатор, так и разговор являются анонимными объектами.)

На начальном этапе изобразим все объекты и связи между ними на диаграмме кооперации при помощи соответствующих обозначений (Рисунок 1). Заметим, что первый телефонный аппарат изображен как активный объект, а второй – как пассивный.

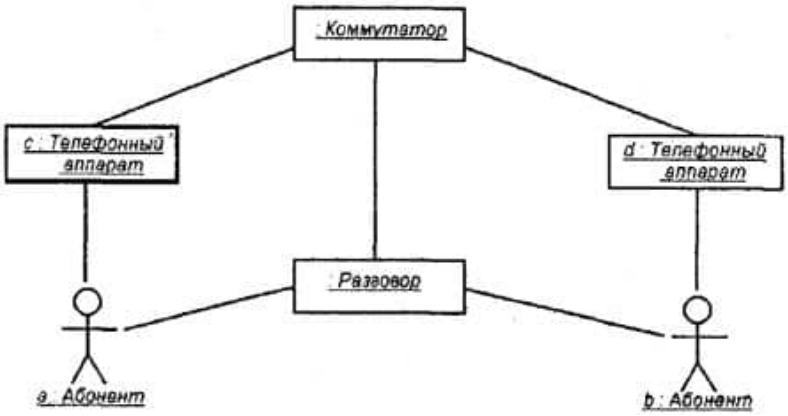


Рисунок 1 - Начальный фрагмент диаграммы кооперации
для примера моделирования обычного телефонного разговора.

В последующем необходимо специфицировать все связи на этой диаграмме, указав на их концах необходимую информацию в форме ролей связей. Дополненный таким образом вариант диаграммы кооперации изображен ниже (Рисунок 2). Заметим, что для объекта «Разговор» указано помеченное значение {transient}, которое означает, что этот объект создается в процессе выполнения объемлющего процесса и уничтожается до его завершения. Напомним, что помеченные значения (tagged values) являются стандартными элементами языка UML.

Рисунок 2 - Фрагмент диаграммы кооперации, дополненный стереотипами ролей связей, именами ассоциаций и помеченным значением объекта.

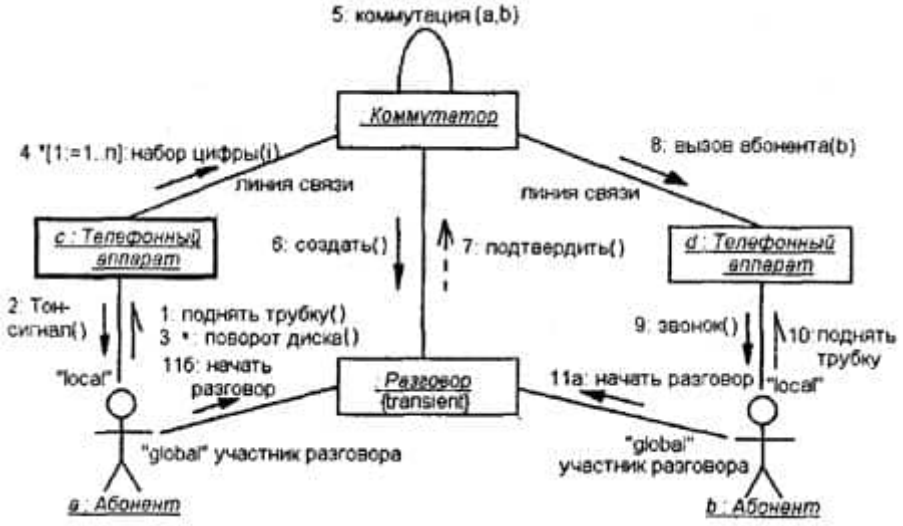


Рисунок 3 - Окончательный вариант диаграммы кооперации для моделирования телефонного разговора.

Наконец, на диаграмму кооперации необходимо нанести все сообщения, указав их порядок и семантические особенности. Окончательный фрагмент диаграммы кооперации изображен на Рисунок 3 и содержит, строго говоря, модель кооперации только для начала разговора. Эта диаграмма может быть дополнена сообщениями, необходимыми для окончания разговора, что читателям предлагается выполнить самостоятельно в качестве упражнения.

Как нетрудно заметить, диаграмма кооперации для примера с телефонным разговором не содержит ни временных особенностей передачи сообщений, ни особенностей жизненного цикла участвующих в данной кооперации объектов. Поэтому может быть принято решение о том, что она является избыточной при наличии построенной диаграммы последовательности. Этот факт не вызывает сомнений в тех случаях, когда структура взаимодействующих объектов является достаточно тривиальной.

Если же взаимодействующие объекты образуют между собой различные типы отношений-ассоциаций (композиция, агрегация), то диаграмма кооперации оказывается необходимым представлением модели на всех ее уровнях.

2. Диаграмма развертывания (Deployment diagram)

Диаграмма развертывания (Deployment diagram) - структурная диаграмма, на которой показаны узлы и отношения между ними.

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Конечно, если разрабатывается простая программа, которая может выполняться локально на компьютере пользователя, не задействуя никаких периферийных устройств и ресурсов, то в этом случае нет необходимости в разработке дополнительных диаграмм. Однако при разработке корпоративных приложений ситуация представляется совсем по-другому.

Во-первых, сложные программные системы могут реализовываться в сетевом варианте на различных вычислительных платформах и технологиях доступа к распределенным базам данных. Во-вторых, интеграция программной системы с Интернетом определяет необходимость решения дополнительных вопросов при проектировании системы, таких как обеспечение безопасности, криптозащищенности и устойчивости доступа к информации для корпоративных клиентов. Эти аспекты в немалой степени зависят от реализации проекта в форме физически существующих узлов системы, таких как серверы, рабочие станции, брандмауэры, каналы связи и хранилища данных.

Наконец, технологии доступа и манипулирования данными в рамках общей схемы "клиент-сервер" также требуют размещения больших баз данных в различных сегментах корпоративной сети, их резервного копирования, архивирования, кэширования для обеспечения необходимой производительности системы в целом. Эти аспекты также требуют визуального представления с целью спецификации программных и технологических особенностей реализации распределенных архитектур.

Первой из диаграмм физического представления является диаграмма компонентов. Второй формой физического представления программной системы является диаграмма развертывания (синоним - диаграмма размещения). Она применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений - маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации. Эта диаграмма, по сути, завершает процесс ООАП для конкретной программной системы и ее разработка, как правило, является последним этапом спецификации модели.

Разработка диаграммы развертывания преследует следующие цели:

- Определить распределение компонентов системы по ее физическим узлам.
- Показать физические связи между всеми узлами реализации системы на этапе ее исполнения.
- Выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Для обеспечения этих требований диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками. Далее рассмотрим отдельные элементы, из которых состоят диаграммы развертывания.

Узел (node) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом. В качестве вычислительного ресурса узла может рассматриваться наличие по меньшей мере некоторого объема электронной или магнитооптической памяти и/или процессора. В последней версии языка UML понятие узла расширено и может включать в себя не только вычислительные устройства (процессоры), но и другие механические или электронные устройства, такие как датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы.

Возможность включения людей (персонала) в понятие узла позволяет создавать средствами языка UML модели самых различных систем, включая бизнес-процессы и технические комплексы. Действительно, реализация бизнес-логики предприятия требует рассматривать в качестве узлов системы организационные подразделения, состоящие из персонала. Автоматизация управления техническими комплексами также требует рассмотрения в качестве самостоятельного элемента человека-оператора, способного принимать решения в нештатных ситуациях и нести ответственность за возможные последствия этих решений.

Графически на диаграмме развертывания узел изображается в форме трехмерного куба. Узел имеет собственное имя, которое указывается внутри этого графического символа. Сами узлы могут представляться как в качестве типов (Рисунок 1, а), так и в качестве экземпляров (Рисунок 1, б).

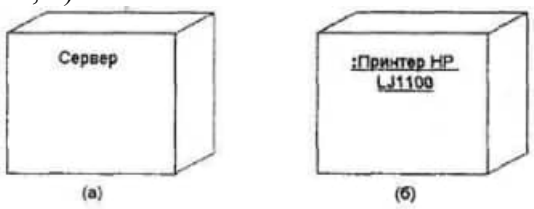


Рисунок 1 - Графическое изображение узла на диаграмме развертывания

В первом случае имя узла записывается без подчеркивания и начинается с заглавной буквы. Во втором имя узла-экземпляра записывается в виде:

<имя узла ':' имя типа узла>.

Имя типа узла указывает на некоторую разновидность узлов, присутствующих в модели системы.

Например, аппаратная часть системы может состоять из нескольких персональных компьютеров, каждый из которых соответствует отдельному узлу-экземпляру в модели. Однако все эти узлы-экземпляры относятся к одному типу узлов, а именно узлу с именем типа "Персональный компьютер". Так, на представленном выше рисунке (Рисунок 1, а) узел с именем "Сервер" относится к общему типу и никак не конкретизируется. Второй же узел (рис.1, б) является анонимным узлом-экземпляром конкретной модели принтера.

Так же, как и на диаграмме компонентов, изображения узлов могут расширяться, чтобы включить некоторую дополнительную информацию о спецификации узла. Если дополнительная информация относится к имени узла, то она записывается под этим именем в форме помеченного значения (рис.2).

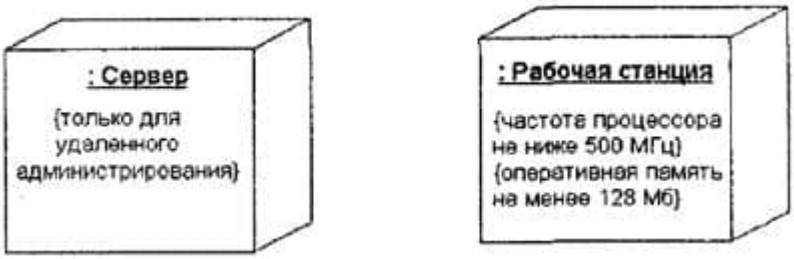


Рисунок 2 - Графическое изображение узла-экземпляра с дополнительной информацией в форме помеченного значения.

Если необходимо явно указать компоненты, которые размещаются на отдельном узле, то это можно сделать двумя способами. Первый из них позволяет разделить графический символ узла на две секции горизонтальной линией. В верхней секции записывают имя узла, а в нижней секции - размещенные на этом узле компоненты (рис.3, а).

Второй способ разрешает показывать на диаграмме развертывания узлы с вложенными изображениями компонентов (рис.3, б). Важно помнить, что в качестве таких вложенных компонентов могут выступать только исполняемые компоненты.

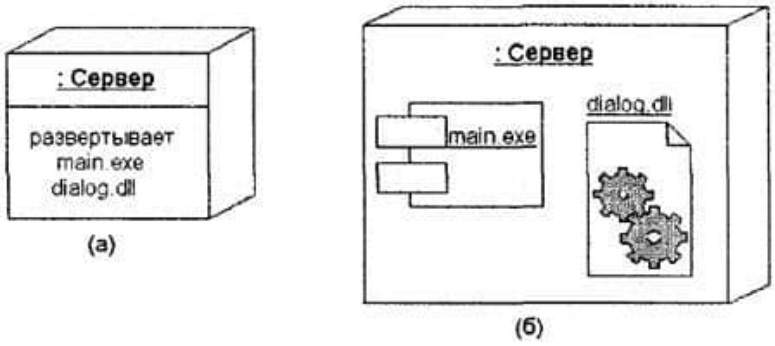


Рисунок 3 - Варианты графического изображения узлов-экземпляров с размещаемыми на них компонентами.

В качестве дополнения к имени узла могут использоваться различные стереотипы, которые явно специфицируют назначение этого узла. Хотя в языке UML стереотипы для узлов не определены, в литературе встречаются следующие их варианты: "процессор", "датчик", "модем", "сеть", "консоль" и др., которые самостоятельно могут быть определены разработчиком. Более того, на диаграммах развертывания допускаются специальные обозначения для различных физических устройств, графическое изображение которых проясняет назначение или выполняемые устройством функции.

Соединения. Кроме собственно изображений узлов на диаграмме развертывания указываются отношения между ними. В качестве отношений выступают физические соединения между узлами и зависимости между узлами и компонентами, изображения которых тоже могут присутствовать на диаграммах развертывания.

Соединения являются разновидностью ассоциации и изображаются отрезками линий без стрелок. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между соответствующими узлами. Характер соединения может быть дополнительно специфицирован примечанием, помеченным значением или ограничением. Так, на представленном ниже фрагменте диаграммы развертывания (Рисунок 4) явно определены не только требования к скорости передачи данных в локальной сети с помощью помеченного значения, но и рекомендации по технологии физической реализации соединений в форме примечания.

Рисунок 4 - Фрагмент диаграммы развертывания с соединениями между узлами.

Кроме соединений на диаграмме развертывания могут присутствовать отношения зависимости между узлом и развернутыми на нем компонентами. Подобный способ является альтернативой вложенному изображению компонентов внутри символа узла, что не всегда удобно, поскольку делает этот символ излишне объемным. Поэтому при большом количестве развернутых на, узле компонентов соответствующую информацию можно представить в форме отношения зависимости (Рисунок 5).

Диаграммы развертывания могут иметь более сложную структуру, включающую вложенные компоненты, интерфейсы и другие аппаратные устройства. На изображенной ниже диаграмме развертывания (Рисунок 6) представлен фрагмент физического представления системы удаленного обслуживания клиентов банка. Узлами этой системы являются удаленный терминал (узел-тип) и сервер банка (узел-экземпляр).

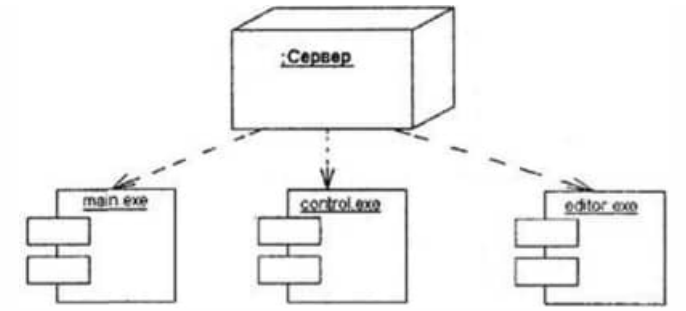


Рисунок 5 - Диаграмма развертывания с отношением зависимости между узлом и развернутыми на нем компонентами.

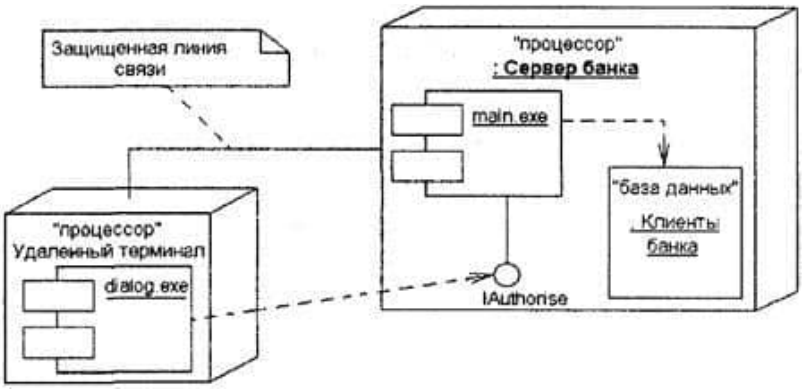


Рисунок 6 - Диаграмма развертывания для системы удаленного обслуживания клиентов банка.

На этой диаграмме развертывания указана зависимость компонента реализации диалога (как пример!)"dialog.exe" на удаленном терминале от интерфейса lAuthorise, реализованного компонентом "main.exe", который, в свою очередь, развернут на анонимном узле-экземпляре "Сервер банка". Последний зависит от компонента базы данных "Клиенты банка", который развернут на этом же узле.

Примечание указывает на необходимость использования защищенной линии связи для обмена данными в данной системе. Другой вариант записи этой информации заключается в дополнении диаграммы узлом со стереотипом "закрытая сеть".

Разработка так называемых встроенных систем предполагает не только создание программного кода, но и согласование между собой всех аппаратных средств и механических устройств. В качестве примера рассмотрим фрагмент модели управления удаленным механическим средством типа транспортной платформы. Такая платформа предназначена для перемещения в агрессивных средах, где присутствие человека невозможно в силу целого ряда физических причин.

Транспортная платформа оснащается собственным микропроцессором, цифровой видеокамерой, датчиками температуры и местоположения, а также управляющими приводами для изменения направления и скорости перемещения платформы. Управляющая и телеметрическая информация от платформы по радиолинии передается в центр управления, оснащенный управляющим компьютером, манипуляторами управления и большим информационным табло.

На микропроцессоре платформы развернуты программные компоненты для реализации простейших управляющих воздействий на приводы, что позволяет дискретно изменять направление и скорость перемещения платформы. На компьютере центра управления развернуты программные компоненты анализа телеметрической информации, характеризующей состояние отдельных устройств' платформы, а также реализованы алгоритмы управления перемещением платформы в целом.

Вариант физического представления этой транспортной системы показан на следующей диаграмме развертывания (Рисунок 7).

Данная диаграмма содержит самую общую информацию о развертывании рассматриваемой системы и в последующем может быть детализирована при разработке собственно программных компонентов управления. Как видно из рисунка, при разработке этой диаграммы развертывания использованы дополнительный стереотип "приемопередатчик", который отсутствует в описании языка UML, и специальные изображения для отдельных аппаратных и механических устройств.

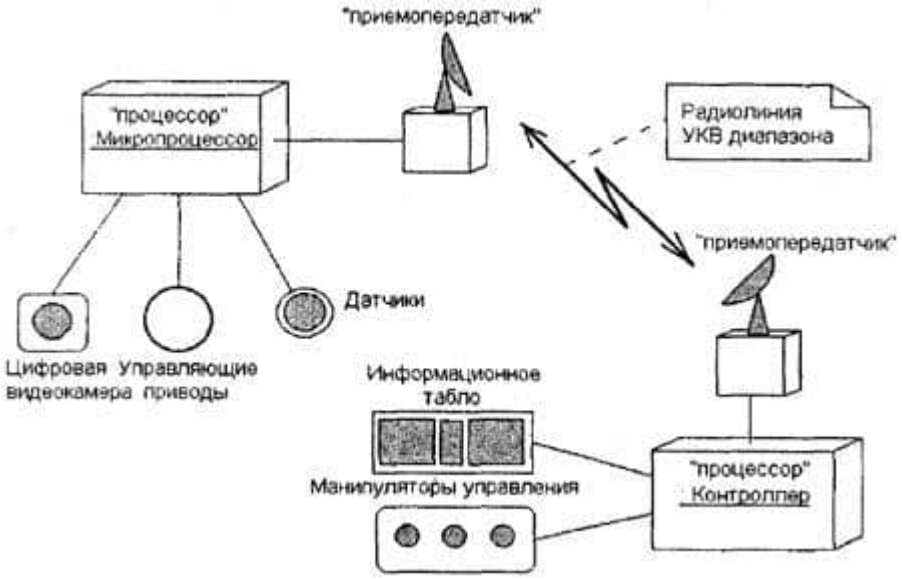


Рисунок 7 - Диаграмма развертывания для модели системы управления транспортной платформой.

Задания на работу

- Предметная область для моделирования и создания UML диаграмм по варианту индивидуального задания из практических работ №№ 1-3 и на основании лабораторной работы №1.
- Для выполнения лабораторной работы может быть использован любой другой редактор UML диаграмм.

1. Создать диаграмму кооперации (Collaboration diagram).
2. Создать диаграмму развертывания (Deployment diagram).
3. Оформить отчет.

Контрольные вопросы

1. Основное назначение Диаграммы кооперации (Collaboration diagram)?
2. Основное назначение диаграммы развертывания (Deployment diagram)?

Лабораторная работа № 3
«ПОСТРОЕНИЕ ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ, ДИАГРАММЫ СОСТОЯНИЙ И ДИАГРАММЫ КЛАССОВ»

Цель работы: изучение основ моделирования предметной области с использованием возможностей диаграмм (деятельности, состояний и классов) языка моделирования UML.

Теоретические сведения

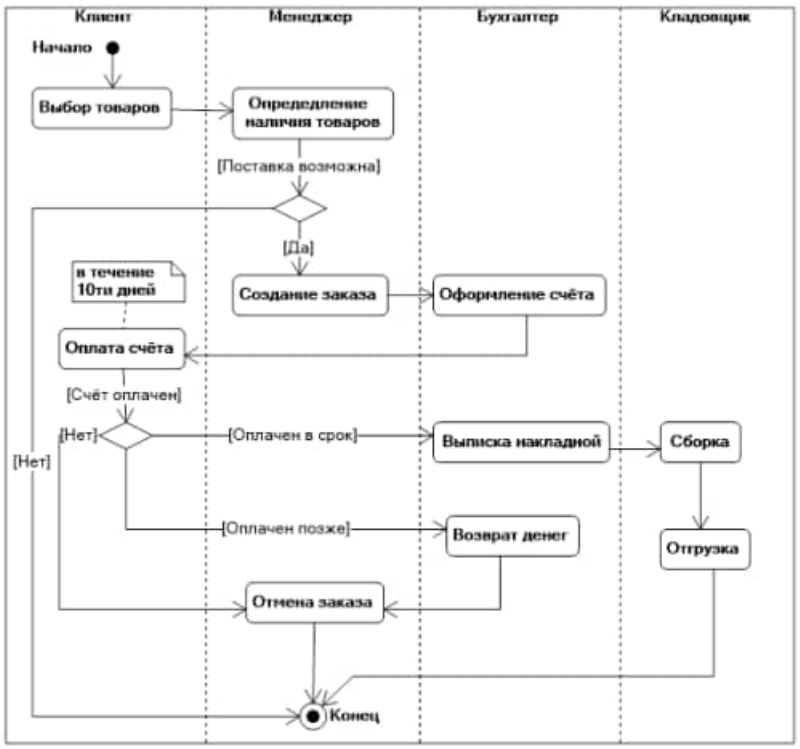
1. *Диаграмма деятельности.*

Диаграмма видов деятельности, как и диаграмма состояний, отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

В качестве примера предлагается рассмотреть укрупненную диаграмму деятельности для описания процесса реализации продукции клиенту. Это позволит лучше разобраться в происходящих действиях.

Порядок построения диаграммы видов деятельности:

1. Создать диаграмму видов деятельности.
2. Например (начало рассмотрено в лабораторной работе №1), в бизнес-процессе участвуют четыре объекта: клиент, менеджер, бухгалтер и кладовщик. Следует создать дорожки (swimlanes) для каждого из объектов. Каждая из дорожек ответственна за выполнение определенных действий тем объектом, с которым она проассоциирована.
3. Далее необходимо расположить на диаграмме все деятельности/действия, которые выполняются тем или иным объектом.
4. Результатом является диаграмма. Например, представленная ниже:



2. *Диаграмма состояний.*

Диаграмма состояний определяет последовательность состояний объекта, вызванных последовательностью событий.

Порядок построения диаграммы:

1. Создать диаграмму состояний. Например, для объектов класса «Заказ».
2. Из спецификации прецедентов следует, что заказ может быть в трех состояниях:

- «Новый»;
- «Оплаченный»;
- «Отмененный».

В состояние «Новый» заказ попадает сразу после своего создания и находится в нем до момента перевода его менеджером в состояние «Оплаченный». Событием к переходу является поступление денег в кассу. Условие перехода – оплата должна производиться не позднее 10 дней со дня оформления заказа. В случае если оплата не производится в течение отведенных 10 дней или производится позже, заказ переходит в состояние «Отмененный». Соответствующая диаграмма состояний представлена на рисунке 1:

Рисунок 1 – Диаграмма состояний для объектов класса «Заказ».

3.Создать диаграмму состояний для объектов следующего класса. Например, «Накладная».

4.Построить диаграмму состояний, например, для товарно-транспортной накладной. Все вновь созданные накладные попадают в состояние «Новая». После печати накладной она переходит в состояние «Выписанная». В этот момент электронная накладная становится доступной кладовщику на складе, и он начинает сборку заказа. По окончании сборки кладовщик переводит накладную в состояние «Готовая». Если по каким-то причинам на складе не оказалось нужного товара (брак в партии, просрочка поставщика и т.п.), что делает невозможным сборку заказа, накладная переходит в состояние «Приостановленная». После того как товар отгружен клиенту, накладная переходит в состояние «Отгруженная». Диаграмма состояний для накладной изображена на рисунке 2:

Рисунок 2 – Диаграмма состояний для объектов класса «Накладная».

3. Диаграмма классов.

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними.

На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. Вид и интерпретация диаграммы классов существенно зависит от точки зрения (уровня абстракции): классы могут представлять сущности предметной области (в процессе анализа) или элементы программной системы (в процессах проектирования и реализации).

Рисунок 1 - Основные элементы диаграммы классов.

Основными элементами являются классы и связи между ними. Классы характеризуются при помощи атрибутов и операций.

Атрибуты описывают свойства объектов класса. Большинство объектов в классе получают свою индивидуальность из-за различий в их атрибутах и взаимосвязи с другими объектами. Однако, возможны объекты с идентичными значениями атрибутов и взаимосвязей. Т.е. индивидуальность объектов определяется самим фактом их существования, а не различиями в их свойствах. Имя атрибута должно быть уникально в пределах класса. За именем атрибута может следовать его тип и значение по умолчанию.

Операция есть функция или преобразование. Операция может иметь параметры и возвращать значения.

Виды связей:

- ассоциация
- агрегация
- наследование.

Ассоциация (association) – представляет собой отношения между экземплярами классов.

Каждый конец ассоциации обладает кратностью (синоним – мощностью, ориг. — multiplicity), которая показывает, сколько объектов, расположенных с соответствующего конца ассоциации, может участвовать в данном отношении. В примере на рисунке каждый Товар имеет *сколь угодно* Записей в накладной, но каждая Запись в накладной обязательно *один* Товар. В общем случае кратность может быть задана любым множеством.

Ассоциации может быть присвоено имя. В качестве имени обычно выбирается глагол или глагольное словосочетание, сообщающие смысл и назначение связи. Также на концах ассоциации под кратностью может указываться имя роли, т.е. какую роль выполняют объекты, находящиеся с данного конца ассоциации.

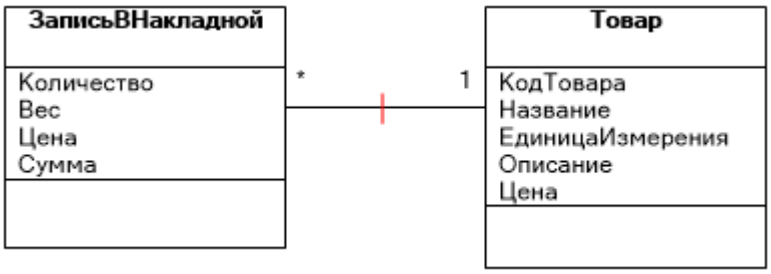


Рисунок 2 – Ассоциации.

Агрегация (aggregation) – это ассоциация типа «целое-часть». Агрегация в UML представляется в виде прямой с ромбом на конце. Ромб на связи указывает, какой класс является агрегирующим (т.е. «состоящим из»); класс с противоположного конца — агрегированным (т.е. те самые «части»).

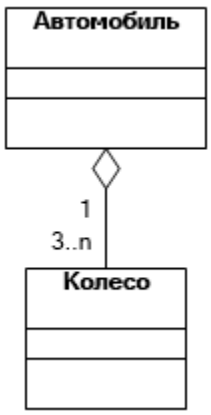


Рисунок 3 – Агрегация.

Композиция (composition) – это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Композиция изображается так же, как ассоциация, только ромбик закрашен. Важно понимать разницу между агрегацией и композицией: при агрегации объекты-части могут существовать сами по себе, а при композиции — нет. Пример агрегации: автомобиль—колесо, пример композиции: дом—комната.

Рисунок 4 – Композиция.

Наследование (inheritance) – это отношение типа «общее-частное». Позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой ряда других классов. При создании производного класса на основе базового (одного или нескольких) возникает иерархия наследования. Реализация принципов наследования является ключевой предпосылкой возможности повторного использования кода, поскольку это основной инструмент достижения полиморфизма.

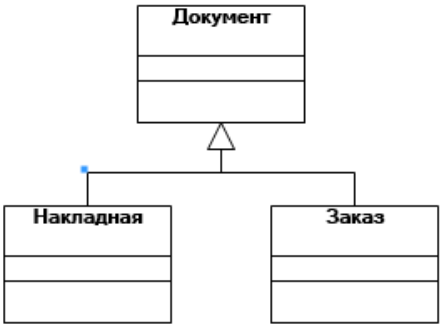
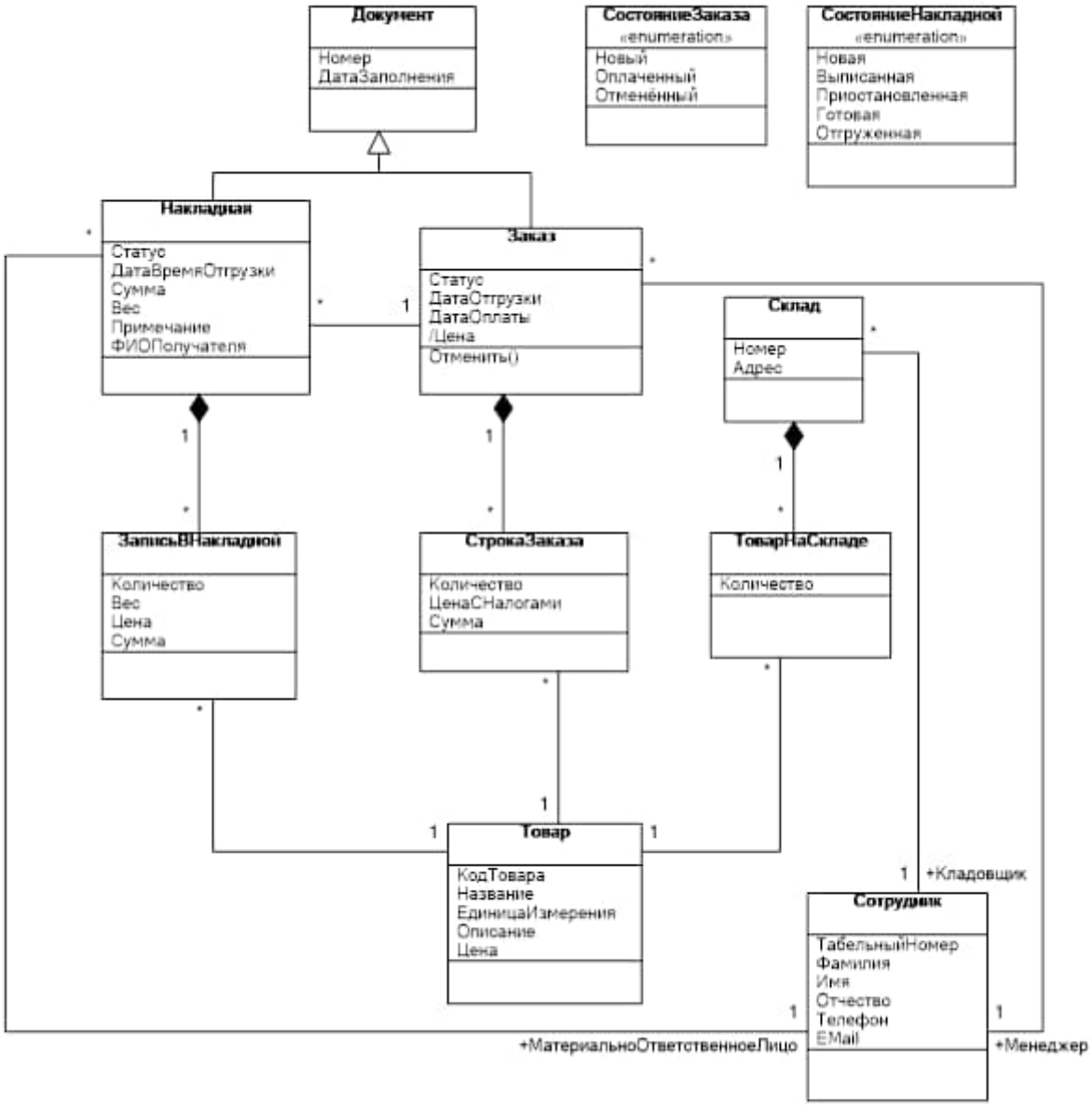


Рисунок 5 – Наследование.

Порядок построения диаграммы классов:

1. Создать новую диаграмму с именем «Сущности».
2. Проанализировать предметную область и построить диаграмму классов. Должна получиться диаграмма, подобная примеру:



Основной сущностью в системе будет являться товар. Как известно из задания на проектирование, товар хранится на складе. Но понятия товара как некоего описания и товара, лежащего непосредственно на складе, отличаются друг от друга. Товар, лежащий на складе, кроме того, что связан со складом отношением композиции (агрегация не совсем подходит, поскольку в данной системе товар является товаром, пока он не покинет склад), ещё характеризуется количеством. Аналогично следует рассуждать и при рассмотрении отношения Товара и Заказа, Товара и Накладной. В связи с тем, что Заказ и Накладная в сущности являются документами и имеют сходные атрибуты, они были объединены с помощью общего класса-предка Документ. Примечательно, что на диаграмме представлены два класса со стереотипом Enumeration (перечисление). Стереотип можно установить из контекстного меню для класса.

Задания на работу

- Предметная область для моделирования и создания UML диаграмм по варианту индивидуального задания из практических работ №№ 1-3 и на основании лабораторных работ №№1,2.

- Для выполнения лабораторной работы может быть использован любой другой редактор UML диаграмм.

1. Создать диаграмму деятельности.
2. Создать диаграмму состояний.
3. Создать диаграмму классов.
4. Оформить отчет.

Контрольные вопросы

1. Основное назначение диаграммы деятельности?
2. Основное назначение диаграммы состояний?
3. Основное назначение диаграммы классов?

Лабораторная работа № 4
«ПОСТРОЕНИЕ ДИАГРАММ КОМПОНЕНТОВ И ПОТОКОВ ДАННЫХ. ПОСТРОЕНИЕ IDEF ДИАГРАММ»

Цель работы: изучение основ моделирования предметной области с использованием возможностей диаграмм (компонентов, потоков данных) языка моделирования UML; IDEF диаграмм.

Теоретические сведения

1. Диаграмма компонентов (Component diagram)

Диаграмма компонентов (Component diagram) - диаграмма поведения, на которой показан автомат и подчеркнуто поведение объектов с точки зрения порядка получения событий.

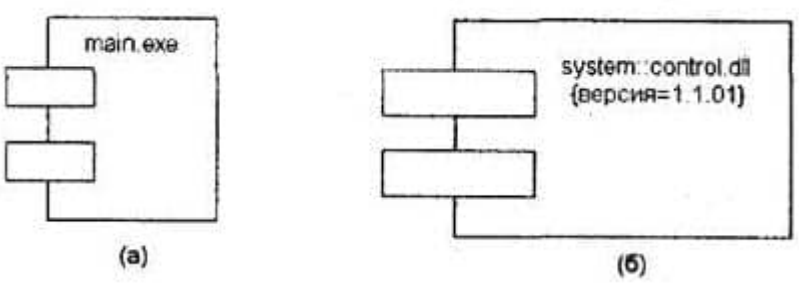
Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Диаграмма компонентов разрабатывается для следующих целей:

- визуализации общей структуры исходного кода программной системы;
- спецификации исполнимого варианта программной системы;
- обеспечения многократного использования отдельных фрагментов программного кода;
- представления концептуальной и физической схем баз данных.

Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода. Одни компоненты могут существовать только на этапе компиляции программного кода, другие — на этапе его исполнения. Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

Для представления физических сущностей в языке UML применяется специальный термин – компонент (component). Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели. Для графического представления компонента может использоваться специальный символ – прямоугольник со вставленными слева двумя более мелкими прямоугольниками (рис.1). Внутри объемлющего прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация. Изображение этого символа может незначительно варьироваться в зависимости от характера ассоциируемой с компонентом информации. В метамодели языка UML компонент является потомком классификатора. Он предоставляет организацию в рамках физического пакета ассоциированным с ним элементам модели. Как классификатор, компонент может иметь также свои собственные свойства, такие как атрибуты и операции.



(а) (б)

Рисунок 1 - Графическое изображение компонента в языке UML.

Так, в первом случае (Рисунок 1, а) с компонентом уровня экземпляра связывается только его имя, а во втором (Рисунок 1, б) — дополнительно имя пакета и помеченное значение.

В языке UML выделяют три вида компонентов.

1) компоненты развертывания, которые обеспечивают непосредственное выполнение системой своих функций (такими компонентами могут быть динамически подключаемые библиотеки с расширением dll (Рисунок 2, а), Web-страницы на языке разметки гипертекста с расширением html (Рисунок 2, б) и файлы справки с расширением hlp (Рисунок 2, в)).

2) компоненты-рабочие продукты (обычно это файлы с исходными текстами программ, например, с расширениями h или cpp для языка C++ (Рисунок 10.2, г)).

3) компоненты исполнения, представляющие исполнимые модули — файлы с расширением exe.

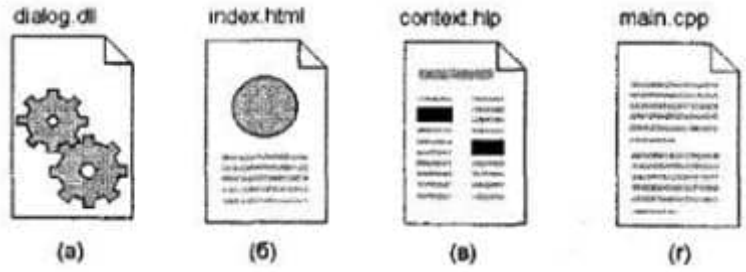


Рисунок 2 - Варианты изображения компонентов на диаграмме компонентов.

Другой способ спецификации различных видов компонентов — явное указание стереотипа компонента перед его именем. В языке UML для компонентов определены следующие стереотипы:

- библиотека (library) – определяет первую разновидность компонента, который представляется в форме динамической или статической библиотеки;
- таблица (table) – также определяет первую разновидность компонента, который представляется в форме таблицы базы данных;
- файл (file) – определяет вторую разновидность компонента, который представляется в виде файлов с исходными текстами программ;
- документ (document) – определяет вторую разновидность компонента, который представляется в форме документа;
- исполнимый (executable) – определяет третий вид компонента, который может исполняться в узле.

Следующим элементом диаграммы компонентов являются интерфейсы. В общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок (рис.3, а). При этом имя интерфейса, которое обязательно должно начинаться с заглавной буквы "I", записывается рядом с окружностью. Семантически линия означает реализацию интерфейса, а наличие интерфейсов у компонента означает, что данный компонент реализует соответствующий набор интерфейсов.

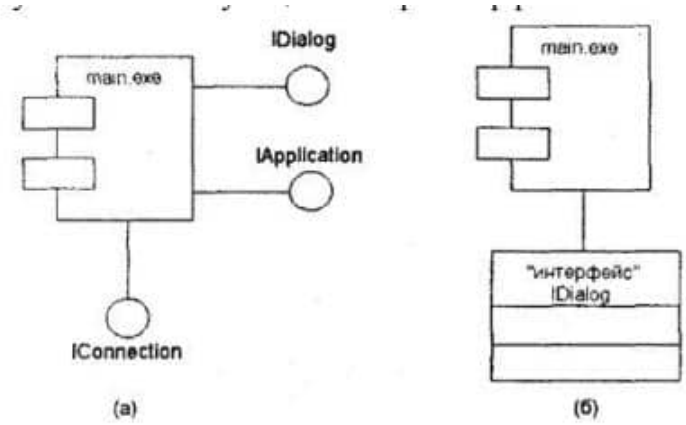


Рисунок 3 - Графическое изображение интерфейсов на диаграмме компонентов

Другим способом представления интерфейса на диаграмме компонентов является его изображение в виде прямоугольника класса со стереотипом «интерфейс» и возможными секциями атрибутов и операций (Рисунок 3, б). Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

При разработке программных систем интерфейсы обеспечивают не только совместимость различных версий, но и возможность вносить существенные изменения в одни части программы, не изменяя другие ее части. Таким образом, назначение интерфейсов существенно шире, чем спецификация взаимодействия с пользователями системы (актерами).

Зависимости могут отражать связи модулей программы на этапе компиляции и генерации объектного кода. В другом случае зависимость может отражать наличие в независимом компоненте описаний классов, которые используются в зависимом компоненте для создания соответствующих объектов. Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

В первом случае рисуют стрелку от компонента-клиента к импортируемому интерфейсу (Рисунок 4). Наличие такой стрелки означает, что компонент не реализует соответствующий интерфейс, а использует его в процессе своего выполнения.

Причем на этой же диаграмме может присутствовать и другой компонент, который реализует этот интерфейс. Так, например, изображенный ниже фрагмент диаграммы компонентов представляет информацию о том, что компонент с именем "main.exe" зависит от импортируемого интерфейса I Dialog, который, в свою очередь, реализуется компонентом с именем "image.java". Для второго компонента этот же интерфейс является экспортируемым.

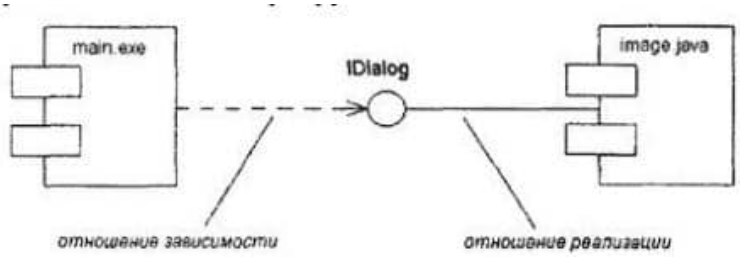


Рисунок 4 - Фрагмент диаграммы компонентов с отношением зависимости.

Другим случаем отношения зависимости на диаграмме компонентов является отношение между различными видами компонентов (Рисунок 5). Наличие подобной зависимости означает, что внесение изменений в исходные тексты программ или динамические библиотеки приводит к изменениям самого компонента. При этом характер изменений может быть отмечен дополнительно.

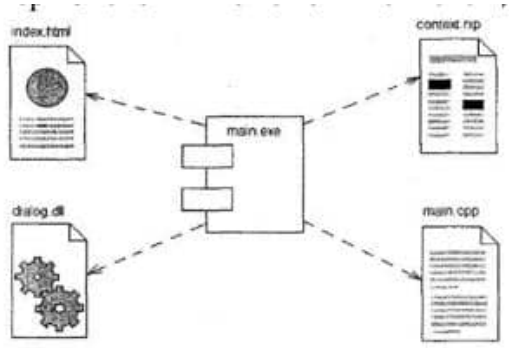


Рисунок 5 - Графическое изображение отношения зависимости между компонентами

На диаграмме компонентов могут быть представлены отношения зависимости между компонентами и реализованными в них классами. Ниже приводится фрагмент зависимости подобного рода, когда некоторый компонент зависит от соответствующих классов.

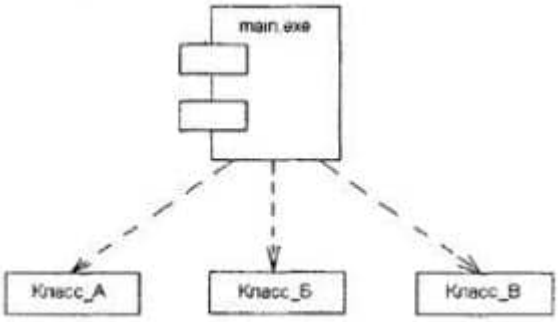


Рисунок 6 - Графическое изображение зависимости между компонентом и классами

Внутри символа компонента могут изображаться другие элементы графической нотации, такие как классы (компонент уровня типа) или объекты (компонент уровня экземпляра). В этом случае символ компонента изображается так, чтобы вместить эти дополнительные символы (Рисунок 7).

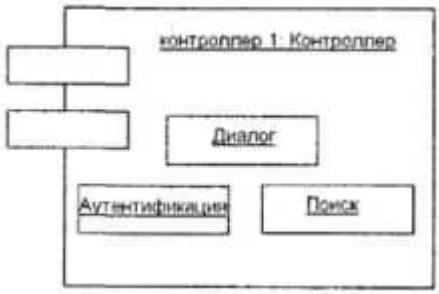


Рисунок 7 - Графическое изображение компонента уровня экземпляра, реализующего отдельные объекты

2. Диаграмма потоков данных.

Диаграммы потоков данных представляют собой информационную модель (DFD), основными компонентами которой являются различные потоки данных, которые переносят информацию от одной подсистемы к другой. Каждая из подсистем выполняет определенные преобразования входного потока данных и передает результаты обработки информации в виде потоков данных для других подсистем.

Основными компонентами диаграмм потоков данных являются: внешние сущности, накопители данных или хранилища, процессы, потоки данных, системы/подсистемы.

Внешняя сущность представляет собой материальный объект или физическое лицо, которое может выступать в качестве источника или приемника информации. Примерами внешних сущностей могут служить: клиенты организации, заказчики, персонал, поставщики. Внешняя сущность обозначается прямоугольником с тенью (рисунок 8), внутри которого указывается ее имя. При этом в качестве имени рекомендуется использовать существительное в именительном падеже.

Рисунок 8 - Изображение внешней сущности на диаграмме потоков данных

Процесс представляет собой совокупность операций по преобразованию входных потоков данных в выходные в соответствии с определенным алгоритмом или правилом. Хотя физически процесс может быть реализован различными способами, наиболее часто подразумевается программная реализация процесса. Процесс на диаграмме потоков данных изображается прямоугольником с закругленными вершинами (рисунок 9), разделенным на три секции или поля горизонтальными линиями. Поле номера процесса служит для идентификации последнего. В среднем поле указывается имя процесса. В качестве имени рекомендовано использовать глагол в неопределенной форме с необходимыми дополнениями. Нижнее поле содержит указание на способ физической реализации процесса.

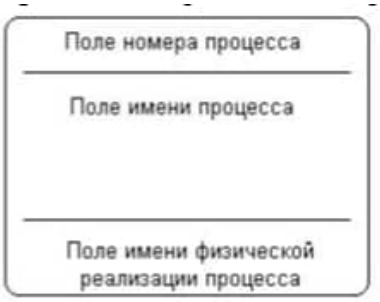


Рисунок 9 - Изображение процесса на диаграмме потоков данных

Накопитель данных или хранилище представляет собой абстрактное устройство или способ хранения информации, перемещаемой между процессами. Предполагается, что данные можно в любой момент поместить в накопитель и через некоторое время извлечь, причем физические способы помещения и извлечения данных могут быть произвольными. Накопитель данных на диаграмме потоков данных изображается прямоугольником с двумя полями (рисунок 10).

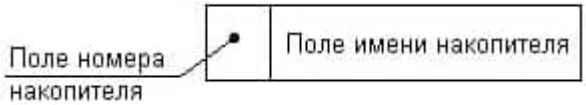


Рисунок 10 - Изображение накопителя на диаграмме потоков данных

Поток данных определяет качественный характер информации, передаваемой через некоторое соединение от источника к приемнику. Поток данных на диаграмме DFD изображается линией со стрелкой на одном из ее концов, при этом стрелка показывает направление потока данных. Каждый поток данных имеет свое собственное имя, отражающее его содержание.

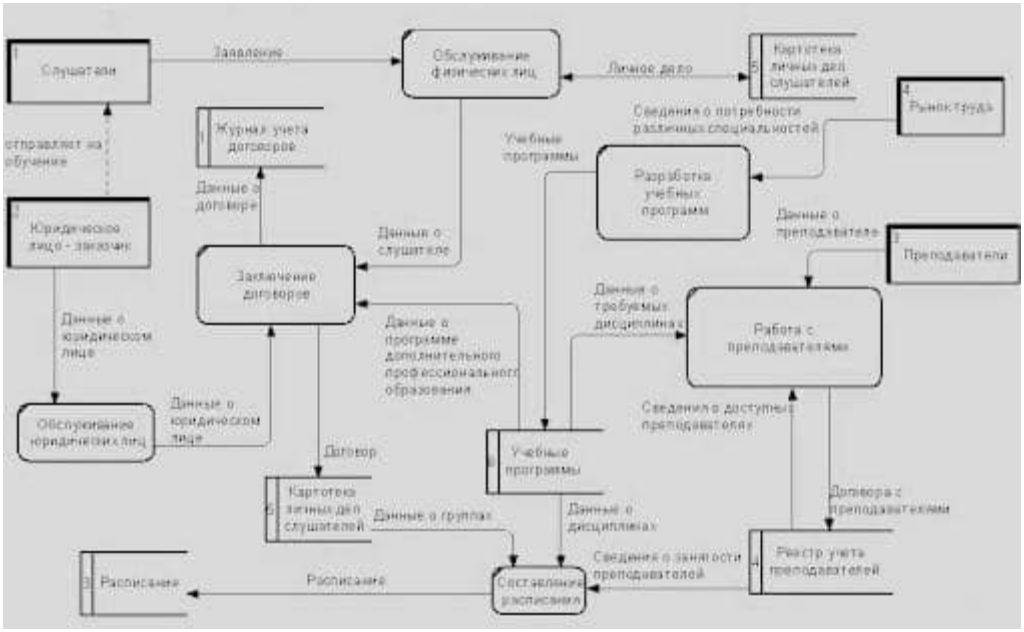


Рисунок 11 – Пример диаграммы потоков данных.

Таким образом, информационная модель системы в нотации DFD строится в виде диаграмм потоков данных, которые графически представляются с использованием соответствующей системы обозначений.

3. Диаграммы IDEF.

IDEF (I-CAM DEFinition или Integrated DEFinition) — методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем, позволяют отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах.

В настоящий момент к семейству IDEF можно отнести следующие стандарты:

IDEF0. Function Modeling (Моделирование бизнес-процесса) — методология функционального моделирования. С помощью наглядного графического языка IDEF0 изучаемая система предстаёт перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков — в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы. Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique).

IDEF1. Information Modeling — методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи.

IDEF2. Simulation Model Design — методология динамического моделирования развития систем. В связи с весьма серьёзными сложностями анализа динамических систем от этого стандарта практически отказались, и его развитие приостановилось на самом начальном этапе. В настоящее время присутствуют алгоритмы и их компьютерные реализации, позволяющие превращать набор статических диаграмм IDEF0 в динамические модели, построенные на базе «раскрашенных сетей Петри» (CPN — Color Petri Nets).

IDEF3. Process Description Capture (Документирование технологических процессов) — методология документирования процессов, происходящих в системе (например, на предприятии), описывает сценарий и последовательность операций для каждого процесса. IDEF3 имеет прямую взаимосвязь с методологией IDEF0 — каждая

функция (функциональный блок) может быть представлена в виде отдельного процесса средствами IDEF3.

IDEF4. Object-Oriented Design — методология построения объектно-ориентированных систем, позволяет отображать структуру объектов и заложенные принципы их взаимодействия и тем самым анализировать и оптимизировать сложные объектно-ориентированные системы.

IDEF5. Ontology Description Capture — Стандарт онтологического исследования сложных систем. С помощью методологии IDEF5 онтология системы может быть описана при помощи определённого словаря терминов и правил, на основании которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени. На основе этих утверждений формируются выводы о дальнейшем развитии системы и производится её оптимизация;

IDEF6. Design Rationale Capture — Обоснование проектных действий. Назначение IDEF6 состоит в облегчении получения «знаний о способе» моделирования, их представления и использования при разработке систем управления предприятиями. Под «знаниями о способе» понимаются причины, обстоятельства, скрытые мотивы, которые обуславливают выбранные методы моделирования. Проще говоря, «знания о способе» интерпретируются как ответ на вопрос: «Почему модель получилась такой, какой получилась?» Большинство методов моделирования фокусируются на собственно получаемых моделях, а не на процессе их создания. Метод IDEF6 акцентирует внимание именно на процессе создания модели;

IDEF7. Information System Auditing — Аудит информационных систем. Этот метод определён как востребованный, однако так и не был полностью разработан;

IDEF8. User Interface Modeling — Метод разработки интерфейсов взаимодействия оператора и системы (пользовательских интерфейсов). Современные среды разработки пользовательских интерфейсов в большей степени создают внешний вид интерфейса. IDEF8 фокусирует внимание разработчиков интерфейса на программировании желаемого взаимного поведения интерфейса и пользователя на трёх уровнях: выполняемой операции (что это за операция); сценарии взаимодействия, определяемом специфической ролью пользователя (по какому сценарию она должна выполняться тем или иным пользователем); и, наконец, на деталях интерфейса (какие элементы управления, предлагает интерфейс для выполнения операции);

IDEF9. Scenario-Driven IS Design (Business Constraint Discovery method) — Метод исследования бизнес-ограничений был разработан для облегчения обнаружения и анализа ограничений в условиях, в которых действует предприятие. Обычно при построении моделей уделяется недостаточное внимание описанию ограничений, оказывающих влияние на протекание процессов на предприятии. Знания об основных ограничениях и характере их влияния, закладываемые в модели, в лучшем случае остаются неполными, несогласованными, распределёнными нерационально, но часто их вовсе нет. Это не обязательно приводит к тому, что построенные модели нежизнеспособны, просто их реализация столкнётся с непредвиденными трудностями, в результате чего их потенциал будет не реализован. Тем не менее, в случаях, когда речь идёт именно о совершенствовании структур или адаптации к предсказываемым изменениям, знания о существующих ограничениях имеют критическое значение;

IDEF10 — Implementation Architecture Modeling — Моделирование архитектуры выполнения.

IDEF11 — Information Artifact Modeling.

IDEF12 — Organization Modeling — Организационное моделирование.

IDEF13 — Three Schema Mapping Design — Трёхсхемное проектирование преобразования данных.

IDEF14 — Network Design — Метод проектирования компьютерных сетей, основанный на анализе требований специфических сетевых компонентов существующих

конфигураций сетей. Также он обеспечивает поддержку решений, связанных с рациональным управлением материальными ресурсами, что позволяет достичь существенной экономии.

Задания на работу

- Предметная область для моделирования и создания UML диаграмм по варианту индивидуального задания из практических работ №№ 1-3 и на основании лабораторной работы №2.
- Для выполнения лабораторной работы может быть использован любой другой редактор UML диаграмм.

1. Создать диаграмму компонентов.
2. Создать диаграмму потоков данных.
3. Оформить отчет.

Контрольные вопросы

1. Основное назначение диаграммы компонентов?
2. Основное назначение диаграммы потоков данных?
3. Основное назначение IDEF диаграмм?
4. Какие стандарты семейства IDEF Вы знаете?
5. Какие основные элементы диаграммы потоков данных?

Лабораторная работа № 5
«ОЦЕНКА НЕОБХОДИМОГО КОЛИЧЕСТВА ТЕСТОВ, РАЗРАБОТКА ТЕСТОВОГО СЦЕНАРИЯ И ТЕСТОВЫХ ПАКЕТОВ»

Цель работы: изучение вопросов оценки качества программных средств: определение необходимого количества, разработка сценария тестирования и тестовых пакетов.

Теоретические сведения

1. ОСНОВНЫЕ ПОНЯТИЯ

Отладка ПС – деятельность по обнаружению и исправлению ошибок в ПС с использованием процессов выполнения его программ. *Тестирование* ПС – процесс выполнения его программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Этот набор данных называется *тестовым* или просто *тестом*.

Отладка = Тестирование + Поиск ошибок + Редактирование.

2. ВИДЫ ОТЛАДКИ ПРОГРАММНОГО СРЕДСТВА

Задачи отладки программных средств.

1. Подготовить такой набор тестов и применить к ним ПС, чтобы обнаружить в нем по возможности большее число ошибок.
2. Определить момент окончания отладки ПС (или отдельной его компоненты).

Для оптимизации набора тестов, необходимо заранее планировать этот набор и использовать рациональную стратегию планирования тестов. Проектирование тестов можно начинать сразу же после завершения этапа внешнего описания ПС. Возможны разные подходы к выработке стратегии проектирования тестов, которые можно условно графически разместить (см. рис.1) между следующими двумя крайними подходами.

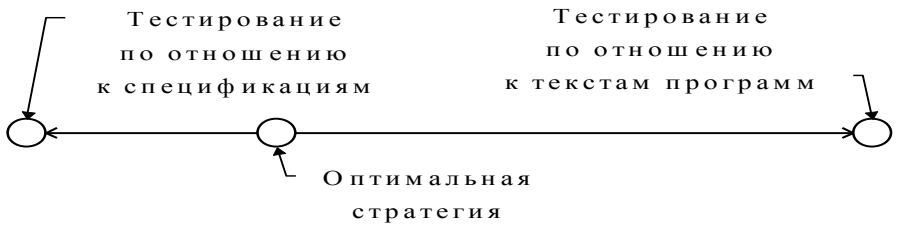


Рисунок 1 - Спектр подходов к проектированию тестов.

Оптимальная стратегия проектирования тестов расположена внутри интервала между этими крайними подходами, но ближе к левому краю. При этом в первом случае эта стратегия базируется на принципах:

- на каждую используемую функцию или возможность – хотя бы один тест,
- на каждую область и на каждую границу изменения какой-либо входной величины – хотя бы один тест,
- на каждую особую (исключительную) ситуацию, указанную в спецификациях, – хотя бы один тест.

Во втором случае эта стратегия базируется на принципе: каждая команда каждой программы ПС должна проработать хотя бы на одном тесте.

В нашей стране различаются два основных вида отладки (включая тестирование):

Автономная отладка ПС - последовательное раздельное тестирование различных частей программ с поиском и исправлением в них фиксируемых при тестировании ошибок. *Комплексная* отладка - тестирование ПС в целом с поиском и исправлением фиксируемых при тестировании ошибок во всех документах.

3. ПРИНЦИПЫ ОТЛАДКИ ПРОГРАММНОГО СРЕДСТВА.

Отметим феномен - по мере роста числа обнаруженных и исправленных ошибок в ПС *растет* также относительная вероятность существования в нем необнаруженных ошибок.

Принцип 1. Считайте тестирование ключевой задачей разработки ПС, поручайте его самым квалифицированным и одаренным программистам; нежелательно тестировать свою собственную программу.

Принцип 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Принцип 3. Готовьте тесты как для правильных, так и для неправильных данных.

Принцип 4. Документируйте пропуск тестов через компьютер; детально изучайте результаты каждого теста; избегайте тестов, пропуск которых нельзя повторить.

Принцип 5. Каждый модуль подключайте к программе только один раз; никогда не изменяйте программу, чтобы облегчить ее тестирование.

Принцип 6. Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПС или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

4. АВТОНОМНАЯ ОТЛАДКА ПРОГРАММНОГО СРЕДСТВА.

При автономной отладке тестируется всегда некоторая программа (*тестируемая программа*), построенная специально для тестирования отлаживаемого модуля. В процессе автономной отладки ПС производится наращивание тестируемой программы отлаженными модулями (*интеграция* программы).

При восходящем тестировании окружение содержит только один отладочный модуль, головной в тестируемой программе - *ведущий* (или драйвер). Ведущий отладочный модуль подготавливает информационную среду для тестирования отлаживаемого модуля, осуществляет обращение к отлаживаемому модулю и выдает необходимые сообщения.

При нисходящем тестировании окружение в качестве отладочных содержит *отладочные имитаторы* (заглушки) некоторых еще не отлаженных модулей. Некоторые из этих имитаторов при отладке одного модуля могут изменяться для разных тестов.

На практике в окружении отлаживаемого модуля могут содержаться отладочные модули обоих типов, если используется смешанная стратегия тестирования.

Достоинства восходящего тестирования:

1) простота подготовки тестов,
2) возможность полной реализации плана тестирования модуля.

Недостатки восходящего тестирования:

1) тестовые данные готовятся, как правило, не в той форме, которая рассчитана на пользователя;
2) большой объем отладочного программирования;
3) необходимость специального тестирования сопряжения модулей.

Достоинства нисходящего тестирования:

1) большинство тестов готовится в форме, рассчитанной на пользователя;
2) во многих случаях относительно небольшой объем отладочного программирования;
3) отпадает необходимость тестирования сопряжения модулей.

Недостатком нисходящего тестирования является то, что тестовое состояние информационной среды перед обращением к отлаживаемому модулю готовится косвенно – оно является результатом применения уже отлаженных модулей к тестовым данным или данным, выдаваемым имитаторами.

Прежде всего, необходимо организовать отладку программы таким образом, чтобы как можно раньше были отлажены модули, осуществляющие ввод данных. Пока модули, осуществляющие ввод данных, не отлажены, тестовые данные поставляются некоторыми имитаторами: они либо включаются в имитатор как его часть, либо вводятся этим имитатором.

При нисходящем тестировании некоторые состояния информационной среды, при которых требуется тестировать отлаживаемый модуль, могут не возникать при выполнении отлаживаемой программы ни при каких входных данных. Чаще же пользуются модифицированным вариантом нисходящего тестирования, при котором отлаживаемые модули перед их интеграцией предварительно тестируются отдельно. Однако, представляется более целесообразной другая модификация нисходящего тестирования: после завершения нисходящего тестирования отлаживаемого модуля для достижимых тестовых состояний информационной среды следует его отдельно протестировать для остальных требуемых состояний информационной среды.

Часто применяют также комбинацию восходящего и нисходящего тестирования, которую называют методом *сандвича*. Сущность этого метода заключается в одновременном осуществлении как восходящего, так и нисходящего тестирования, пока эти два процесса тестирования не встретятся на каком-либо модуле где-то в середине структуры отлаживаемой программы.

Весьма важным при автономной отладке является тестирование

сопряжения модулей. При нисходящем тестировании тестирование сопряжения осуществляется попутно каждым пропускаемым тестом, что считают достоинством нисходящего тестирования. При восходящем тестировании обращение к отлаживаемому модулю производится не из модулей отлаживаемой программы, а из ведущего отладочного модуля.

Автономное тестирование модуля целесообразно осуществлять в четыре последовательно выполняемых шага.

Шаг 1. На основании спецификации отлаживаемого модуля подготовьте тесты для каждой возможности и каждой ситуации, для каждой границы областей допустимых значений всех входных данных, для каждой области изменения данных, для каждой области недопустимых значений всех входных данных и каждого недопустимого условия.

Шаг 2. Проверьте текст модуля, чтобы убедиться, что каждое направление любого разветвления будет пройдено хотя бы на одном тесте. Добавьте недостающие тесты.

Шаг 3. Проверьте текст модуля, чтобы убедиться, что для каждого цикла существуют тесты, обеспечивающие, по крайней мере, три следующие ситуации: тело цикла не выполняется ни разу, тело цикла выполняется один раз и тело цикла выполняется максимальное число раз. Добавьте недостающие тесты.

Шаг 4. Проверьте текст модуля, чтобы убедиться, что существуют тесты, проверяющие чувствительность к отдельным особым значениям входных данных. Добавьте недостающие тесты.

5. КОМПЛЕКСНАЯ ОТЛАДКА ПРОГРАММНОГО СРЕДСТВА

Тестирование при комплексной отладке - применение ПС к конкретным данным, которые могут возникнуть у пользователя, но, возможно, в моделируемой (а не в реальной) среде.

Тестирование архитектуры ПС. Целью тестирования является поиск несоответствия между описанием архитектуры и совокупностью программ ПС. К моменту

начала тестирования архитектуры ПС должна быть уже закончена автономная отладка каждой подсистемы.

Тестирование внешних функций. Целью тестирования является поиск расхождений между функциональной спецификацией и совокупностью программ ПС. Несмотря на то, что все эти программы автономно уже отлажены, указанные расхождения могут быть,

Тестирование качества ПС. Целью тестирования является поиск нарушений требований качества, сформулированных в спецификации качества ПС. Завершенность ПС проверяется уже при тестировании внешних функций.

Тестирование документации по применению ПС. Целью тестирования является поиск несогласованности документации по применению и совокупностью программ ПС, а также выявление неудобств, возникающих при применении ПС. Этот этап непосредственно предшествует подключению пользователя к завершению разработки ПС (тестированию определения требований к ПС и аттестации ПС).

Тестирование определения требований к ПС. Целью тестирования является выяснение, в какой мере ПС не соответствует предъявленному определению требований к нему. Особенность этого вида тестирования заключается в том, что его осуществляет организация-покупатель или организация-пользователь. Обычно производится с помощью контрольных задач – типовых задач, для которых известен результат решения.

6. ОРГАНИЗАЦИЯ ТЕСТИРОВАНИЯ В ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ СИСТЕМАХ

Тестирование является достаточно независимым процессом, применимым к программному обеспечению, разработанному с помощью любого метода проектирования. Тем не менее объектно-ориентированный подход привносит свои особенности.

Традиционно тестирование делится на тестирование элементов, интеграционное тестирование и системное тестирование.

На уровне элементов тестирование объектно-ориентированных программ отличается по следующим показателям:

- определение единиц тестирования;
- тестирование наследования;
- тестирование полиморфизма.

Естественной единицей тестирования является класс. Разбиение его на более мелкие элементы (методы) нецелесообразно, поскольку они не существуют отдельно от классов. Иногда за единицу тестирования принимается тесно связанная группа классов.

Тестирование наследования состоит в тестировании методов, унаследованных классом от своего базового класса. Если базовый класс уже прошел тестирование, нужно ли повторять тестирование для унаследованных методов? Вопреки достаточно распространенным надеждам программистов перетестирование необходимо. Основная причина та, что методы выполняются в новом контексте.

Применимость тестовых сценариев базового класса для тестирования производного класса зависит от того, является ли наследование классификацией. Если нет, то даже для унаследованных методов необходимо разрабатывать новые тестовые сценарии.

Тестирование полиморфизма сходно с тестированием наследования в том, что в тестовых сценариях необходимо предусмотреть все варианты связывания, т.е. все варианты конкретной реализации полиморфизма.

Интеграционное тестирование представляет собой тестирование того, как отдельные элементы программы работают вместе. Свойства объектно-ориентированных языков исключают целый ряд возможных ошибок, прежде всего за счет строгого определения внешних интерфейсов классов и объектов. Однако это не означает, что интеграционное тестирование становится легче. Планирование интеграционного тестирования немного отличается.

При традиционном тестировании имеется такая характеристика, как степень покрытия кода. При тестировании по принципу открытого, или белого ящика (т.е. в случае, когда тестирующему известна структура кода) необходимо обеспечить прохождение управления по всем ответвлениям программы. Иными словами, требуется выполнить как можно больший процент инструкций, имеющихся в программе. Наряду с этой характеристикой планирование тестов для объектно-ориентированной программы должно включать «покрытие состояний».

То, что объект соединяет в себе состояние и поведение, обусловливает необходимость проверки всех возможных переходов из состояния в состояние. В планировании таких тестов должна помочь модель состояний класса, построенная на этапе анализа и проектирования.

Системное тестирование проверяет всю программную систему целиком и строится в большинстве случаев по принципу «черного ящика». Тестирующий знает только внешние характеристики системы, но не знает, как она работает.

Построение требований к системе в форме вариантов использования обеспечивает естественный и простой способ планирования системного тестирования. Фактически система вариантов использования становится основой плана тестов на этапе системного тестирования.

Задания на работу

1. Написать программу решения квадратного уравнения 3ax2 + 2.5bx + c = 0.
2. Найти минимальный набор тестов для программы нахождения вещественных корней квадратного уравнения ax2 + bx + c = 0. Решение представить в таблице следующего вида:

Номер теста	a	b	c	Ожидаемый результат	Что проверяется
..					
..					
..					
..					

3. Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы: имеется консольное приложение (разработайте его самостоятельно), на вход подается 3 строки, на выходе приложение выдает число вхождений второй строки в первую и третью. Набор тестовых сценариев запишите в виде таблицы, например:

Строка 1	Строка 3	Строка 2	Вывод/результат
aabababab	**bbaa**bfbfbf	aa	3
babababab	**aabbaabb**	bb	4

4. Спроектировать тесты по принципу «белого ящика» для программы, разработанной в задании №3. Выбрать несколько алгоритмов для тестирования и обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования. Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную вами программу. Результаты оформить в виде таблиц:

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
..			

..			
..			

5. Проверить все виды тестов и сделать выводы об их эффективности.
6. Оформить отчет.

Контрольные вопросы

1. Дайте определение понятию «Отладка программного средства».
2. Дайте определение понятию «Автономная отладка программного средства».
3. Каковы особенности восходящего тестирования?
4. Каковы особенности нисходящего тестирования?
5. Какие достоинства у восходящего тестирования?
6. Какие недостатки у восходящего тестирования?
7. Какие достоинства у нисходящего тестирования?
8. Какие недостатком у нисходящего тестирования?

Лабораторная работа № 6
«ОЦЕНКА ПРОГРАММНЫХ СРЕДСТВ С ПОМОЩЬЮ МЕТРИК»

Цель работы: изучение вопросов оценки качества программных средств: определение метрик.

Теоретические сведения

Метрики программного обеспечения можно разделить на три категории —

1) *Метрики продукта* — описывает характеристики продукта, такие как размер, сложность, особенности дизайна, производительность и уровень качества.

2) *Метрики процесса* — эти характеристики могут использоваться для улучшения деятельности по разработке и сопровождению программного обеспечения.

3) *Метрики проекта* — эти метрики описывают характеристики и исполнение проекта. Примеры включают число разработчиков программного обеспечения, штатное расписание в течение жизненного цикла программного обеспечения, стоимость, график и производительность.

Метрики продукта — описывает характеристики продукта, такие как размер, сложность, особенности дизайна, производительность и уровень качества.

Метрики процесса — эти характеристики могут использоваться для улучшения деятельности по разработке и сопровождению программного обеспечения.

Метрики проекта — эти метрики описывают характеристики и исполнение проекта. Примеры включают число разработчиков программного обеспечения, штатное расписание в течение жизненного цикла программного обеспечения, стоимость, график и производительность.

Некоторые показатели относятся к нескольким категориям. Например, показатели качества процесса в проекте являются как показателями процесса, так и показателями проекта.

Метрики качества программного обеспечения представляют собой подмножество метрик программного обеспечения, которые фокусируются на аспектах качества продукта, процесса и проекта. Они более тесно связаны с метриками процесса и продукта, чем с метриками проекта.

Метрики качества программного обеспечения можно разделить на три категории:

1) Метрики качества продукции.
2) Показатели качества в процессе.
3) Метрики качества обслуживания.

Метрики качества продукции включают в себя следующее —

- Среднее время до отказа.
- Плотность дефектов.
- Проблемы с клиентами.
- Удовлетворенность клиентов.

Среднее время до отказа. Это время между неудачами.

Плотность дефектов. Он измеряет дефекты относительно размера программного обеспечения, выраженного в виде строк кода или функциональной точки и т. Д., Т. Е. Измеряет качество кода на единицу. Этот показатель используется во многих коммерческих системах программного обеспечения.

Проблемы с клиентами. Он измеряет проблемы, с которыми сталкиваются клиенты при использовании программного продукта. Он содержит взгляд клиента на проблемное пространство программного обеспечения, которое включает в себя проблемы, не связанные с дефектами, а также проблемы с дефектами.

Удовлетворенность клиентов. Удовлетворенность клиентов часто измеряется данными опросов клиентов по пятибалльной шкале —

- Очень доволен
- Довольный
- нейтральный
- неудовлетворенный
- Очень Недовольный

Удовлетворение общим качеством продукта и его конкретными размерами обычно достигается с помощью различных методов опросов клиентов. На основе данных пятибалльной шкалы можно построить и использовать несколько метрик с небольшими отклонениями, в зависимости от цели анализа. Например —

- Процент полностью удовлетворенных клиентов
- Процент довольных клиентов
- Процент недовольных клиентов
- Процент неудовлетворенных клиентов

Обычно этот процент удовлетворения используется.

Показатели качества в процессе. Показатели качества в процессе работы связаны с отслеживанием появления дефектов во время формального машинного тестирования для некоторых организаций. Этот показатель включает в себя —

- Плотность дефектов при машинном тестировании
- Схема прибытия дефекта во время машинного тестирования
- Схема устранения дефектов на основе фазы
- Эффективность устранения дефектов

Плотность дефектов при машинном тестировании. Частота дефектов во время формального машинного тестирования (тестирование после интеграции кода в системную библиотеку) коррелирует с частотой дефектов в полевых условиях. Более высокая частота дефектов, обнаруженная во время тестирования, является показателем того, что в процессе разработки программного обеспечения наблюдается более высокая степень ошибок, если только более высокая частота дефектов не вызвана чрезмерными усилиями по тестированию.

Эта простая метрика дефектов является хорошим показателем качества, пока программное обеспечение все еще тестируется. Это особенно полезно для отслеживания последующих выпусков продукта в той же организации по разработке.

Задания на работу

1. Написать программу, генерирующую случайный массив вещественных чисел в диапазоне от –100 до 100 и определяющую все минимальные положительные элементы.

2. Оценить эффективность разработанной программы, используя метрики:

	Исходная программа		Улучшенная программа	
	Недостатки	Количественная оценка	Улучшения	Количественная оценка
Время				
выполнения				
Оперативная				
память				
Внешняя				
память				

3. Оценить качество разработанной программы, используя метрики:

	Правильность	Универсальность	Проверяемость	Точность результатов
Недостатки				
Оценка				

Контрольные вопросы

1. Дайте определение понятию «Метрики продукта».
2. Дайте определение понятию «Метрики процесса»
3. Дайте определение понятию «Метрики проекта»
4. Дайте определение понятию «Метрики качества программного обеспечения»

Лабораторная работа № 7
«ИНСПЕКЦИЯ ПРОГРАММНОГО КОДА НА ПРЕДМЕТ СООТВЕТСТВИЯ СТАНДАРТАМ КОДИРОВАНИЯ»

Цель работы: изучение вопросов оценки качества программных средств: инспекция программного кода на предмет соответствия стандартам кодирования.

Теоретические сведения

1. Формальные инспекции программного кода

Процесс формальной инспекции программного кода подчиняется всем правилам, определенным для абстрактной формальной инспекции, однако, имеет некоторые особенности, связанные, в первую очередь со структурой инспектируемого программного кода, а также с тем, что обычно инспектируются участки кода, которые невозможно проверить при помощи автоматизированного тестирования, основанного на тестовых примерах.

2. Особенности этапа просмотра инспектируемого кода

Выделение ключевых точек и построение или использование таблиц трассировки. Перед началом просмотра исходного кода рекомендуется отметить пункты требований, на соответствие которым проверяется исходный код, а также записать обоснования того, почему эти требования не могут быть проверены в автоматическом режиме. После этого можно переходить к просмотру собственно исходного кода. Все пометки, которые придется вносить в ходе инспектирования в исходный код необходимо делать не в файле, хранящемся в базе данных проекта, а в его копии, которая потом будет подшита к материалам инспекции. Копия может быть в том же формате, что и исходный файл, либо распечатана на бумаге или выведена в формат DOC, PDF или аналогичный, допускающий комментирование.

При помощи трассировочных таблиц в исходном коде определяются инспектируемые функции или методы, соответствующие необходимым требованиям.

Участки кода выделяются и отмечаются меткой или номером соответствующего требования. Если участок кода соответствует требованиям, то необходимо отметить этот факт либо цветом выделения, либо соответствующим текстовым примечанием. Если участок кода имеет проблемы, этот факт должен быть отражен либо цветом выделения, либо ссылкой на соответствующий пункт списка замечаний в бланке инспекции.

В случае отсутствия трассировочных таблиц требований на исходный код рекомендуется делать пометки, поясняющие, почему именно данный участок кода реализует указанные требования. Такие пометки помогут на этапе обсуждения документа.

Проверка стиля кодирования. Отдельным объектом проверки при формальной инспекции программного кода является стиль кодирования. В большинстве проектов существуют стандарты, описывающие правила оформления исходных текстов программ и файлов данных. Неверный стиль кодирования не влияет на работоспособность программы в целом, но значительно затрудняет сопровождение и поддержку изменений в ходе дальнейшего развития системы. Поэтому отклонения от стиля кодирования в инспектируемых участках кода также должны отмечаться в тексте и в списке замечаний.

В некоторых случаях проводят инспекции, целиком направленные на проверку стиля кодирования.

Проверка надежности кода. В некоторых случаях рекомендуется проверять наличие участков, гарантирующих робастность, даже если требования прямо не определяют необходимости обработки недопустимых значений. В случае, если

потенциально возможна некорректная работа программы из-за отсутствия обработчиков неверных значений, рекомендуется отметить это в списке замечаний.

3. ИЗУЧЕНИЕ ПРОГРАММ С ПОМОЩЬЮ ОТЛАДЧИКОВ

Существуют два способа запуска режима отладки в Visual Studio, первый - запуск из меню Debug и второй - запуск с помощью соответствующей панели инструментов. Оба способа - предоставляют доступ к запуску сеансов отладки, пошаговому прохождению кода, управлению точками останова, а также и ко многим функциональным возможностям отладки в Visual Studio.

Имеются два состояния меню отладки (Debug):

- Состояние покоя (неактивное);
- Режим отладки.

В состоянии покоя меню Debug (рис.1) предоставляет возможности для запуска сеанса отладки, прикрепления к выполняющемуся процессу и для доступа к некоторым из отладочных окон. В табл.1 перечислены все функциональные возможности, имеющиеся в меню Debug в состоянии покоя.

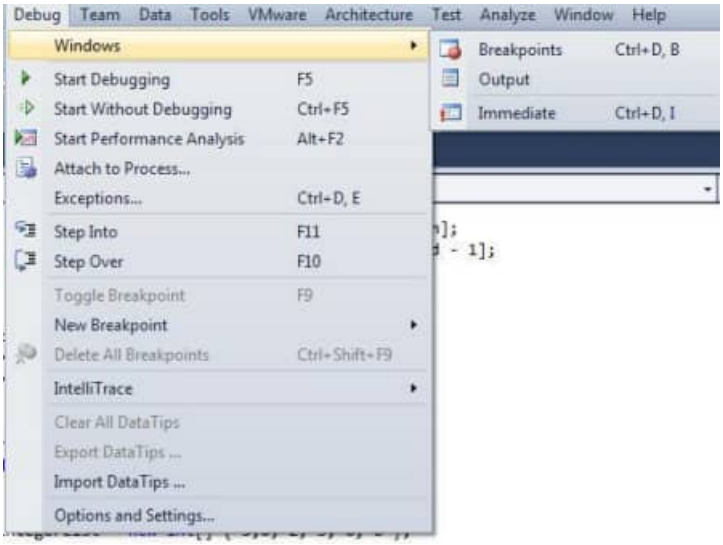


Рисунок 1. Меню Debug в Visual Studio в режиме покоя

Таблица 1. Элементы меню Debug в состоянии покоя

Элемент меню	Описание
Windows | Breakpoints	Открывает в интегрированной среде окно Breakpoints, которое дает доступ ко всем точкам останова данного решения.
Windows | Output	Показывает в интегрированной среде окно Output. Окно Output - это бегущий журнал множества сообщений, выдаваемых интегрированной средой, компилятором и отладчиком. Поэтому эта информация относится не только к сеансу отладки
Windows | Immediate	Открывает в интегрированной среде окно Immediate, которое позволяет выполнять команды.
Start Debugging	Запускает приложение в режиме отладки
Start Without Debugging	Запускает приложение без подключения отладчика к выполняющемуся процессу. В этом режиме разработчик видит то, что увидит пользователь (вместо того, чтобы выходить в интегрированную среду при ошибках и в точках останова)
Attach to Process	Позволяет прикрепить отладчик к выполняющемуся процессу (исполняемому файлу). Например, если запущено приложение без отладки, то можете потом прикрепиться к этому выполняющемуся процессу и начать отладку
Exceptions	Открывает диалоговое окно Exceptions, которое позволяет выбрать способ останова

	отладчика для каждого исключительного состояния
Step Into	Запускает приложение в режиме отладки. Для большинства проектов выбор команды Step Into означает вызов отладчика на первой выполняемой строке приложения. Таким образом, можно войти в приложение с первой строки
Step Over	Когда вы не находитесь в сеансе отладки, то команда Step Over просто запускает приложение точно так же, как это сделала бы кнопка Run
Toggle Breakpoint	Включает или выключает точку останова на текущей (активной) строке кода текстового редактора. Эта опция неактивна, если в интегрированной среде нет активного кодового окна
New Breakpoint | Break at Function	Активирует диалоговое окно New Breakpoint позволяющее указать имя функции, для которой необходимо создать точку останова.
New Breakpoint | New Data Breakpoint	Эта опция доступна только для приложений C++, позволяет определить точку останова, которая выходит в интегрированную среду тогда, когда изменяется значение по определенному адресу в памяти
Delete All Breakpoints	Удаляет все точки останова из текущего решения
Disable All Breakpoints	Деактивирует (без удаления) все точки останова текущего решения

Когда отладчик запущен, то состояние меню Debug изменяется, в нем становятся активными несколько дополнительных опции. Эти опции включают: функции перемещения по коду, перезапуск сеанса и доступ к дополнительным окнам отладки и т.д. На рис.2. показано меню Debug во время сеанса отладки. В табл.2 представлены элементы меню Debug в состоянии отладки.

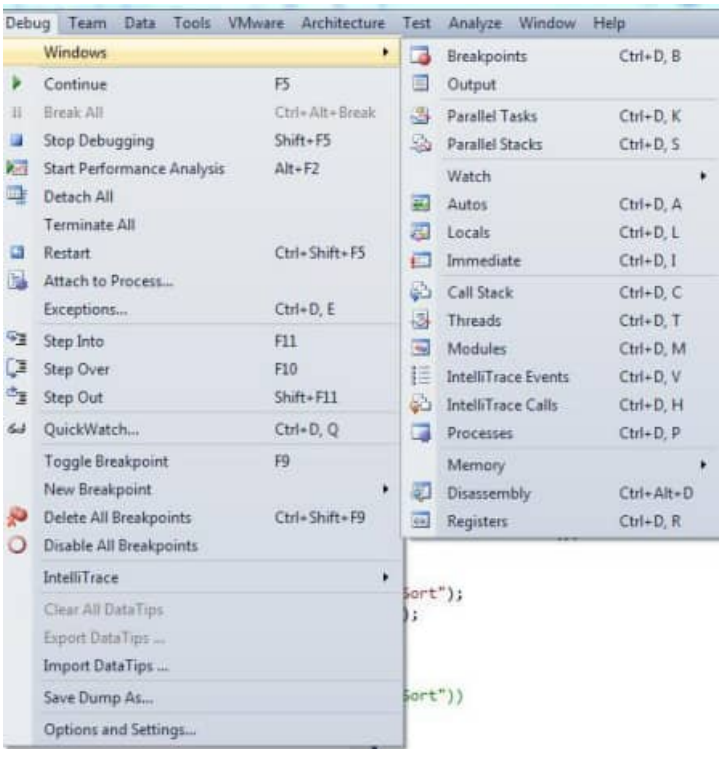


Рисунок 2. Меню Debug в Visual Studio в режиме отладки

Таблица 2. Элементы меню Debug в режиме отладки

Элемент меню	Описание
Windows | Breakpoints	Позволяет открыть окно Breakpoints во время сеанса отладки.
Windows |	Открывает окно Output во время активного сеанса отладки для того, чтобы можно было

Output	читать выходные сообщения, выдаваемые компилятором и отладчиком.
Windows | Parallel Task	Одно из нововведений в Visual Studio 2010. Подробнее о данном нововведение в подразделе лекции "Окно Parallel Task" .
Windows | Parallel Stacks	Одно из нововведений в Visual Studio 2010 Подробнее о данном нововведение в подразделе лекции "Окно Parallel Stacks".
Windows | Watch	Открывает одно из нескольких окон контрольных значений интегрированной среды. Окна контрольных значений представляют элементы и выражения, за которыми вы наблюдаете в течение сеанса отладки.
Windows | Autos	Открывает окно Autos. Это окно показывает переменные (и их значения) в текущей и предыдущей строках кода.
Windows | Locals	Открывает в интегрированной среде окно Locals, которое показывает переменные в локальной области действия (функции).
Windows | Immediate	Открывает окно Immediate, в котором вы можете выполнить команду.
Windows | Call Stack	Открывает список функций, которые имеются в стеке. Также указывает текущий кадр стека (функцию). Выделенный элемент - это то, что определяет содержимое окон Locals. Autos и окон контрольных значений.
Windows | Threads	Показывает в интегрированной среде окно Threads. Здесь можно просматривать потоки отлаживаемого приложения и управлять ими.
Windows | Modules	Показывает в интегрированной среде окно Modules. Это окно дает список dll и exe-файлов, используемых приложением.
Windows | IntelliTrace Events	Одно из нововведений в Visual Studio 2010. По умолчанию IntelliTrace собирает информацию для выбранных событий IntelliTrace. Когда приложение входит в режим приостановки выполнения, можно воспользоваться представлением IntelliTrace Events, чтобы просмотреть собранные события диагностики.
Windows | IntelliTrace Calls	Одно из нововведений в Visual Studio 2010. Когда IntelliTrace выполняет сбор сведений о вызовах, эти сведения можно просматривать в окне IntelliTrace Calls.
Windows | Processes	Показывает в интегрированной среде окно Processes. Это окно отображает список процессов, к которым прикреплен сеанс отладки.
Windows | Memory	Открывает окно Memory для просмотра используемой приложением памяти. Это работает только при включенной (в диалоговом окне Options) отладке на уровне адресов.
Windows | Disassembly	Открывает окно Disassembly. Это окно показывает ассемблерный код, соответствующий командам компилятора. Данная функция работает только при включенной (в диалоговом окне Options) отладке на уровне адресов.
Windows | Registers	Открывает окно Registers, чтобы вы могли видеть изменения значений регистров при прохождении по коду. Данная функция работает только при включенной (в диалоговом окне Options) отладке на уровне адресов.
Continue	Продолжает выполнение приложения после выхода в интегрированную среду разработки. Приложение продолжает выполняться с активной строки кода.
Break All	Позволяет прервать приложение вручную (без использования точки останова) во время сеанса отладки Приложение прервется на следующей исполняемой строке. Эта возможность полезна в том случае, когда нужно получить доступ к отладочной информации.
Stop Debugging	Останавливает режим отладки. Прерывает также и отлаживаемый процесс.
Detach All	Открепляет отладчик от выполняющегося процесса. Это позволяет приложению продолжать выполнение после того, как отладчик свою работу закончил.
Terminate All	Останавливает отладку и прекращает все процессы, к которым вы прикреплены.
Restart	Останавливает и перезапускает его сеанс отладки. Аналогично последовательному нажатию кнопок Stop Debugging и Start Debugging.
Attach to Process	Позволяет прикрепить активный сеанс отладки к одному дополнительному процессу.
Exceptions	Активирует диалог Exceptions, который позволяет управлять выходом в IDE по конкретным типам исключительных состояний .NET Framework и других библиотек.

Step Into	Приводит к продвижению отладчика на одну строку.
Step Over	Работает точно так же, как Step Into, но с одной важной разницей: если используете пропуск функции по Step Over, то строка вызова функции будет выполнена (и функция тоже), и отладчик установит следующую строку за вызовом функции в качестве следующей отлаживаемой строки.
Step Out	Указывает отладчику выполнить текущую функцию, а затем выйти назад в отладчик (после выполнения функции). Эта функция полезна тогда, когда осуществляется вход в функцию, а затем нужно, чтобы эта функция выполнилась и вернула в отладчик после ее завершения.
QuickWatch	Активирует окно QuickWatch. Это окно отображает значение одной, конкретной переменной (или выражения).
Toggle Breakpoint	Включает и выключает активную точку останова.
New Breakpoint	Активирует диалоговое окно New Breakpoint.
Delete All Break - 'points	Удаляет все точки останова в вашем решении.

Установка точки останова

Точка останова (breakpoint) - это сигнал, который указывает отладчику временно остановить выполнение программы в определенной точке. Приостановка выполнения программы в точке останова называется режимом приостановки. Вход в режим приостановки выполнения не приводит к прекращению или завершению работы программы, поэтому выполнение программы может быть продолжено в любое время. В Visual Studio можно помещать на любую строку кода, которая выполняется. Существуют три способа расстановки точек останова в Visual Studio 2010:

- С помощью клавиши F9;
- Через пункт меню Debug - Toggle Breakpoint;
- И самый простой способ - это щелкнуть дважды левой кнопкой мыши на нужной строке, в окне редактора кода внутри затененной области вдоль левого края окна документа.

Все эти способы приводят к размещению в конкретной строке точки останова, которая вызывает прерывание процесса выполнения и передачу управления отладчику. Как и в предыдущих версиях Visual Studio, точка останова обозначается большим кружком слева от соответствующей строки в окне редактора кода.

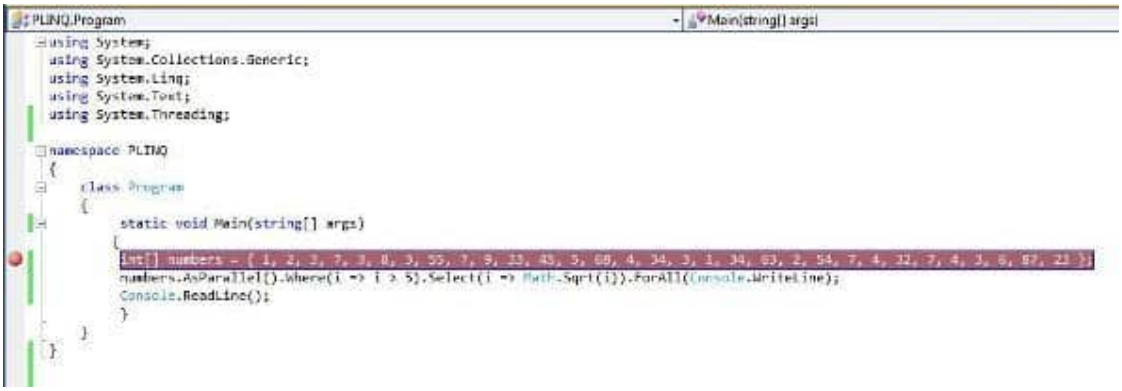


Рисунок 3. Установка точки останова в Visual Studio

Запуск отладчика

Для того что бы запустить режим отладки можно использовать несколько способов.

- Способ первый. С помощью специальной панели инструментов, как показано на рис.4;

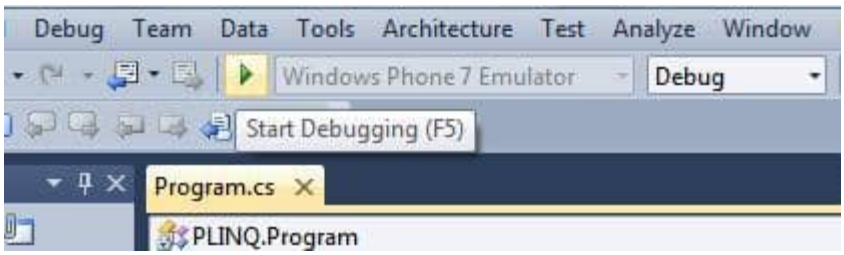


Рисунок 4. Запуск режима отладки с панели инструментов

- Способ второй. Из меню Debug с помощью пункта Start Debugging (рис.5);

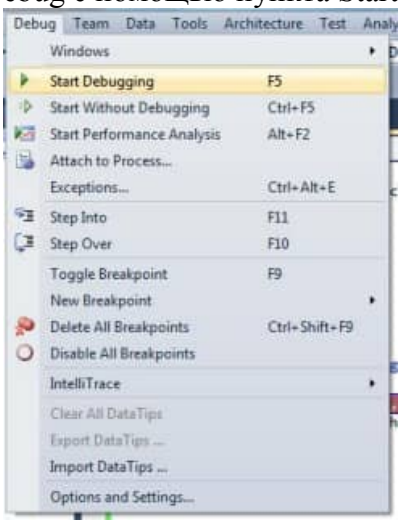


Рисунок 5. Запуск режима отладки из меню Debug

- Способ третий. С помощью клавиши F5.

После запуска режима отладки, появится курсор отладки, который остановится напротив первой точки останова (рис.6).

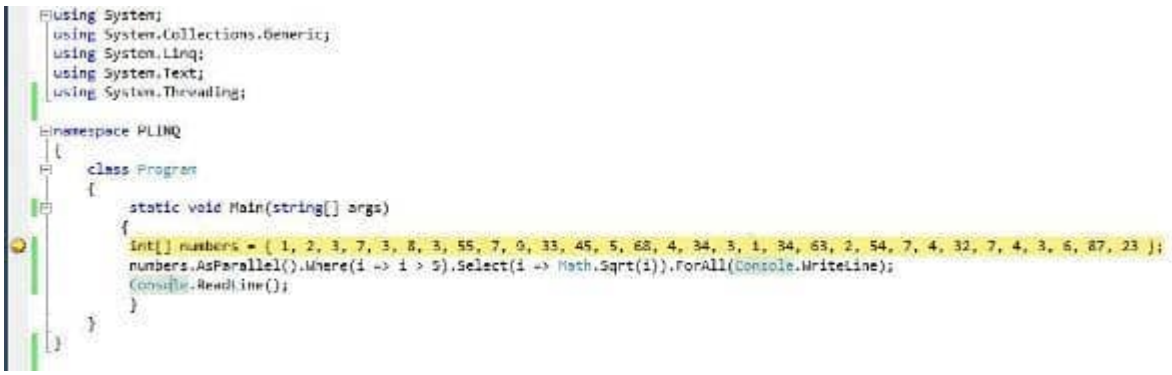


Рисунок 6. Курсор отладки в VIsual Studio

Пошаговое выполнение программы

Для пошаговой отладки используют специальную панель инструментов (рис.7):

Рисунок 7. Специальная панель инструментов отладки

Или набор горячих клавиш (которые дублируют работу специальной панели инструментов), список которых представлен ниже:

- Continue (F5) - продолжить выполнение программы.
- Stop debugging (Shift+F5) - остановить отладку. При этом остановится и выполнение программы.
- Restart (Ctrl+Shift+F5) - перезапустить программу. Выполнение программы будет прервано и запустится заново;
- Show Next Statement (Alt + Num *) - показать следующий оператор, т.е переместить курсор редактора кода в курсор пошагового выполнения;
- Step Into (F11) - выполнить очередной оператор. Если это метод, то перейти в начало этого метода, чтобы начать отладку;
- Step Over (F10) - выполнить очередной оператор. Если это метод, то он будет полностью выполнен, т.е. курсор выполнения не будет входить внутрь метода;
- Step out (Shift + F11) - выйти из метода.

При выполнении пошаговой отладки, разработчик может использовать следующие окна для просмотра значение переменных или если приложение многопоточное, то просматривать состояние потоков или переключаться между ними:

- Autos;
- Locals;
- Watch;
- Immediate;
- Threads;
- Parallel Task;
- Parallel Stacks.

Окно Autos

Окно Autos (рис.8) используется для того чтобы, просматривать значения, связанные с той строкой кода, на которой находится курсор отладки. Это окно отображает значения всех переменных и выражений, имеющихся в текущей выполняющейся строке кода или в предыдущей строке кода. Содержит следующие столбцы:

- Name - название переменной;
- Value - значение переменной;
- Type - тип переменной.

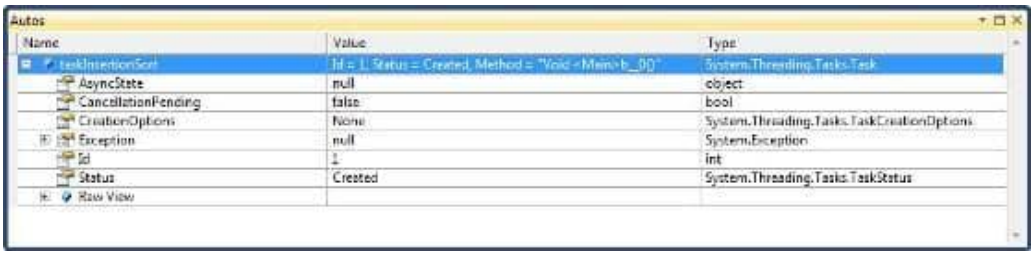


Рисунок 8. Окно Autos

Окно Locals

Окно Locals (рис.9) отображает все переменные и их значения для текущей области видимости отладчика, что дает представление обо всех переменных, которые используются в текущем выполняющемся методе. Переменные в этом окне автоматически настраиваются отладчиком. Данное окно содержит следующие столбцы:

- Name - название переменной;
- Value - значение переменной;
- Type - тип переменной.

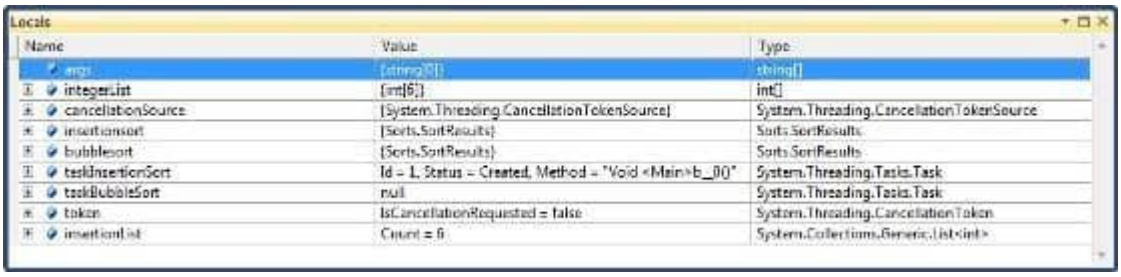


Рисунок 9. Окно Locals

Окно Watch

Окно Watch или окно контрольных значений (рис.10) - позволяет настраивать собственный список переменных и выражений, которые необходимо отслеживать. Всего доступно четыре окна Watch (Watch 1, Watch 2, Watch 3 и Watch 4), что позволяет выделить в четыре списка переменные и выражения, данную возможность удобно использовать в том случае, если каждый список относится к отдельной области видимости приложения. Переменные или выражение в окно Watch добавляются или из редактора кода, или из окна QuickWatch. Если нужно добавить в окно Watch элемент из редактора кода, то нужно выделить нужную переменную или выражение, щелкнуть по ней правой кнопкой мыши и выбирать пункт Add Watch. Также можно перетаскивать, с помощью мыши, выделенный элемент в окно Watch. Данное окно содержит следующие столбцы:

- Name - название переменной;
- Value - значение переменной;
- Type - тип переменной.
-

Рисунок 10. Окно Watch

Окно Immediate

Окно Immediate или непосредственное выполнение (рис.11) - предназначено для ручного ввода и выполнения команд. Это окно появляется автоматически при прерывании работы программы в точках останова программы. Для выполнения команды или оператора необходимо написать команду и нажать клавишу <Enter>.

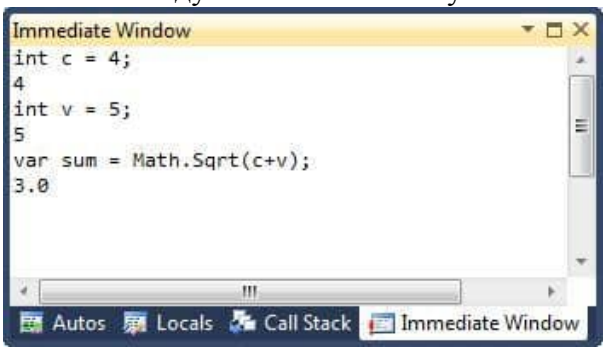


Рисунок 11. Окно Immediate

Окно Threads

Окно Threads (рис.12) - позволяет просматривать и управлять всеми запущенными потокам на различных этапах отладки приложения.

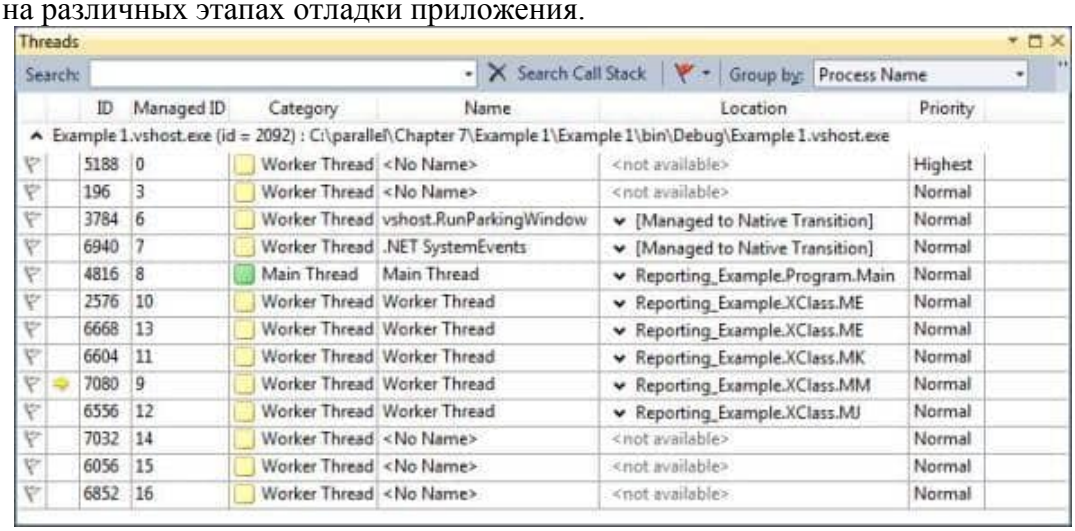


Рисунок 12. Окно Threads

По умолчанию в таблице перечисляются все потоки приложения, но можно фильтровать этот список, чтобы в нем показывались только нужные потоки. В каждом столбце содержится свой тип сведений. Сведения о столбцах окна Threads представлены в табл.3

Таблица 3. Столбцы окна отладки Threads

Имя столбца	**Описание**
Флаги	Показывает, какие потоки помечены, и позволяет помечать потоки и снимать с них метки.
Значки	Желтая стрелка указывает активный поток. Контур стрелки указывает поток, где выполнение было передано в отладчик. Белая стрелка указывает прерванную задачу, т.е. задачу, которая была текущей во время вызова отладчика. Значок паузы указывает поток, замороженный пользователем.
ID	Столбец содержит идентификационные номера всех потоков.
Managed ID	В столбце содержаться управляемые идентификационные номера управляемых потоков.
Category	В данном столбце потоки классифицируются по категориям: потоки пользовательского интерфейса, обработчики удаленного вызова процедур (RPC) и рабочие потоки. Особая категория идентифицирует главный поток приложения.
Name	Столбец, в котором для каждого потока указывается имя, если оно имеется, или значение <No Name>.
Location	В данном столбце показывается, где поток выполняется. Можно развернуть это расположение, чтобы отобразить полный стек вызова для потока.
Priority	Столбец содержит приоритет потока, назначенный системой каждому потоку.
Affinity Mask	Дополнительный столбец, который обычно скрыт. В этом столбце показывается маска сходства процессора для каждого потока. В многопроцессорной системе маска сходства определяет, какой процессор, в каком потоке может работать.
Suspended	Столбец содержит счетчик приостановок. Этот счетчик определяет, может ли поток выполняться.

Окно Parallel Task

Окно Parallel Task (рис.13) или окно параллельных задач - выглядит как окно Threads (потоки), за исключением того, что отображает сведения о каждой задаче или объекте task_handle, вместо сведений о каждом потоке. Как и потоки, задачи представляют асинхронные операции, которые могут выполняться параллельно, однако несколько задач могут выполняться в одном потоке.

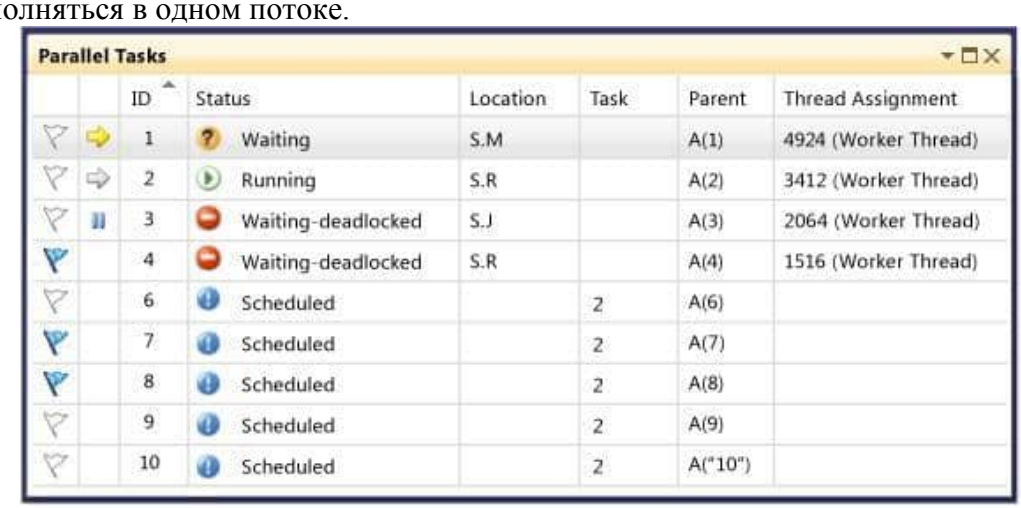


Рисунок 13. Окно Parallel Task

Сведения о столбцах окна Parallel Tasks представлены в табл.4

Таблица 4. Столбцы окна Parallel Task

Имя столбца	Описание
Флаги	Показывает, какие задачи помечены, и позволяет помечать задачи и снимать с них метки.
Значки	Рядом с текущей задачей отображается желтая стрелка. Текущая задача находится на самом верхнем уровне текущего потока. Белая стрелка указывает прерванную задачу, т.е. задачу, которая была текущей во время вызова отладчика. Значок паузы указывает задачу, замороженную пользователем.
ID	Столбец содержит предоставленный системой номер задачи. В машинном коде этот номер является адресом задачи.
Status	Столбец отображает текущее состояние задачи: • Запланированная задача - это задача, которая еще не выполнялась и, следовательно, не имеет стека вызова, назначенного потока и других соответствующих сведений. • Запущенная задача - это задача, которая выполняла код, пока не была прервана в отладчике. • Находящаяся в ожидании задача - это задача, заблокированная вследствие ожидания сигнала события, освобождения блокировки или завершения другой задачи. • Заблокированная задача - это находящаяся в ожидании задача, чей поток заблокирован другим потоком.
Location	Столбец, который отображает текущее расположение в стеке вызова задачи.
Task	Столбец отображает исходный метод и аргументы, которые были переданы в задачу при ее создании.
Parent	Столбец содержит идентификатор задачи, создавшей данную задачу. Если эта ячейка пуста, то у задачи нет родительской задачи. Это применимо только для управляемых программ.
Thread Assignment	Данный столбец содержит идентификатор и имя потока, в котором запущена задача.
AppDomain	Столбец содержит информацию о домене приложения (для управляемого кода), в котором выполняется задача.

task_group	Столбец отображает информацию о адрес объекта task_group, который запланировал задачу. Для асинхронных агентов и упрощенных задач этот столбец содержит значение 0.

Окно Parallel Stacks

Окно Parallel Stacks или параллельные стеки (рис.14) - это окно применяется при отладке многопоточных приложений, и содержит сведения о стеке вызова для всех потоков приложения. Оно также позволяет переходить в различные потоки и кадры стека в потоках.

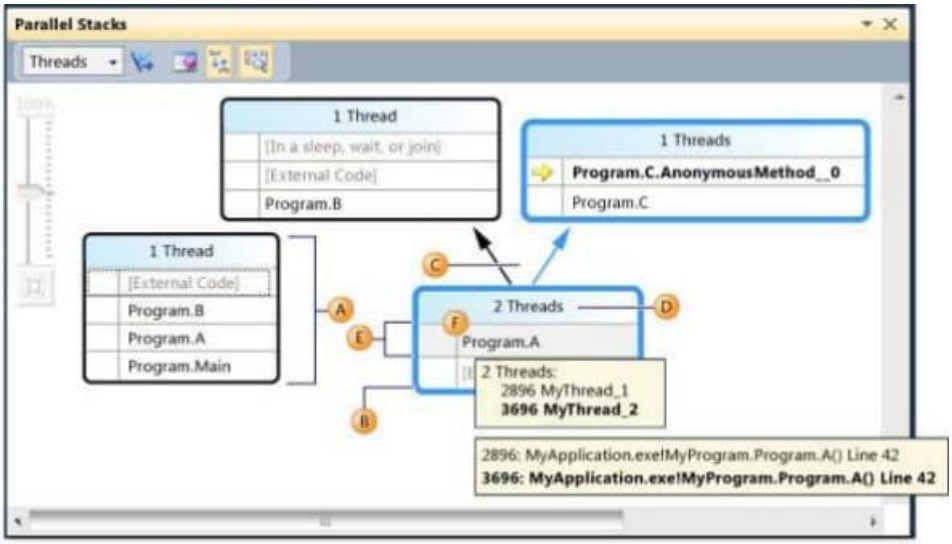


Рисунок 14. Окно Parallel Stacks

На рис.14 - путь вызова текущего потока выделен, синим, а активный кадр стека обозначается желтой стрелкой. Текущий кадр стека можно изменить, выбрав другой метод в окне Parallel Stacks. При этом также может измениться текущий поток в зависимости от того, входит ли выбранный метод в текущий поток или является частью другого потока. В табл.5 представлены компоненты окна Parallel Stacks

Таблица 5. Компоненты окна Parallel Stacks

	Имя элемента	Описание
	Сегмент или узел стека вызова	Содержит последовательности контекстов методов для одного или нескольких потоков. Если узел не имеет линий со стрелками, то он представляет собой единый путь вызова для потоков.
	Синее выделение	Указывает путь вызова текущего потока.
	Линии со стрелками	Соединяют узлы и показывают единый путь вызова для потоков.
	Всплывающая подсказка заголовка узла	Показывает идентификатор и пользовательское имя каждого потока, путь вызова которого использует данный узел.
	Контекст метода	Представляет один или несколько кадров стека одного метода.
	Всплывающая подсказка для контекста метода	Показывает подробные сведения всех кадров стека, которые представляются контекстом метода. Кадры стека для текущего потока отображаются жирным шрифтом.

В табл.6 описываются значки, которые предоставляют сведения об активных и текущих кадрах стека.

Таблица 6. Значки сведений об активных и текущих кадрах стека

Значок	Описание
	Указывает на то, что контекст метода содержит активный кадр стека текущего потока.

	Указывает на то, что контекст метода содержит активный кадр стека потока, который не является текущим.
	Указывает на то, что контекст метода содержит текущий кадр стека. Имя этого метода выделено жирным шрифтом во всех узлах.

На рис.15 и в табл.7 представлены элементы управления, доступные на панели управления "Параллельные стеки".

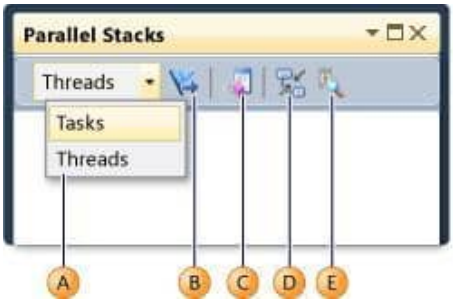


Рисунок 15. Элементы управления окна Parallel Stacks

Таблица 7. Элементы управления окна Parallel Stacks

	Элемент управления	Описание
	Поле со списком "Потоки"/"Задачи"	Переключает отображение между стеками вызова для потоков и стеками вызова для задач.
	Показывать только помеченные	Отображает стеки вызова только для потоков (или задач), помеченных в окнах "Потоки" или "Параллельные задачи".
	Представление метода	Переключает представление стека и представление метода.
	Автопрокрутка к текущему кадру стека	Автоматически прокручивает схему для отображения текущего кадра стека. Этот компонент применяется при изменении текущего кадра стека из других окон или при появлении точки останова в крупных диаграммах.
	Переключить элемент управления масштабом	Отображает или скрывает элемент управления масштабом. Чтобы изменить масштаб, можно также нажать клавишу CTRL и повернуть колесо мыши вне зависимости от того, где находится элемент управления.

Задания на работу

Возможно создание и отладка Windows приложения по индивидуальному варианту из Практической работы №1-2.

Ниже рассмотрим создание приложения «Калькулятор» (в случае выбора не индивидуального задания).

1. Создадим Windows приложение с названием "WindowsDebugApplication":

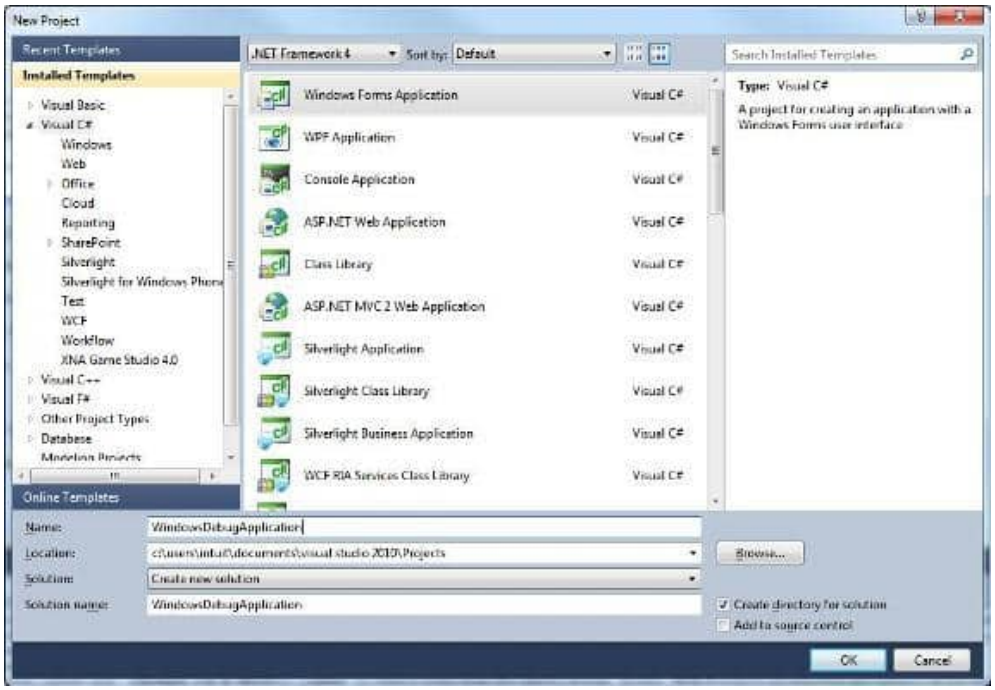


Рисунок 16

2. Создадим простейший калькулятор. Для этого разместим на форме 4 элемента (2 TextBox, 1 ComboBox, 1 Button):

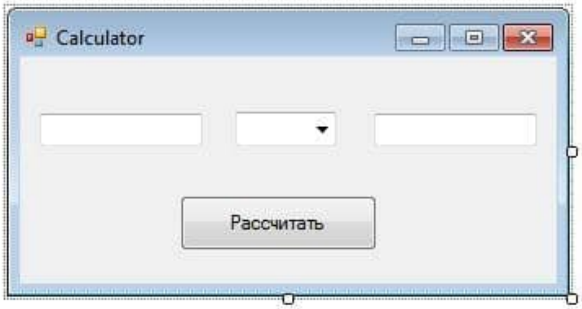


Рисунок 17

3. Добавим в программу следующий код (Коллекция arraysymbol и цикл foreach добавлены для наглядности отладки приложения):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;
namespace WindowsDebugApplication
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
```

```
        private double c =0;
        private void Plus(double a,double b)
        {
          c = a + b;
        }
        private void Minus(double a, double b)
        {
           c = a - b;
        }
        private void Multiply(double a, double b)
        {
           c = a * b;
        }
        private void Split(double a, double b)
        {
           c = a / b;
        }
        private void ResultButton_Click(object sender, EventArgs e)
        {
          string symbol =(string) SymbolcomboBox.SelectedItem;
          ArrayList arraysymbol = new ArrayList();
          foreach (var s in SymbolcomboBox.Items)
          {
             arraysymbol.Add(s);
          }
          double var1 =Convert.ToDouble(FirstVariable.Text);
          double var2 = Convert.ToDouble(SecondVariable.Text);
          switch (symbol)
          {
             case "+": Plus(var1, var1); break;
             case "-": Minus(var1, var1); break;
             case "*": Multiply(var1, var1); break;
             case "/": Split(var1, var1); break;
          }
          MessageBox.Show("Результат: " + c);
          MessageBox.Show("Количество символов в коллекции: "+ arraysymbol.Count);
        }
      }
    }
```

4. Теперь, расставим точки останова (breakpoints) в программе. В примере точки останова расставлены напротив методов математических операций и в событии кнопки:

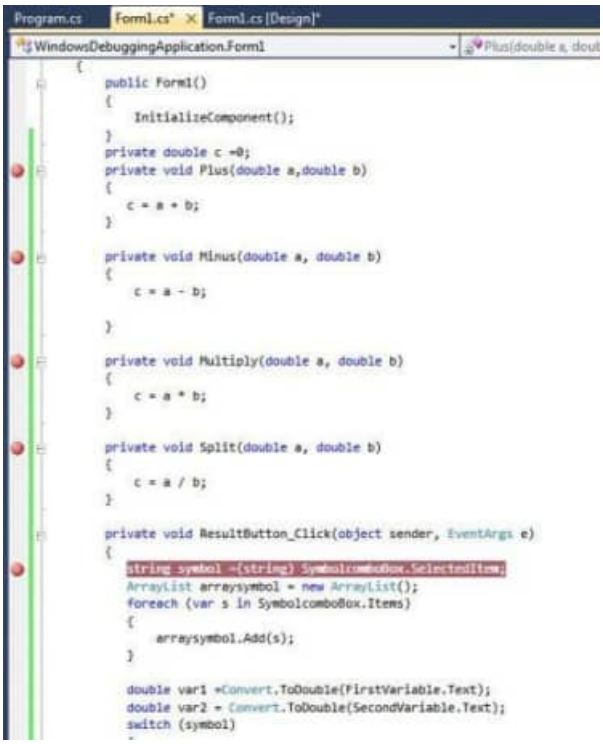


Рисунок 18

Список доступных всех точек останова (Breakpoints), можно посмотреть в специальном окне, которое вызывается из меню Debug пункт меню "Breakpoints" или сочетанием клавиш CTRL+ALT+B:

Рисунок 19

5. Запустите отладку приложения с помощью пункта "Start Debugging" - меню "Debbug" или с помощью клавиши "F5":

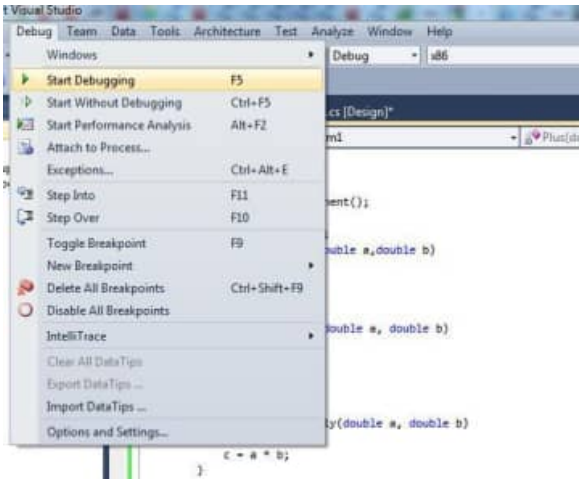


Рисунок 20

6. После ввода значений в поля программы и выбора соответствующей операции (сложение, вычитание и т.д), жмем кнопку "Рассчитать", тем самым вызовется метод ResulButton_Click():

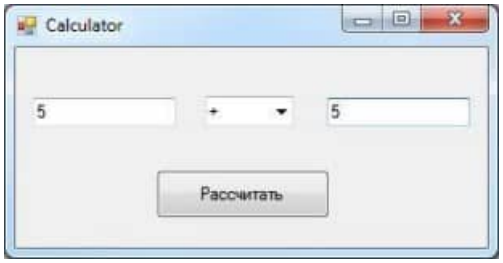


Рисунок 21

7. Запустится пошаговый процесс отладки приложения с точки останова (Breakpoint) в методе ResulButton_Click():

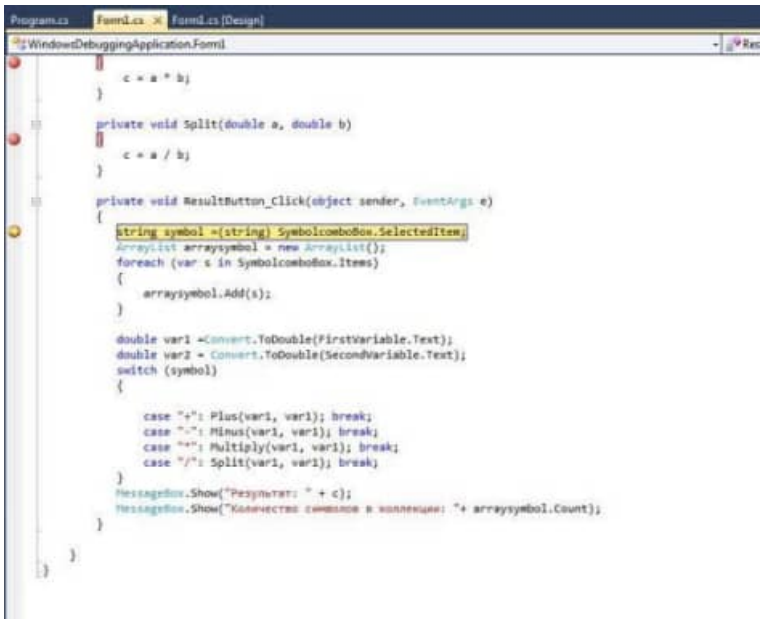


Рисунок 22

8. Добавим для просмотра значений переменных - переменные arraysymbol (коллекция), и переменную "c". Для этого щелкните на нужной переменной правой кнопкой мыши и выберите из списка "Add Watch":

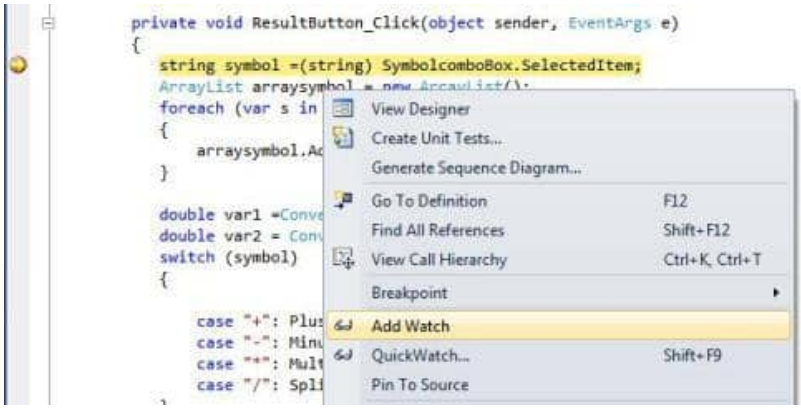


Рисунок 23

Переменные также, можно вручную добавлять в список Watch, для этого, достаточно написать имя нужной переменной в колонке Name:

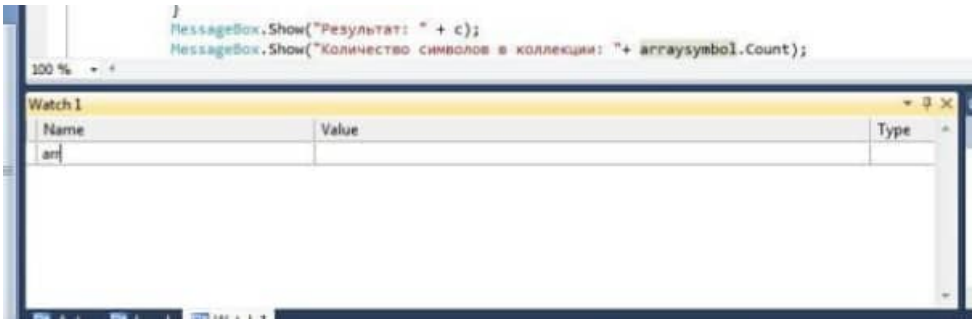


Рисунок 24

Если нужно просматривать состояние всех переменных во время отладки, используется окно Autos (переменные будут появляться в окне автоматически, в зависимости от шага отладчика):

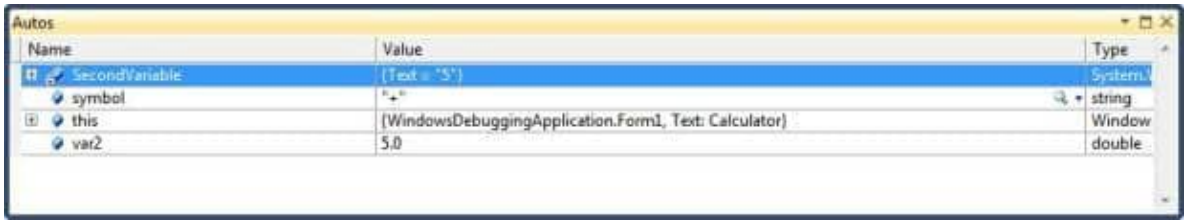


Рисунок 25

9. Используйте кнопку "F10" для пошаговой отладки приложения. В процессе пошаговой отладки, курсор отладчика будет заходить в те методы, которые вызываются в методе ResulButton_Click(), в нашем случае это метод Plus() (т.к была выбрана операция сложения - "+"):

```
}
private double c =0;
private void Plus(double a,double b)
{
    c = a + b;
}

private void Minus(double a, double b)
{
    c = a - b;

}
```

Рисунок 26

10. В процессе отладки приложения, значения переменных, в списке Watch, будут изменяться в зависимости от шага:

```
private void ResultButton_Click(object sender, EventArgs e)
{
    string symbol =(string) SymbolcomboBox.SelectedItem;
    ArrayList arraysymbol = new ArrayList();
    foreach (var s in SymbolcomboBox.Items)
    {
        arraysymbol.Add(s);
    }

    double var1 =Convert.ToDouble(FirstVariable.Text);
    double var2 = Convert.ToDouble(SecondVariable.Text);
    switch (symbol)
    {

        case "+": Plus(var1, var1); break;
        case "-": Minus(var1, var1); break;
        case "*": Multiply(var1, var1); break;
        case "/": Split(var1, var1); break;
    }
    MessageBox.Show("Результат: " + c);
    MessageBox.Show("Количество символов в коллекции: "+ arraysymbol.Count);
}

}
}
```

Watch 1

Name	Value	Type
arraysymbol	Count = 4	System.(
[0]	"+"	object {s
[1]	"-"	object {s
[2]	"*"	object {s
[3]	"/"	object {s
Raw View		
c	10.0	double

Рисунок 27

Всего разработчику доступны четыре списка Watch, которые вызываются из меню Debug или с помощью горячих клавиш "Ctrl+Alt+W,1":

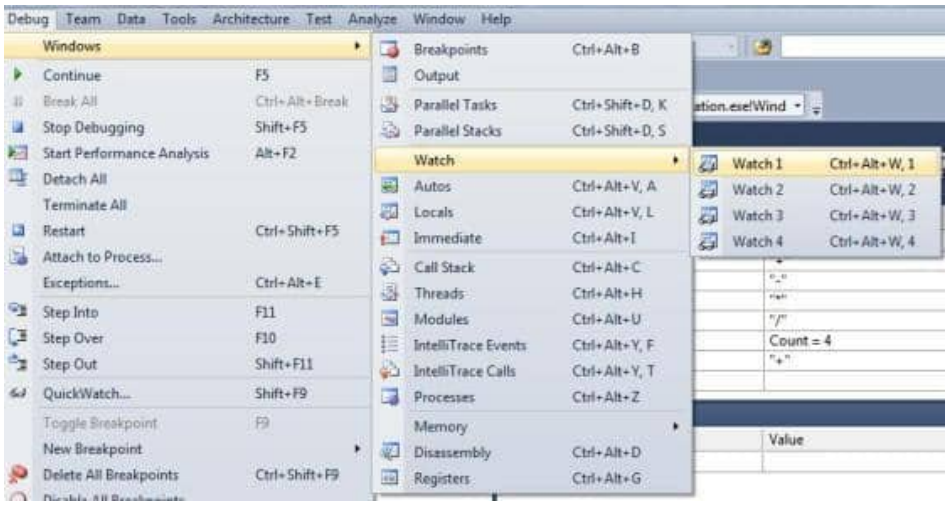


Рисунок 28

11. После завершения отладки (и если не возникло не каких ошибок) программа выдаст результаты:

Рисунок 29

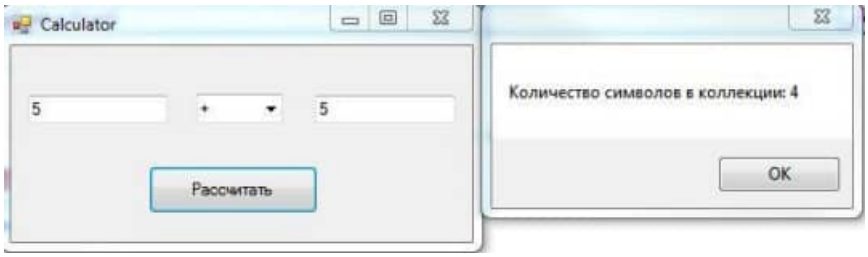


Рисунок 30

12. Повторно запустим отладку приложения и намеренно введем значения, вызывающие исключение. В нашем случае это "1a" и "2":

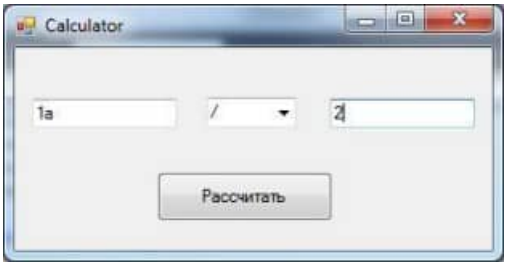


Рисунок 31

13. Если в программе не обрабатываются исключения (блок try, catch), отладчик выдаст ошибку, на строке, где возникает исключение. В нашем случае исключение, связанное с преобразованием формата типа string в формат double:

Рисунок 32

14. Для того что бы остановить отладку используйте пункт меню "Stop Debugging" меню Debug или используя сочетание клавиш "Shift+F5":

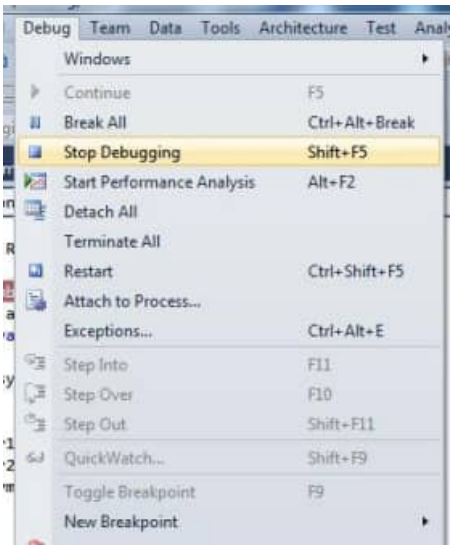


Рисунок 33.

Контрольные вопросы

1. Охарактеризуйте этапа просмотра инспектируемого кода.
2. Каковы особенности этапа просмотра инспектируемого кода.

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

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторно-практических работ

по междисциплинарному курсу
МДК 02.02 Инструментальные средства разработки программного обеспечения

профессионального модуля
ПМ.02. ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕ

специальности СПО

09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЕНЫ

на заседании цикловой комиссии информационных технологий
Протокол от «13» января 2023 г. № 6

Председатель цикловой комиссии И.В. Миляева

Автор: Сафронова М.А., преподаватель, канд. техн. наук

Лабораторная работа №1
«РАЗРАБОТКА СТРУКТУРЫ ПРОЕКТА»

Цель работы: получить практические навыки в разработке структуры программного проекта с использованием инструментальных средств.

Теоретические сведения

1. Общие сведения о разработке программного обеспечения

Необходимость управления программными проектами вытекает из того факта, что процесс создания профессионального ПО всегда является субъектом бюджетной политики организации, где оно разрабатывается, и имеет временные ограничения. Работа *руководителя программного проекта* по большому счету заключается в том, чтобы гарантировать выполнение этих бюджетных и временных ограничений с учетом бизнес-целей организации относительно разрабатываемого ПО.

Руководители проектов призваны спланировать все этапы разработки программного продукта. Они также должны контролировать ход выполнения работ и соблюдения всех требуемых стандартов.

Постоянный контроль за ходом выполнения работ необходим для того, чтобы процесс разработки не выходил за временные и бюджетные ограничения. Хорошее управление не гарантирует успешного завершения проекта, но плохое управление обязательно приведет к его провалу. Это может выразиться в задержке сроков сдачи готового ПО, в превышении сметной стоимости проекта и в несоответствии готового ПО спецификации требований.

Процесс разработки ПО существенно отличается от процессов реализации технических проектов, что порождает определенные сложности в управлении программными проектами:

1. *Программный продукт нематериален*. Программное обеспечение нематериально, его нельзя увидеть или потрогать. Руководитель программного проекта не видит процесс "роста" разрабатываемого ПО. Он может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.
2. *Не существует стандартных процессов разработки ПО*. На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены. Процессы создания большинства технических систем хорошо изучены. Изучением же процессов создания ПО специалисты занимаются только последнее время. Поэтому пока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему программному проекту.
3. *Большие программные проекты - это часто "одноразовые" проекты*. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но постоянные технологические изменения в компьютерной технике и коммуникационном оборудовании обесценивают предыдущий опыт. Знания и навыки, накопленные опытом, могут не востребоваться в новом проекте.

Перечисленные отличия могут привести к тому, что реализация проекта выйдет из временного графика или превысит бюджетные ассигнования. Программные системы зачастую оказываются новинками как в "идеологическом", так и в техническом плане. Поэтому, предвидя возможные проблемы в реализации программного проекта, следует

всегда помнить, что многим из них свойственно выходить за рамки временных и бюджетных ограничений.

2. Процесс управления разработкой программного обеспечения

Невозможно описать и стандартизировать все работы, выполняемые в проекте по созданию ПО. Эти работы весьма существенно зависят от организации, где выполняется разработка ПО, и от типа создаваемого программного продукта.

Но всегда можно выделить следующие:

1) Написание предложений по созданию ПО.
2) Планирование и составление графика работ по созданию ПО.
3) Оценивание стоимости проекта.
4) Подбор персонала.
5) Контроль за ходом выполнения работ.
6) Написание отчетов и представлений.

Первая стадия программного проекта может состоять из написания предложений по реализации этого проекта. Предложения должны содержать описание целей проектов и способов их достижения. Они также обычно включают в себя оценки финансовых и временных затрат на выполнение проекта. При необходимости здесь могут приводиться обоснования для передачи проекта на выполнение сторонней организации или команде разработчиков.

Написание предложений — очень ответственная работа, так как для многих организаций вопрос о том, будет ли проект выполняться самой организацией или разрабатываться по контракту сторонней компанией, является критическим. Не существует каких-либо рекомендаций по написанию предложений, многое здесь зависит от опыт.

На этапе *планирования проекта* определяются процессы, этапы и полученные на каждом из них результаты, которые должны привести к выполнению проекта. Реализация этого плана приведет к достижению целей проекта. Определение стоимости проекта напрямую связано с его планированием, поскольку здесь оцениваются ресурсы, требующиеся для выполнения плана.

Контроль за ходом выполнения работ (мониторинг проекта) — это непрерывный процесс, продолжающийся в течение всего срока реализации проекта. Руководитель должен постоянно отслеживать ход реализации проекта и сравнивать фактические и плановые показатели выполнения работ с их стоимостью. Хотя многие организации имеют механизмы формального мониторинга работ, опытный руководитель может составить ясную картину о стадии развитии проекта просто путем неформального общения с разработчиками.

Неформальный мониторинг часто помогает обнаружить потенциальные проблемы, которые в явном виде могут обнаружиться позднее. Например, ежедневное обсуждение хода выполнения работ может выявить отдельные недоработки в создаваемом программном продукте. Вместо ожидания отчетов, в которых будет отражен факт "пробуксовки" графика работ, можно обсудить со специалистами намечающиеся программистские проблемы и не допустить срыва графика работ.

В течение реализации проекта обычно происходит несколько формальных контрольных проверок хода выполнения работ по созданию ПО. Такие проверки должны дать общую картину хода реализации проекта в целом и показать, насколько уже разработанная часть ПО соответствует целям проекта.

Время выполнения больших программных проектов может занимать несколько лет. В течение этого времени цели и намерения организации, заказавшей программный проект, могут существенно измениться. Может оказаться, что разрабатываемый программный продукт стал уже ненужным либо исходные требования к создаваемому ПО просто

устарели и их необходимо кардинально менять. В такой ситуации руководство организации-разработчика может принять решение о прекращении разработки ПО или об изменении проекта в целом с тем, чтобы учесть изменившиеся цели и намерения организации-заказчика.

Руководители проектов обычно обязаны сами *подбирать исполнителей* для своих проектов. В идеальном случае профессиональный уровень исполнителей должен соответствовать той работе, которую они будут выполнять в ходе реализации проекта. Однако во многих случаях руководители должны полагаться на команду разработчиков, которая далека от идеальной. Такая ситуация может быть вызвана следующими причинами:

1. Бюджет проекта не позволяет привлечь высококвалифицированный персонал. В таком случае за меньшую плату привлекаются менее квалифицированные специалисты.
2. Бывают ситуации, когда невозможно найти специалистов не-обходимой квалификации как в самой организации-разработчике, так и вне ее. Например, в организации "лучшие люди" могут быть уже заняты в других проектах.
3. Организация хочет повысить профессиональный уровень своих работников. В этом случае она может привлечь к участию в проекте неопытных или недостаточно квалифицированных работников, чтобы они приобрели необходимый опыт и поучились у более опытных специалистов.

Таким образом, почти всегда подбор специалистов для выполнения проекта имеет определенные ограничения и не является свободным. Вместе с тем необходимо, чтобы хотя бы несколько членов группы разработчиков имели квалификацию и опыт, достаточные для работы над данным проектом. В противном случае невозможно избежать ошибок в разработке ПО.

Руководитель проекта обычно обязан посылать *отчеты* о ходе его выполнения как заказчику, так и подрядным организациям. Это должны быть краткие документы, основанные на информации, извлекаемой из подробных' отчетов о проекте. В этих отчетах должна быть та информация, которая позволяет четко оценить степень готовности создаваемого программного продукта.

В рамках дисциплины выделены следующие роли в группе по разработке ПО:

Руководитель – общее руководство проектом, написание документации, общение с заказчиком ПО.

Системный аналитик – разработка требований (составление технического задания, проекта программного обеспечения).

Тестер – составление плана тестирования и аттестации готового ПО (продукта), составление сценария тестирования, базовый пример, проведение мероприятий по плану тестирования.

Разработчик – моделирование компонент программного обеспечения, кодирование.

3. Планирование проекта разработки программного обеспечения

Эффективное управление программным проектом напрямую зависит от правильного планирования работ, необходимых для его выполнения. План помогает руководителю предвидеть проблемы, которые могут возникнуть на каких-либо этапах создания ПО, и разработать превентивные меры для их предупреждения или решения. План, разработанный на начальном этапе проекта, рассматривается всеми его участниками как руководящий документ, выполнение которого должно привести к успешному завершению проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

Процесс планирования начинается, исходя из описания системы, с определения проектных ограничений (временные ограничения, возможности наличного персонала, бюджетные ограничения и т.д.). Эти ограничения должны определяться параллельно с

оцениванием проектных параметров, таких как структура и размер проекта, а также распределением функций среди исполнителей. Затем определяются этапы разработки и то, какие результаты документация, прототипы, подсистемы или версии программного продукта) должны быть получены по окончании этих этапов. Далее начинается циклическая часть планирования. Сначала разрабатывается график работ по выполнению проекта или дается разрешение на продолжение использования ранее созданного графика. После этого проводится контроль выполнения работ и отмечаются расхождения между реальным и плановым ходом работ.

Далее, по мере поступления новой информации о ходе выполнения проекта, возможен пересмотр первоначальных оценок параметров проекта. Это, в свою очередь, может привести к изменению графика работ. Если в результате этих изменений нарушаются сроки завершения проекта, должны быть пересмотрены (и согласованы с заказчиком ПО) проектные ограничения.

Конечно, большинство руководителей проектов не думают, что реализация их проектов пройдет гладко, без всяких проблем. Желательно описать возможные проблемы еще до того, как они проявят себя в ходе выполнения проекта. Поэтому лучше составлять "пессимистические" графики работ, чем "оптимистические". Но, конечно, невозможно построить план, учитывающий все, в том числе случайные, проблемы и задержки выполнения проекта, поэтому и возникает необходимость периодического пересмотра проектных ограничений и этапов создания программного продукта.

План проекта должен четко показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. В некоторых организациях план проекта составляется как единый документ, содержащий все виды планов, описанных выше. В других случаях план проекта описывает только технологический процесс создания ПО. В таком плане обязательно присутствуют ссылки на планы других видов, но они разрабатываются отдельно от плана проекта.

Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации-разработчика. Но в любом случае большинство планов содержат следующие разделы.

1. *Введение*. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом.
2. *Организация выполнения проекта*. Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.
3. *Анализ рисков*. Описание возможных проектных рисков, вероятности их проявления и стратегий, направленных на их уменьшение.
4. *Аппаратные и программные ресурсы, необходимые для реализации проекта*. Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта. Если аппаратные средства требуется закупать, приводится их стоимость совместно с графиком закупки и поставки.
5. *Разбиение работ на этапы*. Процесс реализации проекта разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов ("выходов") каждого этапа и контрольные отметки.
6. *График работ*. В этом графике отображаются зависимости между отдельными процессами (этапами) разработки ПО, оценки времени их выполнения и распределение членов команды разработчиков по отдельным этапам.
7. *Механизмы мониторинга и контроля за ходом выполнения проекта*. Описываются предоставляемые руководителем отчеты о ходе выполнения работ, сроки их предоставления, а также механизмы мониторинга всего проекта.

План должен регулярно пересматриваться в процессе реализации проекта. Одни части плана, например график работ, изменяются часто, другие более стабильны. Для

внесения изменений в план требуется специальная организация документопотока, позволяющая отслеживать эти изменения.

4. Общие сведения о требованиях к информационным системам

Проблемы, которые приходится решать специалистам в процессе создания программного обеспечения, очень сложны. Природа этих проблем не всегда ясна, особенно если разрабатываемая программная система инновационная. В частности, трудно чѐтко описать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на систему, называется требованиями к этой системе, а сам процесс формирования, анализа, документирования и проверки этих функциональных возможностей и ограничений – разработкой требований.

Требования подразделяются на пользовательские и системные. Пользовательские требования – это описание на естественном языке (плюс поясняющие диаграммы) функций, выполняемых системой, и ограничений, накладываемых на неѐ. Системные требования – это описание особенностей системы (архитектура системы, требования к параметрам оборудования и т.д.), необходимых для эффективной реализации требований пользователя.

Первые шаги по разработке требований к информационным системам - анализ осуществимости.

Разработка требований — это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Для новых программных систем процесс разработки требований должен начинаться с анализа осуществимости. Началом такого анализа является общее описание системы и ее назначения, а результатом анализа — отчет, в котором должна быть четкая рекомендация, продолжать или нет процесс разработки требований проектируемой системы. Другими словами, анализ осуществимости должен осветить следующие вопросы.

1) Отвечает ли система общим и бизнес-целям организации-заказчика и организации-разработчика?
2) Можно ли реализовать систему, используя существующие на данный момент технологии и не выходя за пределы заданной стоимости?
3) Можно ли объединить систему с другими системами, которые уже эксплуатируются?

Критическим является вопрос, будет ли система соответствовать целям организации. Если система не соответствует этим целям, она не представляет никакой ценности для организации. В то же время многие организации разрабатывают системы, не соответствующие их целям, либо не совсем ясно понимая эти цели, либо под влиянием политических или общественных факторов.

Выполнение анализа осуществимости включает сбор и анализ информации о будущей системе и написание соответствующего отче-та. Сначала следует определить, какая именно информация необходима, чтобы ответить на поставленные выше вопросы.

Например, эту информацию можно получить, ответив на следующее:

1) Что произойдет с организацией, если система не будет введена в эксплуатацию?

2) Какие текущие проблемы существуют в организации и как новая система поможет их решить?

3) Каким образом система будет способствовать целям бизнеса?

4) Требует ли разработка системы технологии, которая до этого не использовалась в организации?

Далее необходимо определить источники информации. Это могут быть менеджеры отделов, где система будет использоваться, разработчики программного

обеспечения, знакомые с типом будущей системы, технологи, конечные пользователи и т.д.

После обработки собранной информации готовится отчет по анализу осуществимости создания системы. В нем должны быть даны рекомендации относительно продолжения разработки системы. Могут быть предложены изменения бюджета и графика работ по созданию системы или предъявлены более высокие требования к системе.

5. Работа в программе MS Project.

Новый проект в программе MS Project может быть создан как с нуля, так и используя один из предлагаемых стандартных шаблонов. Шаблон представляет собой особенный тип файла проекта, содержащий набор информации, призванной упростить работу над проектом. В состав шаблона обычно входит список заранее организованных и размещенных определенным образом задач, а также информация о ресурсах, пользовательские представления, календари, отчеты, макросы и т.д. Любая информация, предлагаемая шаблоном, может быть изменена в соответствии с требованиями конкретного проекта. В качестве шаблона также может быть использован созданный ранее проект. При создании проекта из шаблона необходимо выбрать на панели *Консультанта* ссылку *Общие шаблоны*. Далее на вкладке *Шаблоны проектов* выбирается необходимый шаблон.

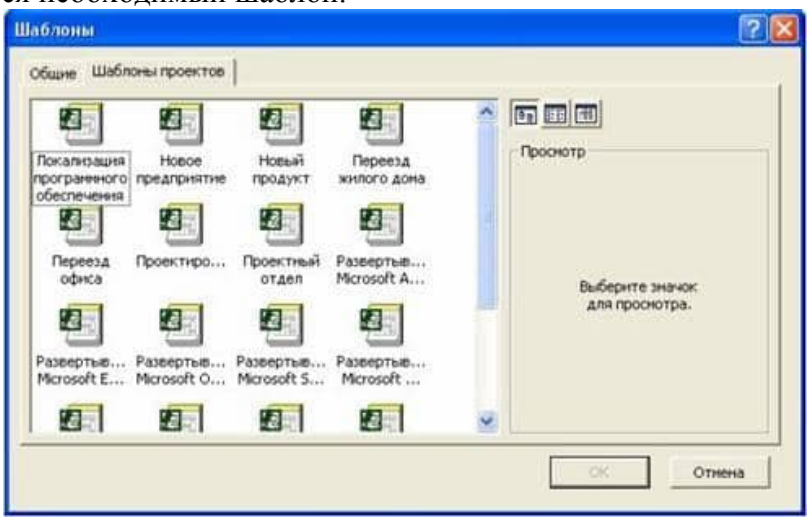


Рисунок 1- Выбор шаблона проекта

Рабочее пространство программы называется видом или представлением. По умолчанию после создания проекта активен вид *Диаграмма Ганта* (рис.2). Данная диаграмма служит для отображения последовательности задач проекта как в текстовом так и в графическом виде.

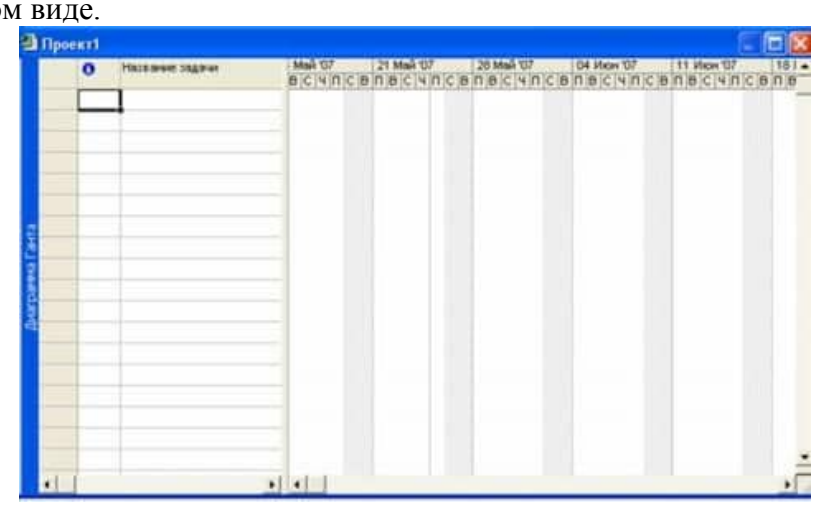


Рисунок 2 - Окно диаграммы Ганта

После создания проекта необходимо настроить его основные параметры. Для этого

удобно использовать мастер *Новый проект.* Для этого нажимаем кнопку *Задачи* на панели *Консультанта* и выбираем ссылку *Определение проекта*. Ответив на вопросы о дате начала проекта и совместной работе над проектом и сохранив результат, выбираем ссылку *Определение рабочего времени проекта* для запуска мастера *Рабочее время проекта*. Таким образом мы можем настроить календарь проекта. Следующим решением, которое необходимо принять на стадии создания, является выбор исходной даты проекта. План проекта может быть составлен от даты начала или завершения проекта. Для настройки планирования от начальной даты выберите в меню Проект пункт сведения о проекте. В появившемся окне (рис.3) выбираем планирование *От даты начала проекта* и ставим *Дату начала*. Да окончания будет рассчитана далее автоматически. В случае планирования от конечной даты выбираем *От даты окончания проекта* и ставим *Дату окончания*. В этом случае автоматически будет рассчитываться дата начала.

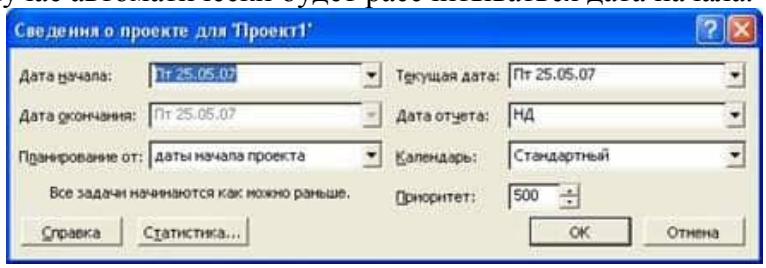


Рисунок 3 - Настройка сведений о проекте

Также в этом окне мы можем выбрать календарь для проекта. В состав пакета MS

Project входит три базовых календаря – стандартный, ночная смена и 24 часа. В *стандартном календаре* рабочий день начинается с 8:00 и заканчивается в 17:00 с обеденным перерывом с 12:00 до 13:00. Рабочая неделя начинается с понедельника и заканчивается в пятницу. Это календарь, применяемый по умолчанию. В *календаре ночной смены* рабочий день начинается с 23:00 и заканчивается в 8:00 с часовым перерывом с 03:00 до 04:00. В *календаре «24 часа»* рабочее время продолжается круглые сутки без выходных и обеденных перерывов. Базовые календари можно редактировать для этого в меню *Сервис* необходимо выбрать пункт *Изменение рабочего времени*. В появившемся окне (рис.4.) выбираем базовое расписание, которое мы хотим отредактировать. Для изменения рабочего времени одного дня необходимо выбрать этот день в календаре. Далее, если необходимо сделать этот день выходным, мы выбираем параметр *нерабочее время*, если же мы хотим только изменить временные рамки рабочего дня, то выбираем параметр *нестандартное рабочее время* и в полях ниже вводим время начала и завершения рабочего дня.

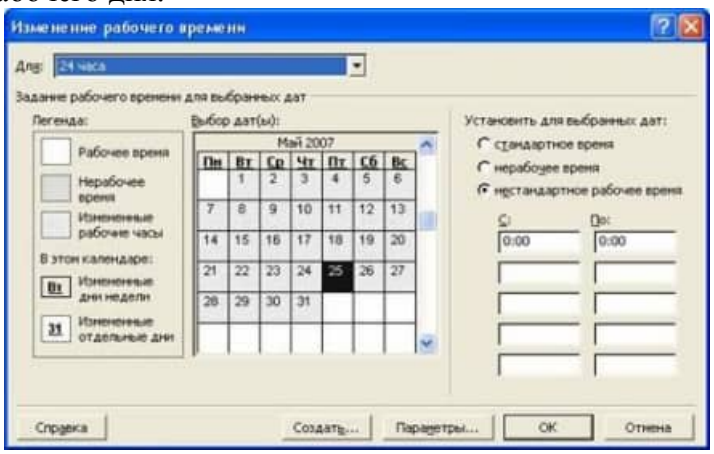


Рисунок 4 - Изменение рабочего времени.

Можно также создать новое базовое расписание. Для этого в окне *Изменение рабочего времени* нажимаем кнопку *Создать*. В появившемся окне (рис.5) выбираем создание нового календаря на основе стандартного или создание копии любого другого

календаря. Значения рабочего времени для вновь созданного календаря могут также быть отредактированы через окно *Изменение рабочего времени*.

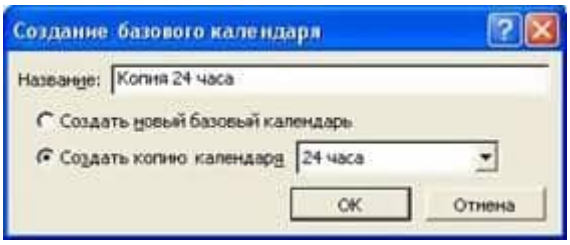


Рисунок 5 - Создание базового календаря.

Создаваемый проект может быть использован в качестве хранилища проектной документации, например, обзора проекта, результатов проведенных анализов или спецификации создаваемого продукта. Для присоединения такой документации целесообразно использовать т.н. суммарную задачу проекта, содержащую итоговую информацию о датах и стоимости проекта. Для отображения суммарной задачи на диаграмме Ганта необходимо в меню *Сервис* выбрать пункт *Параметры* и перейти на вкладку *Вид*. На данной вкладке необходимо выбрать параметр *Показать суммарную задачу проекта* под заголовком *Парметры структуры для проекта*. Суммарная задача появится в нулевом ряду диаграммы Ганта. Проектная документация может как включаться в файл проекта, так и быть доступной через гиперссылки. Для включения документов в файл проекта необходимо выбрать суммарную задачу проекта и нажать кнопку *Сведения о задаче,* расположенную на стандартной панели задач. В открывшемся окне (рис.6) выбираем вкладку Заметки. На вкладке нажимаем кнопку *Вставить объект.* В открывшемся окне необходимо выбрать опцию *Создать из файла*. После этого указываем путь к файлу документа, который предполагается включить в проект.

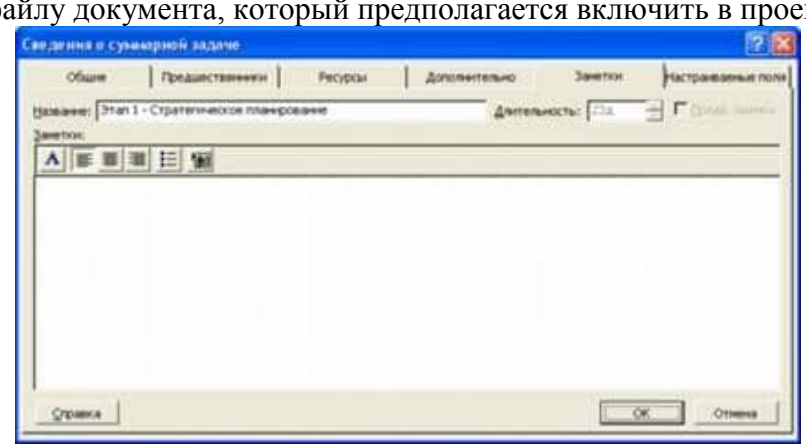


Рисунок 6 - Сведения о задаче.

После закрытия окна сведений о суммарной задаче в диаграмме Ганта появится индикатор примечаний. Для создания гиперссылки к документу необходимо нажать кнопку Гиперссылка на панели задач. В поле Текст открывшегося диалогового окна *Добавление гиперссылки* (рис.7) введите название связываемого документа, затем выберите документ в списке. В поле индикаторов диаграммы Ганта появится индикатор гиперссылок.

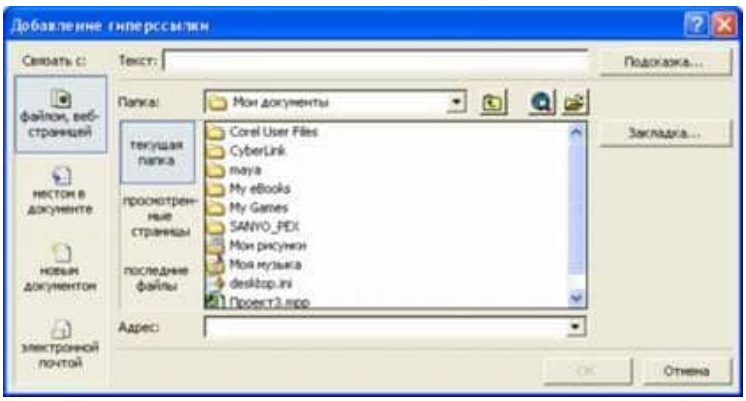


Рисунок 7 - Добавление гиперссылки.

Этапы работы в программе для созданию проекта:

1. С помощью меню кнопки Пуск вызвать приложение Microsoft Project на рабочий стол.

2. Вызвать на поле рабочего окна приложения ранее подготовленный проект, используя библиотеку шаблонов: в области задач Создание проекта перейти на поле группы Создание с помощью шаблона и выбрать гиперссылку Общие шаблоны. На вкладке Шаблоны открыть лист Шаблоны проектов, выбрать шаблон Новый продукт и щелкнуть по кнопке ОК.

3. Сохранить шаблон проекта под новым именем: открыть меню Файл, активизировать команду Сохранить как и перейти на поле диалогового окна Сохранение документа; в диалоговом окне перейти на рабочую папку, записать в поле имени файла новое имя проекта, например FIO_Project, и щелкнуть по кнопке Сохранить.

4. Закрыть пустой проект, т.е. перейти на поле проекта Проект 1, активизировать команду Закрыть в меню Файл.

5. На поле текущего проекта убрать Область задач, для чего следует щелкнуть по кнопке Закрыть, размещенной в правом верхнем углу области.

6. Разместить на рабочем поле различные представления о состоянии проекта: вызвать Панель представлений, для чего в меню Вид щелкнуть по соответствующей команде. Панель представлений позволяет вызвать различные формы представления информации о проекте с помощью соответствующих кнопок; настроить комбинированное представление, используя команду Разделить в меню Окно и соответствующие кнопки панели инструментов, например Диаграмма Ганта и График ресурсов, Использование задач и Использование ресурсов, Диаграмма Ганта и Использование задач. Найти необходимую информацию об использовании ресурсов на Графике ресурсов, данные об их использовании (временной загрузке).

7. Настройка таблицы диаграммы Ганта: снять разделение на рабочей области с помощью команды Разделить из меню Окно и вызвать представление Диаграмма Ганта. Для вызова соответствующего столбца используется меню команды Таблица. Далее следует ознакомиться с основными опциями этого меню; перейти на поле меню Вид и раскрыть меню опций с помощью команды Таблица.

В исходном положении выбрана опция Ввод, которая устанавливает рядом с диаграммой первые два столбца таблицы: Наименование задачи (постоянный столбец) и столбец Длительность задачи; выбрать опцию Гиперссылка — рядом со столбцом задач появится столбец Гиперссылка. В ячейках этого столбца можно записать вспомогательные сведения о задачах путем составления заметок, вложения файлов или формирования гиперссылок на сопутствующую информацию, находящуюся в файле проекта или в других местах. Это позволяет подготовить библиотеки документов и связать их с проектами и задачами. После этого руководители проекта и другие заинтересованные стороны смогут просматривать сопровождающие документы в своих веб-обозревателях; выбрать опцию Затраты. В этом случае появляются три столбца затрат: Фиксированные

затраты, Начисления фактических затрат и Общие затраты. перейти на опцию Использование. На рабочем поле таблицы появятся два столбца: Трудозатраты и Длительность; просмотреть опцию Календарный план. Она вызывает четыре столбца: Начало, Окончание, Позднее начало, Позднее окончание; активировать опцию Отклонение и убедиться, что последние два столбца на поле будут заменены на столбцы Базовое начало и Базовое окончание; вызвать опцию Отслеживание, что позволяет вызвать другую группу из четырех столбцов: Фактическое начало, Фактическое окончание, % завершения и Физический % завершения; щелкнуть по опции Суммарные данные, что позволит установить такую последовательность столбцов: Длительность, Начало, Окончание и % завершения; использование опции Трудозатраты, чтобы одновременно увидеть следующие данные: Трудозатраты, Базовые, Отклонения, Фактические. 8. Настроить таблицу, добавляя необходимые и удаляя лишние столбцы: добавить новые столбцы в таблицу следует в меню Вставка, выбрать команду Столбец и в поле диалога Определение столбца с помощью раскрывающегося списка Имя поля выбрать новое поле, например, Трудозатраты; удалить установленный столбец с помощью контекстного меню, которое следует вызвать щелчком правой клавиши мыши по полю удаляемого столбца. В контекстном меню следует активизировать команду Скрыть столбец.

9. Выполнить фильтрацию данных диаграммы Ганта: выбрать кнопку Другие представления на панели представлений. На поле диалогового окна в списке Представления выбрать строку Подробная Диаграмма Ганта и щелкнуть по кнопке Применить; раскрыть список Фильтр, размещенный на панели форматирования, и щелкнуть по строке Вехи.

10. Выполнить сортировку задач проекта по длительности: в меню Проект выбрать команду Сортировка, в меню опций которой выбрать Сортировать по; в диалоговом окне Сортировка раскрыть список Сортировать по и выбрать в нем строку Длительность. Установить флажок По возрастанию и щелкнуть по кнопке Сортировать; отодвинув поле диаграммы так, чтобы видеть столбец Длительность, убедиться в правильности выполненной операции.

11. Настроить изображение диаграммы Ганта: для настройки формы и цвета отрезков в меню Формат выбрать команду Стили отрезков. В верхней части вкладки выбрать тип задачи и стиль отрезка, который следует изменить. На нижней части вкладки перейти на лист Отрезки, где выполнить операции по изменению стиля отрезка, его формы, узора и цвета; показать текст, который следует разместить рядом с отрезком (информация, отображаемая соответствующим элементом диаграммы). Для этого на вкладке Стили отрезков раскрыть лист Текст и показать, где (слева, справа, снизу, сверху или внутри отрезка) следует разместить текст; настроить шкалу времени диаграммы. Перевести курсор на поле шкалы времени и вызвать контекстное меню, где выбрать опцию Шкала времени. На соответствующей вкладке выбрать уровень шкалы времени (Верхний, Средний, Нижний) и в раскрывающемся списке Отображать выбрать строку Три уровня. Шкала времени может состоять из трех уровней (например, год, квартал, месяц), но обязателен только Средний уровень; перейти на лист Верхний уровень и установить Год в строке Единицы, показать текстовое обозначение года в раскрывающемся списке Надписи, выбрать необходимый стиль форматирования надписи и др.; перейти на лист Средний уровень и выбрать в качестве единицы измерения Квартал; перейти на лист Нижний уровень и установить в качестве единицы измерения времени Месяцы; проверить полученный результат настройки диаграммы Ганта.

12. Сохранить проект в рабочей папке и закрыть приложение.

Задания на работу

Задание 1. Изучить предлагаемый теоретический материал.

Задание 2. Составить подробное описание информационной системы по индивидуальному варианту.

№№ варианта	*Название проекта*
1	ИС «Поликлиника»
2	
3	
4	ИС «Склад»
5	
6	
7	ИС «Аптека»
8	
9	
10	ИС «Предприятие»
11	
12	
13	ИС «Сессия»
14	
15	
16	ИС «Туристическое агентство»
17	
18	
19	ИС «Железнодорожное расписание»
20	
21	
22	ИС «Ремонт квартир»
23	
24	
25	ИС «Квартплата»
26	
27	

Задание 3. На основании описания системы провести анализ осуществимости проекта. 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 не использовалась в организации?*

Результатом анализа должно явиться заключение о возможности реализации проекта.

Задание 4. Создать подгруппу (по 3 человека в каждой) и распределить роли в подгруппе (руководитель проекта-разработчик, системный аналитик-разработчик, тестер-разработчик).

Задание 5. Разработать структуру проекта любым способом и средствами.

Задание 6. Заполнить разделы плана:

- *Введение.*
- *Организация выполнения проекта.*
- *Анализ рисков.*

Разделы должны содержать рекомендации относительно разработки системы, базовые предложения по объему требуемого бюджета, числу разработчиков, времени и требуемому программному обеспечению и т.п.

Задание 7. Оформить отчет (Указать краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.), которые важны для управления проектом,

представить описание информационной системы (ПО) - наличие заключения о возможности реализации проекта, содержащего рекомендации относительно разработки системы, базовые предложения по объему требуемого бюджета, числу разработчиков, времени и требуемому программному обеспечению; представить анализ осуществимости, указать возможные проблемы и пути их решения; описать роли участников группы разработки ПО; программно-аппаратные средства, используемые при выполнении работы).

Контрольные вопросы

1. Что произойдет с организацией, если система не будет введена в эксплуатацию?
2. Какие текущие проблемы существуют в организации и как новая система поможет их решить?
3. Каким образом система будет способствовать целям бизнеса?
4. Требует ли разработка системы технологии, которая до этого не использовалась в организации?

Лабораторная работа №2
«РАЗРАБОТКА МОДУЛЬНОЙ СТРУКТУРЫ ПРОЕКТА (ДИАГРАММЫ МОДУЛЕЙ)»

Цель работы: получить практические навыки в разработке модульной структуры проекта (диаграммы модулей) с использованием инструментальных средств.

Теоретические сведения

1. Общие сведения.

Чтобы добиться декомпозиции на модули максимальной прочности и минимального сцепления, необходимо спроектировать модульную структуру в виде дерева, в том числе и со сросшимися ветвями. В узлах такого дерева размещаются программные модули, а направленные дуги (стрелки) показывают статическую подчинённость модулей, т.е. каждая дуга показывает, что в тексте модуля, из которого она исходит, имеется ссылка на модуль, в который она входит.

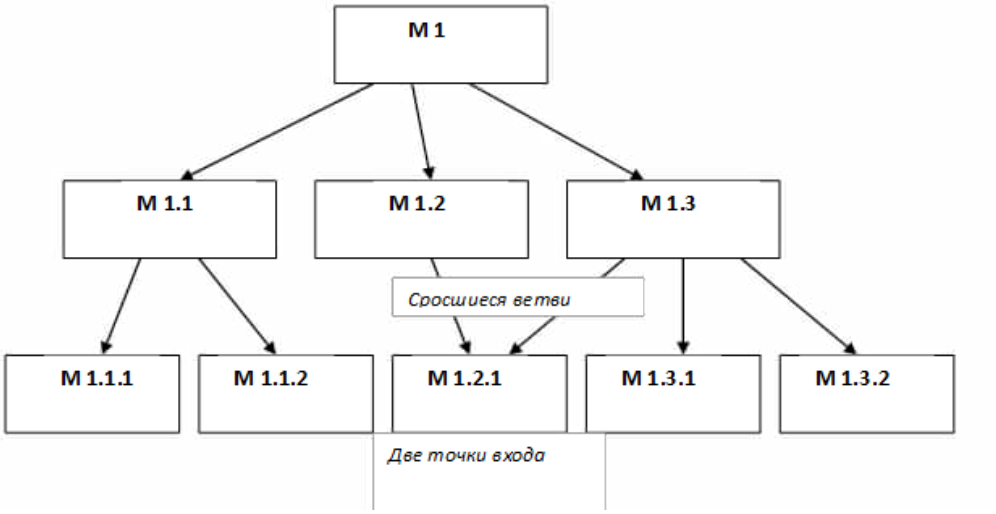


Рисунок 1 - Пример иерархического дерева модулей

При этом модульная структура программы должна, помимо картинки, включать спецификацию программного модуля.

Спецификация программного модуля должна содержать:

- синтаксическую спецификацию его входов (имя модуля, типы передаваемых ему параметров, типы возвращаемых результатов, синтаксис обращения к любому ему входов)
- функциональную спецификацию (описание семантики функций, выполняемых модулем по каждому из его входов).

В процессе разработки модульная структура может по-разному использоваться для определения порядка программирования модулей — восходящая и нисходящая разработка.

В восходящей разработке модули программируются, начиная с нижних уровней, и сразу тестируются, исходя из функциональных спецификаций. Такой порядок представляется естественным, т.к. каждый новый модуль выражается через уже запрограммированные и проверенные модули. Однако современная технология не рекомендует этот прием, т.к. при этом трудно обеспечить концептуальную целостность ПС.

Концептуальная целостность предполагает общие принципы реализации, предположения, структуры данных, - а они могут быть ещё не ясны в начальных стадиях разработки.

Перепрограммирование же модулей нижних уровней связано с большими затратами, т.к. требует не только повторной разработки текстов, но и повторного тестирования.

Предпочтительной является нисходящая разработка. В этой технологии программирование начинается с модуля с самого верхнего уровня. При этом для тестирования модули нижних уровней заменяются простыми по конструкции имитаторами, которые либо моделируют работу нижних уровней (например, реализуют таблицы; вход-отклик), либо просто сообщают о своём вызове и завершаются признаком успеха. Такая реализация обеспечивает большую концептуальную целостность и меньший объём разрабатываемых тестов, каждый модуль здесь тестируется при т.н. «естественном» состоянии информационной среды, т.к. он вызывается реальным (оттестированным) модулем верхнего уровня.

2. Архитектура приложений на Python — модули и пакеты, инструментарий PyCharm Community Edition.

Модульное программирование есть процесс разбиения большой и громоздкой задачи на отдельные, более маленькие, управляемые подзадачи и **модули**. Все это по научному называется декомпозицией. Далее отдельные модули могут быть скомпонованы вместе, как строительные блоки, для создания более крупного приложения, решающего вашу задачу.

У такого, ***модульного*** подхода при проектировании кода больших приложений есть сразу несколько преимуществ:

- ***Простота:*** Вместо того, чтобы думать о всей проблеме в целом, обычно, в модуле фокусируются на решении одной, относительно небольшой, части программы. Работая над одним модулем, сужается область размышлений, что делает разработку проще и менее подверженной ошибкам.
- ***Модифицируемость:*** Обычно, модули имеют логические границы между различными задачами проблемы в целом. Если в модулях свести к минимуму взаимозависимости, то снижается вероятность того, что модификации одного модуля окажут влияние на другие части программы. Возможно, вы даже сможете вносить изменения в модуль, не зная ничего о приложении, для которого он написан. Таким образом, над одним приложением может работать большая группа программистов, что есть совместная разработка.
- ***Повторное использование кода:*** Функциональность, определенная в одном модуле, может быть легко использована повторно (через соответствующий интерфейс) другими приложениями, что избавляет от необходимости дублирования.
- ***Область действия:*** Обычно, в модуле определяется отдельное пространство имен, что помогает избежать коллизий между идентификаторами в разных областях программы.

Есть все конструкции *функций*, *модулей* и *пакетов* в Python, которые способствуют модульному программированию на Python.

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django

PyCharm — это кроссплатформенная среда разработки, которая совместима с Windows, macOS, Linux. PyCharm Community Edition (*бесплатная версия*) находится под лицензией Apache License, а PyCharm Professional Edition (*платная версия*) является проприетарным ПО.

Задания на работу

Задание 1. Определить необходимые программные модули, реализующие ИС в соответствии с индивидуальным заданием из Лабораторной работы №1 и разработать

модульную структуру ИС в среде PyCharm Community Edition (Python) и (или) NetBeans (Java).

Задание 2. Оформить отчет.

Контрольные вопросы

1. В чем заключается процесс разработки модульной структуры проекта (диаграммы модулей)?
2. В каких инструментальных средствах и на каких языках программирования можно создавать программные модули?

Лабораторная работа №3
«РАЗРАБОТКА ПЕРЕЧНЯ АРТЕФАКТОВ И ПРОТОКОЛОВ ПРОЕКТА»

Цель работы: получить практические навыки в разработке перечня артефактов и протоколов проекта с использованием инструментальных средств.

Теоретические сведения

1. Основные сведения по разработке перечня Артефактов.

Разработка ИС включает в себя несколько этапов.

Однако начальным этапом создания системы всегда является изучение, анализ и моделирование бизнес-деятельности организации. На этом этапе вводится и отображается в модели ряд понятий, свойственных объектно-ориентированному подходу.

Исполнитель (Действующее лицо, Actor) – личность, организация или система, взаимодействующая с ИС; различают внешнего исполнителя (который использует или используется системой, т.е. порождает прецеденты деятельности) и внутреннего исполнителя (который обеспечивает реализацию прецедентов деятельности внутри системы). На диаграмме исполнитель представляется стилизованной фигуркой человека.

Для того чтобы описать взаимодействие компании на верхнем уровне с внешними контрагентами, сначала необходимо выяснить, кто (или что) является внешними контрагентами и какие у них основные функции. Для получения ответов на эти вопросы в наглядной форме средствами UML можно составить диаграмму, которую условно назовем «физической», отличающуюся от диаграммы прецедентов (Use Case Diagram) отсутствием прецедентов, наличием только внешних (по отношению к компании) исполнителей (контрагентов) и наименований функций.

Рассмотрим пример разработки перечня Артефактов.

В качестве предметной области будем использовать компанию «Медицина», которая закупает медицинские препараты различный производителей и реализует их через собственную дистрибьюторскую сеть и сеть аптек. Компания осуществляет доставку товаров, как собственным транспортом, так и с помощью услуг сторонних организаций.

Организационная структура компании «Медицина» имеет следующий вид:

Основные бизнес-процессы компании - закупки, складирование запасов, продажи, взаиморасчеты с поставщиками и клиентами.

Ключевые функциональные требования к информационной системе:

1. Управление запасами. Оперативное получение информации об остатках на складе.
2. Управление продажами. Контроль лимита задолженности с возможностью блокировки формирования отгрузочных документов.
3. Управление закупками. Планирование закупок в разрезе поставщиков.
4. Получение управленческих отчетов в необходимых аналитических срезах - как детальных для менеджеров, так и агрегированных для руководителей подразделений, и директоров фирмы.
5. Полный контроль взаиморасчетов с поставщиками и клиентами.

Ограничения проекта. В рамках проекта не рассматривается автоматизация учета основных средств, расчета и начисления заработной платы, управления кадрами. Развертывание новой системы предполагается осуществить только в следующих подразделениях компании:

- Отдел закупок;
- Отдел приемки;
- Отдел продаж;
- Отдел маркетинга;
- Группа планирования и маркетинга;
- Группа логистики;
- Учетно-операционный отдел;
- Учетный отдел;
- Отдел сертификации (в части учета сертификатов на медикаменты);
- Бухгалтерия (только в части учета закупок, продаж, поступлений и платежей).

Описание автоматизируемых бизнес-процессов**.** Бизнес-процессы компании, подлежащие автоматизации, приведены в следующей таблице:

№ п.п	**Код бизнес-процесса**	**Наименование бизнес-процесса**
1.	Закуп-11	Закупки
2.	Склад-22	Запасы-Склад
3.	Прод-33	Продажи
4.	Врасч-44	Взаиморасчеты с поставщиками и клиентами

Каждый бизнес-процесс имеет свой уникальный номер. Нумерация бизнес-процессов построена по следующему принципу: "префикс-номер", где префикс обозначает группу описываемых бизнес-процессов, а номер - порядковый номер бизнес-процесса в списке.

1) Как отмечалось в описании предметной области, компания осуществляет закупки у отечественных и зарубежных производителей, следовательно, контрагентами компании являются отечественные и зарубежные поставщики медикаментов.

2) Компания пользуется услугами транспортных компаний для доставки медикаментов. Следовательно, внешними контрагентами также являются транспортные компании.

3) Кроме того, компания реализует медикаменты через дистрибьюторскую сеть и сеть аптек. Следовательно, контрагентами компании являются различные покупатели.

Таким образом, внешними контрагентами компании являются поставщики (отечественные, зарубежные), покупатели (дистрибьюторы, аптеки) и транспортные компании.

На физической диаграмме компанию изобразим прямоугольником. Для отображения контрагентов используются графический символ Actor (фигурка человечка). Для изображения взаимодействия между компанией и внешними контрагентами используются соединительные линии, поименованные для того, чтобы были понятны функции контрагентов по отношению к компании.

Создание «физической» диаграммы в MS Visio:

1. Запустите MS Visio.

2. Появится окно, в котором необходимо выбрать папку (категорию шаблонов) Программное обеспечение и базы данных / Схема модели UML.

3. В открывшемся списке форм (Фигуры) выбрать пункт Сценарий выполнения UML (т.е. диаграмму Use Case).

В результате проделанных действий на экране появится окно, в левой части которого будет отображен набор графических символов, а в правой части - лист для рисования диаграммы. Общий вид этого окна аналогичен представленному на рис.1, на котором (как и на остальных рисунках) интерфейс этого окна не русифицирован и соответствует ранним версиям MS Visio (см. рис.1).

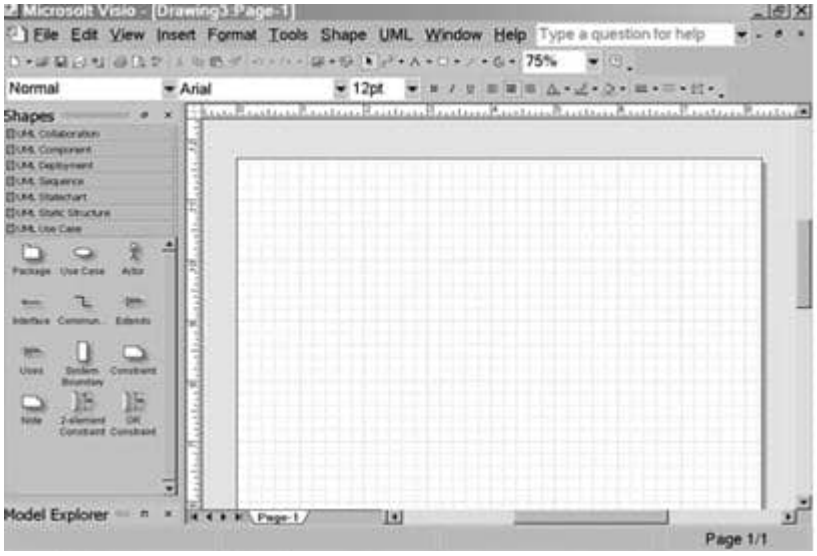


Рисунок 1 - Общий вид окна MS Visio

1. Для изображения границ компании «МЕД» выберите из набора графических элементов, представленных в левой части окна MS Visio, пиктограмму прямоугольника с надписью **«Границы системы»** и переносите ее на рабочее поле мышкой при нажатой правой клавише, Отрегулируйте размеры прямоугольника согласно рис. 2.

2. Для изображения на диаграмме контрагентов следует воспользоваться графическим символом с изображением человечка с надписью **«Актер»** . и так же перенести его на рабочее поле при нажатой правой клавише мышки.

Примечание. Для последующего перемещения графических символов по рабочему полю необходимо зафиксировать пиктограмму «**Указатель**» с изображением стрелки, размещенную на панели инструментов "Форматирование" (в верхней части окна). Только после этого графический символ будет доступен для перемещения его мышкой.

3. Соедините линиями изображение каждого контрагента с прямоугольником. Для этого можно использовать пиктограмму «**Сообщение**», расположенную там же, где и «**Актер**», либо на панели инструментов "Стандартная" щелчком мыши зафиксируйте пиктограмму с изображением линии «**Соединительная линия**» и при нажатой левой клавише мышки осуществите соединение фигур.

4. Внесите наименования контрагентов "Покупатели (аптеки)", "Покупатели (дистрибьюторы)", "Поставщики (Россия)", "Поставщики (импорт)", "Транспортные компании". Для того чтобы внести надписи на диаграмме, необходимо на панели инструментов "Форматирование" зафиксировать пиктограмму «**Текст**» (символ буквы "А"). Щелкните мышкой на изображении человечка, курсор установится на поле с надписью **Актер**. Введите в это поле наименование контрагента.

5. Введите наименование компании "МЕД" в нарисованный прямоугольник, щелкнув мышкой по прямоугольнику. Обратите внимание на то, что при этом должна быть активна пиктограмма «**Текст**» (символ буквы "А").

6. Аналогичным образом внесите надписи к линиям соединения фирмы и контрагентов.

Физическая диаграмма компании представлена на рисунке 2.

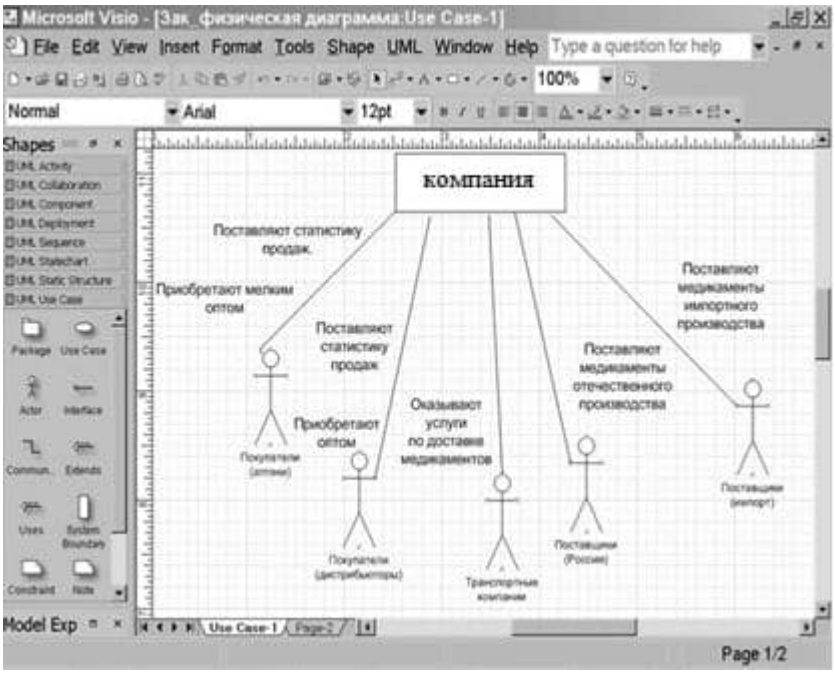


Рисунок 2 - Физическая диаграмма компании.

Построение диаграммы прецедентов. Используя навыки, полученные при выполнении задания 1, построить диаграмму прецедентов, отображающую прецеденты (варианты использования) компании и внутренних исполнителей, обеспечивающих реализацию этих прецедентов внутри системы (см. рис. 3).

Рисунок 3 - Диаграмма прецедентов (вариантов использования) компании.

2. Основные сведения по работе с протоколами.

Для проведения успешного проекта нужно оценить объем предстоящих работ, возможный риск, требуемые ресурсы, предстоящие задачи, определить контрольные точки, стоимость и план работ, которому желательно следовать. Процесс руководство программным проектом включает решение вышеперечисленных задач. Этот процесс начинается перед технической работой, продолжается по мере развития ПО от идея к реальности и достигает наибольшей интенсивности к концу работ. Основной задачей при планировании является определение структуры распределения работ WBS (Work Breakdown Structure) с помощью средств планирования работ (например, MS Project). План выполнения работ составляется на основе декомпозиции проекта вплоть до постановки элементарных задач, которые могут быть выяснены по результатам предварительного анализа. При этом возможно применение содержательных моделей системного анализа. Например, использование модели декомпозиции типа «жизненный цикл» позволит разбивать отдельные задачи на подзадачи путем определения последовательности действий.

Процесс декомпозиции будет определяться принятой моделью жизненного цикла разработки программного обеспечения.

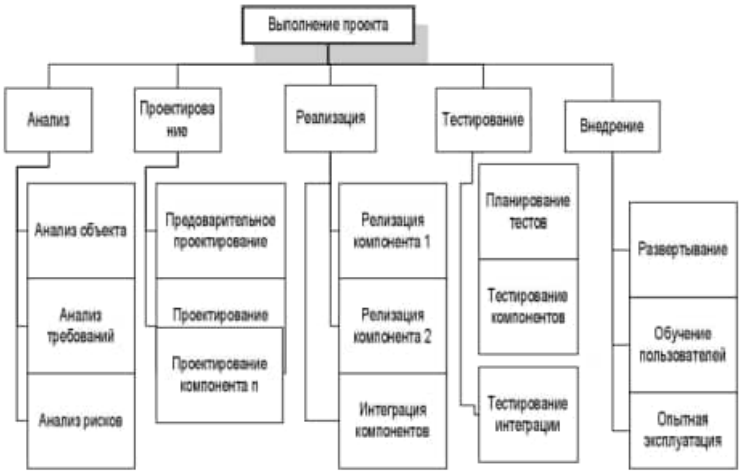


Рисунок 4 - Декомпозиция задач, которые необходимо решить в процессе выполнения проекта по разработке программного обеспечения.

Для каждой элементарной задачи должны быть определены:

- ресурсы, необходимые для решения задачи (в том числе трудовые);
- объем работ, выраженный в принятой системе метрик;
- стоимость работ (может быть вычислена на основе объема работ и стоимости привлекаемых ресурсов);
- длительность работ (может быть вычислена на основе объема работ, количества привлекаемых рудовых ресурсов и принятых нормативов производительности).

Между отдельными элементарными задачами могут быть определенные зависимости, заключающиеся в том, что одни задачи могут выполняться параллельно, другие — в строгой последовательности (для выполнения одних задач могут требоваться результаты выполнения других).

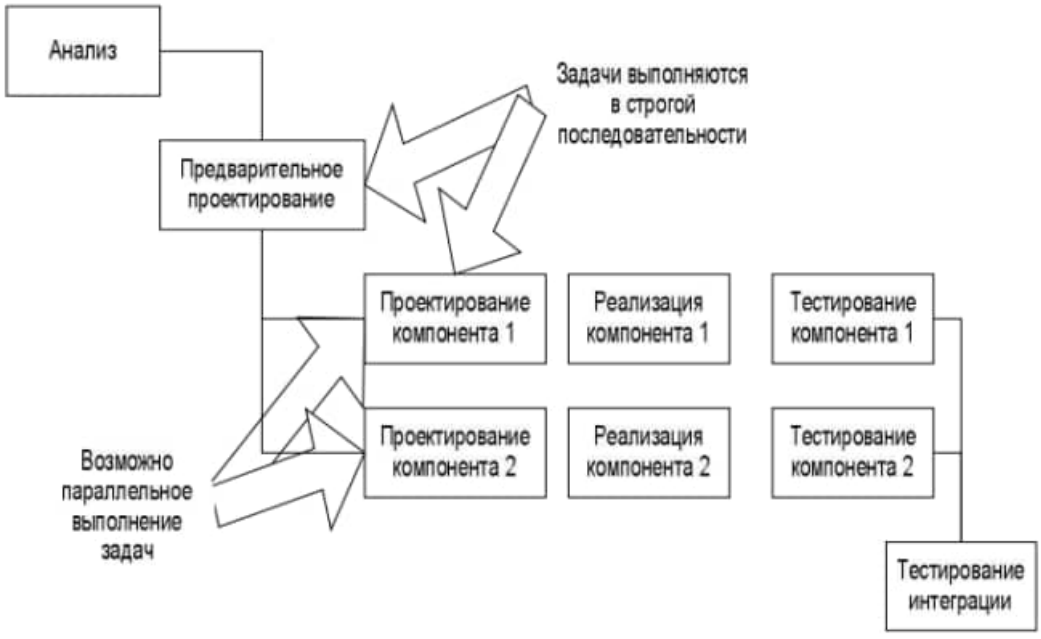


Рисунок 5 - Параллельное и последовательное выполнение задач.

После определения зависимостей можно приступать к распределению элементарных задач по времени. При этом особое внимание следует остановить на задачах, выполняемых параллельно. Параллельность действий повышает требования к планированию.

Необходимо четко отследить наличие ресурсов, необходимых для выполнения каждой задачи. Если план предусматривает использование ресурса 1 для выполнения задач А и Б, то эти задачи не могут выполняться параллельно, даже если между ними нет концептуальной зависимости. Кроме того, руководитель проекта должен знать задачи, лежащие на критическом пути. Для того чтобы весь проект был выполнен в срок, необходимо выполнять в срок все критические задачи.

Календарный план помимо распределения задач и ресурсов по времени должен предусматривать процедуры контроля промежуточных результатов. Такие процедуры обычно называют вехами. Очень важно, чтобы вехи были расставлены через регулярные интервалы вдоль всего процесса разработки программного обеспечения. Кроме того, желательно чтобы вехи совпадали со сроком выполнения критических задач. Это даст руководителю возможность не только регулярно получать информацию о текущем положении дел, но и объективно оценивать риски срыва сроков выполнения проекта, принимать оперативные решения по снижению этих рисков. В первую очередь определите основные фазы выполнения проекта.

В основу может быть положена принятая модель жизненного цикла процесса разработки программного обеспечения. Например, при использовании каскадной модели основными фазами будут являться анализ, проектирование, реализация, тестирование, внедрение. Далее определите состав работ внутри каждой фазы, в соответствии с сутью разработки.

Таким образом будет определен состав работ по проекту. Каждая фаза должна заканчиваться вехой – специальной единицей работы, подразумевающей контроль выполнения работ по проекту и достижение некоторого промежуточного или окончательного результата. Далее определите длительность каждой работы, входящей в план работ.

Для определения длительности могут быть использованы различные регрессионные модели (например, COCOMO II), или же может применяться прямой метод оценки. Следующим этапом является определение связей между задачами. В большинстве средств планирования (например, в MS Project), существует четыре вида связей: Связь типа окончание – начало означает, что задача Б не может начаться раньше окончания задачи А (например, если в процессе выполнения задачи Б используются результаты, получаемые при решении задачи А).

Связь типа начало – начало означает, что задача Б не может начаться до тех пор, пока не началось выполнение задачи А. Например, тестирование программного компонента не может начинаться до того, как была начата его разработка, но, в то же время, для написания тестов не обязательно дожидаться окончания разработки этого компонента. Связь типа окончание – окончание означает, что работа Б не может окончиться до тех пор, пока не завершится выполнение работы А. Например, проектирование базы данных не может быть закончено до того, как будет завершено семантическое моделирование предметной области.

Связь окончание – начало означает, что задача Б не может закончиться до того, как началась задача А. Обычно такая связь используется в том случае, когда А является задачей с фиксированной датой начала, которую нельзя изменить. После того, как определен состав работ, нужно определить, кто эти задачи будет выполнять и какое оборудование должно использоваться.

Сформируйте список ресурсов, для каждого ресурса определите название и стоимость его использования. Далее назначьте ресурсы на выполнение конкретных задач. При первом назначении ресурса будут автоматически рассчитаны трудозатраты. В тех

случаях, когда необходимо ускорить выполнение тех или иных задач, на них могут быть назначены дополнительные ресурсы. После распределения ресурсов необходимо выполнить выравнивание их нагрузки. В тех случаях, когда на параллельно выполняемые задачи назначается один и тот же ресурс, нагрузка на него может превысить максимально допустимую. Для выравнивания нагрузки установите дополнительные связи между задачами таким образом, чтобы задачи, использующие один и тот же ресурс, выполнялись последовательно.

Задания на работу

Задание 1. Определить перечень артефактов в соответствии с индивидуальным заданием из Лабораторной работы №1 используя инструментальные средства.

Задание 2. На основании индивидуального задания из Лабораторной работы №1 выберите модель жизненного цикла процесса разработки и внедрения ПО, которая, по вашему мнению, в наибольшей степени соответствует рассматриваемой ситуации; выделите основные этапы работ; выделите основные задачи внутри отдельных этапов работ; определите зависимости между задачами; определите порядок выполнения отдельных задач; назначьте исполнителей на решаемые задачи; сбалансируйте нагрузку исполнителей.

Задание 3. Оформите отчет.

Контрольные вопросы

1. Назовите сходства и различия диаграмм прецедентов и контекстных диаграмм?
2. Назовите сходства и различия экторов и внешних сущностей.
3. Для чего используются диаграммы прецедентов (вариантов использования)?
4. Что отображает (представляет) «прецедент» на Диаграмме прецедентов?
5. Что такое «эктор» (актер, действующее лицо), что он отображает на диаграмме прецедентов?
6. Назовите основные типы «экторов».
7. Какие типы отношений (связей) между экторами и прецедентами используются на диаграммах прецедентов?
8. Совпадает ли понятие «эктор» с понятием «физический пользователь»?
9. Что представляет (описывает, отображает) прецедент?
10. Какие типы связей (отношений) допускаются между экторами?
11. Что такое жизненный цикл ПО?
12. В чем разница между параллельным и последовательным выполнением задач?

Лабораторная работа №4

«НАСТРОЙКА РАБОТЫ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ (ТИПОВ ИМПОРТИРУЕМЫХ ФАЙЛОВ, ПУТЕЙ, ФИЛЬТРОВ И ДР. ПАРАМЕТРОВ ИМПОРТА В РЕПОЗИТОРИЙ)»

Цель работы: получить практические навыки настройки работы с системой контроля версий Git (GitHub).

Теоретические сведения

1. О контроле версий

Система контроля версий *(СКВ)* — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Программисты обычно помещают в систему контроля версий исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

Если вы графический или вебдизайнер и хотели бы хранить каждую версию изображения или макета, то пользоваться системой контроля версий будет очень мудрым решением. СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какуюто ошибку, и многое другое. Вообще, если, пользуясь СКВ, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы будут очень маленькими.

2. Локальные системы контроля версий

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, программисты уже давно разработали локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (см. рис. 1).

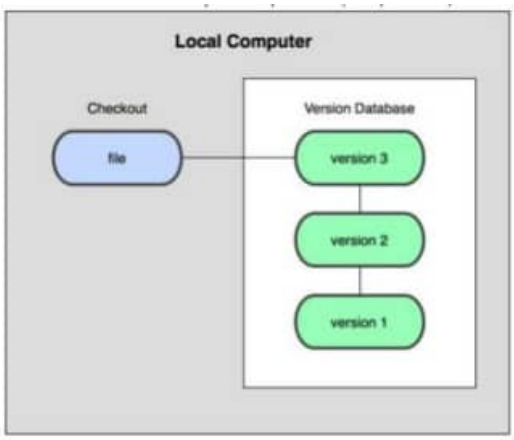


Рисунок 1. Схема локальной СКВ

Одной из наиболее популярных СКВ такого типа является ***rcs***, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий (патч — файл, описывающий различие между файлами), которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

3. Централизованные системы контроля версий

Следующей основной проблемой оказалась необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например ***CVS***, ***Subversion*** и ***Perforce***, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (см. рис. 2).

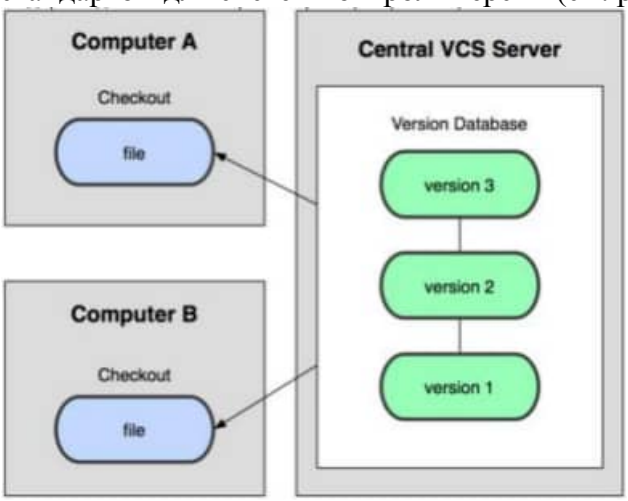


Рисунок 2. Схема централизованного контроля версий

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный — централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё — всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

4. Распределённые системы контроля версий

И в этой ситуации в игру вступают распределённые системы контроля версий (РСКВ). В таких системах как ***Git***, ***Mercurial***, ***Bazaar*** или ***Darcs*** клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий (репозиторий место, где хранятся и поддерживаются какиелибо данные, в данном случае, данные, находящиеся под версионным контролем). Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (см. рис. 3).

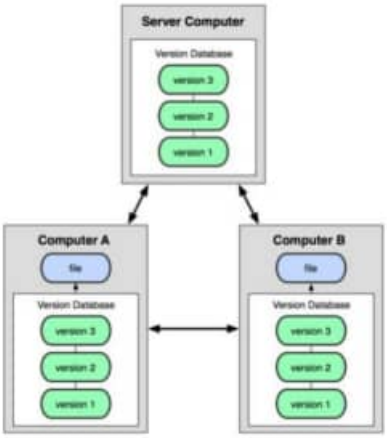


Рисунок 3. Схема распределённой системы контроля версий

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать поразному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

5 Основы Git

Так что же такое Git в двух словах? Эту часть важно усвоить, поскольку если вы поймёте, что такое Git, и каковы принципы его работы, вам будет гораздо проще пользоваться им эффективно. Изучая Git, постарайтесь освободиться от всего, что вы знали о других СКВ, таких как Subversion или Perforce. В Git'е совсем не такие понятия об информации и работе с ней как в других системах, хотя пользовательский интерфейс очень похож. Знание этих различий защитит вас от путаницы при использовании Git'а.

6. Слепки вместо патчей

Главное отличие Git'а от любых других СКВ (например, Subversion и ей подобных) — это то, как Git смотрит на свои данные. В принципе, большинство других систем хранит информацию как список изменений (патчей) для файлов. Эти системы (CVS, Subversion, Perforce, Bazaar и другие) относятся к хранимым данным как к набору файлов и изменений, сделанных для каждого из этих файлов во времени, как показано на рисунке 4.

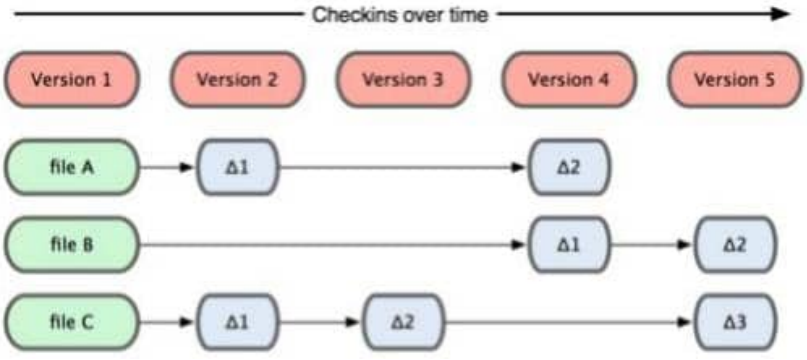


Рисунок 4. Другие системы хранят данные как изменения к базовой версии для каждого файла

Git не хранит свои данные в таком виде. Вместо этого Git считает хранимые данные набором слепков небольшой файловой системы. Каждый раз, когда вы фиксируете текущую версию проекта, Git, по сути, сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, Git не сохраняет

файл снова, а делает ссылку на ранее сохранённый файл. То, как Git подходит к хранению данных, похоже на рисунок 5.

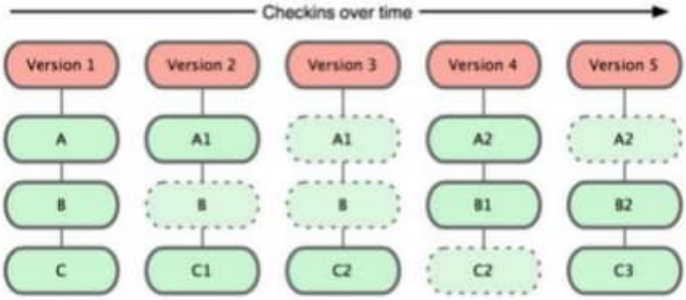


Рисунок 5. Git хранит данные как слепки состояний проекта во времени

Это важное отличие Git'a от практически всех других систем контроля версий. Изза него Git вынужден пересмотреть практически все аспекты контроля версий, которые другие системы переняли от своих предшественниц. Git больше похож на небольшую файловую систему с невероятно мощными инструментами, работающими поверх неё, чем на просто СКВ.

7. Почти все операции — локальные

Для совершения большинства операций в Git'e необходимы только локальные файлы и ресурсы, т.е. обычно информация с других компьютеров в сети не нужна. Поскольку вся история проекта хранится локально у вас на диске, большинство операций кажутся практически мгновенными.

К примеру, чтобы показать историю проекта, Git'у не нужно скачивать её с сервера, он просто читает её прямо из вашего локального репозитория. Поэтому историю вы увидите практически мгновенно. Если вам нужно просмотреть изменения между текущей версией файла и версией, сделанной месяц назад, Git может взять файл месячной давности и вычислить разницу на месте, вместо того чтобы запрашивать разницу у СКВсервера или качать с него старую версию файла и делать локальное сравнение.

Кроме того, работа локально означает, что мало чего нельзя сделать без доступа к Сети или VPN. Если вы в самолёте или в поезде и хотите немного поработать, можно спокойно делать коммиты, а затем отправить их, как только станет доступна сеть. Если вы пришли домой, а VPNклиент не работает, всё равно можно продолжать работать. Во многих других системах это невозможно или же крайне неудобно. Например, используя Perforce, вы мало что можете сделать без соединения с сервером. Работая с Subversion и CVS, вы можете редактировать файлы, но сохранить изменения в вашу базу данных нельзя (потому что она отключена от репозитория).

8. Git следит за целостностью данных

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Эта функциональность встроена в сам фундамент Git'a и является важной составляющей его философии. Если информация потеряется при передаче или повредится на диске, Git всегда это выявит.

Механизм, используемый Git'ом для вычисления контрольных сумм, называется SHA1 хешем. Это строка из 40 шестнадцатеричных символов (09 и af), вычисляемая в Git'e на основе содержимого файла или структуры каталога. SHA1 хеш выглядит примерно так:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

Работая с Git'ом, вы будете встречать эти хеши повсюду, поскольку он их очень широко использует. Фактически, в своей базе данных Git сохраняет всё не по именам файлов, а по хешам их содержимого.

9. Чаще всего данные в Git только добавляются

Практически все действия, которые вы совершаете в Git'е, только добавляют данные в базу. Очень сложно заставить систему удалить данные или сделать чтото неотменяемое. Можно, как и в любой другой СКВ, потерять данные, которые вы ещё не сохранили, но как только они зафиксированы, их очень сложно потерять, особенно если вы регулярно отправляете изменения в другой репозиторий.

10. Три состояния

Это самое важное, что нужно помнить про Git, если вы хотите, чтобы изучение шло гладко. В Git'е файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. "Зафиксированный" значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Таким образом, в проектах, использующих Git, есть три части: каталог Git'а (Git directory), рабочий каталог (working directory) и область подготовленных файлов (staging area).

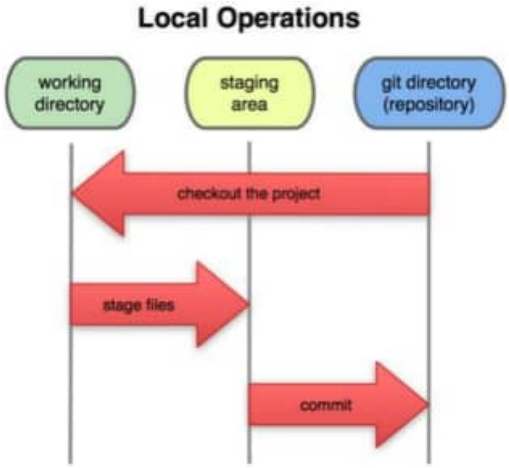


Рисунок 6. Рабочий каталог, область подготовленных файлов, каталог Git'а

Каталог Git'а — это место, где Git хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть Git'а, и именно она копируется, когда вы клонируете репозиторий с другого компьютера.

Рабочий каталог — это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге Git'а и помещаются на диск для того, чтобы вы их просматривали и редактировали.

Область подготовленных файлов — это обычный файл, обычно хранящийся в каталоге Git'а, который содержит информацию о том, что должно войти в следующий коммит. Иногда его называют индексом (index), но в последнее время становится стандартом называть его областью подготовленных файлов (staging area).

Стандартный рабочий процесс с использованием Git'а выглядит примерно так:

1.Вы вносите изменения в файлы в своём рабочем каталоге.

2.Подготавливаете файлы, добавляя их слепки в область подготовленных файлов.

3.Делаете коммит, который берёт подготовленные файлы из индекса и помещает их в каталог Git'а на постоянное хранение.

Если рабочая версия файла совпадает с версией в каталоге Git'a, файл считается зафиксированным. Если файл изменён, но добавлен в область подготовленных данных, он подготовлен. Если же файл изменился после выгрузки из БД, но не был подготовлен, то он считается изменённым.

11. GitHub

GitHub — самый крупный вебсервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git и разработан компанией GitHub, Inc (ранее Logical Awesome).

Сервис абсолютно бесплатен для проектов с открытым исходным кодом и предоставляет им все возможности, а для частных проектов предлагаются различные платные тарифные планы.

Регулярные выражения

Регулярное выражение — это маска или образец последовательности символов. Они используются для поиска и изменения подстрок в тексте. В качестве примеров типичных задач, которые решаются с использованием регулярных выражений, можно привести:

- валидация строки, которая должна представлять собой ip адрес, email или логин;
- замена устаревшего названия на новое;
- подсчет того, сколько раз некоторая конструкция встречается в тексте;
- нахождение в директории файлов с определенным расширением, например, txt.

Таблица регулярных выражений (шпаргалка)

выражение	**значение**
. (точка)	любой символ
аб	последовательность символов
а|б	любой из символов
[абв]	любое из значений в списке
[^абв]	все, кроме значений в списке
[аг]	любое из значений в промежутке
[агАГ]	любой из промежутков
^выражение	начало строки
выражение$	конец строки

\d	цифра
\D	все, кроме цифр
\s	пробельный символ, эквивалентно [\t\n\x0B\f\r]
\S	все, кробе пробельных символов
выражение{количество}	сколько раз встречается, например, а{2}
выражение{мин,макс}	сколько раз встречается, промежуток
выражение{мин,}	сколько раз встречается, минимум
выражение*	0 или более раз
выражение+	1 или более раз
выражение?	0 или 1 раз
(выражение)	использовать группу как единое выражение, например, (аб)+

Основные элементы регулярных выражений

Метасимволы

Большинство символов не изменяют свое значение в регулярном выражении. Например, выражения «ю» или «123» выражают сами себя, то есть строки текста «ю» и «123». Однако, если бы регулярные выражения обозначали только сами себя, от них не было бы большой пользы. Для обозначения сразу группы символов используются метасимволы, например:

- «.» — обозначает любой символ
- \d — обозначает любой символ цифры
- \s — обозначает пробельные символы, например, пробел или табуляция

Из-за того что «.» является метасимволом, для обозначения обычного символа в тексте её и другие метасимволы нужно экранировать с помощью косой черты, «\.». С символами «d» и «s» ситуация обратная, без косой черты они значат сами себя, а с ней становятся метасимволами. Может возникнуть вопрос, а как же быть с самой косой чертой «\»? Для использования косой черты как саму себя её нужно выделять косой чертой. Еще больше осложняет ситуацию то, что в Java, «\» в строке, тоже является специальным символом, для того чтобы передать его методам работы с регулярными выражениями, его тоже нужно выделить косой чертой. Например, регулярное выражение обозначающее цифру «\d» в Java строке будет выглядеть «String regExp = «\\d». Обратите внимание на это “удвоение” числа косых черт при составлении регулярного выражения.

Начало и конец строки

Еще два полезных метасимвола обозначают позицию подстроки в строке:

- ^ — обозначает начало строки
- $ — обозначает конец строки

Например, регулярное выражение «^\d» обозначает цифру в самом начале строки, а «\.$»

обозначает строки оканчивающиеся точкой.

Последовательность и выбор

- a\d – последовательность, сначала символ «a», а потом цифра
- a|\d – выбор, или символ «a», или цифра

Перечисления и промежутки и отрицание

●[абв] — обозначает любой из символов внутри квадратных скобок, «а», «б» или «в»

●[аг] — обозначает любое из значений в промежутке от «а» до «г» то есть «а», «б», «в» или «г»

●[агАГ] — обозначает любой из символов в промежутках от «а» до «г» и от «А» до «Г»

●[^абв] — любые символы кроме «а», «б» и «в»

Группы

Группа позволяет оперировать сразу с целой конструкцией, она обозначается с помощью круглых скобок (), например, «(2\d)» обозначает группу из двух цифр, первая из которых является двойкой. Группы (а также отдельные символы) можно использовать как выражения в квантификаторах, о которых пойдет речь далее.

Квантификаторы

Квантификаторы позволяют указывать сколько раз повторяется выражение

●выражение{количество} — указать сколько раз повторяется выражение, например, (01){3} будет сокращением для «010101»

●выражение{мин,макс} — указать промежуток сколько раз повторяется выражение, например

(010){1,3} обозначает «010», «0100101» или «010010010»

●выражение{мин,} — указать только минимальное количество, максимальное не ограниченно

●выражение* — 0 или более раз

●выражение+ — 1 или более раз

●выражение? — 0 или один раз

Жадные, ленивые и собственнические квантификаторы

Эти определения выражают то, как будет осуществляться поиск регулярного выражения, что необходимо для того, чтобы разрешить возникающие неоднозначности.

●Жадные квантификаторы начинают с того что сопоставляют с выражением как можно большую подстроку. Если это не приводит к успеху, то они на каждом следующем шаге уменьшают свой “аппетит”. Жадность является состоянием по умолчанию для квантификаторов, поэтому нам не нужно её специально указывать.

●Ленивые квантификаторы начинают с как можно меньшей подстроки, постепенно увеличивая её, пока не добьются успеха. Для их обозначения используется символ «?» после квантификатора, например, «выражение*?» или «выражение{3}?».

●Собственнические квантификаторы, как и жадные начинают с наибольшей подстроки, но, в отличии от жадных, на этом и заканчивают. Если сопоставление с наибольшей подстрокой не привело к разрешению регулярного выражения то возвращается несовпадение. Собственнические квантификаторы обозначаются с помощью символа «+», например, «выражение++» или «выражение{1,2}+».

Рассмотрим пример. Пусть у нас есть строка «00183295200». Рассмотрим регулярное выражение «.*00», оно использует жадный квантификатор и найдет в нашей подстроку являющуюся всей строкой, т.е. «00183295200». Ленивое регулярное выражение «.*?00» вернет «00». Собственническое же выражение «.*+00» не вернет вообще ничего, «.*» заберет все символы и «00» сопоставить не удастся.

Обратные ссылки

Круглые скобки помимо логического отделения выражений играют ещё одну роль, а именно создают т.н. группы. Они полезны, когда ваше регулярное выражение состоит из

нескольких одинаковых частей. Тогда достаточно описать один раз однотипную часть шаблона, а потом ссылаться на неё.

Пример:

```
public static void main(String[] args){
Pattern p = Pattern.compile("(якороль).+(\\1)");
Matcher m = p.matcher("регулярные выражения это круто регулярные выражения это круто регулярные выражения это круто якороль Я СЕГОДНЯ ЕЛ БАНАНЫ якороль регулярные выражения это круто" );
if(m.find()){
System.out.println(m.group());
}
}
```

Результатом будет:

```
якороль Я СЕГОДНЯ ЕЛ БАНАНЫ якороль
```

На месте первой группы (якороль) могло содержаться более сложное выражение, тогда обратная ссылка \\1 значительно сократила бы размер регулярного выражения.

Классы для работы с регулярными выражениями

Java классы, предназначенные для работы с регулярными выражениями это ***java.util.regex.Pattern*** и ***java.util.regex.Matcher***. Однако некоторые операции с регулярными выражениями можно выполнить с помощью методов класса java.util.String:

●boolean matches(String regexp)— проверяет, удовлетворяет ли строка регулярному выражению

●String replaceFirst(String regexp, String replacement)— заменяет первую найденную подстроку удовлетворяющую выражению regexp на replacement

●String replaceAll(String regexp, String replacement)— заменяет все подстроки удовлетворяющие выражению regexp на replacement

●String[] split(String regex)— разбивает строку в массив, разделителями служат подстроки, удовлетворяющие регулярному выражению

●String[] split(String regex, int limit)— то же, что и метод выше, плюс позволяет установить предел разбиения

Класс ***java.util.regex.Pattern*** предназначен для хранения скомпилированного регулярного

выражения. Объекты этого класса создаются с помощью статического метода compile:

```
Pattern p = Pattern.compile("\\d+");
```

Также он содержит метод matcher для создания объекта класса java.util.regex.Matcher:

```
Matcher m = p.matcher("a120a0bs");
```

Основные методы класса Matcher:

●boolean find()— найти следующую подстроку, удовлетворяющую выражению

●boolean find(int start)— вариация предыдущего метода, начинает поиск заново, а также позволяет указать с какого по счету символа начать

●String group()— возвращает предыдущую найденную подстроку (группу)

●String group(int group)— возвращает найденную подстроку по индексу

●int groupCount()— возвращает количество найденных подстрок

●int start()— возвращает начальную позицию предыдущей найденной подстроки

●int start(int group)— вариация предыдущего метода, позволяющая указать индекс подстроки (группы)

●int end()— возвращает конечную позицию предыдущей найденной подстроки

●int end(int group)— вариация предыдущего метода, позволяющая указать индекс подстроки (группы)

●Matcher reset()— очищает поиск, позволяет начать все заново

●Matcher reset(CharSequence input)— вариация преддущего метода позволяющая поменять строку в которой ведется поиск

●Matcher usePattern(Pattern newPattern)— указать новый паттерн

Пример, найдем в строке цифр подстроки, состоящие из единичек и выведем начальные и конечные позиции этих подстрок в исходной строке:

```
import java.util.regex.Matcher; import java.util.regex.Pattern;
public class FindOnes {
public static void main(String[] args) { Pattern pattern = Pattern.compile("1+"); String s = "302130032111239021130"; Matcher matcher = pattern.matcher(s); while(matcher.find()){
System.out.println("found: " + matcher.group()); System.out.println("start position: " + matcher.start()); System.out.println("end position: " + matcher.end()); System.out.println();
}
}
}
```

Программа выведет следующее:

```
found: 1
start position: 3 end position: 4
found: 111
start position: 9 end position: 12
found: 11
start position: 17 end position: 19
```

12. Первоначальная настройка Git

Теперь, когда Git установлен в вашей системе, необходимо настроить среду для работы с Git'ом под себя. Это нужно сделать только один раз — при обновлении версии Git'а настройки сохранятся. Но вы можете поменять их в любой момент, выполнив те же команды снова.

В состав Git'а входит утилита git config, которая позволяет просматривать и устанавливать параметры, контролирующие все аспекты работы Git'а и его внешний вид.

Имя пользователя

Первое, что вам следует сделать после установки Git'а, — указать ваше имя и адрес электронной почты. Это важно, потому что каждый коммит в Git'е содержит эту информацию, и она включена в коммиты, передаваемые вами, и не может быть далее изменена:

```
$ git config global user.name "John Doe"
$ git config global user.email johndoe@example.com
```

Проверка настроек

Если вы хотите проверить используемые настройки, можете использовать команду git config list, чтобы показать все, которые Git найдёт:

```
$ git config list user.name=Scott Chacon user.email=schacon@gmail.com color.status=auto
```

13. Создание Git-репозитория

Для создания Gitрепозитория существуют два основных подхода. Первый подход — импорт в Git уже существующего проекта или каталога. Второй — клонирование уже существующего репозитория с сервера.

Создание репозитория в существующем каталоге

Если вы собираетесь начать использовать Git для существующего проекта, то вам необходимо перейти в проектный каталог и в командной строке ввести

$ git init

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git содержащий все необходимые файлы репозитория — основу Gitрепозитория. На этом этапе ваш проект ещё не находится под версионным контролем.

Если вы хотите добавить под версионный контроль существующие файлы (в отличие от пустого каталога), вам стоит проиндексировать эти файлы и осуществить первую фиксацию изменений. Осуществить это вы можете с помощью нескольких команд git add указывающих индексируемые файлы, а

затем commit:

$ git add *.c

$ git add README

$ git commit m 'initial project version'

14. Клонирование существующего репозитория

Если вы желаете получить копию существующего репозитория Git, например, проекта, в котором вы хотите поучаствовать, то вам нужна команда git clone. Если вы знакомы с другими системами контроля версий, такими как Subversion, то заметите, что команда называется clone, а не checkout. Это важное отличие — Git получает копию практически всех данных, что есть на сервере. Каждая версия каждого файла из истории проекта забирается (pulled) с сервера, когда вы выполняете git clone. Фактически, если серверный диск выйдет из строя, вы можете использовать любой из клонов на любом из клиентов, для того чтобы вернуть сервер в то состояние, в котором он находился в момент клонирования.

Клонирование репозитория осуществляется командой git clone [url]. Например, если вы хотите клонировать библиотеку Ruby Git, известную как Grit, вы можете сделать это следующим образом:

```
$ git clone git://github.com/schacon/grit.git
```

Эта команда создаёт каталог с именем grit, инициализирует в нём каталог .git, скачивает все данные для этого репозитория и создаёт (checks out) рабочую копию последней версии. Если вы зайдёте в новый каталог grit, вы увидите в нём проектные файлы, пригодные для работы и использования. Если вы хотите клонировать репозиторий в каталог, отличный от grit, можно это указать в следующем параметре командной строки:

```
$ git clone git://github.com/schacon/grit.git mygrit
```

Эта команда делает всё то же самое, что и предыдущая, только результирующий каталог будет назван mygrit.

В Git'е реализовано несколько транспортных протоколов, которые вы можете использовать. В предыдущем примере использовался протокол git://, вы также можете встретить http(s):// или user@server:/path.git, использующий протокол передачи SSH.

15. Определение состояния файлов

Основной инструмент, используемый для определения, какие файлы в каком состоянии находятся

— это команда git status. Если вы выполните эту команду сразу после клонирования, вы увидите чтото вроде этого:

```
$ git status
# On branch master
nothing to commit, working directory clean
```

Это означает, что у вас чистый рабочий каталог, другими словами — в нём нет отслеживаемых изменённых файлов. Git также не обнаружил неотслеживаемых файлов, в противном случае они бы были перечислены здесь. И наконец, команда сообщает вам на какой ветке (branch) вы сейчас находитесь. Пока что это всегда ветка master — это ветка по умолчанию; в этой лабораторной работе это не важно.

16. Отслеживание новых файлов

Для того чтобы начать отслеживать (добавить под версионный контроль) новый файл, используется команда git add. Чтобы начать отслеживание файла README, вы можете выполнить следующее:

```
$ git add README
```

Если вы выполните команду status, то увидите, что файл README теперь отслеживаемый и индексированный:

```
$ git status
#On branch master
#Changes to be committed:
#(use "git reset HEAD <file>..." to unstage)
# new file: README
#
```

Вы можете видеть, что файл проиндексирован по тому, что он находится в секции “Changes to be committed”. Если вы выполните коммит в этот момент, то версия файла, существовавшая на момент выполнения вами команды git add, будет добавлена в историю снимков состояния. Как вы помните, когда вы ранее выполнили git init, вы затем выполнили git add (файлы) — это было сделано для того, чтобы добавить файлы в вашем каталоге под версионный контроль. Команда git add принимает параметром путь к файлу или каталогу, если это каталог, команда рекурсивно добавляет (индексирует) все файлы в данном каталоге.

Задания на работу

Задание 1. Установите систему контроля версий Git (GitHub).

Задание 2. Выполните ее настройку.

Задание 3. Изучите основные режимы работы с системой контроля версий (работу с различными типами файлов, импортированием файлов, установкой фильтров и т.п.)

Задание 4. Оформите отчет.

Контрольные вопросы

1. Опишите процесс создание Git-репозитория.
2. Опишите процесс создание репозитория в существующем каталоге.
3. Опишите процесс клонирования существующего репозитория.
4. Опишите процесс записи изменений в репозиторий.
5. Опишите процесс определения состояния файлов.
6. Опишите процесс отслеживания новых файлов.

Лабораторная работа №5

«РАЗРАБОТКА И ИНТЕГРАЦИЯ МОДУЛЕЙ ПРОЕКТА (КОМАНДНАЯ РАБОТА)»

Цель работы: получить практические навыки разработки и интеграции модулей проекта в команде с использование инструментальных средств.

Теоретические сведения

1. Язык программирования Python и среда программирования

По синтаксису они очень схожи, по крайней мере, гораздо больше чем с C++. Семантика этих языков чуть ли не одинаковая, а синтаксис большинства операторов различается незначительно.

Программа на Python пишется в файлах с расширением *py*.

Переменные не надо определять/описывать. Тип переменной (например, бывают целые и вещественные переменные) задается присваиванием ей значения. Тип переменной не фиксируется. Новое присваивание может сменить тип переменной. На самом деле, присваивание не присваивает значение, а делает левую часть оператора присваивания ссылкой на объект, создающийся в правой части.

Блоки (в циклах, операторах *if* и т.д.) выделяются одинаковым отступом слева (например, табуляцией).

Пусть последовательность целых чисел записана в файле по одному числу в строке. **Для работы с файлом** его надо открыть (по аналогии с C), последовательно прочитать строки (как строки!), преобразовать каждую строку в целое число и делать с ним что угодно.

В следующем примере распечатываются все числа последовательности, заданной в файле (по одному числу в строке):

```
f=open('1.txt','r') #открыть файл
for line in f:      #цикл по всем строкам файла
    i=int(line)     #преобразовать строку в целое число
    print(i)        #напечатать число
```

Оператор *print* выводит по умолчанию аргументы через пробел с переходом на следующую строку в конце. Это поведение можно изменить:

i=1;j=2;k=3;print(i,j,k, sep=',',end='.')

Будет выведено: *1,2,3.* (без перехода на следующую строку).

Также в функции *print* можно указать, что вывод должен производиться **в файл**:

fout=open("out.txt","w"); print("Hello!\n",file=fout);fout.close();

Арифметические операции, практически, такие же, как в C, но нет операций ++, --. Возведение в степень как в Фортране (радость для тех, кто понимает J): 2**3 дает 8.

Операции сравнения совпадают с C.

Логические операции: *and, or, not*

Пример поиска минимума последовательности:

```
f=open('1.txt','r') #открыть файл
SetFirst=True       #обратите внимание, как пишется True
for line in f:      #цикл по всем строкам файла
    vmin=int(line) if SetFirst else min(vmin,int(line))    #для первой строки взять число в ней, иначе - минимум...
    SetFirst=False
f.close()           #файл, в конце работы, конечно, надо закрыть
print("min=",vmin)
```

Здесь использован оператор

Выражение1 if Логич.Выражение else Выражение2

его значение равно *Выражение1* если *Логич.Выражение* истинно, иначе выражение равно *Выражение2* .

Синтаксис **оператора *if*** можно понять из следующего примера решения задачи поиска суммы всех чисел последовательности, кратных 3:

```
f=open('1.txt','r') #открыть файл
sum=0               #сумма эл-тов, кратных 3, если они есть
n=0                 #количество чисел, кратных 3
for line in f:      #цикл по всем строкам файла
    i=int(line)
    if i%3==0:
        n+=1
        sum+=i
f.close()
if n>0:
    print("sum=",sum)
else:
    print("There are no elements")
```

Не забывайте ставить двоеточие!

Исключения (по аналогии с C++) являются основным средством работы с ошибками. Пример ловли исключения, выбрасывающегося в случае невозможности открытия файла:

```
try:
    f=open('1.txt','r') #открыть файл
    SetFirst=True       #обратите внимание, как пишется True
    for line in f:      #цикл по всем строкам файла
        vmin=int(line) if SetFirst else min(vmin,int(line))    #для п
        SetFirst=False
    f.close()           #файл, в конце работы, конечно, надо закрыть
    print("min=",vmin)
except FileNotFoundError:
    print("File '1.txt' not found")
```

Конечно, желательно решать задачи внутри отдельных **функций**. Пример оформления определения и вызова функции для вышеприведенной задачи:

```
#-------------------------------------
def GetSum3(FileName):
    try:
        f=open(FileName,'r') #открыть файл
        sum=0                #сумма эл-тов, кратных 3, ес
        n=0                  #количество чисел, кратных 3
        for line in f:       #цикл по всем строкам файла
            i=int(line)
            if i%3==0:
                n+=1
                sum+=i
        f.close()
        if n>0:
            return 0,sum
        else:
            print("There are no elements")
            return -1,0
    except FileNotFoundError:
        return -2,0
#-------------------------------------
rez=0;err=0;
err,rez=GetSum3('1.txt')
if err:
    print("Error!")
else:
    print("sum=",rez)
#-------------------------------------
```

В приведенном примере интересно использование множественного присваивания: мы возвращаем из функции сразу несколько значений через запятую и слева от знака присваивания пишем аналогичную конструкцию. Например:

a,b,c,d=1,2,3,4

Подключить модуль из стандартной библиотеки можно с помощью инструкций *import* и *from* :

```
from math import sqrt
a=sqrt(2)
```

или

```
import math
a=math.sqrt(2)
```

Абсолютно аналогично можно подключить собственный модуль (=файл) и использовать подпрограммы из него. Файл с модулем можно размещать в той же папке, что и основная выполняемая программа.

Если числа заданы в файле не по одному числу в строке и/или если между числами возможны различные разделителя, то требуется **разбиение строки на слова**. Для этого используется член класса строки *split* с последующим перебором всех слов из получившегося списка. Например, в случае предыдущей задачи это можно сделать так:

```
#-------------------------------------
def GetSum3(FileName):
    try:
        f=open(FileName,'r') #открыть файл
        sum=0                #сумма эл-тов, кратных 3, если они есть
        n=0                  #количество чисел, кратных 3
        for s in f:      #цикл по всем строкам файла
           for s1 in s.split(' '): #разбиваем строку пробелами
             for s2 in s1.split('\n'): #разбиваем подстроки концами строк
               for s3 in s2.split('\t'): #разбиваем подстроки табуляциями
                 for s4 in s3.split(','):
                   if(s4!=''):
                     i=int(s4)
                     if i%3==0:
                       n+=1
                       sum+=i
        f.close()
        if n>0:
            return 0,sum
        else:
            return -1,0
    except FileNotFoundError:
        return -2,0
#-------------------------------------
rez=0;err=0;
err,rez=GetSum3('2.txt')
if err:
    print("Error!")
else:
    print("sum=",rez)
#-------------------------------------
```

То же самое можно сделать более коротко и с игнорированием ошибочных слов:

```
#-----------------------------------
def GetSum3(FileName):
  sum=0; n=0
  try:
    f=open(FileName,'r')
    for s in [s5 for s1 in f for s2 in s1.split(' ') for s3 in s2.split('\n') for s4 in s3.split(',') for s5 in s4.split('\t') if s5!='']:
      try:
        i=int(s)
        if i%3==0:
          n+=1; sum+=i
      except:
        print('word ',s,' ignored',sep='')
    return 0,sum
  except FileNotFoundError:
    return -2,0
#-----------------------------------
rez=0;err=0;
err,rez=GetSum3('2.txt')
if err:
    print("Error!")
else:
    print("sum=",rez)
#-----------------------------------
```

Для понимания **работы со строками** приведем пример решения следующей задачи:

Текстовый файл содержит строки, состоящие из слов, разделенных пробелами и табуляциями. Написать функцию, возвращающую самое короткое слово, состоящее из одной цифры в первой позиции слова и далее из букв английского языка, и его длину. Также функция должна возвращать код ошибки.

```
def f(sf):
 try:
  MinLen=-1
  f=open(sf);
  for str in f:
    for s in [s for s1 in str.split(' ') for s2 in s1.split('\t') for s in s2.split('\n') if s!='']:
      if s[1:].isalpha() and s[0].isdigit():
        if MinLen<0:
          ShortestS=s; MinLen=len(s)
        elif len(s) < MinLen:
          ShortestS=s; MinLen=len(s)
  f.close();
  if MinLen<0:
    return "No correct words",0,""
  else:
    return "",MinLen,ShortestS
 except FileNotFoundError:
  return "File not found",0,""

err,rez,word=f("t.txt")
if err!="":
  print(err)
else:
  print("minl=",rez,", word=<",word,">",sep="")
```

Здесь следует обратить внимание, что строки индексируются с 0. Обращаться к символу строки с индексом *k* можно *str[k]*. Обращаться к подстроке в интервале индексов от *from* до *to* (*to* не включается!) следует как *str[from:to]* .

Для строк доступны функции вида *is**, аналогичные функциям языка С.

Аналогом массивов в Python являются *списки*.

Все необходимое для работы с массивами.

Создать пустой **список** *m* (=массив) можно командой

m=[]

Добавить элемент в конец списка:

m.append(v)

Удалить элемент с индексом *i* из списка:

m.pop(i)

Если у вас есть список *m*, то можно обращаться к его элементам так же, как и в языке С. Например: *m[0]=m[1]*

Список является итерируемым объектом, т.е. можно писать:

for x in m:

...

В предыдущем примере фраза

[s for s1 in str.split(' ') for s2 in s1.split('\t') for s in s2.split('\n') if s!=""]

на самом деле, создает список непустых слов в строке, разделенных пробелами, табуляциями и символом перехода на следующую строку.

Ничто не мешает создавать список списков.

Приведем пример программы, которая вводит из файла список отрезков на прямой и проверяет, принадлежит ли отрезок из первой строки файла объединению отрезков, записанных в последующих строках файла.

Для проверки мы напишем функцию, которая проверяет принадлежность отрезка, заданного в первом списке отрезков, объединению отрезков, заданному во втором списке отрезков.

Функция последовательно исключает отрезки, заданные во втором списке отрезков, изо всех отрезков первого списка (изначально в нем содержится один отрезок). При каждом исключении может получаться либо 0, либо 1, либо 2 отрезка. Эти отрезки заносятся в третий список. После исключения одного отрезка второго списка изо всех отрезков первого списка, третий список копируется в первый.

```
from os import _exit as exit
from sys import exit as exit0

def clr(v_:list,l:list):
    v=[v_]
    for x in l:
        v2=[]
        for y in v:# delete x in y
            if x[1]>=y[0] and x[1]<y[1]: v2.append([x[1],y[1]])
            if x[0]>y[0] and x[0]<=y[1]: v2.append([y[0],x[0]])
            if x[0]>y[1] or x[1]<y[0]: v2.append([y[0],y[0]])
        v=v2
    return 1 if len(v)==0 else 0,v



def main():
  l=[]
  try:
    f=open("p.txt","r")
    try:
      for str in f:
        s=[s for s1 in str.split(' ') for s2 in s1.split('\t') for s in s2.split('\n') if s!='']
        if(len(s)>0):
          l.append([float(s[0]),float(s[1])])
    except:
      print("incorrect data");exit(-1);
  except:
    print("can't open file");exit(-2);
  v=l.pop(0)
  rez,v2=clr(v,l)
  if rez:
      print(v," in ",l)
  else:
      print(v," not in ",l," : ",v2)

main()
```

Сортировки. Отсортировать список можно с помощью метода *sort()*:

```
l.sort()
```

В функции сортировки можно задавать функцию, возвращающую ключ сортировки по каждому сортируемому элементу:

```
def K(v):return –v;
l.sort(key=K)
```

В качестве функции, возвращающей ключ, можно использовать **lambda-функции**:

```
l.sort(key=(lambda x:-x))
```

Аналогично можно использовать функцию сортировки *sorted()* , возвращающую отсортированный массив. Функцию можно применять не только для списков, но и для кортежей и словарей:

```
d={1:"one", 2:"two"}; print(sorted(d,key=lambda x:-x))
```

В любом случае функция возвращает список (в случае словаря – список ключей).

Кортежи (*tuple*) полностью аналогичны спискам, за тем исключениям, что их нельзя изменять. Вместо квадратных скобок при задании кортежей используются круглые скобки, или скобки вообще не пишутся.

Преимуществом кортежей над списками является то, что (в отличии от списков) кортежи можно использовать в качестве индексов для **словарей**. Словари можно рассматривать как аналоги списков, но индексировать их можно не только целыми индексами, но и строками, вещественными числами, кортежами. Простейший способ инициализации словаря: *v={key1:value1, key2:value2,…}*

Приведем пример программы, в которой задаются два двумерных массива (=две матрицы), распечатываются значения матриц (для этого создается функция распечатывания матрицы), первая матрица умножается на вторую (с помощью созданной функции) и распечатывается результат. Матрицы задаются с помощью словарей, ключами которых являются кортежи из двух элементов.

```
def print_m(str,a,m,n):
    print(str)
    for i in range(m):
        for j in range(n):
            print(a[i,j]," ",end='')
        print()

def mlt(a,m,n, b):
    r={}
    for i in range(m):
        for j in range(m):
            r[i,j]=0
            for k in range(n):
                r[i,j]+=a[i,k]*b[k,j]
    return r


def main():
    m=5;n=4;a={};b={}# a и b - словари
    for i in range(m):
        for j in range(n):
            a[i,j]=i*j# i,j = кортеж
            b[j,i]=2
    print("a=",a)
    print_m("a=",a,m,n)
    print_m("b=",b,n,m)
    r=mlt(a,m,n, b)
    print_m("a*b=",r,m,m)

main()
```

Здесь используется правило хорошего тона: главная функция оформляется в виде функции *main()*, которая потом вызывается.

В словарях можно обращаться к элементам по ключу по аналогии с массивами. Если элемента с данным ключом нет, то выбрасывается исключение. Присваивание значения элементу с заданным ключом добавляет в словарь соответствующую пару *(ключ,значение)*. Метод *get()* позволяет получить значение по ключу без выбрасывания исключения (если элемента с данным ключом нет, то метод возвращает *None*).

В следующем классическом примере программа выводит количество появлений в файле различных слов, разделенных пробелами:

```
l={}
f=open("p.txt","r");
for str in f:
    for s in [s for s1 in str.split(' ') for s in s1.split('\n') if s!='']:
        l[s]=1 if l.get(s)==None else l[s]+1#get() возвращает значение по ключу
f.close();
print(l)
```

Простая работа с изображениями.

Наиболее просто работать с графическими файлами в рамках библиотеки SciPy. С помощью данной библиотеки изображение можно загрузить как список списков списков (= список списков пикселов, где пиксел = список из трех значений RGB). Например, в следующем примере изображение (я использовал изображение 32 бита на пиксел) транспонируется и слегка снижается яркость (по странному алгоритму).

```
import os
from scipy import misc
im= misc.imread('32.bmp', flatten= 0)
h=len(im)    # высота изображения = длина списка строк изображения
w=len(im[0]) # ширина изображения = длина списка одной строки
# распечатаем размеры изображения и приблизительный размер BMP-файла (4bpp)
print('width=',w,'height=',h, 'size=',w*h*4)
im2=[i for i in range(0,w)] # создаем список для строк для транспонированного изображения
misc.imsave('32x0.bmp', im, format='bmp') # пересохраняем исходное изображение
for ix in range(0,w):
  im2[ix]=[i for i in range(0,h)] # в каждом элементе списка создаем список для строки
  for iy in range(0,h):
    v=(im[iy][ix][0]+0+im[iy][ix][1]+im[iy][ix][2])//3 # яркость пиксела
    im2[ix][iy]=[v//1,v//2,v//3] # помещаем красновато-серое изображение пиксела в трансп-ное изобр-ние
misc.imsave('32x1.bmp', im2, format='bmp') # сохраняем транспонированное изображение
```

Стоит отметить, что работоспособность данного примера зависит от версии библиотеки SciPy. По крайней мере, это работает в первой версии библиотеки. Например, в последней версии Python под Windows для работоспособности примера требуется установить (если он не установлен) *pip* и в командной строке запустить команды

pip install scipy
pip install Pillow
pip install imageio

После этого данный код лучше использовать в следующем виде:

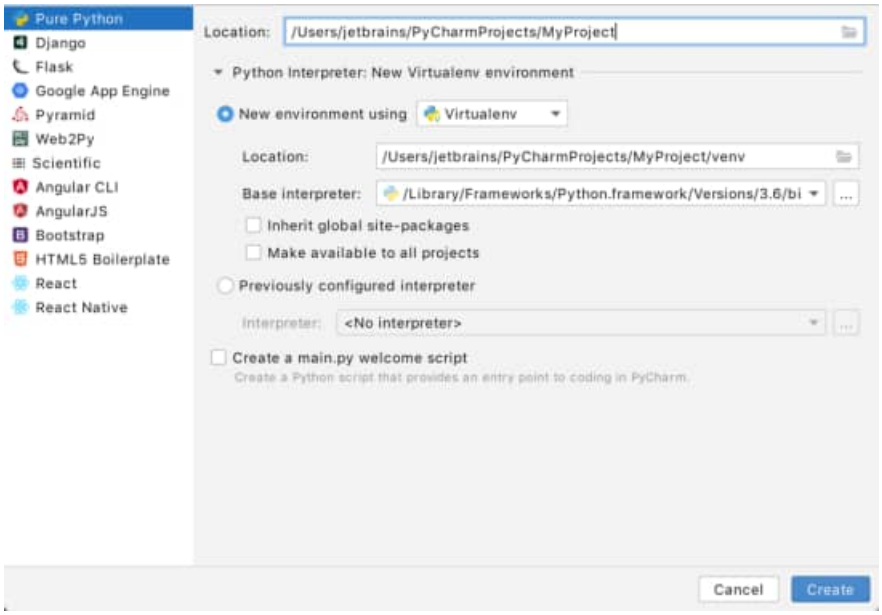
```
import os
from scipy import misc
from numpy import uint8 as u8
from imageio import imread
from imageio import imwrite
im= imread('32.bmp')
h=len(im)    # высота изображения = длина списка строк изображения
w=len(im[0]) # ширина изображения = длина списка одной строки
# распечатаем размеры изображения и приблизительный размер BMP-файла (4bpp):
print('width=',w,'height=',h, 'size=',w*h*4)
im2=[i for i in range(0,w)] # создаем список для строк для транспонированного изображения
imwrite('32x0.bmp', im, format='bmp') # пересохраняем исходное изображение
for ix in range(0,w):
  im2[ix]=[i for i in range(0,h)] # в каждом элементе списка создаем список для строки
  for iy in range(0,h):
    v=((im[iy][ix][0]+0+im[iy][ix][1]+im[iy][ix][2])//3) # яркость пиксела
    # помещаем красновато-серое изображение пиксела в трансп-ное изобр-ние:
    im2[ix][iy]=[u8(v//1),u8(v//2),u8(v//3)]
imwrite('32x1.bmp', im2, format='bmp') # сохраняем транспонированное изображение
```

После установки указанных пакетов старый код тоже будет работать, но с кучей замечаний.

2. Работа в среде PyCharm

Создать проект Python

1. Если вы находитесь на экране приветствия, нажмите « **Новый проект»** . Если у вас уже открыт какой-либо проект, выберите **Файл | Новый проект** из главного меню.
2. Хотя в PyCharm можно создавать проекты различных типов, в этом руководстве давайте создадим простой проект на **чистом Python** . Этот шаблон создаст пустой проект.

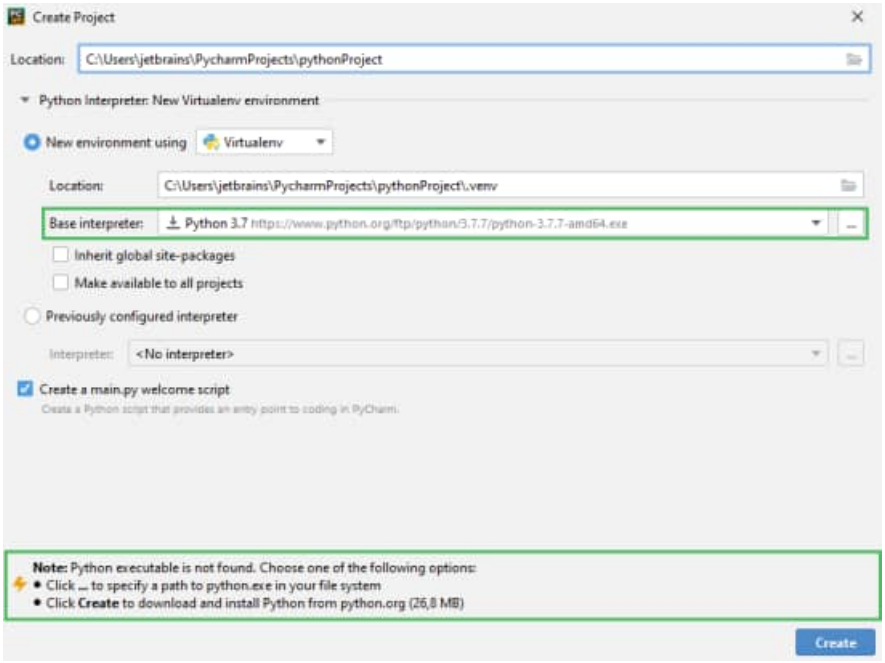


3. Выберите место для проекта. Нажмите кнопку рядом с полем « **Местоположение»** и укажите каталог для вашего проекта.

4. Кроме того, снимите флажок « **Создать приветственный сценарий main.py»,** потому что вы создадите новый файл Python для этого руководства.

5. Лучшая практика Python - создавать виртуальный каталог для каждого проекта. В большинстве случаев PyCharm создает новую виртуальную среду автоматически, и вам не нужно ничего настраивать. Тем не менее, вы можете предварительно просмотреть и изменить параметры venv. Разверните узел **Python Interpreter: New Virtualenv Environment** и выберите инструмент, используемый для создания новой виртуальной среды. Давайте выберем инструмент **Virtualenv** и укажем расположение среды и базового интерпретатора Python, используемого для новой виртуальной среды.

При настройке *базового интерпретатора* необходимо указать путь к исполняемому файлу Python. Если PyCharm не обнаруживает Python на вашем компьютере, он предоставляет два варианта: загрузить последние версии Python с python.org или указать путь к исполняемому файлу Python (в случае нестандартной установки).

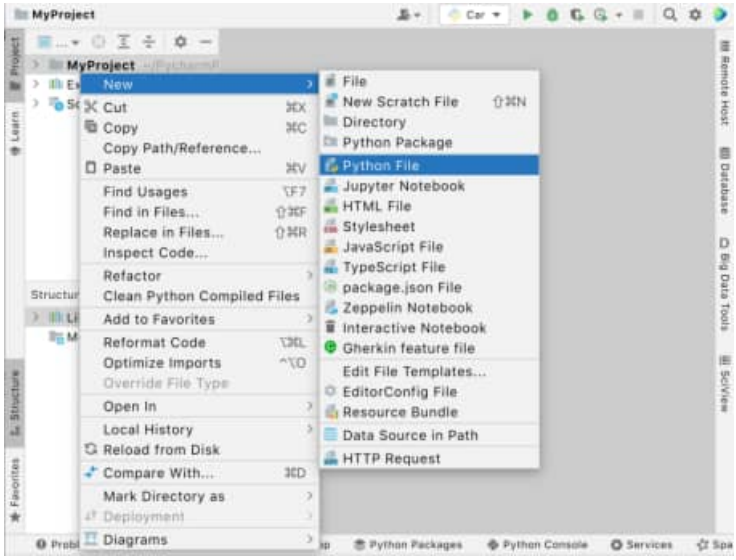


Теперь нажмите кнопку « **Создать»** в нижней части диалогового окна « **Новый проект** ».

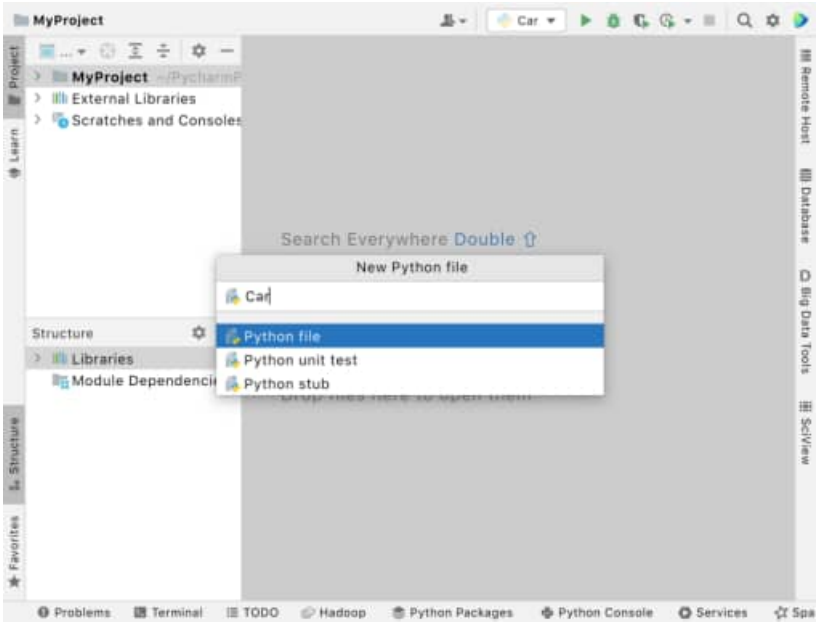
Если у вас уже открыт проект, после нажатия кнопки « **Создать»** PyCharm спросит вас, нужно ли открывать новый проект в текущем окне или в новом. Выберите « **Открыть в текущем окне»** - это закроет текущий проект, но вы сможете открыть его позже.

Создайте файл Python

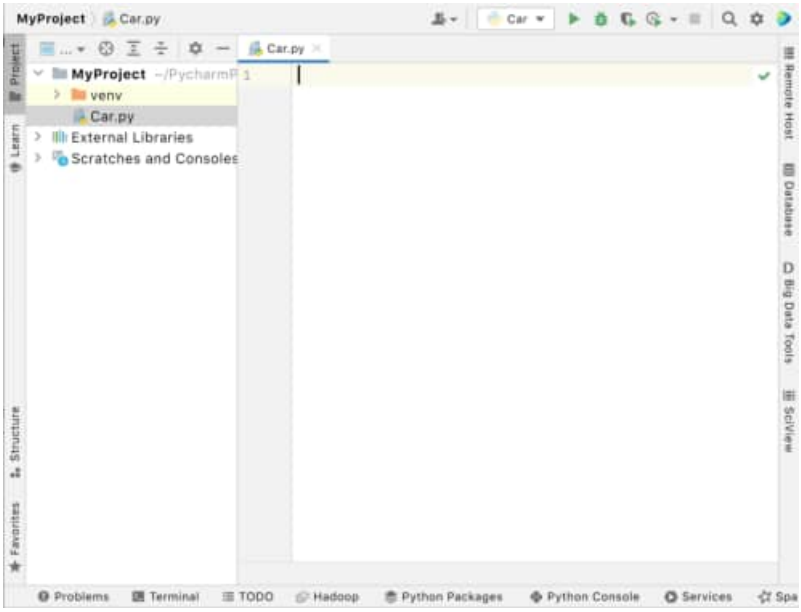
1. В окне инструмента « **Проект»** выберите *корень проекта* (обычно это корневой узел в дереве проекта), щелкните его правой кнопкой мыши и выберите « **Файл» | Новый ...** .



2. Выберите параметр « **Файл Python»** в контекстном меню, а затем введите новое имя файла.



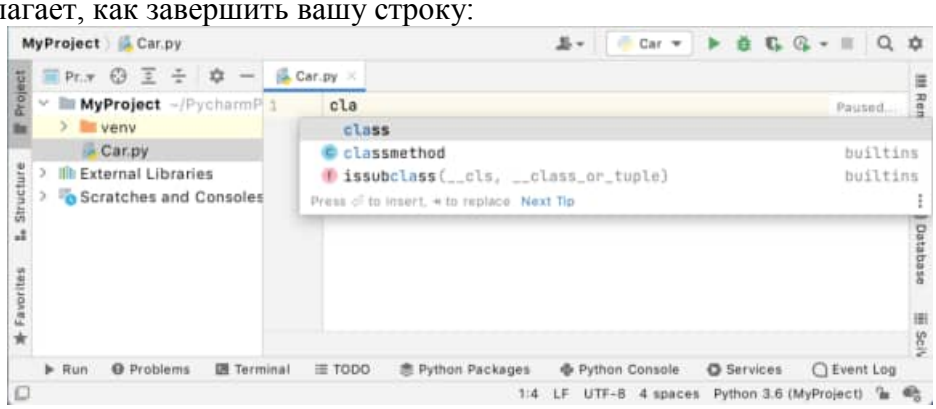
PyCharm создает новый файл Python и открывает его для редактирования.



Редактировать код Python

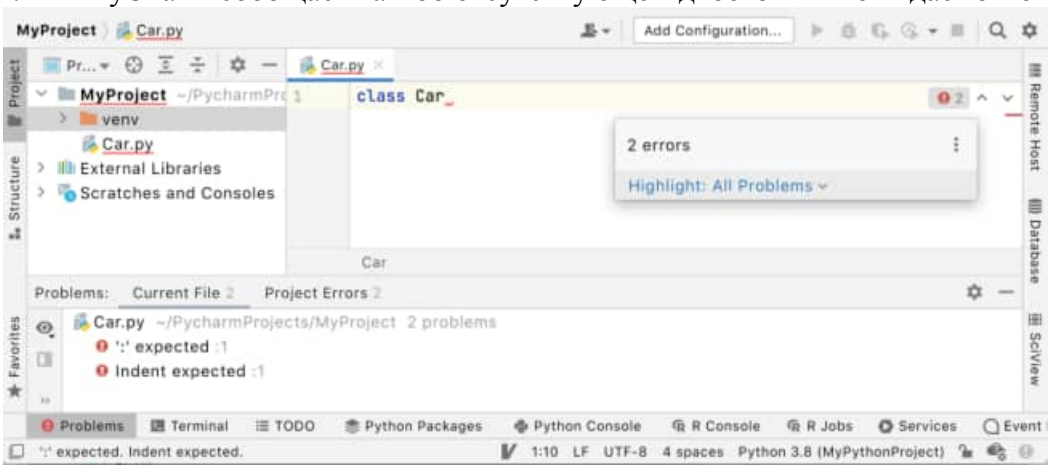
Приступим к редактированию только что созданного файла Python.

1. Начнем с объявления класса. Сразу после того, как вы начнете печатать, PyCharm предлагает, как завершить вашу строку:



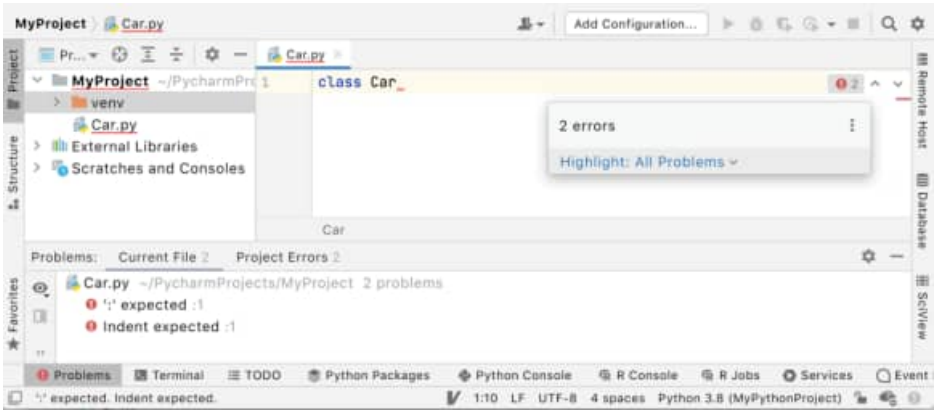
Выберите ключевое слово classи введите имя класса Car.

2. PyCharm сообщает вам об отсутствующем двоеточии и ожидаемом отступе:

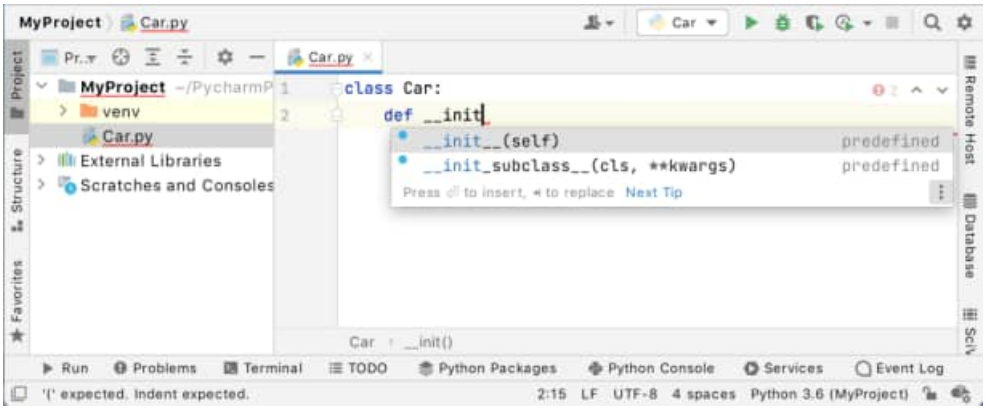


Обратите внимание, что PyCharm анализирует ваш код на лету, результаты немедленно отображаются в индикаторе проверки в правом верхнем углу редактора. Этот индикатор проверки работает как светофор: когда он зеленый, все в порядке, и вы можете

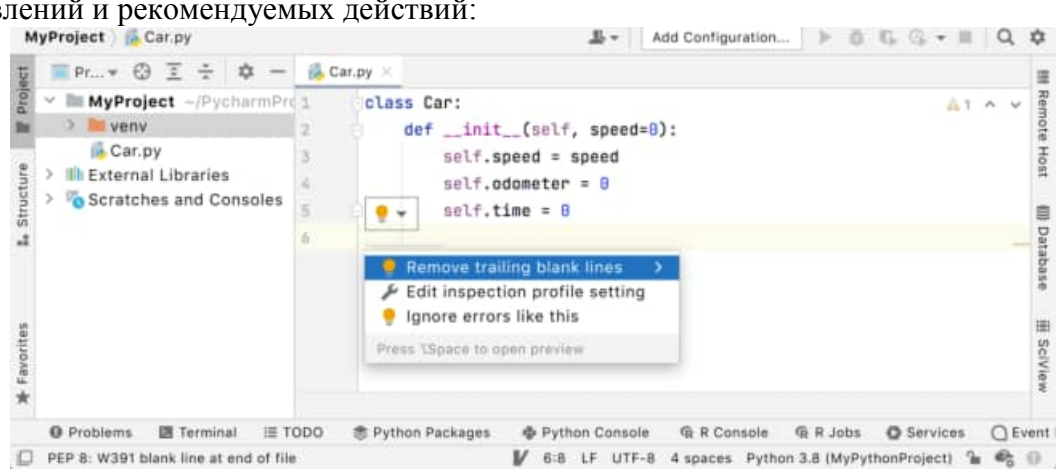
продолжать работу со своим кодом; желтый свет означает незначительные проблемы, которые, однако, не повлияют на компиляцию; но когда горит красный свет, это означает, что у вас серьезные ошибки. Щелкните его, чтобы просмотреть подробные сведения в окне инструмента « **Проблемы»** .



3. Давайте продолжим создание функции __init__: когда вы просто вводите открывающую скобку, PyCharm создает всю конструкцию кода (обязательный параметр self, закрывающую скобку и двоеточие) и обеспечивает правильный отступ.



4. Если вы заметили какие-либо предупреждения проверки при редактировании кода, щелкните значок лампочки, чтобы просмотреть список возможных исправлений и рекомендуемых действий:



5. Скопируем и вставим весь образец кода. Нажмите кнопку копирования в правом верхнем углу блока кода здесь, на странице справки, затем вставьте его в редактор PyCharm, заменив содержимое файла **Car.py** :

Это приложение предназначено для Python 3

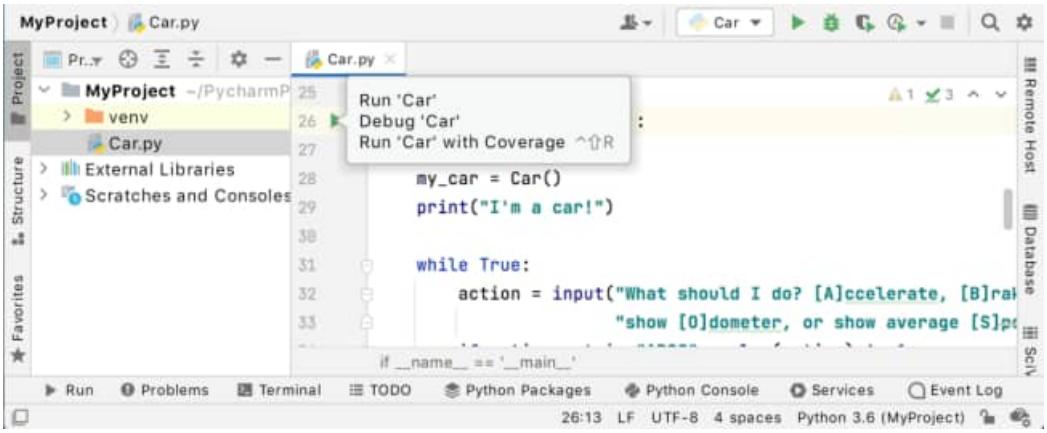
```
class Car:

    def __init__(self, speed=0):
        self.speed = speed
        self.odometer = 0
        self.time = 0

    def say_state(self):
        print("I'm going {} kph!".format(self.speed))

    def accelerate(self):
        self.speed += 5

    def brake(self):
        if self.speed < 5:
            self.speed = 0
        else:
            self.speed -= 5

    def step(self):
        self.odometer += self.speed
        self.time += 1

    def average_speed(self):
        if self.time != 0:
            return self.odometer / self.time
        else:
            pass


if __name__ == '__main__':

    my_car = Car()
    print("I'm a car!")
    while True:
        action = input("What should I do? [A]ccelerate, [B]rake, "
                "show [O]dometer, or show average [S]peed?").upper()
        if action not in "ABOS" or len(action) != 1:
            print("I don't know how to do that")
            continue
        if action == 'A':
            my_car.accelerate()
        elif action == 'B':
            my_car.brake()
        elif action == 'O':
            print("The car has driven {} kilometers".format(my_car.odometer))
        elif action == 'S':
            print("The car's average speed was {} kph".format(my_car.average_speed()))
        my_car.step()
        my_car.say_state()
```

На этом этапе вы готовы запустить свое первое приложение Python в PyCharm.

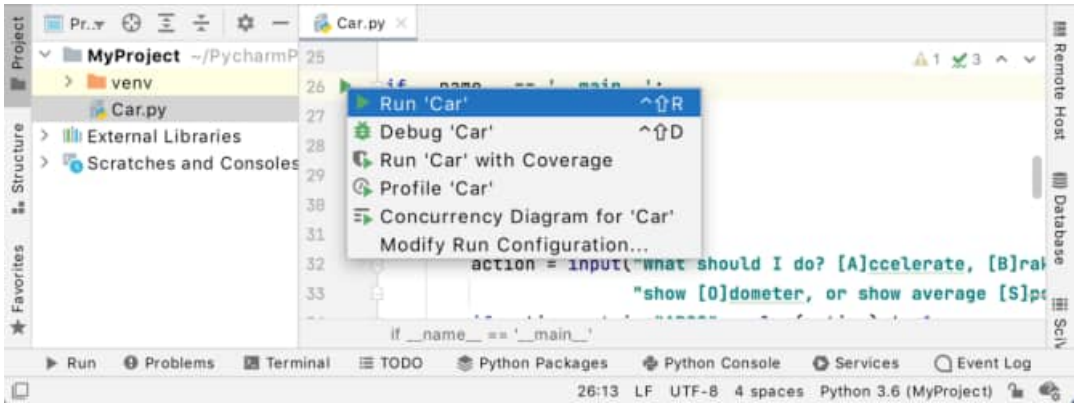
Запустите ваше приложение

Используйте любой из следующих способов для запуска вашего кода:

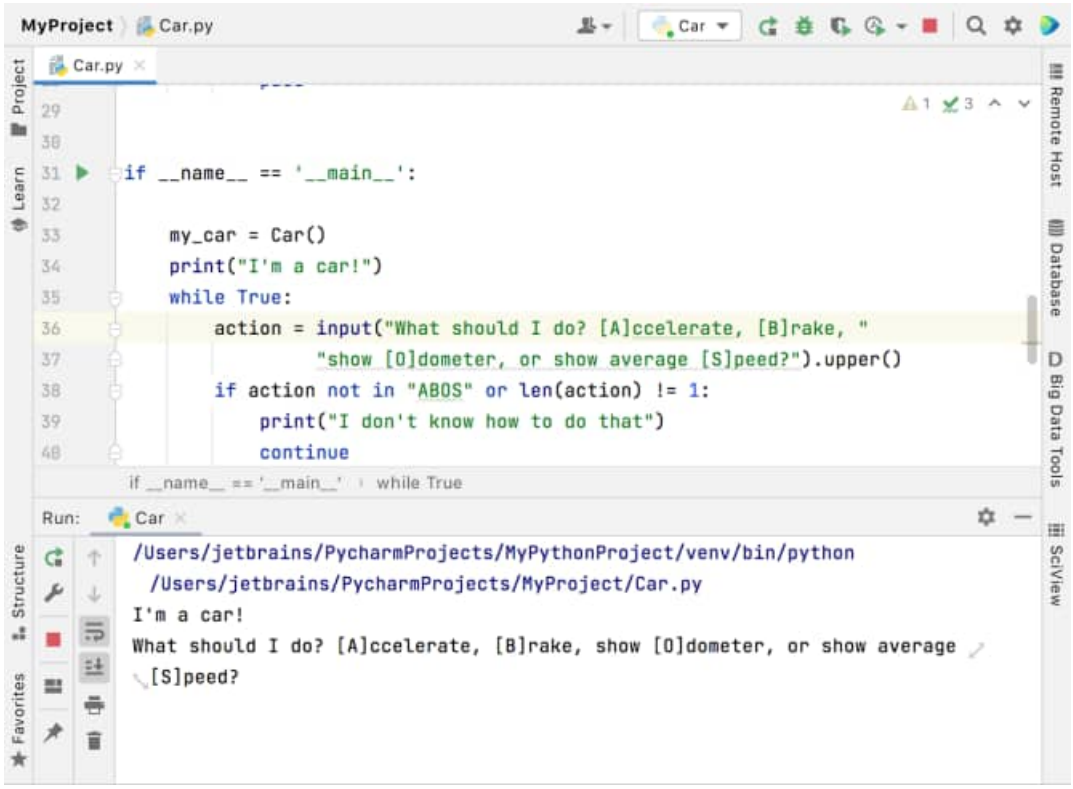
- Щелкните правой кнопкой мыши редактор и выберите в контекстном меню команду « **Выполнить …»** .
- Нажмите Ctrl+Shift+F10.
- Поскольку этот скрипт Python содержит основную функцию, вы можете щелкнуть значок ▶ в желобе. Если вы наведете на него указатель мыши, появятся доступные команды:



Если вы щелкните этот значок, вы увидите всплывающее меню с доступными командами. Выберите **Run 'Car'** :

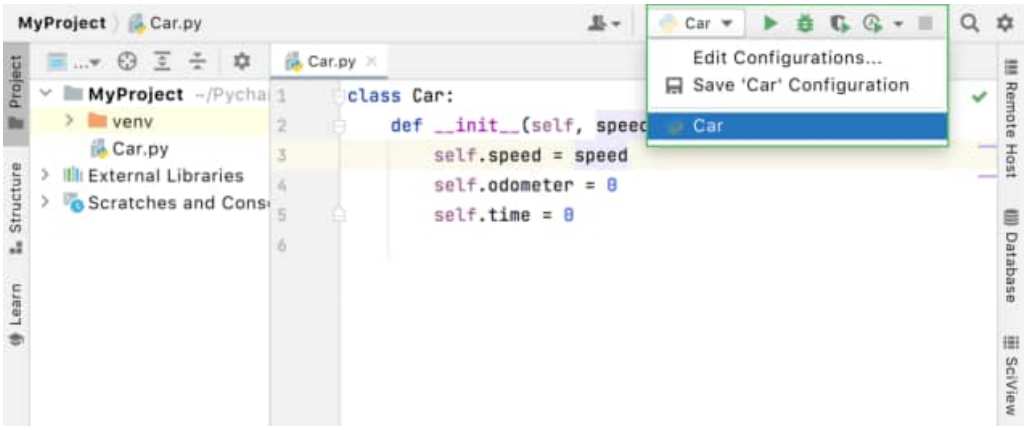


PyCharm выполняет ваш код в окне инструмента «Выполнить» .



Здесь вы можете ввести ожидаемые значения и предварительно просмотреть вывод скрипта.

Обратите внимание, что PyCharm создал временную конфигурацию *запуска / отладки* для файла **Car** .



Конфигурация запуска / отладки определяет способ выполнения вашего кода PyCharm. Вы можете сохранить его, чтобы сделать его *постоянной* конфигурацией, или изменить его параметры.

3. Язык программирования Java и среда программирования

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада и ряда других.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведётся независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

Последние версии NetBeans IDE поддерживают рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun JDK или J2EE SDK подходящей версии. Среда разработки NetBeans по умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 NetBeans поддерживает разработку для мобильных платформ J2ME, C++ (только g++) и PHP без установки дополнительных компонентов.

Язык Java — это объектно-ориентированный язык программирования, ведущий свою историю от известного языка C++. Но в отличие от последнего Java является языком интерпретируемым, программы, написанные на нем, способны работать в разных местах сети и не зависят от платформы, на которой выполняются написанные на нем приложения.

Изучая Java, вы будете приятно удивлены тем, что его синтаксис близок к синтаксису языка C++. Унаследовав самое лучшее от языка программирования C++, язык Java при этом избавился от некоторых недостатков C++, в результате чего на нем стало проще программировать. В этом языке нет, например, указателей, которые сложны в использовании и потенциально могут послужить причиной доступа программы к не принадлежащей ей области памяти. Нет множественного наследования и шаблонов, хотя функциональные возможности языка Java от этого не пострадали. Если вы умеете программировать на C++, для вас не составит особого труда изучить язык Java.

Огромное преимущество Java заключается в том, что на этом языке можно создавать приложения, способные работать на различных платформах. К сети Internet подключены компьютеры самых разных типов - Pentium PC, Macintosh, рабочие станции Sun и так далее. Даже в рамках компьютеров, созданных на базе процессоров Intel, существует несколько платформ, например, Microsoft Windows 2000, Windows NT, Windows XP, OS/2, Solaris, различные разновидности операционной системы UNIX с графической оболочкой XWindows. Приложения Java предназначены для работы на различных платформах и не зависят от конкретного типа процессора и операционной системы.

Программы, составленные на языке программирования Java, можно разделить по своему назначению на две большие группы.

К первой группе относятся приложения Java, предназначенные для автономной работы под управлением специальной интерпретирующей машины Java. Реализации этой машины созданы для всех основных компьютерных платформ.

Вторая группа - это так называемые апплеты (applets). Апплеты представляют собой разновидность приложений Java, которые интерпретируются виртуальной машиной Java, встроенной практически во все современные браузеры.

Приложения, относящиеся к первой группе (мы будем называть их просто приложениями Java), - это обычные автономные программы. Так как они не содержат машинного кода и работают под управлением специального интерпретатора, их производительность заметно ниже, чем у обычных программ, составленных, например, на языке программирования C++. Однако не следует забывать, что программы Java без перетрансляции способны работать на любой платформе, что само по себе имеет большое значение в плане разработок для Internet.

Для повышения производительности приложений Java в современных браузерах используется компиляция "на лету"- Just-In-Time compilation (JIT). При первой загрузке апплета его код транслируется в обычную исполнимую программу, которая сохраняется на диске и запускается. В результате общая скорость выполнения апплета Java увеличивается в несколько раз.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Так же как и библиотеки классов систем разработки приложений на языке C++, библиотеки классов Java значительно упрощают разработку приложений, представляя в распоряжение программиста мощные средства решения распространенных задач. Поэтому программист может больше внимания уделить решению прикладных задач, а не таких, как, например, организация динамических массивов, взаимодействие с операционной системой или реализация элементов пользовательского интерфейса.

Под жизненным циклом мы будем понимать процесс, необходимый для создания работающего приложения. Для программ на Java он отличается от жизненного цикла программ на других языках программирования. (рис.1).

Из рисунка видно, что исходная Java-программа должна быть в файле с расширением java. Программа транслируется в байт-код компилятором javac.exe. Оттранслированная в байт-код программа имеет расширение class. Для запуска программы нужно вызвать интерпретатор java.exe, указав в параметрах вызова, какую программу ему следует выполнять. Кроме того, ему нужно указать, какие библиотеки нужно использовать при выполнении программы. Библиотеки размещены в файлах с расширением jar.

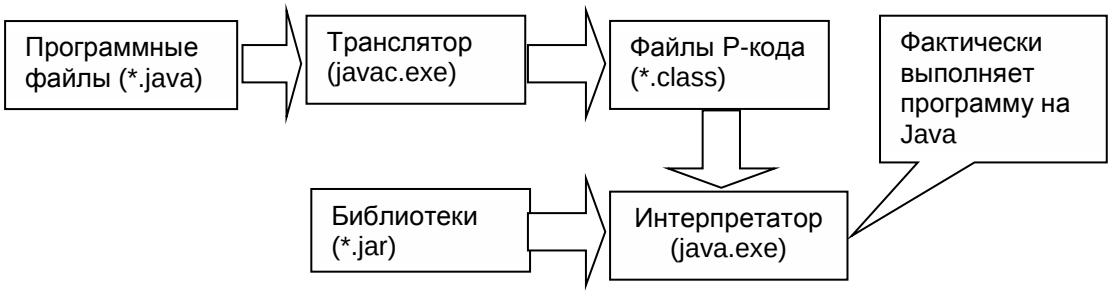


Рисунок 1 - Жизненный цикл программ на языке Java.

Как уже было отмечено, Java - объектно-ориентированный язык. Это дало возможность зафиксировать достаточно компактное ядро языка, ограничив его сравнительно небольшим числом различных синтаксических конструкций, а большую часть возможностей языка вводить с помощью классов. Так, нити включены в язык с помощью классов Thread и ThreadGroup. В виде классов реализованы и такие базовые языковые понятия, как функции обработки строк и обработка исключений. Развитие языка Java тоже ведется путем включения в него новых классов и пакетов (пакеты заменяют в системе программирования Java файлы-заголовки окружения C/C++, однако, в отличие от файлов-заголовков, пакеты содержат как спецификацию классов, так и их реализацию; подробнее о пакетах см. [14]). Такой способ расширения языка удобен тем, что старые компиляторы остаются пригодными и для расширенного языка. Многие новые свойства языка, введенные в него через новые классы и пакеты

Что же было исключено из C++ при разработке Java? Прежде всего, были исключены указатели. Указатели или адреса в памяти - наиболее мощное средство написания высокоэффективных программ в окружении C/C++, но это и наиболее опасное средство этих языков. И дело даже не в том, что, как отмечают авторы языка Java, при недостаточно аккуратном обращении с указателями могут возникать трудно устранимые ошибки в C++-программе. Более существенные трудности в работе с указателями выявляются при разработке распределенных программ, когда требуется осуществить удаленный вызов функции (метода), среди параметров которой есть указатели.

Java является объектно-ориентированным языком программирования. Если сравнивать в этом смысле Java и C++, то между ними есть существенные различия. В ней нет средств, позволяющих писать не объектно-ориентированные программы. Из этого сразу следует один вывод. Нельзя научиться программировать на Java, не овладев основами объектно-ориентированного подхода. Отметим 5 принципов ООР -

● Все является объектом. Все данные программы хранятся в объектах. Каждый объект создается (есть средства для создания объектов), существует какое-то время, потом уничтожается

● Программа есть группа объектов, общающихся друг с другом. Кроме того, что объект хранит какие-то данные, он умеет выполнять различные операции над своими данными и возвращать результаты этих операций. Теоретически эти операции выполняются как реакция на получение некоторого сообщения данным объектом. Практически это происходит при вызове метода данного объекта

● Каждый объект имеет свою память, состоящую из других объектов и/или элементарных данных. Объект хранит некоторые данные. Эти данные - это другие объекты, входящие в состав данного объекта и/или данные элементарных типов, такие как целое, вещественное, символ, и т.п

● Каждый объект имеет свой тип (класс). Т.е. в объектно-ориентированном подходе не рассматривается возможность создания произвольного объекта, состоящего из того, например, что мы укажем в момент его создания. Все объекты строго типизированы. Мы должны сначала описать (создать) тип (класс) объекта, указав в этом описании из каких элементов (полей) будут состоять объекты данного типа. После этого мы можем создавать объекты этого типа. Все они будут состоять из одних и тех же элементов (полей).

● Все объекты одного и того же типа могут получать одни и те же сообщения. Кроме описания структуры данных, входящих в объекты данного типа, описание типа содержит описание всех сообщений, которые могут получать объекты данного типа (всех методов данного класса). Более того, в описании типа мы должны задать не только перечень и сигнатуру сообщений данного типа, но и алгоритмы их обработки

3.1. Базовые типы данных

В языке Java определено восемь базовых типов данных. Для каждого базового типа данных отводится конкретный размер памяти. Этот размер, не зависит от платформы, на которой выполняется приложение Java:

Тип данных	Размер занимаемой области памяти	Значение по умолчанию
boolean	8	false
byte	8	0
char	16	'x0'
short	16	0
int	32	0
long	64	0
float	32	0.0F
double	64	0.0D

Все базовые типы данных по умолчанию инициализируются, поэтому программисту не нужно об этом беспокоиться. Вы можете также инициализировать переменные базовых типов в программе или при их определении.

3.2. Операции (operators) в языке Java

Большинство операций Java просты и интуитивно понятны. Это такие операции, как +, -, *, /, <,> и др. Операции имеют свой порядок выполнения и приоритеты. В выражении сначала выполняется умножение, а потом сложение, поскольку приоритет у операции умножения выше, чем у операции сложения. Но операции Java имеют и свои особенности. Не вдаваясь в детальное описание простейших операций, остановимся на особенностях.

Начнем с присваивания. В отличии от ряда других языков программирования в Java присваивание - это не оператор, а операция. Операция присваивания обозначается символом '='. Она вычисляет значение своего правого операнда и присваивает его левому операнду, а также выдает в качестве результата присвоенное значение. Это значение может быть использовано другими операциями. Последовательность из нескольких операций присваивания выполняется справа налево.

В простейшем случае все выглядит как обычно.

```
x = a + b;
```

Здесь происходит именно то, что мы интуитивно подразумеваем, - вычисляется сумма a и b, результат заносится в x. Но вот два других примера.

```
a = b = 1;    a+b;
```

В первом сначала 1 заносится в b, результатом операции является 1, потом этот результат заносится в a. Во втором примере вычисляется сумма a и b и результат теряется. Это бессмысленно, но синтаксически допустимо

Операции сравнения

Это операции >, <, >=, <=, != и ==. Следует обратить внимание, что сравнение на равенство обозначается двумя знаками '='. Операндами этих операций могут быть арифметические данные, результат - типа boolean.

Операции отношений != и == работают для всех объектов, но их смысл может быть не сразу очевиден. Вот пример

```
Public class Op{
……….
Integer n1 = new Integer(47);
Integer n2 = new Integer(47);
System.out.print (n1 == n2);
System.out.print (n1 != n2);
}
```

Выражение System.out.print (n1 ==n2) выведет результат булевского сравнения, содержащегося в скобках. Конечно, результат должен быть истина (true), а во втором случае – false, так как оба целых имеют одинаковые значения. Но в то время, как содержимое объектов одинаковое, ссылки на них разные а операторы != и == сравнивают именно ссылки. Поэтому результат первого выражения false, а второго – true. Как же действительно сравнить содержимое объектов? Вы должны использовать метод equals(), существующий для всех объектов.

Операции инкремента, декремента

Это операции ++ и -- Так y++ (инкремент) является сокращенной записью y = y +1, аналогично и с операцией декремента (--). Но с этими операциями есть одна тонкость. Они существуют в двух формах - префиксной (++y) и постфиксной (y++). Действие этих операций одно и то же - они увеличивают (операции декремента - уменьшают) свой операнд на 1, а вот результат у них разный. Префиксная форма в качестве результата

выдает уже измененное на 1 значение операнда, а постфиксна - значение операнда до изменения.

3.3. Литералы (константы)

Существуют арифметические (для указания типа константы применяются суффиксы: l (или L) - long, f (или F) - float, d (или D) – double), логические (это true (истина) и false (ложь)), строковые (записываются в двойных кавычках), символьные (записываются в апострофах, например 'F', 'ш')

3.4. Операторы

Оператор	Синтаксис
Выражение	<выражение>
Условный оператор	**if** (<условие>) <оператор1> [**else** <оператор2>]
цикла по предусловию (**while**)	**while** (<условие>) <оператор>
цикла по постусловию (**do while**)	**do** <оператор> **while** (<условие>);
цикла "со счетчиком" (**for**)	**for** (<инициализация>; <условие>; <инкремент>) <оператор>
Операторы **break** и **continue**	Операторы **break** и **continue** являются структурированными аналогами goto. Они могут применяться в циклах, а **break** еще и в операторе выбора (**switch**). Выполнение оператора **break** приводит к немедленному завершению цикла. Оператор **continue** вызывает окончание текущего витка цикла и начало нового
Оператор выбора (**switch**)	**switch** (<выражение>) { **case** <константа1>: <операторы1> **case** <константа2>: <операторы2> . . . [**default**: <операторы_D>]}

3.5. Массивы в Java

В Java есть как одномерные, так и многомерные массивы. Но реализация массивов в Java имеет свои особенности. Во-первых массив в Java это объект. Этот объект состоит из размера массива (поле length) и собственно массива.

Рассмотрим сначала простейшие одномерные массивы базовых типов - int intAry[]; или int[] intAry;. Это описание переменной (или поля) intAry - ссылки на массив. Соответственно, в этом описании размер массива не указывается и сам массив не создается. Как и любой другой объект массив должен быть создан операцией new - intAry = new int[10];

Для массивов допустима инициализация списком значений. int intAry[] = {1, 2, 3, 4}; Здесь описан массив из 4-х элементов и сразу определены их начальные значения. Элементы массивов в Java нумеруются с 0. При обращении к элементу массивы его индексы задаются в квадратных скобках. Java жестко контролирует выход за пределы массива. При попытке обратиться к несуществующему элементу массива возникает IndexOutOfBoundException.

Для двумерных массивов ставится не одна пара скобок, а две, для трехмерных - три и т.д. Например. s = sAray[i][0]; tAray[i][j][k] = 10; Двумерный массив - это массив ссылок на объекты-массивы. Трехмерный массив - это массив ссылок на массивы, которые, в свою очередь, являются массивами ссылок на массивы. Как уже указывалось, массив является объектом, который, в частности, хранит поле length - размер массива. Это позволяет задавать обработку массивов произвольно. Они строятся по принципу "массив массивов".

Возможные способы инициализации массивов:

1. явное создание

```
int ary[][] = new int[3][3];
```

2. использование списка инициализации

```
int ary[][] = new int[][] {
                        {1, 1, 1},
                        {2, 2, 2},
                        {1, 2, 3}, };
```

3. массивы в языке Java являются объектами некоторого встроенного класса, для этого класса существует возможность определить размер массива, обратившись к элементу данных класса с именем length, например:

```
int[] nAnotherNumbers;
nAnotherNumbers = new int[15];
for(int i = 0; i < nAnotherNumbers.length; i++)
{
  nAnotherNumbers[i] = nInitialValue;
}
```

3.6. Работа со строками

Создание строк

Строка в Java является объектом, поэтому ее можно создать, как и любой другой объект, при помощи оператора **new.**

```
String str1 = new String("Строка созданная при помощи оператора new");
```

Также строку можно создать при помощи литерала (фразы заключенной в кавычки) следующим образом.

```
String str2 = "Эта строка создана при помощи литерала.";
```

Обе строки, независимо от способа создания являются объектами — экземплярами класса String.

Важный момент: создание объектов при помощи литерала возможно только в классе String. Объекты любого другого класса при помощи литерала создать нельзя.

Можно также создать массив строк. Например, так:

```
String[] auto = {"Маша", "Петя", "Вася", "Коля"};
```

Конкатенация или слияние строк в Java

Для того, чтобы объединить несколько разных строк в одну, в Java можно использовать перегруженные (специально для объектов String) операторы «+» и «=+».

Еще один важный момент: операторы «+» и «=+», перегруженные для String, являются единственными перегруженными операторами в Java. Программист здесь не имеет возможности самостоятельно перегружать какие-либо операторы (как, например, в C++ и некоторых других языках).

Пример 1:

```
String str1 = "Мама ";
String str2 = "мыла ";
String str3 = "раму";

String result = str1 + str2 + str3;
System.out.print(result);
```

На консоль будет выведено «Мама мыла раму»

Пример 2:

```
String[] animals = {"Хаски","Морж"};// массив строк 1
String[] food = {"колбаски","корж"}; // массив строк 2
//составляем строки из элементов массивов и связующего слова
String result1 = animals[0] + " ест " + food[0];
String result2 = animals[1] + " ест " + food[1];
//выводим на консоль
System.out.println(result1);
System.out.println(result2);
```

Пример 3:

```
String[] auto = {"Волга", "Чайка", "Жигули"}; //задан массив строк
String result = "В гараже стоят машины: "; //задана строка
//прибавляем к строке элементы массива
for(int i = 0; i < auto.length; i++){
  //если элемент не последний, разделяем запятой
  if(i != auto.length-1 )
    result += auto[i] + ", ";
  //если последний, ставим после него точку
  else
    result += auto[i] + ".";
}
//выводим результат
System.out.print(result);
```

Наиболее употребительные методы класса String

При использовании IDE можно легко увидеть, какие методы есть у класса и получить подсказку по их использованию. На примере **IDE Eclipse**: для того, чтобы открыть список методов и быстро выбрать подходящий, нужно после имени переменной поставить точку и нажать комбинацию клавиш **CTRL + Space (пробел).** После этого появится окно, как на рисунке 2, где будут перечислены все доступные методы класса.

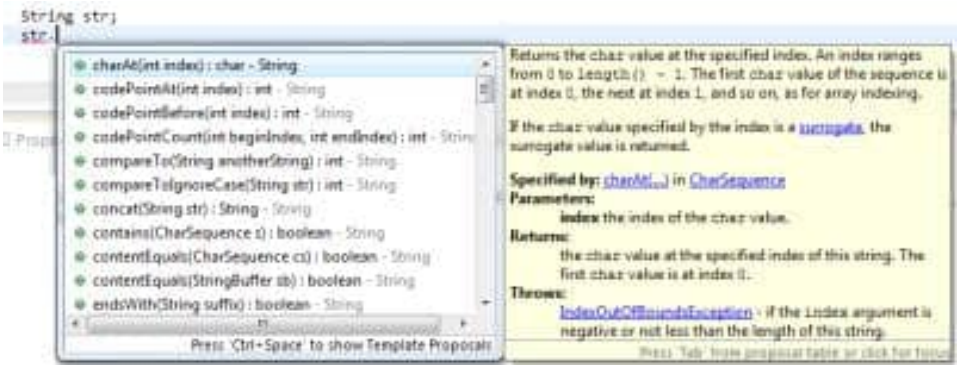


Рисунок 2 – Доступные методы класса

При выборе метода из этого списка, справа (или слева) появится желтое окно с подсказкой по его использованию. При помощи нажатия Enter или двойного клика мыши метод можно вставить в ваш код, не прибегая к ручному набору. Также после имени переменной и точки можно начать набирать вручную имя метода и после введения нескольких первых букв нажать **CTRL + Space (пробел)**. При этом, если метод, начинающийся на эти буквы один, то он автоматически подставится в код, если методов несколько, то откроется окно, как на рисунке 14.1, где будут перечислены только те методы, которые начинаются с этих введенных вами букв. Это было лирическое отступление о том, как облегчить себе жизнь. Далее рассмотрим методы, которые чаще всего используются при работе со строками. Некоторые задачи можно решить и без применения этих методов, но их знание значительно облегчает процесс программирования. В дальнейшем описании, первое слово, которое стоит перед названием метода — тип значения, которое возникнет в результате работы метода (значение, которое метод возвращает).

Конкатенация строк

String concat(String str) — производит ту же конкатенацию, что была описана выше, но использование этого метода из класса String положительно влияет на производительность и скорость программы. На небольших примерах это незаметно и не существенно, но в более серьезных приложениях стоит использовать этот метод. Результатом работы метода будет строка. Параметр, который нужно передавать в метод для конкатенации — тоже строка, о чем нам говорит значение в скобках (String str).

Перепишем пример 2, при помощи concat():

```
String[] animals = {"Хаски","Морж"};// массив строк 1
String[] food = {"колбаски","корж"}; // массив строк 2
//составляем строки из элементов массивов и связующего слова
String result1 = animals[0].concat(" ест ").concat(food[0]);
String result2 = animals[1].concat(" ест ").concat(food[1]);
//выводим на консоль
System.out.println(result1);
System.out.println(result2);
```

Определение количества символов в строке

Для того чтобы определить количество символов в строке, используется метод ***length*.**

int length() **—** возвращает длину строки. Длина равна количеству символов Unicode в строке.

Пример 4:

```
String str = "Строка из букв, цифр 492 и специальных символов %*;№?";
int length = str.length();
System.out.println("Длина строки = " + length);
```

Извлечение символов из строки

Если нам требуется **узнать, какой символ находиться в строке на конкретной позиции**, можем использовать метод ***charAt.***

char charAt(int index) — возвращает символ, находящийся по указанному индексу в строке. Результатом работы метода будет символ типа char. Параметр, который передается в метод — целое число. Первый символ в строке, подобно массивам, имеет индекс 0.

Пример 5: определить последний символ в строке.

```
String str = "Последний символ в этой строке - о";
int last = str.length()-1;//длина строки - 1, так как отсчет начинается с 0
char ch = str.charAt(last);
System.out.println(ch);
```

Если мы хотим **работать со строкой, как с массивом символов**, можем конвертировать строку в массив при помощи метода ***toCharArray***.

char[] toCharArray() — преобразует строку в новый массив символов.

Пример 6: поменять в строке символы пробела на точки при помощи преобразования в массив символов (для этой задачи есть более простое решение, нежели преобразование в массив, но об этом чуть позже).

Примечание: в данном случае мы не сможем использовать метод ***charAt.*** При помощи этого метода мы бы смогли только найти пробелы в строке, но не поменять их.

```
String str = "1 000 000 000";
//преобразовываем строку в массив
char[] chArray = str.toCharArray();
//перебираем все элементы массива
for(int i = 0; i<chArray.length; i++){
      //находим пробел
   if(chArray[i] == ' '){
            //заменяем на точку
               chArray[i] = '.';

      }
}
//выводим результат
System.out.println(chArray);
```

Извлечение подстроки из строки

String substring(int beginIndex, int endIndex) или ***substring(int beginIndex)*** — возвращает новую строку, которая является подстрокой используемой строки. В параметрах метода нужно указать индекс строки, с которого начинается подстрока и

индекс, которым заканчивается. Также возможно указывать только начальный индекс. В этом случае будет возвращена подстрока от начального индекса и до конца строки.

Пример 7.

```
String s = "www.mysite.com";
String name = s.substring(4, s.length()-4);
System.out.println(name); // на консоль выведет "mysite"

String domain = s.substring(4);
System.out.println(domain); // на консоль выведет "mysite.com"
```

Разбиение строк

Для разбиения строк на части используется метод ***String[] split(String regex),*** который разбивает строку на основании заданного регулярного выражения. О регулярных выражениях поговорим в одном из следующих уроков. Здесь покажем пример простого разбиения строки заданного одним символом.

Пример 8.

```
String isbn = "978-3-16-148410-0";
String[] isbnParts = isbn.split("-");

System.out.println("префикс EAN.UCC: " + isbnParts[0]);
System.out.println("номер регистрационной группы: " + isbnParts[1]);
System.out.println("номер регистранта: " + isbnParts[2]);
System.out.println("номер издания: " + isbnParts[3]);
System.out.println("контрольная цифра: " + isbnParts[4]);
```

Поиск в строке

boolean contains(CharSequence s) — проверяет, содержит ли строка заданную последовательность символов и возвращает true или false.

Пример 9.

```
String s = "www.mysite.com";
boolean isContain1 = s.contains("mysite");
System.out.println(isContain1);// нашел - выведет true

boolean isContain2 = s.contains("mysite.ru");
System.out.println(isContain2);// не нашел - выведет false
```

boolean endsWith(String suffix) — проверяет завершается ли строка определенными символами и возвращает true или false.

Пример 10.

```
String s = "www.mysite.com";

//проверяем заканчивается ли строка суффиксом "com"
boolean isComEnding = s.endsWith("com");
System.out.println(isComEnding);//выведет true

//проверяем заканчивается ли строка суффиксом "ru"
boolean isRuEnding = s.contains("ru");
```

```
System.out.println(isRuEnding);//выведет false
```

boolean startsWith(String prefix) или ***startsWith(String prefix, int toffset)*** — проверяет, начинается ли строка с определенных символов. Во втором случае можно указать позицию с которой необходимо начать поиск префикса.

Пример 11

```
String s = "www.mysite.com";

//Проверяем, начинается ли адрес с www
boolean isWWW = s.startsWith("www");

if(isWWW){
/* Если да, проверяем начинается ли имя сайта
с "my". Поскольку адрес начинается с www
проверку начинаем с 4 символа*/
boolean isNameStarts = s.startsWith("my", 4);
}else{
/* Если нет, проверяем начинается ли имя сайта
с "my". Поскольку адрес не начинается с www
проверку производим с начала строки*/
boolean isNameStarts = s.startsWith("my");
}
```

int indexOf(int ch), ***indexOf(int ch, int fromIndex)***, ***indexOf(String str), indexOf(String str, int fromIndex)*** — метод **indexOf** применяется для поиска первого вхождения указанного символа в строке или первого вхождения указанной подстроки. Поиск также можно произвести с указанием позиции в строке от которой нужно начинать искать. Для поиска нужно указать соответствующие параметры. Метод возвращает число соответствующее индексу первого вхождения символа или подстроки. В случае отсутствия указанного символа или подстроки в строке, будет возвращена -1.

Пример 12

```
String data = "name:Igor\nsurname:Kolashnikov\nage:14\ntime:14:55";
//разбиваем строку на несколько подстрок на основании
// встречаемого символа новой строки
String[]lines=data.split("\n");

//проходим каждую подстроку
for (String line : lines){
    //находим индекс первого вхождения символа ":" в подстроке
    int pos = line.indexOf(":");
    //вычленяем имя атрибута из подстроки
    String attributeName= line.substring(0,pos);
    //вычленяем значение атрибута
    String value = line.substring(pos+1,line.length());
    //вывод на экран вычлененных значений в нужном нам формате.
    System.out.println(attributeName + " - " + value);
}
```

int lastIndexOf*(int ch),* ***lastIndexOf****(int ch, int fromIndex),* ***lastIndexOf****(String str),* ***lastIndexOf****(String str, int fromIndex)* — аналогично предыдущему случаю, только ищется последнее вхождение символа или подстроки в строке.

Модификация строк

Модификация строк не является модификацией как таковой. Дело в том, что объекты класса String после создания уже нельзя изменять. Но можно создать копию строки с изменениями. Именно это и делают следующие методы.

toLowerCase() — преобразовать строку в нижний регистр;
toUpperCase() — преобразовать строку в верхний регистр;
trim() — отсечь на концах строки пустые символы;

Пример 13

```
String str = " Я помню ЧУДНОЕ мгновенье ";

//убрали символы пробела в начале и конце строки
str = str.trim();

//я помню чудное мгновенье
System.out.println(str.toLowerCase());

//Я ПОМНЮ ЧУДНОЕ МГНОВЕНЬЕ
System.out.println(str.toUpperCase());
```

String replace(char oldChar, char newChar), replace(CharSequence target, CharSequence replacement) — замена в строке одного символа или подстроки на другой символ или подстроку.

Вспомним **пример 6**, где нужно было поменять в строке символы пробела на точки и перепишем его с использованием ***replace***:

```
String str = "1 000 000 000";
String newStr = str.replace(" ", ".");
System.out.println(newStr);
```

Сравнение строк

boolean equals(Object anObject) — проверяет идентичность строк. Возвращает *true* только в том случае, если в строках представлена одинаковая последовательность символов одной величены.

Пример 14

```
String str = "Я помню ЧУДНОЕ мгновенье";
String str2 = "я помню чудное мгновенье";
//строки не идентичны
System.out.println(str.equals(str2)); //false

//строки идентичны после перевода первой строки
//в нижний регистр
System.out.println(str.toLowerCase().equals(str2)); // true
```

int compareTo(String anotherString) — так же проверяет идентичность строк, однако, в отличии от метода ***equals*** возвращает:

- нулевое значение, если строки равны,
- целое отрицательное число, если первая строка предшествует второй
- целое положительное число, если первая строка следует за второй

Данный метод предназначен для упорядочивания строк. Он позволяет сравнить строки между собой и определить предшествующую строку. Для того, чтобы реализовать такое сравнение метод сравнивает числовые значения букв.

Рассмотрим пример с именами «Маша» и «Миша». При сравнении этих двух имен (пример 15), метод ***compareTo*** укажет, что имя «Маша» предшествует имени «Миша» (выдав отрицательное число) или наоборот, «Миша» следует за «Маша» (выдав положительное число). При упорядочивании имен по алфавиту мы бы упорядочили эти имена именно так. Метод в данном случае определяет, что числовое значение буквы «а» в «Маша» меньше, чем числовое значение «и» в Миша.

Пример 15

```
String name1 = "Маша";
String name2 = "Миша";
System.out.println(name1.compareTo(name2)); //-8
System.out.println(name2.compareTo(name1)); //8
```

Однако, в случае, если мы напишем «маша» с маленькой буквы и попробуем сравнить с «Миша», то получим положительное число.

```
System.out.println("маша".compareTo("Миша")); //32
```

То есть в данном случае имя «Миша» предшествует имени «маша». Это происходит потому, что в таблице символов Юникода буквы верхнего регистра предшествуют нижнему.

Для сравнения строк без учета регистра символов используется функция ***int compareToIgnoreCase(String str)***

```
System.out.println("маша".compareToIgnoreCase("Миша")); //-8
```

Как мы видим, при сравнивании «маша» с «Миша» мы снова получаем отрицательное значение, то есть «маша» предшествует имени «Миша».

На этом закончим знакомство с методами класса String. Это далеко не все методы. Все методы класса String и их описание можно найти в официальной документации http://docs.oracle.com/javase/8/docs/api/java/lang/String.html#method.summary

3.7. Комментарии

В языке Java используются однострочные и блочные комментарии // и /* */, аналогичные комментариям, применяемым в C++ . Введен также новый вид комментария /** */, который может содержать дескрипторы вида:

@author - задает сведения об авторе;
@exception - задает имя класса исключения;
@param - описывает параметры, передаваемые методу;
@return - описывает тип, возвращаемый методом;
@throws - описывает исключение, генерируемое методом.

Из java-файла, содержащего такие комментарии, соответствующая утилита javadoc.exe может извлекать информацию для документирования классов и сохранения ее в виде HTML-документа.

3.8. Первая программа на языке Java

Элементарные строительные блоки в Java называются классами (как и в C++). Класс состоит из данных и кода для работы с ними. В средствах для разработки на языке

Java все стандартные классы, доступные программисту, объединены для удобства в упаковки — еще одни элементарные блоки Java-программ.

Вот простейшая программа, приводимая во многих учебниках по Java:

```
class JavaTest
{
public static void main(String args[])
        System.out.println("Hello, World!");
}}
```

Откомпилируем программу. Если Вы набрали текст правильно, то в результате компиляции на экран будет выведено одно слово Hello. Если же нет, то на экран будут выданы сообщения об ошибках.

Данный пример примитивен, но, тем не менее, на этом примере можно познакомиться с очень важными понятиями.

Рассмотрим, что он демонстрирует:

● весь программный код в Java заключен внутри классов. Не может быть никакого программного текста (за исключением нескольких специальных директив) вне класса (или интерфейса).

● Каждый файл с именем **Name.java** должен содержать класс с именем **Name** (причем, учитывается регистр). Каждый **public**-класс с именем **Name** должен быть в своем файле **Name.java**.

● Внутри указанного файла могут быть и другие классы, но их имена должны отличаться от **Name** и они не должны быть **public**.

● Внутри класса может быть конструкция

```
public static void main(String[] args) {
  . . .
}
```

Это метод класса. Здесь main - имя метода, **public, static, void** - это описатели, Для описания ограничений доступа используются ключевые слова **public, private, protected**. Они являются опциональными описателями и дают нам три варианта ограничений доступа плюс четвертый вариант, если не указан не один из этих описателей.

.● Указанный метод main является специальным случаем. При запуске Java-программы мы указываем имя класса, и Java-машина ищет этот класс среди всех доступных ей файлов *.class, и в этом классе запускает на выполнение метод main

● Описание метода main должно быть в точности таким, как приведено в примере (можно разве что изменить имя args на какое-то другое).

● В скобках после имени метода указываются параметры метода. Для main-метода параметры должны быть такими как указано. Это - массив строк. При вызове программы на Java можно задать параметры вызова. Java-машина обработает их и сформирует массив строк, который будет передан в main-метод в качестве параметра.

4. АБСТРАКТНЫЕ КЛАССЫ И ИНТЕРФЕЙСЫ JAVA

4.1 Абстрактные классы

Абстрактным называется класс, на основе которого не могут создаваться объекты. При этом наследники класса могут быть не абстрактными, на их основе объекты создавать, соответсвенно, можно.

Для того, чтобы превратить класс в абстрактный перед его именем надо указать модификатор abstract.

Рассмотрим пример абстрактного класса A:

```
abstract class A {
int p1;
A() {
```

```
        p1 = 1;
    }
    void print() {
        System.out.println(p1);
    }
}
class B extends A {
}
public class Main {
    public static void main(String[] args) {
        A ob1;
        // ошибка: ob1 = new A();
        B ob2 = new B(); // будет вызван конструктов по умолчанию из A
        ob2.print();
    }
}
```

Java разрешит описать конструкторы в классе A, но не разрешит ими воспользоваться (потому что запрещено создавать объекты абстрактного класса).

Обратите внимание на то, что объявление переменной ob1 как ссылки, на объект класса A тоже не запрещается.

4.2. Приведение классов

Зачем же может потребоваться ссылка ob1, какой объект с ней удастся связать? Ну, например, объект класса-потомка B. Дело в том, что класс A, как родитель, является более универсальным, чем потомок B. Это значит, что любой объект класса потомка может быть явно или даже автоматически приведён к классу родителю.

То есть следующее содержание метода main было бы вполне корректным:

```
        A ob1;
B ob2 = new B();
ob1 = (A) ob2; // явное приведение
ob1.print();
```

Более того, приведение могло быть и неявным (автоматическим):

```
        ob1 = ob2; // автоматическое приведение
```

Как для встроенных типов, так и для классов автоматическое приведение всегда возможно, когда переменную или объект мы пытаемся привести к более универсальному типу (целые числа к вещественным, объект потомка B к классу родителю A и пр.).

4.3. Абстрактные методы

Абстрактным называется метод, который не имеет реализации в данном классе. После круглых скобок, где перечислены его аргументы, ставится не открывающая фигурная скобка, чтобы начать блок описания метода, а точка с запятой. То есть описание у абстрактноно метода отсутствует. Перед именем метода указывается при этом модификатор abstract.

Какой смысл в создании метода без реализации? Ведь его нельзя будет использовать. Для объектов того класса, где метод описан – конечно же использовать нельзя, но вот если унаследовать класс и в потомках переопределить метод, задав там его описание, то для объектов классов потомков метод можно будет вызывать (и работать будут описанные в классах потомках реализации).

Чтобы исключить возможность использования абстрактного метода, в Java введено следующее требование: класс имеющий хоть один абстрактный метод обязан быть абстрактным классом.

Когда же уместно использовать абстрактные методы и классы? Сначала рассмотрим пример иерархии классов домашних животных, где нет ни абстрактных классов, ни абстрактных методов.

```
        class Pet {
    String name;
    int age;
    boolean hungry;
    void voice() {
    }
    void food() {
        hungry = false;
    }
}
class Snake extends Pet {
    double length;
    void voice() {
        System.out.println("Шшш-ш-ш");
    }
}
class Dog extends Pet {
    void voice() {
        System.out.println("Гав-гав");
    }
}
class PatrolDog extends Dog {
    void voice() {
        System.out.println("Ppp-p-p");
    }
}
class Cat extends Pet {
    void voice() {
        System.out.println("Мяу-мяу");
    }
}
class Fish extends Pet {
}
public class Main {
    public static void main(String[] args) {
        Pet zorka = new Pet();
        zorka.food();
        Fish nemo = new Fish();
        nemo.voice();
    }
}
```

Поскольку нет какого-то общего звука, который издавали бы все домашние животные, то мы в классе Pet не стали задавать какую-то реализауию методу voice(), внутри метода не делается совсем ничего, но тем не менее у него есть тело, обособленное блоком из фигурных скобок. Метод voice() хороший претендент на то, чтобы стать абстрактным.

Кроме того, вряд ли можно завести себе домашнее животного неопределенного вида, то есть у вас дома вполне могли бы жить Cat, Dog или даже Snake, но вряд ли вы бы смогли завести животное Pet, являющееся непонятно кем.

Соответсвенно, в рамках реальной задачи вряд ли потребуется создавать объекты класса Pet, а значит его можно сделать абстрактным (после чего, правда, мы даже при делании не сможем создать объекты на его основе).

Рассмотрим пример с участием абстрактного класса и абстрактного метода:

```
abstract class Pet {
   String name;
   int age;
   boolean hungry;
   abstract void voice();
   void food() {
      hungry = false;
   }
}
class Snake extends Pet {
   double length;
   void voice() {
      System.out.println("Шшш-ш-ш");
   }
}
class Dog extends Pet {
   void voice() {
      System.out.println("Гав-гав");
   }
}
class PatrolDog extends Dog {
   void voice() {
      System.out.println("Ppp-p-p");
   }
}
class Cat extends Pet {
   void voice() {
      System.out.println("Мяу-мяу");
   }
}
class Fish extends Pet {
   void voice() {
   }
}
public class Main {
   public static void main(String[] args) {
      // ошибка: Pet zorka = new Pet();
      Fish nemo = new Fish();
      nemo.voice();
   }
}
```

Обратите внимание, что теперь, во-первых, мы не можем создавать объекты абстрактного класса Pet, а, во-вторых, реализация метода voice() должна иметься во всех его потомках (хотя бы пустая реализация), не являющихся абстрактными классами.

Хотя, мы могли бы создать абстрактного потомка:

```
abstract class Fish extends Pet {
}
```

Но не могли бы создавать объектов класса Fish, нам пришлось бы расширять класс, чтоб в итоге получить не абстрактный и создавать на его основе объекты. Например:

```
        class GoldenFish extends Fish {
    void voice() {
    }
}
```

4.4. Интерфейсы

Интерфейс это консутрукция языка программирования Java, в рамках которой могут описываться только абстрактные публичные (abstract public) методы и статические константyst свойства (final static). То есть также, как и на основе абстрактных классов, на основе интерфейсов нельзя порождать объекты.

Один интерфейс может быть наследником другого интерфейса.

Классы могут реализовывать интерфейсы (т. е. получать от интерфейса список методов и описывать реализацю каждого из них), притом, что особенно важно, один класс может реализовывать сразу несколько интерфейсов.

Перед описанием интерфейса указывается ключевое слово interface. Когда класс реализуемт интерфейс, то после его имени указывается ключевое слово implements и далее через запятую перечисляются имена тех интерфейсов, методы которых будут полностью описаны в классе.

Пример:

```
        interface Instruments {
    final static String key = "До мажор";
    public void play();
}
class Drum implements Instruments {
    public void play() {
        System.out.println("бум бац бац бум бац бац");
    }
}
class Guitar implements Instruments {
    public void play() {
        System.out.println("до ми соль до ре до");
    }
}
```

Поскольку все свойства интерфейса должны быть константными и статическими, а все методы общедоступными, то соответсвующие модификаторы перед свойствами и методами разрешается не указывать. То есть интерфейс можно было описать так:

```
        interface Instruments {
    static public String key = "До мажор";
    void play();
}
```

Но когда метод play() будет описываться в реализующем интерфейс классе, перед ним всё равно необходимо будет явно указать модификатор public.

4.5. Множественное наследование интерфейсов

Java не поддерживает множественное наследование классов. Это объясняется тем, что такое наследование порождает некоторые проблемы.

Чаще всего указываются неоднозначности, возникающие при так называемом «ромбовидном» наследовании, когда один класс «A» является наследником двух других классов «B» и «C», которые в свою очередь оба являются наследниками класса «D». Проблема допустимости множественного наследования кроется в следующем.

Предположим, что в родителе A определялся какой-то метод m1(). И этот же метод мы вызываем для объекта класса D. А что если m1() был переопределён в классах B и C. Какая реализация из трёх будет работать при вызове метода m1() для объекта класса D? От неоднозначности можно было бы избавиться потребовав в описанном случае при вызове уточнять при вызове, какой из методов требуется (так и сделано в некоторых языках), но в Java от множественного наследования классов решили отказаться.

Вместо множественного наследования классов в Java введено множественное наследование интерфейсов, которое частично решает проблемы (но, как будет показано в примере далее, к сожалению, не все).

Рассмотрим пример, где реализовано два интерфейса с методами доступными для грузового и для легкового транспорта. Класс Pickup (пикап) должен обладать как возможностью перевозки грузов, так и пассажиров, поэтому он реализует сразу оба интерфейса:

```
        interface PassangersAuto {
   void transportPassangers();
}
interface CargoAuto {
   void transportCargo();
}
class Truck implements CargoAuto {
   final static int a = 1;
   public void transportCargo() {
      System.out.println("Везу груз");
   }
}
class Sedan implements PassangersAuto {
   public void transportPassangers() {
      System.out.println("Везу пассажиров");
   }
}
class Pickup implements CargoAuto, PassangersAuto {
   public void transportCargo() {
      System.out.println("Везу груз");
   }
   public void transportPassangers() {
      System.out.println("Везу пассажиров");
   }
}
```

Часто всю открытую часть класса (т. е. общедоступные методы) предопределяю как раз в интерфейсе. Тогда взглянув на один лишь интерфейс можно поять какие же методы должны использоваться для взаимодействия с объектами данного класса. То есть интерфейсы вполне соответствуют принципам инкапсуляцуии. Как, впрочем, и принципу полиморфизма. Ведь в нескольких классах метод некоторого интерфейса может быть реализован по-разному, хотя и с одним и тем же именем.

Но интерфейсы, как говорилось выше, не являются совершенным инструментом лишенным всяких недостатков. Рассмотрим пример, когда у нас имеются два интерфейса, в каждом из которых есть свойства с одинаковыми именами (но, возможно, разными значениями) и методы с одинаковыми именами.

Унаследовав класс от пары этих интерфейсов мы не сможем обращаться к свойству его объектов напрямую, без указания того, какой из двух интерфейсов мы имели в виду. Это ограничение существует потому, что в интерфейсах свойствам может даваться разное

начальное значение и, соответсвенно, программа не сможет определить какое же значение выбрать.

Также к свойству можно обратиться как к статическому свойству одного из интерфейсов (разумеется, это можно делать и если у свойств были бы разные имена).

Проблема исчезнет, если перед обращением к свойству мы приведём объект к одному из родительских интерфейсов (напомним, что любой объект можно явно привести к классу или интерфейса его родителя прямого или транзитивного).

К сожалению, создать отдельные реализации для двух одноимённых методов из разных интерфейсов в классе наследнике не получится (чтобы потом ими можно было пользоваться через то же приведение объектов к нужному интерфейсу). Если класс реализует несколько интерфейсов, в которых есть одноимённые методы, то в нём может задаваться лишь одна общая для всех реализация этих методов (и это уже ограничивает полиморфизм при множественном наследовании через интерфейсы в Java).

Итак, код примера:

```
        interface Interface1 {
   int someField = 100;
   String someMethod();
}
interface Interface2 {
   int someField = 200;
   String someMethod();
}
class SomeClass implements Interface1, Interface2 {
   public String someMethod() {
      return "It Works";
   }
}
public class Main {
   public static void main(String[] args) {
      SomeClass a = new SomeClass();
      System.out.println( a.someMethod() ); // It works
      System.out.println( a.someField ); // ошибка
      System.out.println( ( (Interface1) a).someField ); // 100
      System.out.println( Interface1.someField ); // 100
   }
}
```

5. Интеграций модулей Python с Java

Python — это объектно-ориентированный скриптовый язык, что делает его хорошей парой для Java. В сочетании с интерпретатором Python, полностью написанным на Java, таким как Jython, вы можете писать целые апплеты на Python, которые могут затем работать в любом JDK-совместимом браузере, с почти таким же быстрым выполнением кода, как и в C/CPython. Интерпретатор Jython переводит исходный код Python непосредственно в байт-код Java, что дает ему невероятную скорость. А вот другие решения Java-сценариев (Java/TCL, Java/Perl и др.) присоединяют JVM к реализации C на этих языках, что не только создает проблему переносимости, но и сами эти решения не так легки, как хотелось бы.

Есть целый ряд проверенных инструментов, которые реализуют Python на Java или наоборот, так что вы можете выполнять команды одного языка на другом. Вот краткий список некоторых лучших инструментов для интеграции Python с Java:

- Jython — Python, реализованный на Java.
- JPype — позволяет с помощью Python запускать команды Java.
- Jepp — Java, встроенный в Python.
- JCC — генератор кода C ++ для вызова Java из C ++/Python.
- Javabridge — пакет для запуска и взаимодействия с JVM от CPython.
- Py4j — позволяет запускать на Python команды Java.
- Voc — элемент инструмента BeeWare. Преобразует код Python в байт-код Java.
- p2j — конвертирует код Python в Java. Больше не разрабатывается.

Существует несколько подходов к использованию этих инструментов. Каждый имеет свои преимущества и недостатки. Например, вы можете создать прототип всего приложения в Jython, а после нескольких циклов тестирования и перепроектирования переписать все на Java. Это позволяет вам воспользоваться преимуществами повышенной гибкости и скорости разработки скриптовых языков на ранних этапах вашего проекта.

Задания на работу

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Создайте БД MySQL (в соответствии с индивидуальным заданием из Лабораторной работы №1).

Задание 3. Разработайте командой модули проекта с графическим интерфейсом (в соответствии с индивидуальным заданием из Лабораторной работы №1) на языках программирования Python и Java (разные модули на разных языках программирования), используя инструментальные средства NetBeans, PyCharm (или их аналоги).

Задание 4. Осуществите сборку проекта.

Задание 5. Оформите отчет.

Контрольные вопросы

1. В чем заключаются механизмы интеграции программных модулей?
2. Какой функционал присущ языку программирования Python?
3. Какие Вы знаете инструментальные средства для создания приложений на языке Python?
4. Какой функционал присущ языку программирования Java?
5. Какие Вы знаете инструментальные средства для создания приложений на языке Java?
6. Для каких целей используется система контроля версий?

Лабораторная работа №6
«ОТЛАДКА ОТДЕЛЬНЫХ МОДУЛЕЙ ПРОГРАММНОГО ПРОЕКТА, ОРГАНИЗАЦИЯ ОБРАБОТКИ ИСКЛЮЧЕНИЙ»

Цель работы: получить практические навыки отладки отдельных модулей программного проекта, организацию обработки исключений

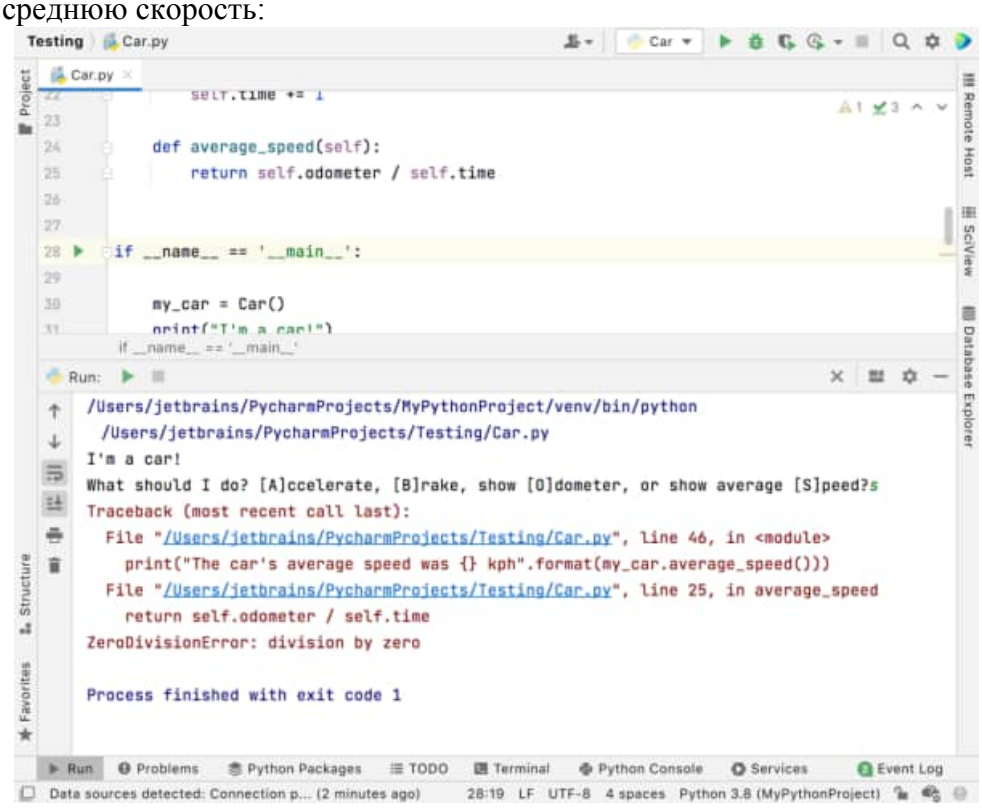
Теоретические сведения

1. Отладка в среде PyCharm
1.1. Выяснение причины проблемы

В предыдущей лабораторной работе создали и запустили сценарий **Car**. Давайте изменим average_speed функцию следующим образом:

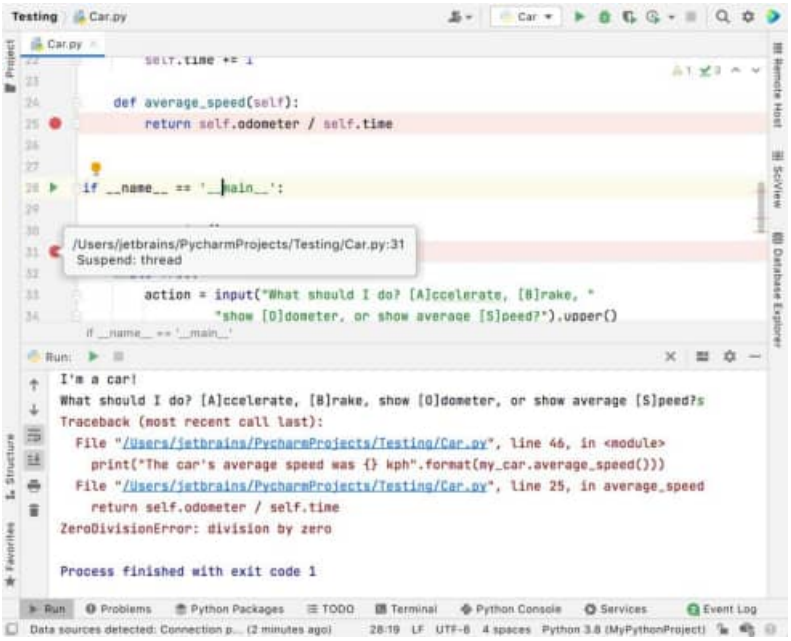
```
def average_speed(self):
    return self.odometer / self.time
```

Давайте посмотрим, что произойдет, когда мы запустим наш скрипт, и попробуем узнать нашу среднюю скорость:

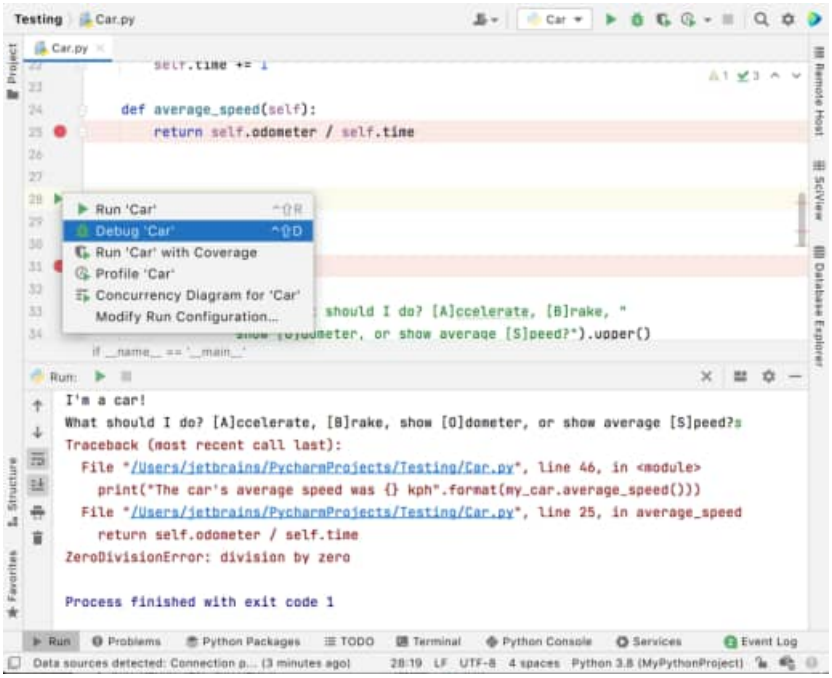


К сожалению, PyCharm сообщает об ошибке выполнения: a ZeroDivisionError.

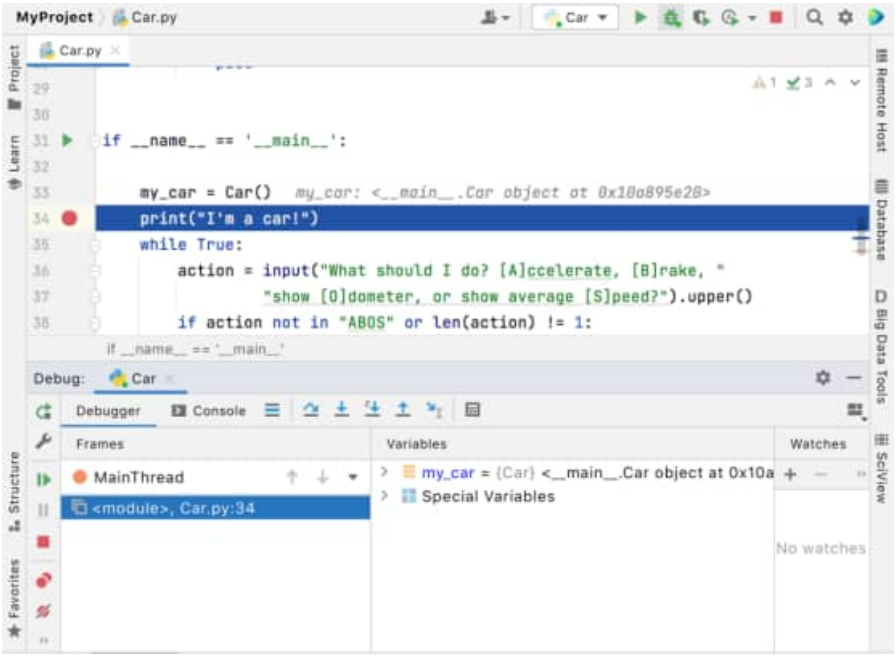
Давайте углубимся в наш код, чтобы выяснить, что идет не так. Мы можем использовать отладчик PyCharm, чтобы точно увидеть, что происходит в нашем коде. Чтобы начать отладку, вы должны сначала установить несколько точек останова . Чтобы создать точки останова, просто щелкните в желобе



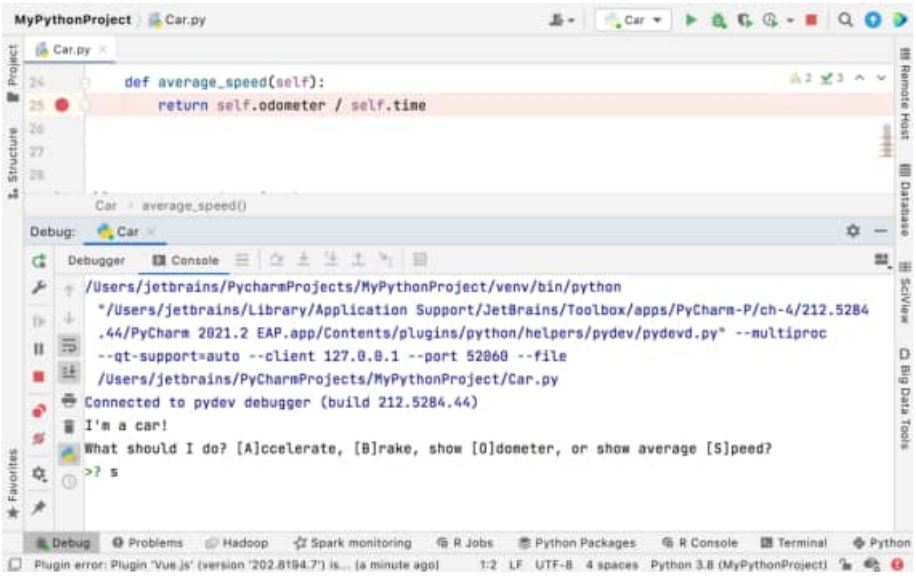
Затем щелкните значок в желобе рядом с main предложением и выберите **Отладка «Автомобиль»**.



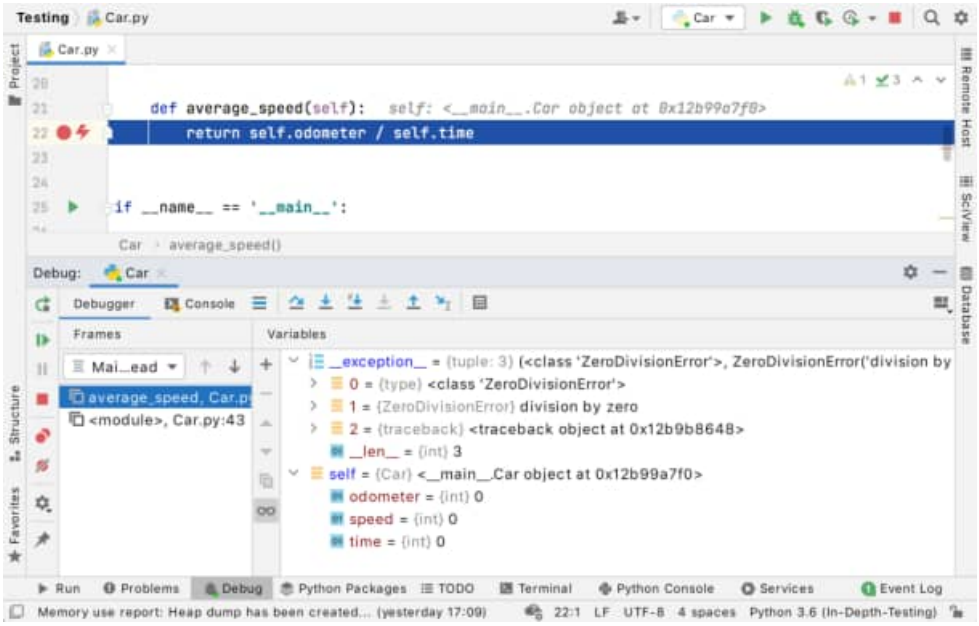
PyCharm запускает сеанс отладки и показывает [окно инструмента отладки.]



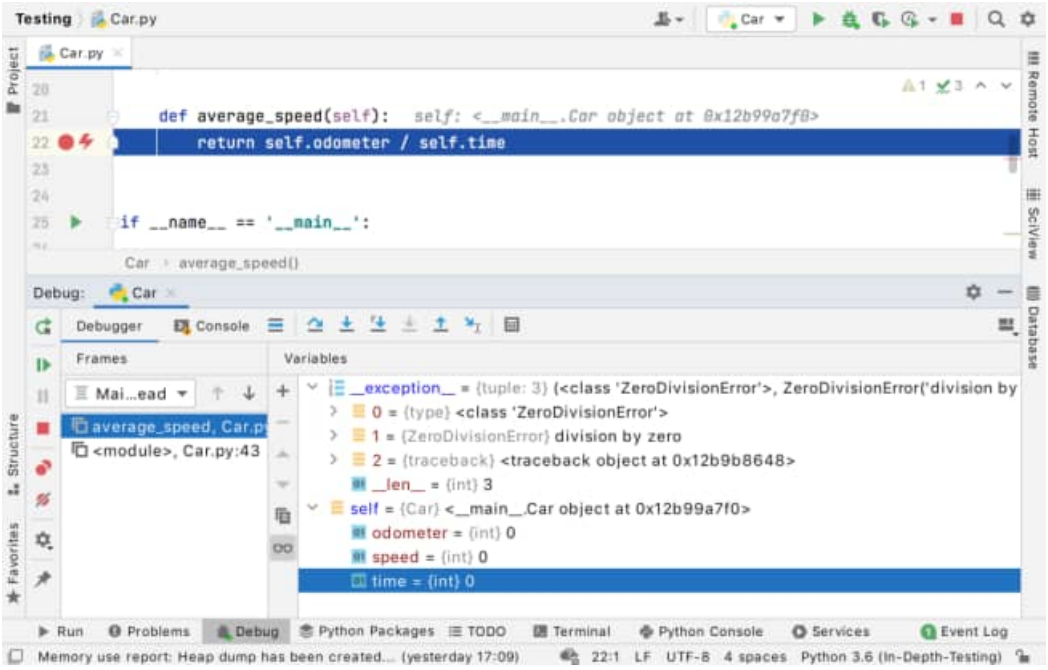
Нажмите кнопку, чтобы продолжить выполнение скрипта, и на вкладке «Консоль» введите S и нажмите Enter:



Нажмите кнопку, чтобы возобновить выполнение скрипта. Исключение здесь. Также появилась другая точка останова: по умолчанию PyCharm останавливается для любого исключения, которое не было обнаружено в вашем коде, и отображает значок точки останова с молнией.

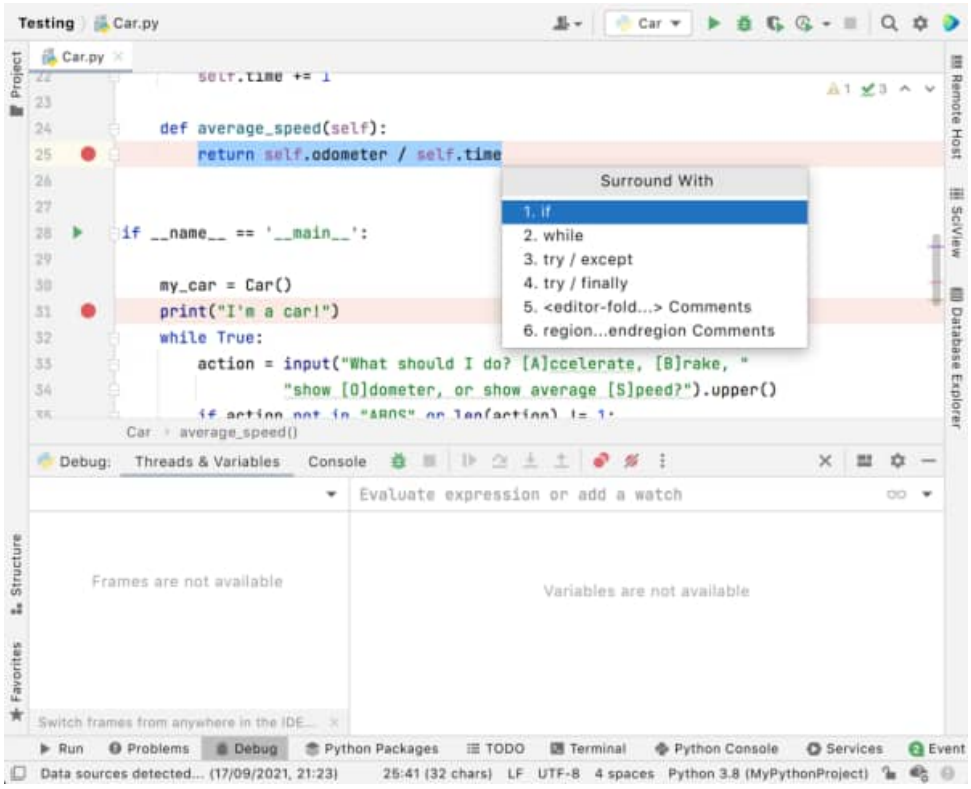


Отладчик также показывает сообщение об ошибке. Итак, мы нашли свою проблему. Вы также можете увидеть в отладчике, что значение self.time равно нулю:

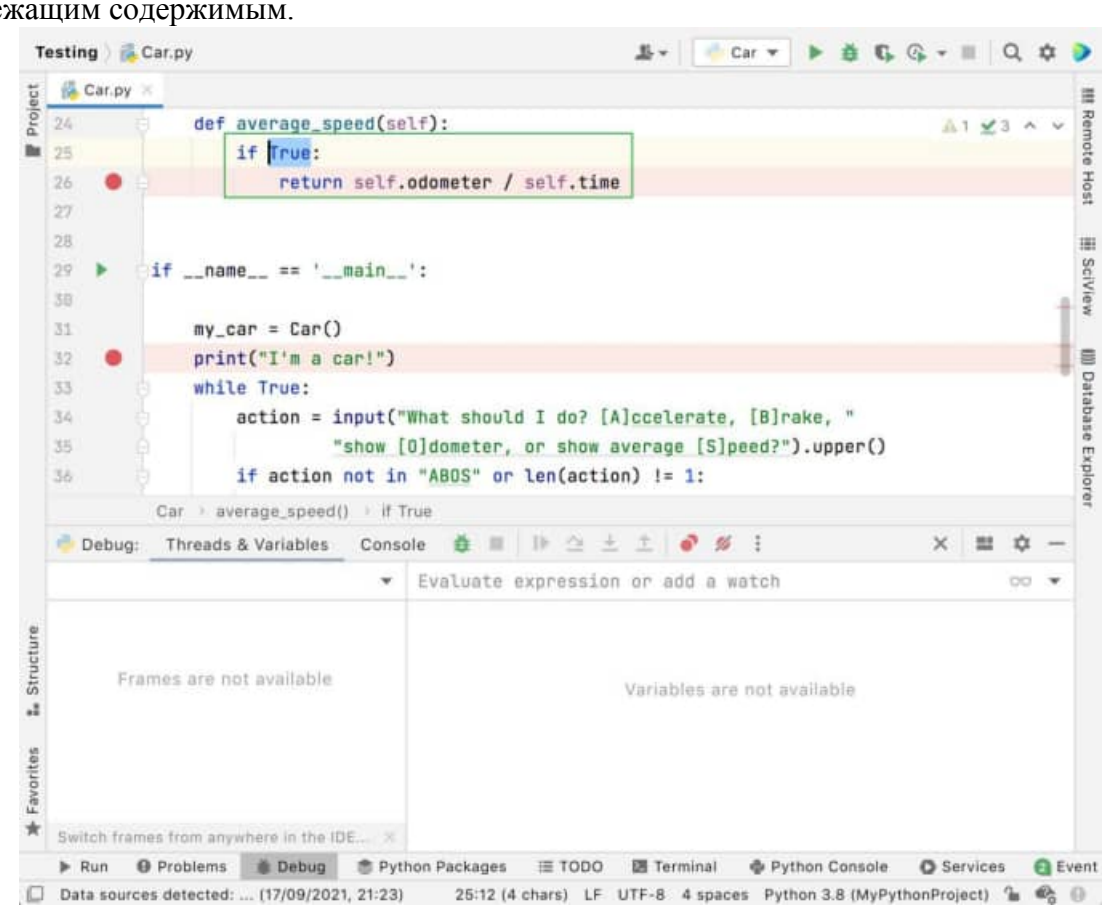


1.2. Окружающий код

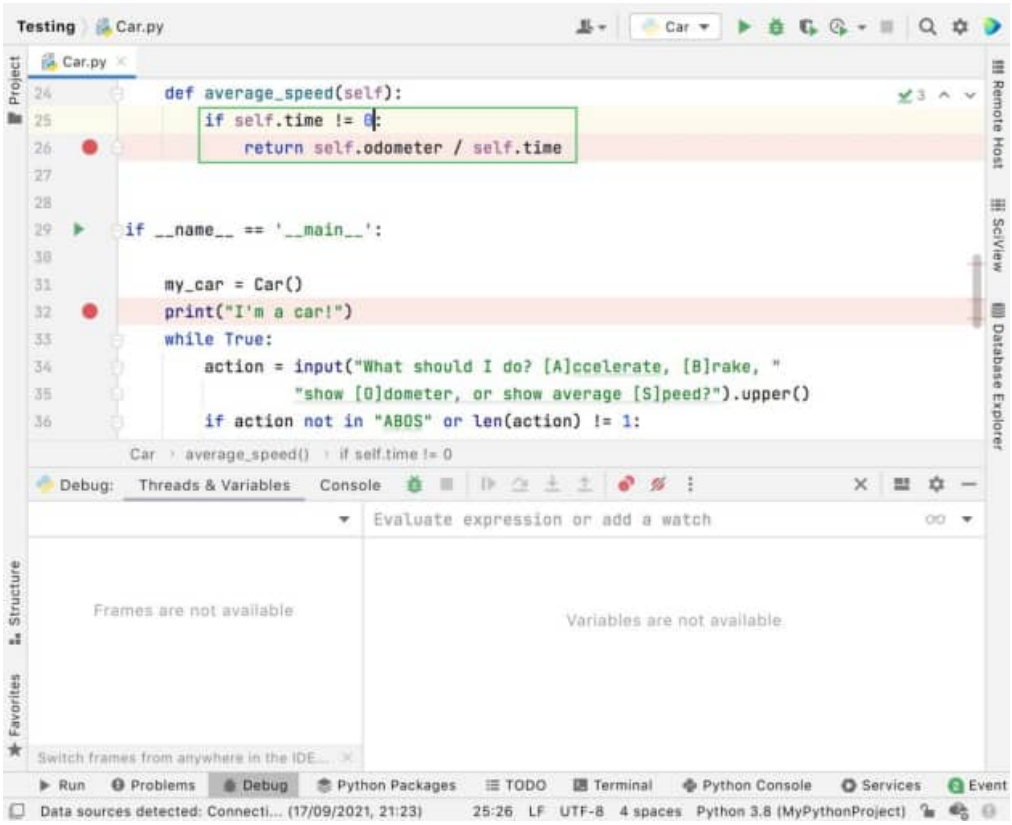
Чтобы не столкнуться с той же проблемой снова, давайте добавим if инструкцию, чтобы проверить, равно ли время нулю. Для этого выберите оператор return self.odometer / self.time в методе average_speed и нажмите Ctrl+Alt+T (**Code | Surround with**):



PyCharm создает ifконструкцию- заглушку , оставляя вам задачу наполнить ее надлежащим содержимым.



После редактирования получаем следующее:



Давайте подробнее рассмотрим, как отладчик может показать вам, что делает ваш код.

1.3. Подробная отладка

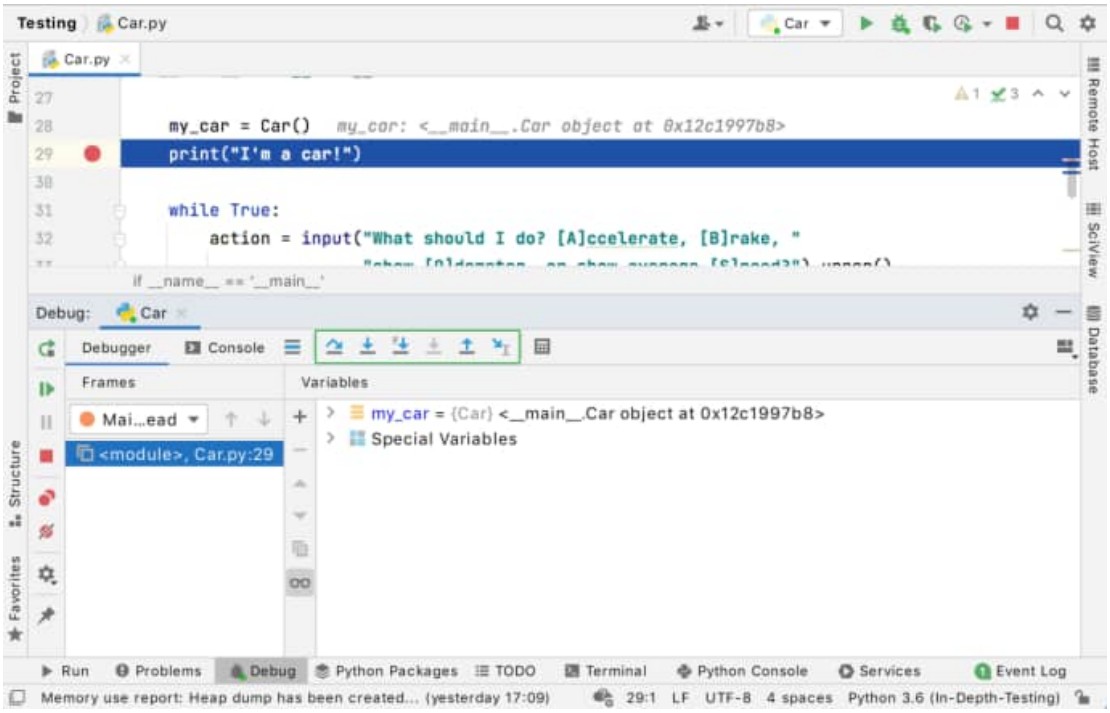
В окне инструмента отладки отображаются выделенные панели для фреймов , переменных и часов , а также консоль , где отображается вся входная и выходная информация. Если вы хотите, чтобы консоль была всегда видимой, вы можете перетащить ее к одному из краев окна PyCharm.

Шагая

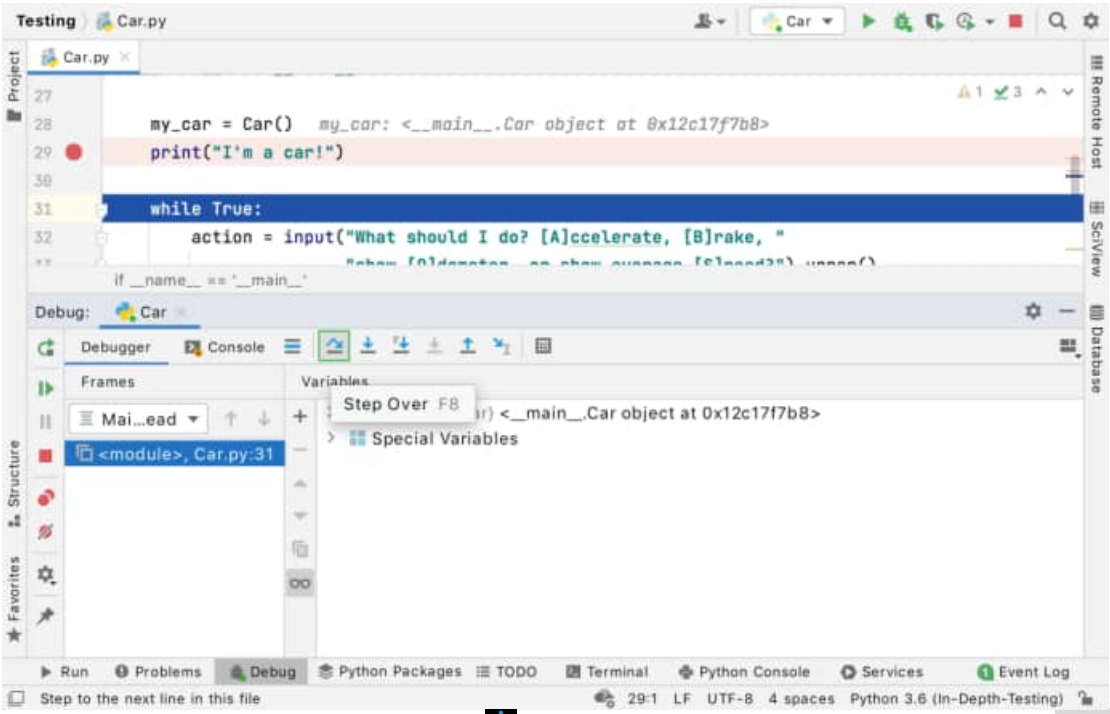
Если вы хотите видеть, что делает ваш код построчно, нет необходимости ставить точку останова на каждой строке, вы можете пошагово выполнять свой код.

Давайте посмотрим, как выглядит пошаговое выполнение нашего примера программы: нажмите кнопку, перейдите в консоль, чтобы запросить среднюю скорость автомобиля (введите «S»), и мы увидим, что мы нажимаем нашу точку останова.

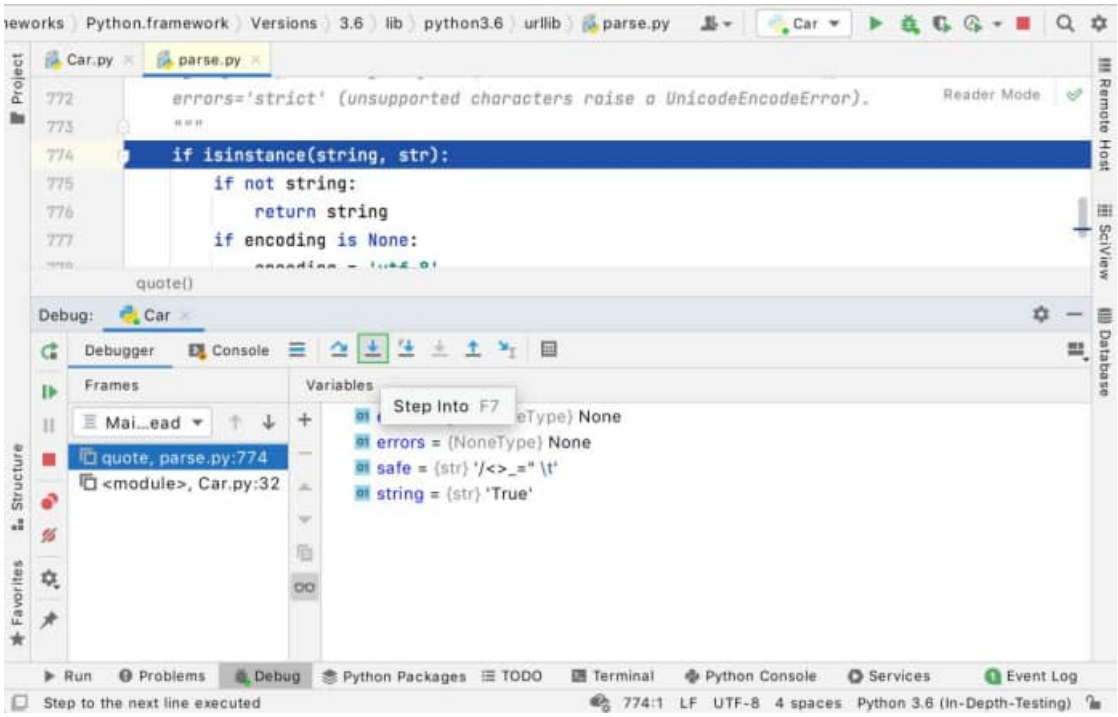
Мы можем использовать кнопки пошаговой панели инструментов, чтобы выбрать, на какой строке мы хотели бы остановиться следующей.



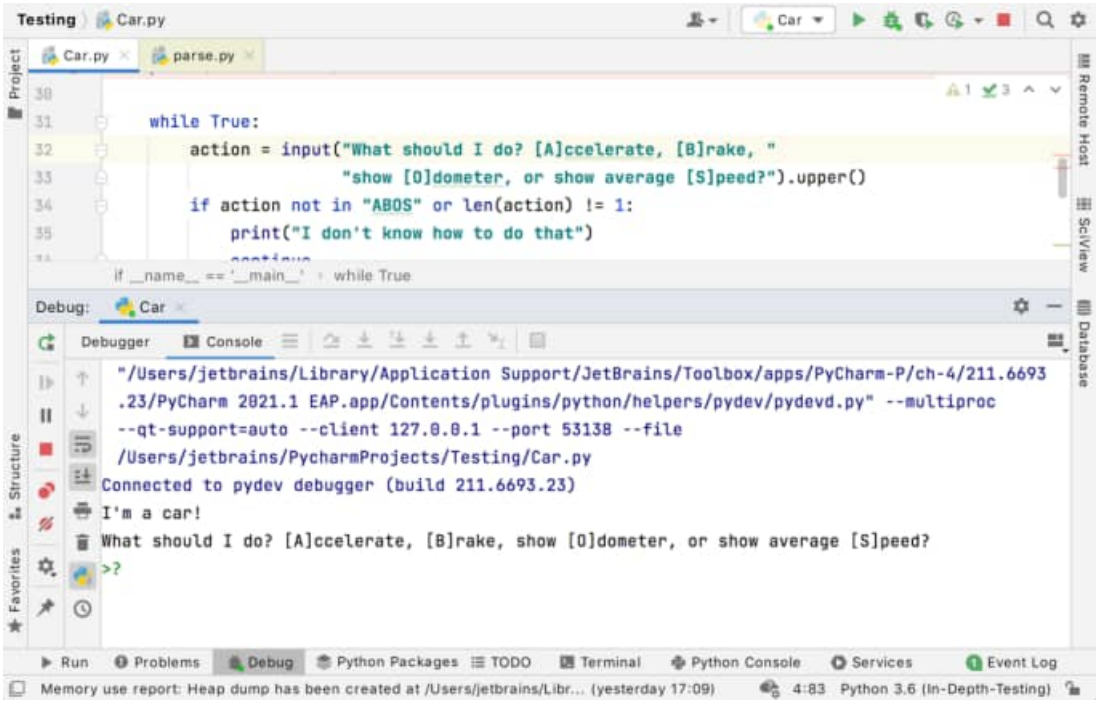
Например, нажмите **Step Over** кнопку и увидеть синий маркер движущуюся на следующую строку кода:



Если нажать **Step Into** кнопку , вы увидите , что после строки `action = input("What should I do? [A]ccelerate, [B]rake, " "show [O]dometer, or show average [S]peed?").upper()` отладчик переходит в файл **parse.py** :



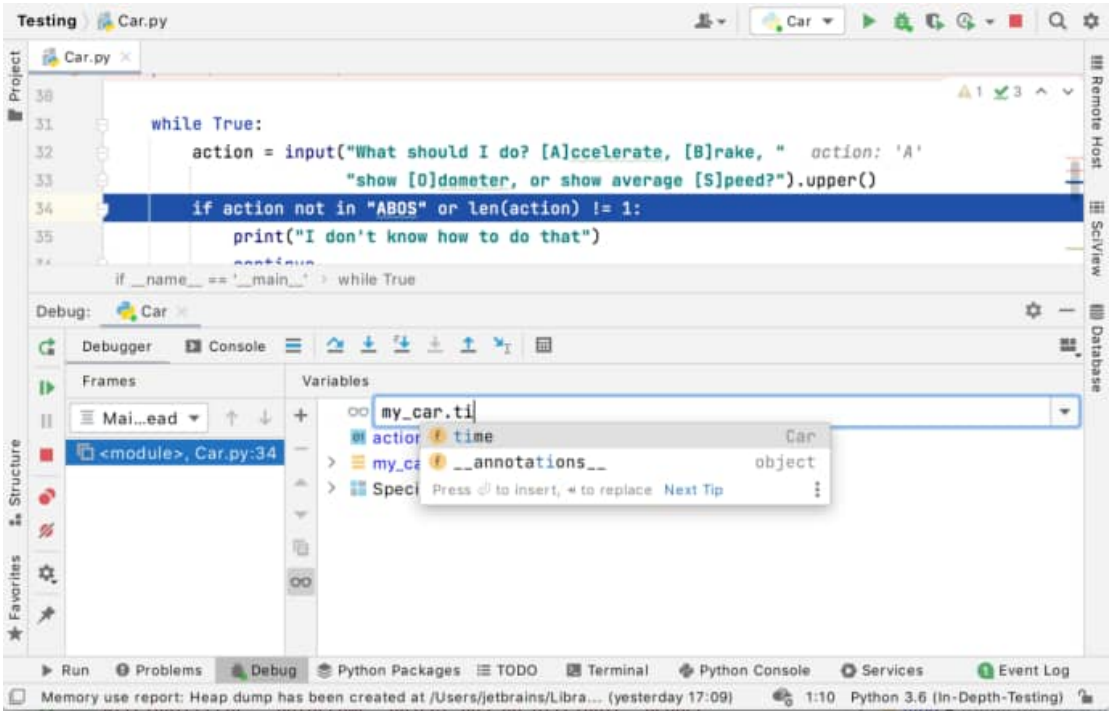
Однако, если вы продолжите использовать , вы увидите, что ваше приложение просто переходит к следующему циклу:



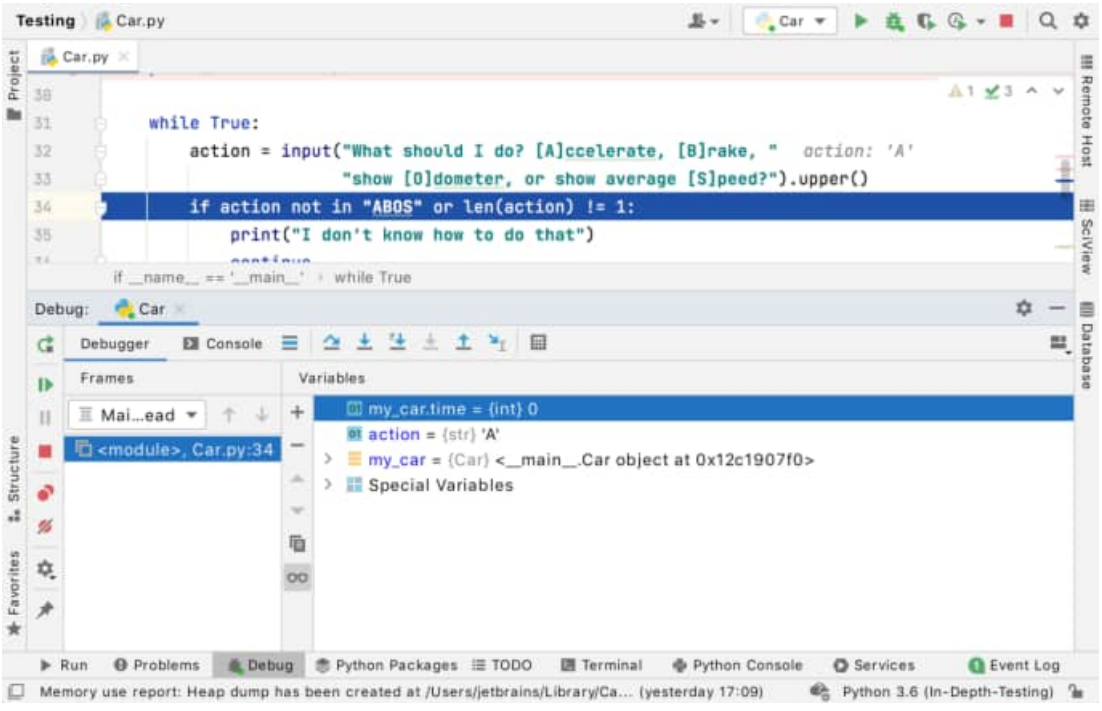
Если вы хотите сконцентрироваться на собственном коде, используйте кнопку « **Шаг в мой код**» - таким образом вы избежите перехода в классы библиотеки.

Смотреть

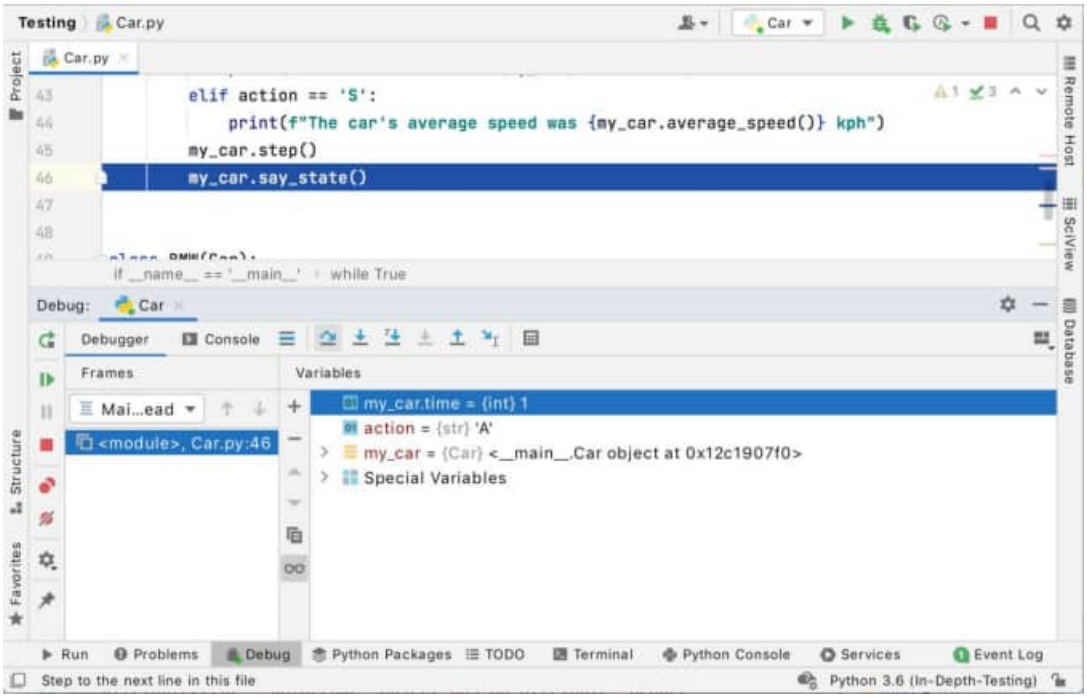
PyCharm позволяет вам наблюдать за любой переменной. Просто введите имя переменной, которую вы хотите отслеживать, в поле **Evaluate and Watch** - пусть будет my_car.time. Обратите внимание, что автозавершение кода доступно здесь. Затем щелкните рядом с полем.



Сначала вы видите, что время равно нулю - это означает, что переменная еще не определена:

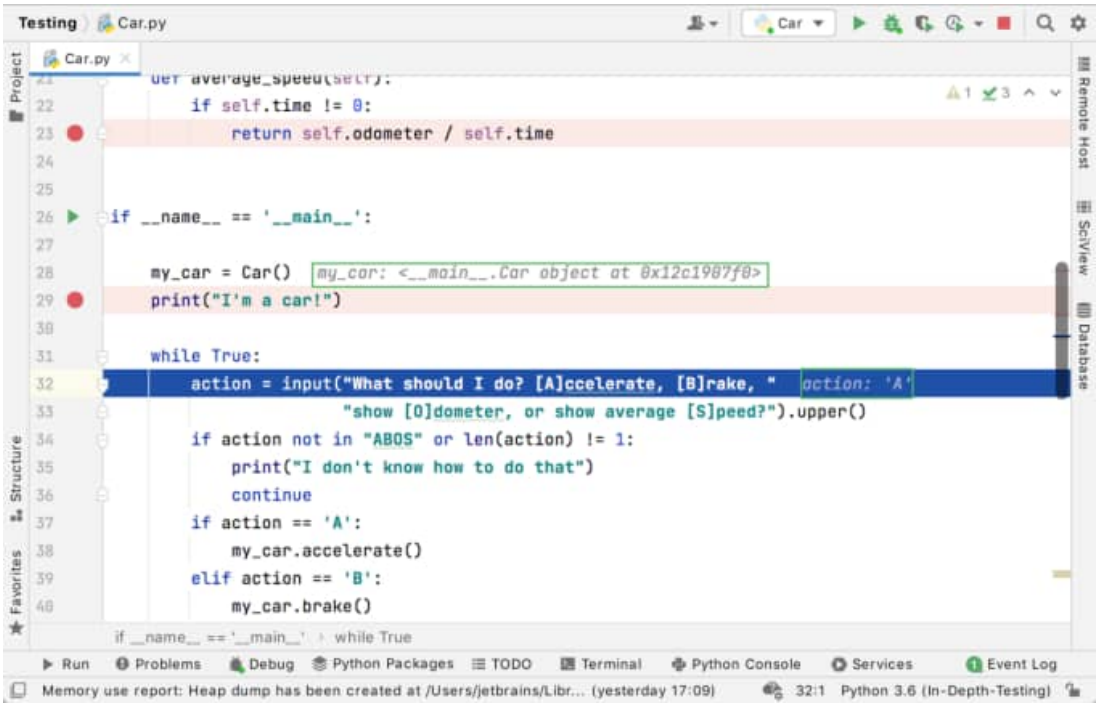


Однако, когда выполнение программы продолжается до области, определяющей переменную, часы получают следующее представление:

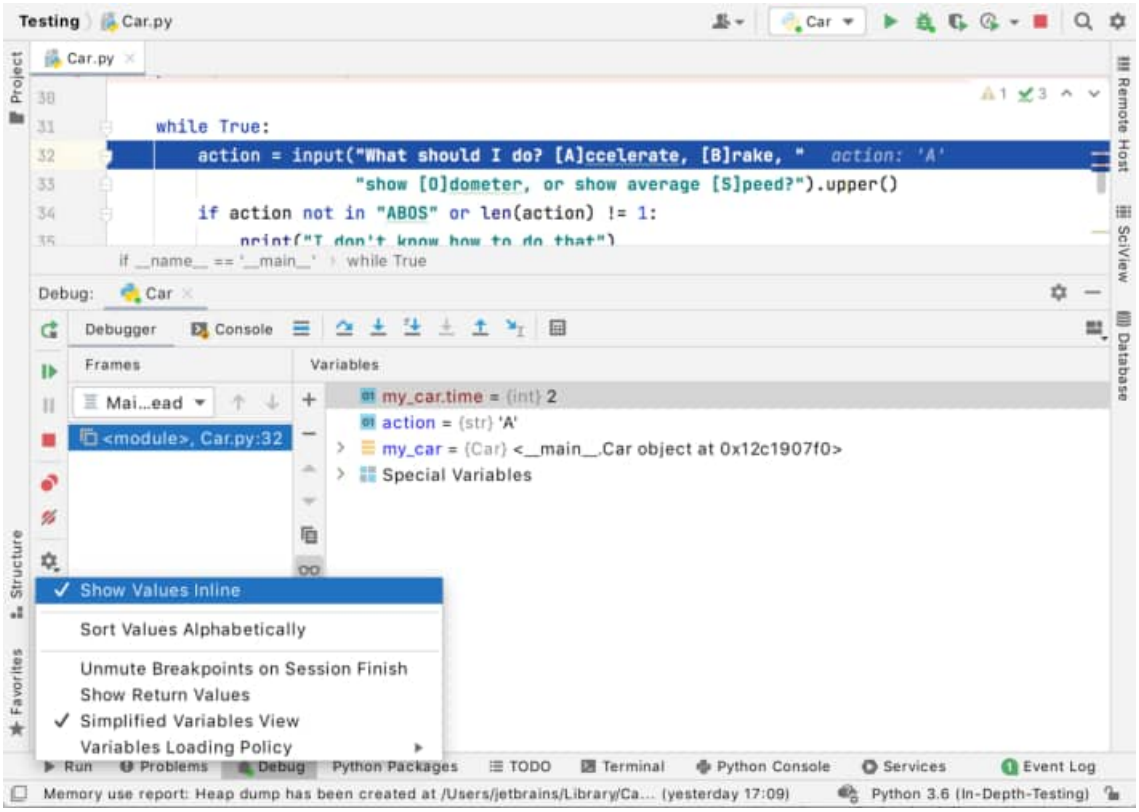


1.4. Встроенная отладка

Возможно, вы заметили еще одну функцию PyCharm, которая позволяет легко увидеть, что делает ваш код: встроенный отладчик . Как только вы нажмете любую точку останова, PyCharm покажет вам значение многих ваших переменных прямо в редакторе:



Эта функция встроенной отладки включена по умолчанию. Если вы не видите встроенные значения отладки, убедитесь, что он включен, используя значок настроек на панели инструментов отладки:



2. Использование визуального отладчика в NetBeans IDE

После создания проекта вы начнете процесс отладки и сделаете снимок графического интерфейса приложения.

1. Выберите 'Файл' > 'Новый проект' (Ctrl-Shift-N; ⌘-Shift-N в Mac) в главном меню, чтобы открыть мастер создания проектов.

2. В категории "Примеры" > Java выберите "Игра Анаграммы". Нажмите кнопку "Далее".

3. Укажите местоположение проекта. Нажмите 'Готово'.

При нажатии на кнопку 'Готово' IDE создает проект, который открывается в окне 'Проекты'.

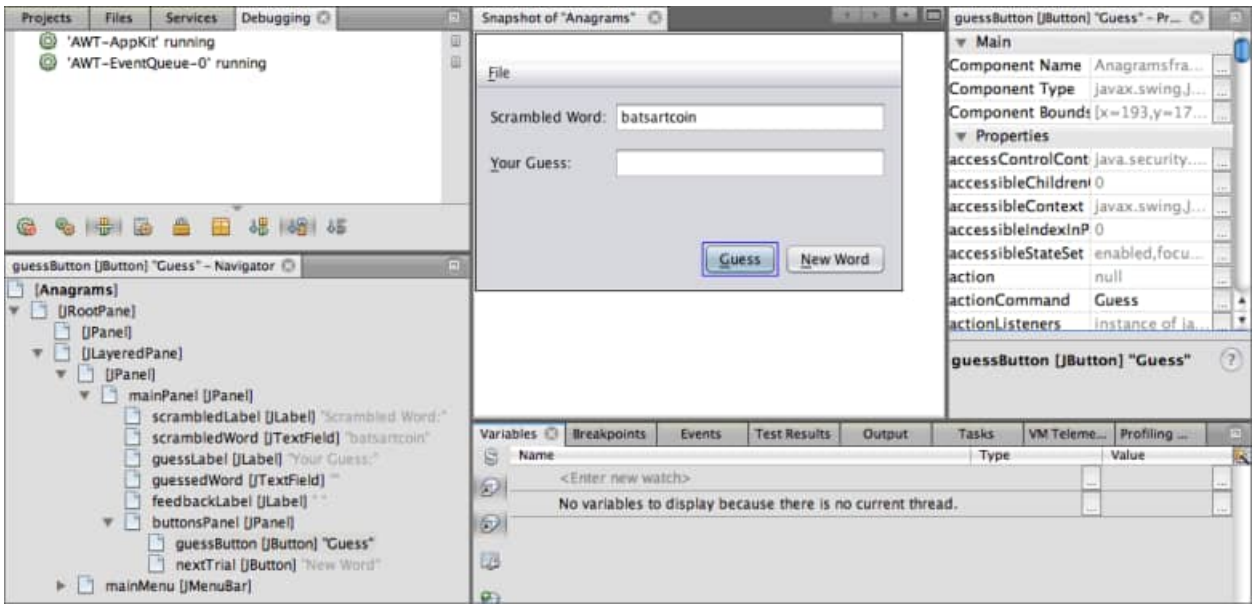
1. Нажмите кнопку 'Отладка' на панели инструментов (Ctrl-F5; ⌘-F5 в Mac) для запуска сеанса отладки.

Также вы можете щелкнуть правой кнопкой мыши узел проекта в окне 'Проекты' и выбрав 'Отладка'.

Когда начнется сеанс, среда IDE запустит игру "Анаграммы" и откроет окно "Отладка".

1. В главном меню выберите "Отладка" > "Сделать снимок графического интерфейса пользователя".

При выборе пункта "Сделать снимок графического интерфейса пользователя" среда IDE сделает моментальный снимок графического интерфейса пользователя и откроет снимок в главном окне.



2.1 Работа с визуальным отладчиком

Снимок графического интерфейса пользователя — это средство визуальной отладки, помогающее найти исходный код для компонентов графического интерфейса пользователя. Иногда исходный код для компонентов графического интерфейса пользователя бывает трудно найти, а снимок позволяет найти исходный код по графическому интерфейсу, вместо поиска во всём коде проекта. Можно выбирать компоненты из снимка и вызывать задачи из всплывающего меню, чтобы просматривать исходный код компонента, показывать прослушиватели и создавать точки останова в компонентах.

2.2. Поиск исходного кода компонентов

В этом упражнении будет продемонстрировано использование снимка графического интерфейса пользователя для поиска строк исходного кода, в которых объявляется и определяется компонент. При выборе компонента на моментальном снимке графического интерфейса пользователя можно использовать всплывающее меню для вызова различных команд.

Команды также можно вызвать из окна 'Навигатор'. Для этого щелкните правой кнопкой мыши компонент и выберите команду во всплывающем меню.

На снимке графического интерфейса пользователя нажмите кнопку "Определить".

При выборе компонента на снимке среда IDE отобразит подробные сведения о выбранном компоненте в окне "Свойства". Если окно 'Свойства' не отображается, выберите 'Окно > Свойства' в главном меню.

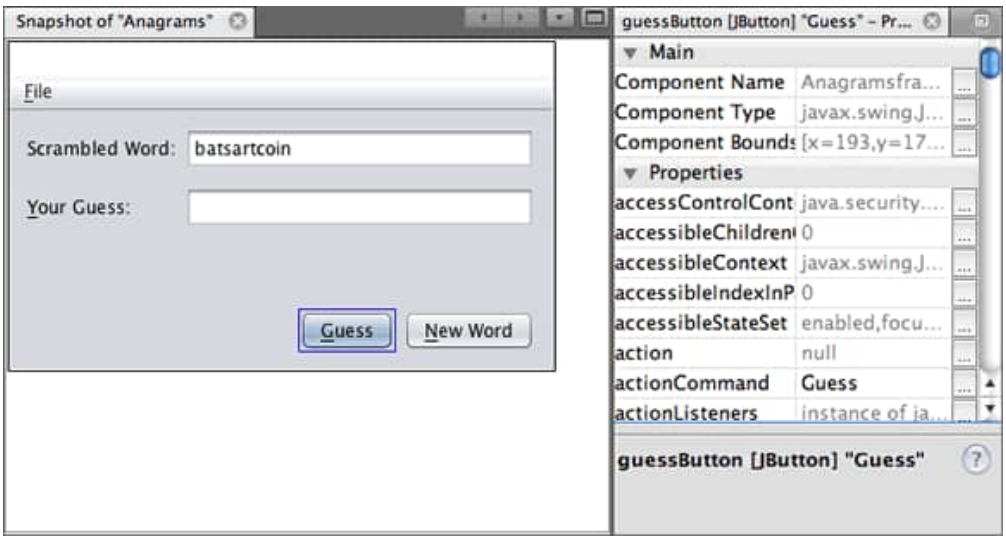


Рисунок . Снимок графического интерфейса пользователя

Среда IDE также отображает местоположение компонента в иерархии формы в окне навигатора.

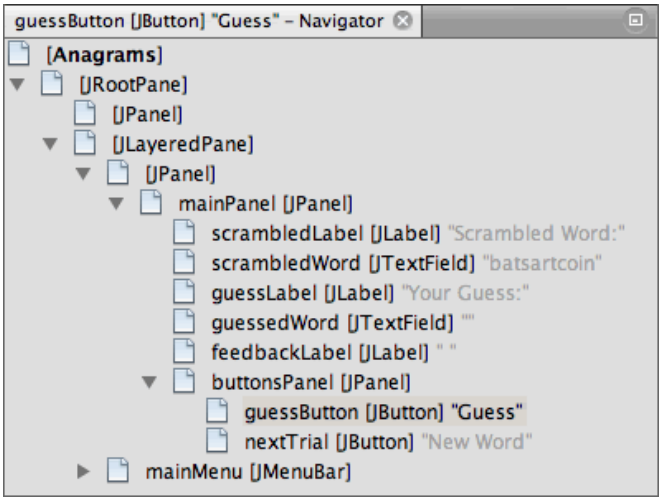


Рисунок. Снимок графического интерфейса пользователя

Щелкните кнопку "Определить" правой кнопкой мыши и во всплывающем меню выберите пункт "Перейти к объявлению компонента".

При выборе команды "Перейти к объявлению компонента" среда IDE открывает файл исходного кода в редакторе и перемещает курсор на ту строку в коде, в которой объявляется guessButton .

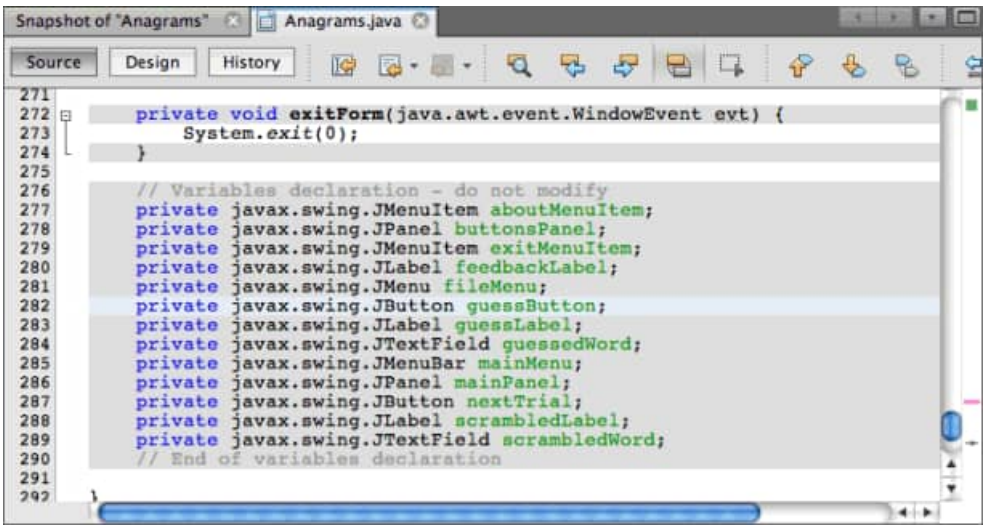


Рисунок. Строка кода, в которой объявлен компонент

Щелкните кнопку "Определить" правой кнопкой мыши на моментальном снимке и выберите "Перейти к исходному коду компонента".

При выборе команды "Перейти к исходному коду компонента" среда IDE открывает файл исходного кода в редакторе и перемещает курсор на строку исходного кода компонента JButton.

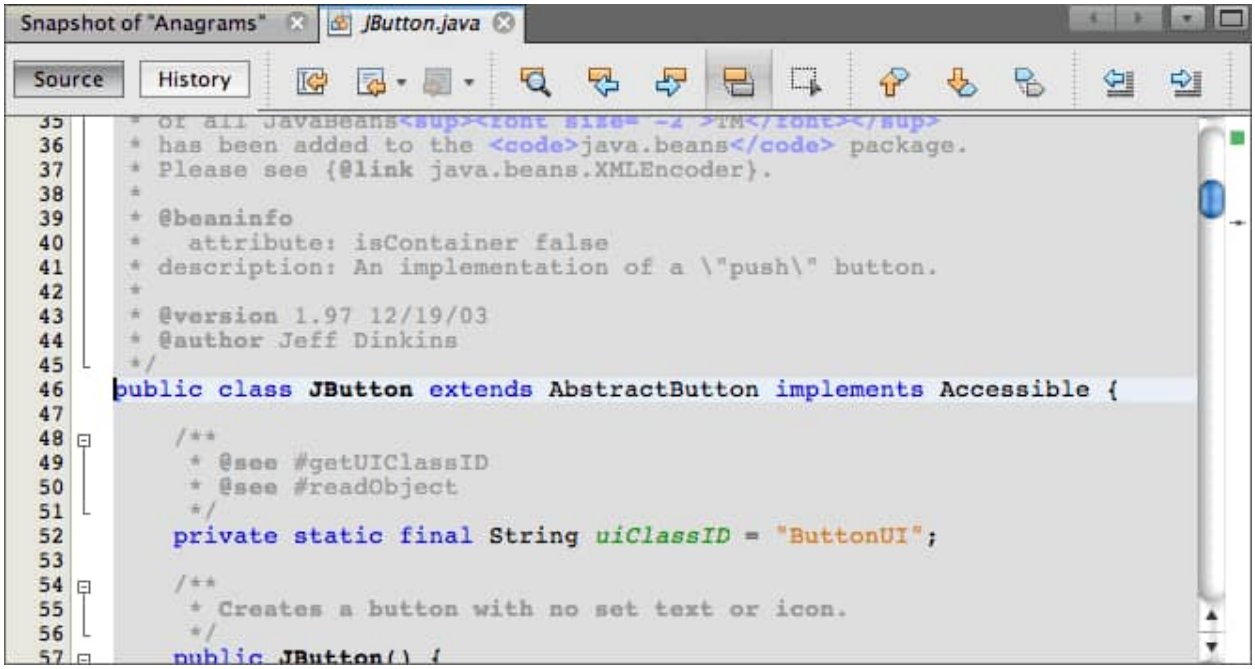


Рисунок. Строка исходного кода компонента

На снимке графического интерфейса можно воспользоваться командой 'Перейти к добавлению иерархии', чтобы найти строку исходного кода, где компонент добавляется в соответствующий контейнер. По умолчанию команда 'Перейти к добавлению иерархии' отключена. Включить эту команду можно в окне 'Параметры'.

1. Откройте окно 'Параметры'.
2. В окне 'Параметры' перейдите на вкладку 'Отладчик Java' в категории Java.

Примечание. В NetBeans IDE 7.1 вкладка 'Отладчик' находится в категории 'Разное' в том же окне 'Параметры'.

1. В списке категорий выберите "Визуальная отладка", а затем выберите **Отслеживать местоположения изменений в иерархии компонентов.** Нажмите кнопку "ОК".

2. Остановите сеанс отладки (если он запущен).

Примечание. После включения команды 'Перейти к добавлению иерархии' в окне 'Параметры' потребуется начать сеанс отладки заново и сделать новый снимок графического интерфейса. После этого можно будет использовать команду.

1. Начните новый сеанс отладки и сделайте снимок графического интерфейса.

2. Щелкните компонент правой кнопкой мыши на моментальном снимке графического интерфейса пользователя и выберите "Перейти к добавлению иерархии".

Среда IDE откроет исходный код в редакторе на строке, в которой добавляется компонент.

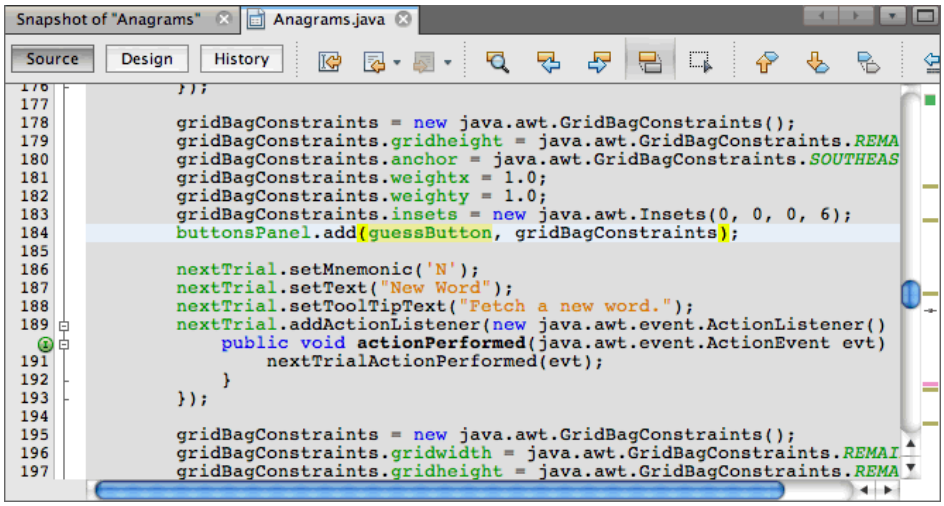


Рисунок. Строка исходного кода, в которой компонент добавлен в контейнер

2.3 Экспорт событий компонента

В этом упражнении вы научитесь использовать снимок графического интерфейса пользователя и окно "События" для просмотра событий компонента, искать прослушиватели компонентов и события, запускаемые компонентами.

Щелкните правой кнопкой мыши кнопку "Определить" на моментальном снимке и во всплывающем меню выберите "Показать прослушиватели".

При выборе команды "Показать прослушиватели" среда IDE открывает окно "События". Как видно, узел "Особые прослушиватели" развернут.

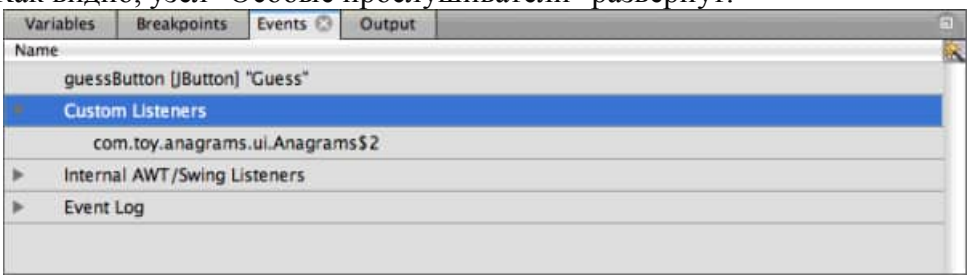


Рисунок. Строка исходного кода, в которой компонент добавлен в контейнер

1. Правой кнопкой мыши щелкните **com.toy.anagrams.ui.Anagrams$3** в узле "Особые прослушиватели" и во всплывающем выберите команду "Перейти к исходному тексту компонента".

Исходный код открывается в редакторе на той строке, в которой определяется прослушиватель.

1. Выберите пустое текстовое поле на снимке.

Также можно выбрать текстовое поле guessedWord в окне "Навигатор".

При выборе текстового поля элементы в окне "События" автоматически изменятся, и в окне будут отображены прослушиватели выбранного компонента.

1. В окне "События" дважды щелкните узел "Журнал событий". При этом откроется окно "Выбор прослушивателя".

Также можно щелкнуть правой кнопкой мыши узел Event Log и выбрать 'Задать события журналирования' во всплывающем меню.

1. В диалоге выберите прослушивающий процесс java.awt.event.KeyListener . Нажмите кнопку "ОК".

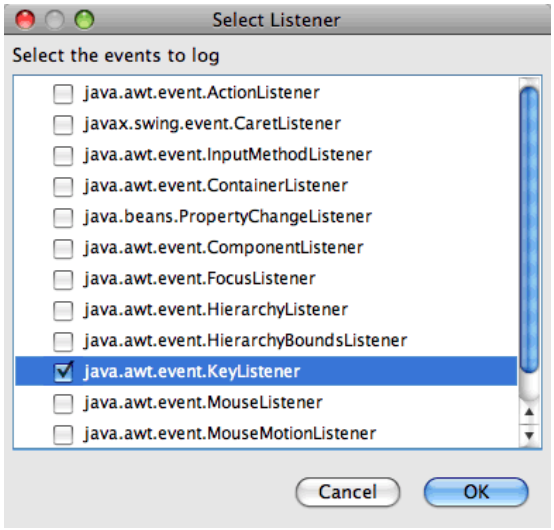


Рисунок. Строка исходного кода, в которой компонент добавлен в контейнер

Теперь прослушиватель прослушивает события клавиатуры в текстовом поле.

В приложении "Анаграммы" в текстовом поле введите несколько символов.

При вводе вами символа это событие записывается в журнал. Развернув узел "Журнал событий", вы увидите, что каждое нажатие клавиши записано. Новые события появляются при каждом вводе символов в текстовое поле игры "Анаграммы". Развернув отдельное событие, например keyPressed , вы увидите свойства этого события в журнале.

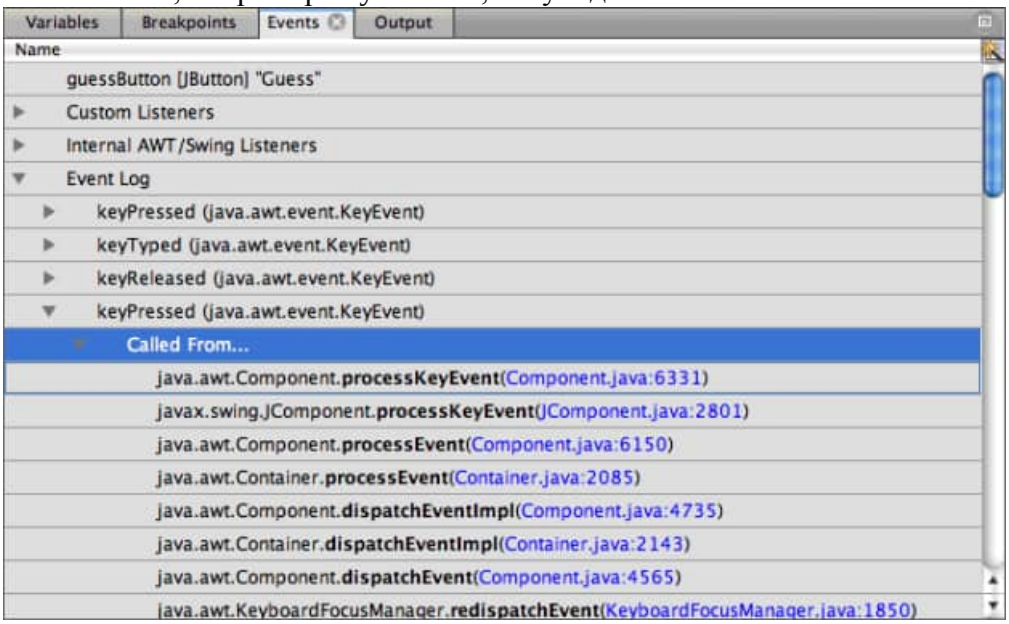


Рисунок. Строка исходного кода, в которой компонент добавлен в контейнер

Развернув узел "Вызов из…" в каком-либо событии, вы увидите трассировку стека данного события.

Задания на работу

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Осуществите отладку модулей, созданных в Лабораторной работе №5, с использованием инструментальных средств.

Задание 3. Оформите отчет.

Контрольные вопросы

1. Какие инструментарии Вы знаете для отладки программного модуля?
2. Какие виды отладки программного модуля Вы знаете?

Лабораторная работа №7
«ПРИМЕНЕНИЕ ОТЛАДОЧНЫХ КЛАССОВ В ПРОЕКТЕ, ОТЛАДКА ПРОЕКТА»

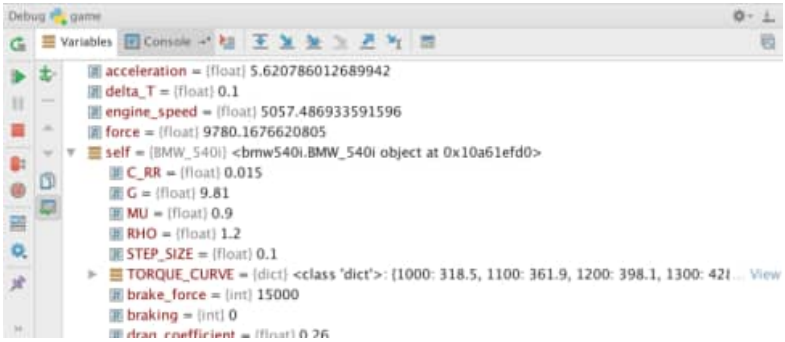
Цель работы: получить практические навыки применения отладочных классов в проекте и отладки проекта.

Теоретические сведения

1. Отладка Python-кода с PyCharm

1.1 Визуальный отладчик

Некоторые разработчики до сих пор отлаживают программы, используя операторы print, потому что концепция сложна, а PDB выглядит пугающе. Графический отладчик PyCharm делает процесс максимально простым, наглядно визуалируя отладку в удобном формате. С PyCharm вы легко освоитесь и сможете быстро начать пользоваться основными функциям отладки.

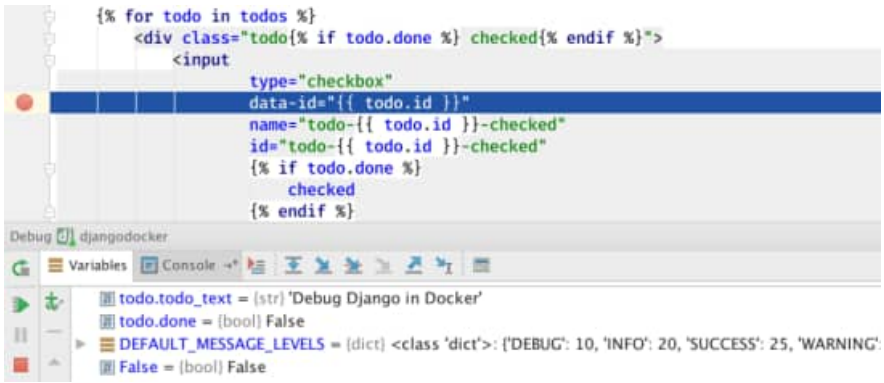


1.2. Debug Everywhere

Конечно, PyCharm умеет отлаживать код, который выполняется на локальном компьютере в системном окружении, virtualenv, Anaconda или Conda. С PyCharm Professional Edition вы также сможете отлаживать код, который запускается в Docker-контейнере, на виртуальной машине или на удаленном хосте через SSH.

1.3. Отладка внутри шаблонов ТОЛЬКО PRO

При работе с шаблонами иногда в них может закрасться ошибка. Такие ошибки бывает трудно исправить, если не видно, что происходит внутри. Отладчик PyCharm позволяет устанавливать точки останова внутри шаблонов Django и Jinja2, чтобы вы могли легко избавиться от проблем.



Обратите внимание: для отладки шаблонов сначала необходимо сконфигурировать язык шаблонов.

PyCharm позволяет создавать и отображать шаблоны, написанные на одном из поддерживаемых языков шаблонов:

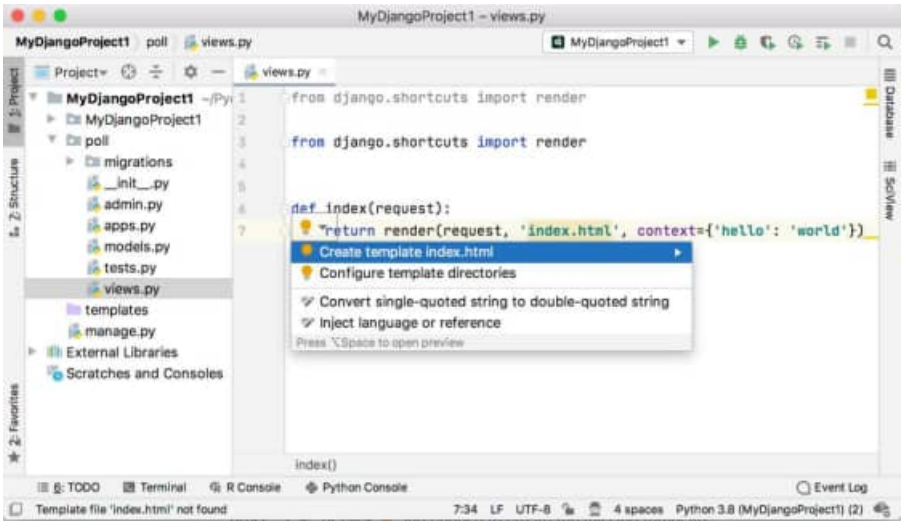
- Джанго
- Jinja2

Важно отметить, что можно редактировать шаблоны без фактической установки языков шаблонов. Однако для того, чтобы создавать или отображать шаблоны, а также перемещаться между представлениями и шаблонами, соответствующий язык шаблонов должен быть правильно установлен.

Создать шаблон для представления

Предположим, вы ссылаетесь на еще не существующий файл шаблона. PyCharm отмечает такую ссылку как неразрешенную и предоставляет действие намерения для создания файла шаблона «из использования».

1. Поместите курсор в неразрешенную ссылку на шаблон.
2. Нажмите Alt+Enterили щелкните желтую лампочку, чтобы отобразить список доступных действий намерения.
3. В списке предложений выберите действие *Создать шаблон <имя>* :



Появится диалоговое окно « **Создать шаблон** », в котором отображается имя **шаблона, доступное** только для чтения (поле « **Путь к** шаблону»), и список возможных расположений шаблона (**корневое** поле « **Шаблоны»**).

Обратите внимание, что PyCharm автоматически обнаруживает каталоги, указанные в полях TEMPLATE_DIRSили файла **settings.py** . Вы можете дополнительно указать другие каталоги.TEMPLATE_LOADERS

4. В диалоговом окне « **Создать шаблон** » выберите каталог шаблонов, в котором будет создан новый шаблон.

В **корневом** поле **шаблона** представлен список возможных мест для нового шаблона. В этот список входят каталоги шаблонов, указанные на странице « Языки шаблонов» диалогового окна «Параметры», а также каталоги, которые автоматически обнаруживаются
в переменных TEMPLATE_DIRSили файла **settings.py** .TEMPLATE_LOADERS

Щелкните **ОК** . В указанном месте создается пустой файл **.html** с указанным именем.

Перед использованием определенного языка шаблонов настройте его на странице « Языки шаблонов» диалогового окна «Настройки» и убедитесь, что типы файлов любых существующих шаблонов правильно распознаются.

Настроить язык шаблона для проекта

1. Откройте диалоговое окно «Настройки / Предпочтения» и щелкните узел « **Языки шаблонов Python» .**
2. На странице " Языки шаблонов" сделайте следующее:

- В списке « **Язык шаблона»** выберите конкретный язык шаблона, который будет использоваться в проекте.
- В **типы файлов шаблона** области, указать типы файлов, в которых будут признаны теги шаблонов.

Обратите внимание, что в файлах HTML, XHTML и XML шаблоны всегда распознаются.

Используйте кнопки « **Добавить»** и « **Удалить»,** чтобы составить желаемый список типов файлов.

Вы можете сделать так, чтобы поддержка Django пропускала некоторые папки (отключая замену языка Django), если вы отметили их как *ресурсы*.

При создании приложения Django вы можете сразу указать папку, в которой будут храниться шаблоны.

Определите каталоги шаблонов

1. В диалоговом окне « **Настройки / Предпочтения** » (Ctrl+Alt+S) щелкните страницу « Структура проекта» .
2. Выберите каталог, который будет отмечен как корень шаблона.
3. Выполните одно из следующих действий:

- Щелкните на панели инструментов панели « **Корни содержимого» .**
- В контекстном меню каталога выберите « **Шаблоны** ».

Отметить папку как каталог шаблона

1. В окне инструмента "Проект" щелкните правой кнопкой мыши нужный каталог.
2. В контекстном меню выберите « **Пометить каталог как»** , а затем щелкните отмеченную команду « **Каталог шаблонов» .**

Это приводит к добавлению отмеченного каталога в список каталогов шаблонов на странице структуры проекта .

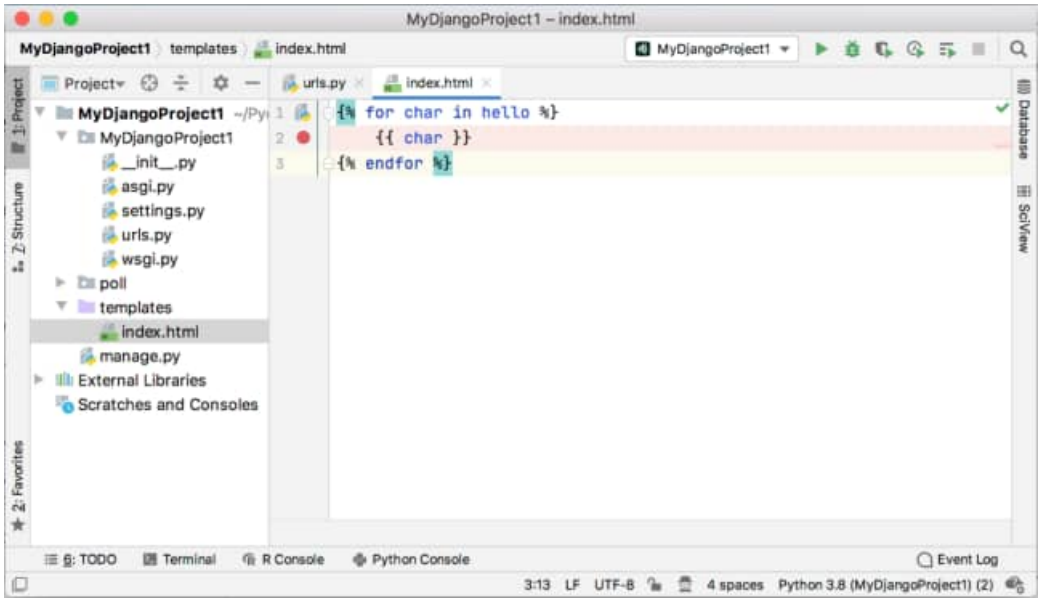
По умолчанию каталог, который был определен как папка шаблона при создании проекта , помечен как . Если каталог указан **TEMPLATES_DIR** из **settings.py** файла, а затем, на первом открытии проекта, он автоматически помечается как корень шаблона.

Вы можете сделать так, чтобы поддержка Django пропускала некоторые папки (отключая замену языка Django), если вы отметили их как *ресурсы* .

PyCharm позволяет размещать точки останова в строках файлов шаблонов Django, в строках с тегами или выражениями Django. Если вы хотите отследить исключения, которые возникают в процессе отладки шаблона, откройте диалоговое окно « **Точки останова** » и на вкладке « **Точки останова для исключений Django** » установите флажок « **Приостановить» .** Отладка шаблонов Jinja не поддерживается.

Отладка шаблона Django

1. Разместите точки останова в нужных строках шаблонов Django:



2. Запустите сервер Django в режиме отладки.

Вот как это делается:

1. В главном меню выберите « **Выполнить» | Редактировать конфигурации** .
2. Если желаемая конфигурация запуска / отладки Django Server существует, выберите ее, в противном случае создайте новую, как описано в разделе Создание и редактирование конфигурации запуска / отладки .
3. Затем нажмите кнопку на главной панели инструментов или нажмите Shift+F9.

PyCharm откроет шаблон в вашем браузере и приостановит выполнение в установленных вами точках останова.

Начнется сеанс отладки, и появится окно инструмента отладки.

3. В окне инструмента отладки вы можете:

- Изучите контексты рендеринга на панели «Переменные» .
- Пройдите через точки останова, определенные в шаблоне Django.
- Используйте консоль отладки .

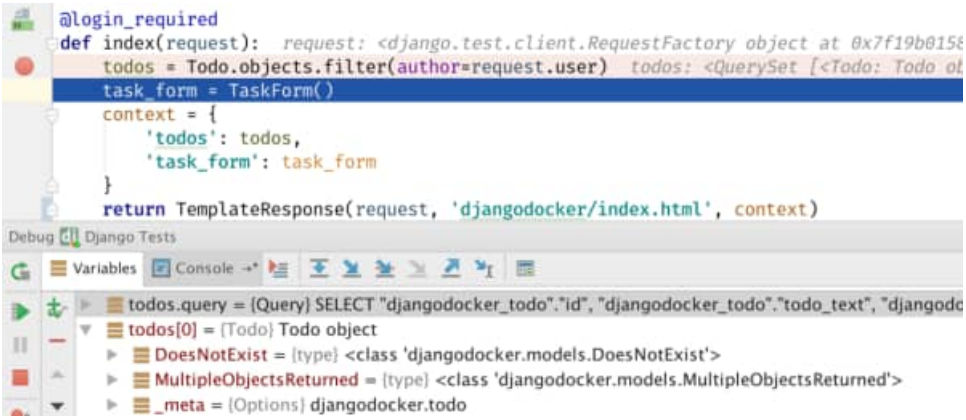
1.3. Отладка при TDD

Разработка через тестирование (TDD) подразумевает исследование кода при написании тестов. Используйте для этого отладчик, устанавливая точки останова в контексте, который вы исследуете.

```
def test_fed_dog_gives_paw(self):  self: <test_dog.TestDog object at 0x10e227908>
    # Arrange
    dog = Dog(name='Lassie')  dog: <Dog Lassie>
    dog.feed()

    # Act
    result = dog.give_paw()  result: 'Offers paw'

    # Assert
    assert result == 'Offers paw'

def test_unfed_dog_gives_no_paw(self):
    # Arrange
    dog = Dog(name='Scooby')

    with pytest.raises(HungryDogError):
        dog.give_paw()
```

Исследование может относиться к коду тестов или к тестируемому коду, что очень полезно при интеграционном тестировании Django-приложений (поддержка Django

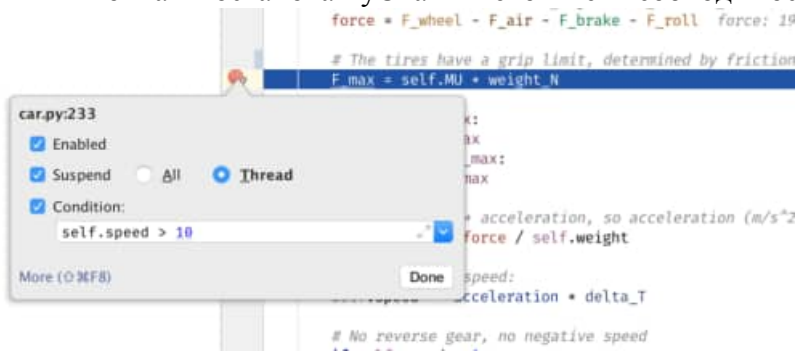
доступна только в редакции Professional Edition). Используйте точку останова, чтобы узнать, что возвращает запрос в тестовом сценарии.



PDB — отличный инструмент, но необходимость менять код может привести к случайной записи вызовов pdb.set_trace() в ваш Git-репозиторий.

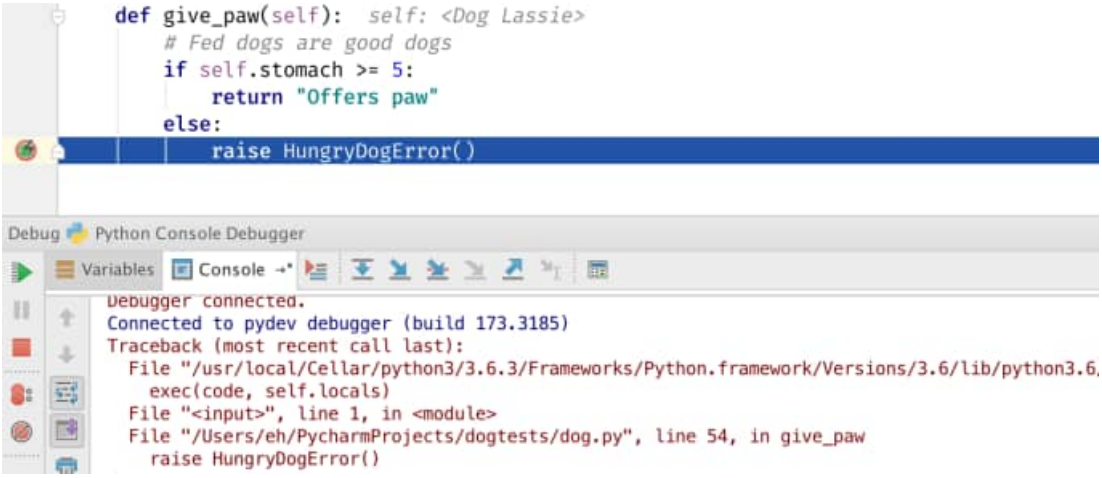
1.4 Точки останова

Точки останова есть во всех отладчиках, но только некоторые из них могут предложить гибкие точки останова. Вам наверняка приходилось много раз нажимать Continue, пока вы наконец не добрались до итерации цикла, в которой возникает ошибка. С условными точками останова PyCharm в этом нет необходимости.

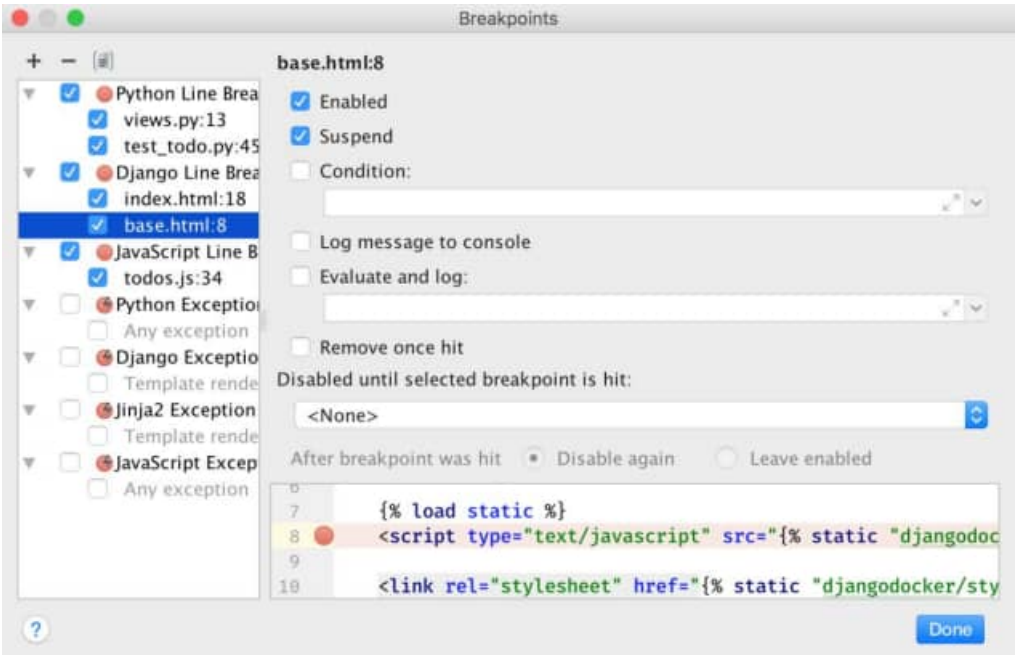


зачастую все, что нужно сделать, — это посмотреть, какое значение имеет определенная переменная во время выполнения кода. Вы можете настроить точки останова PyCharm таким образом, что они не будут останавливать выполнение кода, а только подготовят для вас сообщение.

Исключения могут испортить ваш день, поэтому отладчик PyCharm умеет останавливаться на исключениях, даже если вы не совсем уверены, откуда они взялись.

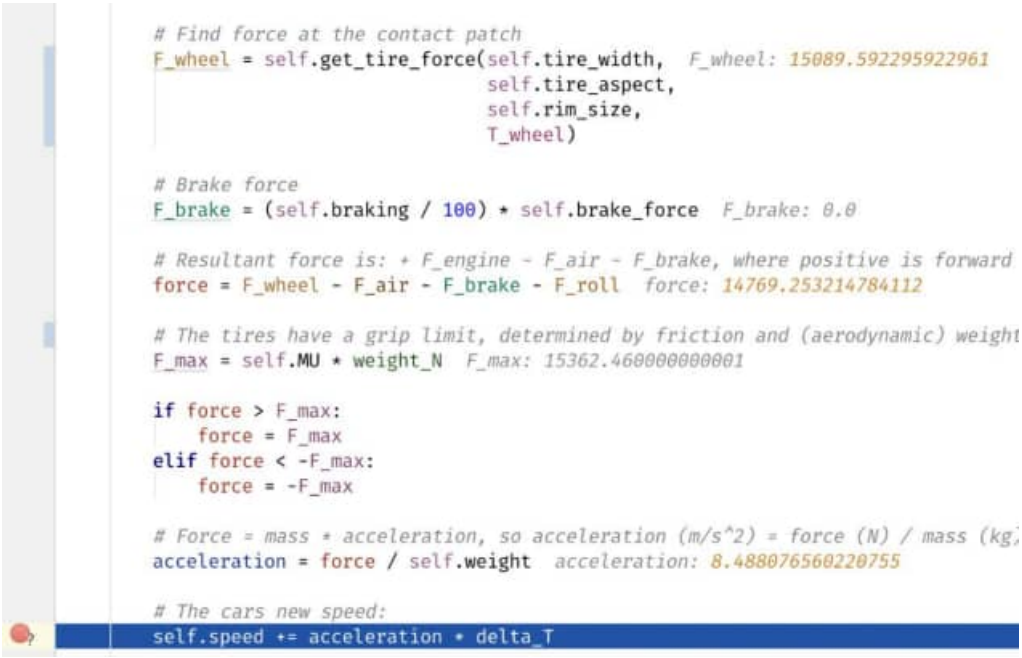


Чтобы помочь вам контролировать процесс отладки, в PyCharm предусмотрено специальное окно, в котором вы можете увидеть все точки останова и отключить некоторые из них, используя флажки. Кроме того, можно временно отключить все точки останова, пока они снова вам не понадобятся.



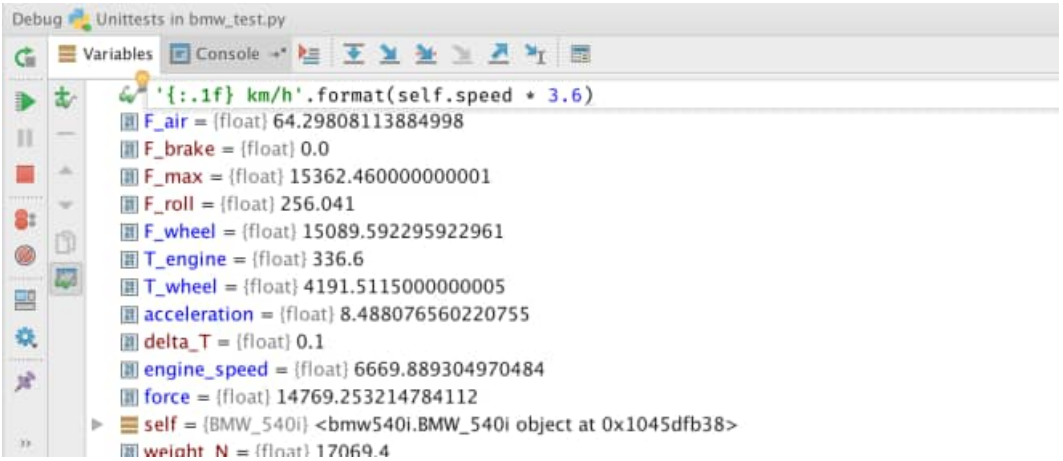
1.5. Просмотр значений переменных

Как только PyCharm достигнет точки останова, вы увидите все значения переменных прямо в коде. Чтобы было проще понять, какие значения поменялись с момента последнего срабатывания точки останова, изменившиеся значения подсвечиваются.



1.6. Watches

Вы можете настроить отображение переменных, добавив watches (отслеживание значений). Независимо от их сложности, PyCharm покажет именно то, что вам нужно.



Если вы хотите знать, как работает ваш код, не обязательно везде ставить точки останова. Вы можете следить за всем, что происходит, шагая по коду с помощью отладчика.

Наглядное представление процесса отладки:

1.7. Выполнение пользовательского кода

В некоторых случаях самый простой способ воспроизведения — принудительно установить определенное значение для переменной. PyCharm предлагает вычислить выражение для быстрых изменений, а также использовать консоль, если вы хотите лучше контролировать процесс. Консоль может использовать оболочку ipython, если она установлена.

2. Отладка Java-кода с IDE Netbeans

Как только проект будет успешно построен, код будет готов к выполнению. Для реализации проекта существует множество способов. Самый простой способ — это сочетание клавиш **F6** . Он запускает код со своими конфигурациями по умолчанию. Другой возможный выход — использовать зеленую кнопку воспроизведения для выполнения кода. При выполнении кода он должен отображать вывод, как показано ниже.

Рисунок - Netbeans Java Code Output

Вот как вы можете запустить любой простой проект Java в IDE Netbeans. Давайте дополнительно изменим код, добавив немного больше строк.

MainClass.java

```
package com.javacodegeeks;

import java.util.Scanner;

/**
 *
 * @author abhishekkothari
 */
public class MainClass {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String name = in.next();
        sayHello(name);
    }

    private static void sayHello(String name){
        System.out.println("Hello "+name);
```

```
    }

}
```

Теперь разберемся с процессом отладки. Отладка заключается в пошаговом выполнении кода для анализа возможных причин ошибки. Чтобы отладить код из определенной строки, нам нужно добавить точку останова. Чтобы добавить точку останова, просто нажмите на номер строки в левой части редактора, как показано ниже.

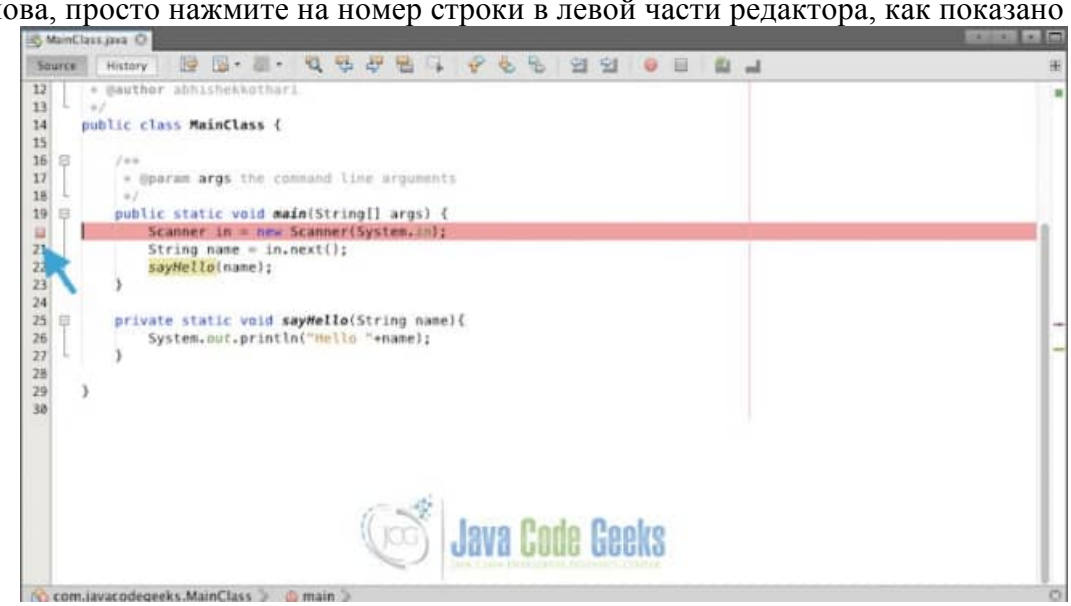


Рисунок - Отладка NetBeans

Чтобы начать процесс отладки, нажмите кнопку отладки, показанную ниже, или используйте сочетание клавиш **Ctrl + F5** .

Рисунок - Кнопка отладки NetBeans

После запуска отладки код останавливается на операторе, который вы пометили для отладки. В случае отладки существует 5 полезных операций, каждая из которых описана ниже.

1. Step Over (Shortcut-F8): переходит по текущему оператору и игнорирует любое выражение или вызов функции, которые могут существовать в операторе.

2. Выражение «Перешагнуть» (Shortcut-Shift + F8). Как следует из названия, оно просто переходит по выражению внутри оператора и позволяет проверить значение после вычисления выражения.

3. Step Into (Shortcut-F7): входит в вызов функции и переходит на первую строку соответствующего вызова функции в выражении.

4. Step Out (Shortcut-Cmd / Ctrl + F7): выход из функции после того, как в нее вступили.

5. Запуск до курсора (Shortcut-F4): выполняет код до оператора, на который указывает курсор в данный момент.

Кнопки показаны на рисунке ниже. Используйте кнопку для пошагового выполнения кода.

Рисунок - Кнопки отладки

Процесс отладки снабжен множеством окон, которые помогут быстрее решить проблему. Список этих окон можно найти, перейдя в **Window -> Debugging,** как показано ниже.

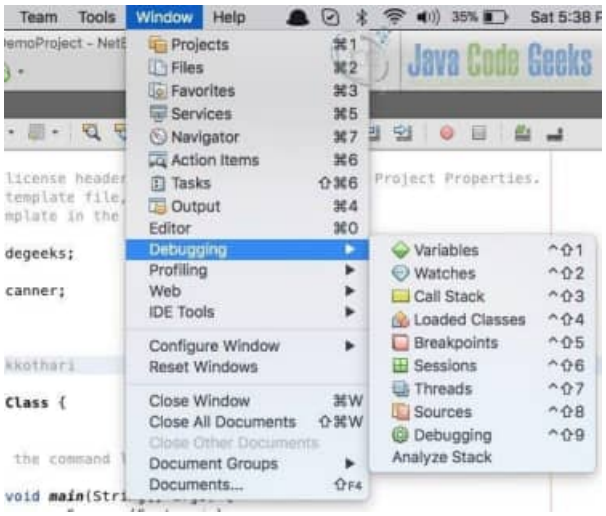


Рисунок - Параметры окна отладки

Окно переменных позволяет отслеживать изменения, которые происходят в переменных отлаживаемого кода, в то время как наблюдение позволяет следить за переменными, а также за выражениями. Окно Call Stack и Loaded классов позволяет отслеживать ход отладки при переходе между файлами внутри проекта. Окно точек останова отображает список точек останова в полном проекте. Сеансы и потоки предоставляют список запущенных потоков для текущего сеанса отладки. Источники и окна отладки используются для просмотра соответствующих источников и потока отладки.

Задания на работу

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Осуществите отладку проекта с графическим интерфейсом, созданного в Лабораторной работе №5, с использованием инструментальных средств.

Задание 3. Оформить отчет, описав этапы, методы и инструменты отладки проекта.

Контрольные вопросы

1. Что такое отладка?
2. Какие инструменты отладки вам известны?
3. Методы отладки.

Лабораторная работа №8
«ИНСПЕКЦИЯ КОДА МОДУЛЕЙ ПРОЕКТА»

Цель работы: получить практические навыки проведения инспекции кода модулей проекта.

Теоретические сведения

Не во всех случаях возможна разработка автоматических или хотя бы четко формализованных ручных тестов для проверки функциональности программной системы. В некоторых случаях выполнение программного кода, подвергаемого тестированию, невозможно в условиях, создаваемых тестовым окружением (например, во встроенных системах, если программный код предназначен для обработки исключительных ситуаций, создаваемых только при установке системы на реальное оборудование). В других случаях верифицируется не программный код, а проектная документация на систему, которую нельзя "выполнить" или создать для нее отдельные тестовые примеры. И в тех и в других случаях обычно прибегают к методу экспертных исследований программного кода или документации на *корректность* или *непротиворечивость*.

Такие экспертные исследования обычно называют инспекциями или просмотрами. Существует два типа инспекций – неформальные и формальные.

Формальная инспекция является четко управляемым процессом, структура которого обычно четко определяется соответствующим стандартом проекта. Таким образом, все формальные инспекции имеют одинаковую структуру и одинаковые выходные документы, которые затем используются при разработке.

Факт начала формальной инспекции четко фиксируется в общей базе данных проекта. Также фиксируются документы, подвергаемые инспекции, списки замечаний, отслеживаются внесенные по замечаниям изменения. Этим формальная инспекция похожа на автоматизированное тестирование – списки замечаний имеют много общего с отчетами о выполнении тестовых примеров.

В ходе формальной инспекции группой специалистов осуществляется независимая проверка соответствия инспектируемых документов исходным документам. Независимость проверки обеспечивается тем, что она осуществляется инспекторами, не участвовавшими в разработке инспектируемого документа. Входами процесса формальной инспекции являются инспектируемые документы и исходные документы, а выходами – материалы инспекции, включающие *список* обнаруженных несоответствий и решение об изменении статуса инспектируемых документов.

Рисунок иллюстрирует *место* формальной инспекции в процессе разработки программных систем.

Рисунок - Место формальной инспекции в процессе разработки программных систем

Инспекция кода (ревизия кода, рецензия кода, Code review) – это систематическая проверка исходного кода программы. Цель проверки – обнаружение и исправление ошибок, которые были пропущены, остались незамечены при разработке. Результат инспекции как правило – улучшение качество ПО и навыки разработчика.

В ходе инспекции, когда могут быть найдены, с последующим устранением, такие уязвимости, как ошибки в форматировании строк, утечка памяти, переполнение буфера, что улучшает безопасность ПО. Системы контроля версий также дают возможность проведения совместной инспекции исходного кода. Помимо этого, есть инструменты, которые предназначены именно для совместной инспекции.

Для проектов, в которых нет модульных тестов, инспекция — это, по большому счету, единственная эффективная методика предотвращения дефектов в коде. По крайней мере ничего другого на ум не приходит. Поэтому если в вашем проекте нет модульных тестов, в первую очередь следует внедрять инспекции и только потом — практику написания модульных тестов.

Список контрольных вопросов и этапов для облегчения процесса инспектирования:

1. Корректность реализации предметной области.
   - Функциональность реализована в полном объеме.
   - Соответствие спецификациям.
   - Верность заданных констант.
   - Обоснованность принятых допущений и соглашений.
   - Верная последовательность обработки.

2. «Читабельность» кода.

Тут главный контрольный вопрос нужно задать самому себе: «Смогу ли я в случае необходимости добавить новую элемент в инспектируемом коде?». Ответ должен быть положительным, в противном случае автору следует изменить код.

3. Наличие и полнота тестового покрытия.
   - Все публичные классы и методы тестируются.
   - Тестируются граничные условия.
   - Достаточность тестирования «положительных веток» выполнения.
   - Достаточность тестирования «отрицательных веток» выполнения.

4. Стоит добавить в свою копилку хорошие практики из чужого кода.

5. Корректность многопоточного кода.
   - Доступ к общим данным синхронизирован.
   - Отсутствуют deadlocks.
   - Для синхронизации используется идиома RAII.
   - Корректно обрабатывается возврат ресурсов в обработчиках ошибок.
   - Нет утечки ресурсов.

6. Код обрабатывает возможные ошибки корректно.

7. Отсутствие видимых нежелательных побочных эффектов в других частях системы.

8. Корректность языковых конструкций.
   - Применяются верные идиомы используемого языка и фреймворка.
   - Использование корректных библиотек.

- Отсутствие вновь изобретенных «велосипедов».
- Отсутствие дублирования кода.

9. Все переменные проинициализированы правильными значениями.

Задания на работу

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2. Провести инспекцию программного кода проекта (созданного в Лабораторной работе №5) согласно Списку контрольных вопросов (см. Теоретические сведения).

Задание 3. Оформить отчет зафиксировав ответы на вопросы их контрольного списка.

Контрольные вопросы

1. Что такое инспекция кода?
2. Какие виды инспекции кода вы знаете?
3. Определите понятия класс, экземпляр класса, объект.
4. На какие этапы можно разбить процесс создания экземпляров класса.
5. Как создать базовый файл класса, какой код он содержит.

Лабораторная работа №9

«ТЕСТИРОВАНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ СРЕДСТВАМИ ИНСТРУМЕНТАЛЬНОЙ СРЕДЫ РАЗРАБОТКИ, РАЗРАБОТКА ТЕСТОВЫХ МОДУЛЕЙ ПРОЕКТА ДЛЯ ТЕСТИРОВАНИЯ ОТДЕЛЬНЫХ МОДУЛЕЙ»

Цель работы: получить практические навыки проведения тестирования интерфейса пользователя, разработки тестовых модулей проекта для тестирования отдельных модулей.

Теоретические сведения

Выделяют четыре этапа разработки пользовательского интерфейса, а именно:

- Сбор и анализ информации от пользователей;
- Разработка пользовательского интерфейса;
- Построение пользовательского интерфейса;
- Подтверждение качества пользовательского интерфейса.

Первый шаг – определение профиля пользователя. Профиль пользователя отвечает на вопрос: «Что представляет наш пользователь?». Он позволяет нам составить представление о возрасте, образовании, предпочтениях пользователей.

Второй шаг – анализ стоящих перед пользователями задач.

Анализ стоящих перед пользователями задач – это определение того, чего хотят пользователи и каким образом они собираются решать свои задачи.

Концептуальное проектирование есть определение общей структуры и взаимодействия продукта. По определению Алана Купера, концептуальные принципы проектирования «помогают определить сущность продукта и его место в более широком контексте использования, который требуется пользователям».

Концептуальное проектирование включает:

- Определение типа интерфейса будущего приложения (монопольный, временный, фоновый);
- Организацию инфраструктуры взаимодействия;

Согласно определению Алана Купера, тип интерфейса определяет поведенческую сущность продукта, то есть то, как он предъявляет себя пользователю. Тип интерфейса – это способ описать то, как много внимания пользователь будет уделять взаимодействию с продуктом, и каким образом продукт будет реагировать на это внимание.

Следует отметить зависимость типа интерфейса от используемой технической платформы: персонального компьютера, Интернет, информационный киоск, мобильное устройство, бытовая техника.

Применительно к программам, которые разрабатываются для современных персональных компьютеров, в литературе также используется термин «настольное приложение».

Нефункциональное тестирование – это проверка характеристик программы. Иначе говоря, когда проверяется не именно правильность работы, а какие-либо свойства (внешний вид и удобство пользования, скорость работы и т.п.).

1. *Тестирование пользовательского интерфейса (GUI)* – тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя.

- навигация
- цвета, графика, оформление

- содержание выводимой информации
- поведение курсора и горячие клавиши
- отображение различного количества данных (нет данных, минимальное и максимальное количество)
- изменение размеров окна или разрешения экрана

2. Тестирование удобства использования (Usability Testing) – тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации.

- визуальное оформление
- навигация
- логичность

3. Тестирование доступности (Accessibility testing) – тестирование, которое определяет степень легкости, с которой пользователи с ограниченными способностями могут использовать систему или ее компоненты.

4. Тестирование интернационализации – тестирование способности продукта работать в локализованных средах (способность изменять элементы интерфейса в зависимости от длины и направления текста, менять сортировки/форматы под различные локали и т.д.). (Максим Черняк). Интернационализация – это процесс, упрощающий дальнейшую адаптацию продукта к языковым и культурным особенностям региона, отличного от того, в котором разрабатывался продукт. Это адаптация продукта для потенциального использования практически в любом месте.

Интернационализация производится на начальных этапах разработки, в то время как локализация — для каждого целевого языка.

5. Тестирование локализации (Localization testing) – тестирование, проводимое с целью проверить качество перевода продукта с одного языка на другой.

6. Тестирование производительности или нагрузочное тестирование – процесс тестирования с целью определения производительности программного продукта.

Интерфейс настольных приложений можно отнести к одному из трёх типов: монопольный, временный и фоновый.

К приложениям *монопольного типа* относятся программы, которые полностью завладевают вниманием пользователя на длительные периоды времени. Для продуктов с монопольным интерфейсом характерна длительная работа в течение длительных отрезков времени. В процессе работа пользователя монопольный продукт является его основным инструментом и преобладает над остальными.

Приложение *временного типа* приходит и уходит, предлагая одну функцию и ограниченный набор связанных с этой функцией элементов управления. Приложение этого типа вызывается при необходимости, делает свою работу и быстро исчезает, позволяя пользователю продолжить прерванную (как правило, в окне монопольного приложения) деятельность. Типичный пример сценария работы с временным приложением – вызов Проводника Windows для поиска и открытия другого файла в то время, когда пользователь уже редактирует один файл в MS Word.

Фоновыми называют приложения, которые в нормальном «рабочем» состоянии не взаимодействуют с пользователем. Такие программы выполняют задачи, которые в целом важны, но не требуют вмешательства пользователя. Примеры: драйвер принтера, подключение к сети.

Инфраструктура взаимодействия включает варианты поведения приложения. Создание инфраструктуры взаимодействия предполагает выполнение шести шагов:

Шаг 1. Определение форм-фактора, типа приложения и способов управления.

Шаг 2. Определение функциональных и информационных элементов.

Шаг 3. Определение функциональных групп и иерархических связей между ними.

Шаг 4. Макетирование общей инфраструктуры взаимодействия.

Шаг 5. Создание ключевых сценариев.

Шаг 6. Выполнение проверочных сценариев для верификации решений.

Форм-фактор – это зависимость вида пользовательского интерфейса от используемой технической платформы.

Функциональные и информационные элементы – это зримые представления функций и данных, доступные пользователю посредством интерфейса. Это конкретные проявления функциональных и информационных потребностей, выявленных на стадии выработки требований.

Информационные элементы – это, как правило, фундаментальные объекты интерактивных продуктов.

Функциональные элементы – это операции, которые могут выполняться над информационными объектами и представляющими эти объекты элементами интерфейса. В большинстве случаев функциональные элементы представляют собой инструменты, работающие с информационными элементами, а также контейнеры, содержащие информационные элементы.

Макетирование общей инфраструктуры взаимодействия Аланом Купером охарактеризовано как «фаза прямоугольников», поскольку эскизы будущего интерфейса начинаются с разделения каждого представления на прямоугольные области, соответствующие панелям, элементам управления и другим высокоуровневым контейнерам. При этом каждому прямоугольнику даётся своё название и показывается, каким образом одна группа элементов может влиять на другие. Содержательно этот шаг предназначен для исследования различных вариантов представления информации и функциональности в интерфейсе, при этом затраты на внесение изменений должны быть минимальны.

Известны два вида макетов: с жёсткой компоновкой и без компоновки.

При этом макет с жёсткой компоновкой:

- содержит взаимное расположение элементов и визуальную информацию о приоритетах;
- ограничивает работу графического дизайнера.

Для макета без компоновки характерно то, что он:

- не содержит графического представления элементов;
- содержит текстовое описание элементов и их приоритетов;
- не ограничивает работу графического дизайнера.

Сценарий определяется Аланом Купером как средство описания идеального для пользователя взаимодействия. Истоки этого понятия восходят к публикациям сообщества HCI (Human-Computer Interaction – взаимодействие человека и компьютера), где оно увязывалось с указанием на метод решения задач проектирования через конкретизацию, которая понималась как использование специально составленного рассказа, чтобы одновременно конструировать и иллюстрировать проектные решения. Применение сценарного подхода к проектированию сосредоточено на описании того, как пользователи решают задачи. Такое описание включает характеристику обстановки рабочей среды, а также агентов, или действующих лиц, которые являются абстрактными представителями пользователей.

Сценарии, основанные на персонажах, есть краткие описания одного или более персонажей, применяющих программный продукт для достижения конкретных целей. Сценарии позволяют начинать проектирование с рассказа, описывающего идеальный с точки зрения персонажа опыт, при этом фокусируя внимание на людях, их образе мысли и поведении.

Процесс выработки требований с использованием персонажей и сценариев состоит из следующих пяти шагов:

Шаг 1. Постановка задач и определение образа продукта.

Шаг 2. Мозговой штурм.

Шаг 3. Выявление ожиданий персонажей.

Шаг 4. Разработка контекстных сценариев.

Шаг 5. Выявление требований.

Ключевой сценарий описывает взаимодействие персонажа с системой в терминах лексикона инфраструктуры взаимодействия. Он отражает магистральные пути внутри интерфейса, используемые персонажем чаще всего (например, ежедневно). Ключевые сценарии сосредоточены на задачах. Например, в случае приложения для работы с электронной почтой ключевые действия – это просмотр и создание новых сообщений, а не настройка нового почтового сервера.

Ключевые сценарии, как правило, являются результатом развития контекстных сценариев, но целенаправленно описывают взаимодействие персонажа с различными функциональными и информационными элементами, составляющими общую инфраструктуру взаимодействия.

Контекстные сценарии сосредоточены на целях, же ключевые сценарии больше сосредоточены на задачах, намеки на которые или описания которых содержатся в контекстных сценариях.

Оптимизация – преобразование программы, сохраняющее ее семантику (конструкции языка программирования), но уменьшающие ее размер и время выполнения.

Виды оптимизация программы:

- глобальная (всей программы);
- локальная (нескольких соседних операторов, образующих линейный участок);
- квазилокальная (фрагментов программы фиксированной структуры, например, циклов).

Способы оптимизации:

1. Разгрузка участков повторяемости: вынесение вычислений из многократно проходимых исполняемых участков программы на участки программы, редко проходимые. Таким образом, это преобразование тела цикла или рекурсивных процедур.

2. Упрощение действий: улучшение программы за счет замены групп вычислений на группу вычислений, дающих тот же результат с точки зрения всей программы, но имеющих меньшую сложность.

а) упрощение действий происходит при замене сложных операций в выражениях более простыми:

```
x / 0.4 -> x*0.25;
```

б) преобразование по объединению или расчленению циклов, по перестановке заголовков циклов, по удалению избыточных выражений (замене их на переменную).

3. Реализация действия: действия над константами заменяются на константы; ликвидация константных распознавателей -замена условного оператора на одну из его ветвей, если его выбирающее условие-выражение имеет постоянное значение; удаление из программы ненужных пересылок вида:

```
Y=F(W), X=Y на X=F(W)
```

4. Чистка программы (удаление ненужных конструкций): недостижимых операторов, существенных операторов, неиспользуемых переменных, видов, операций.

5. Сокращение размера программы: вынесение одинаковых конструкций в начальную или конечную точку программы; поиск в программе похожих объектов и формирование их в виде процедуры.

6. Экономия памяти -уменьшение объема памяти, отводимые под информационные объекты программы (например, параметры процедуры).

Задания на работу

В качестве основы для выполнения данной лабораторной работы предлагается использовать приложение-проект, разработанное в Лабораторной работе №5.

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2.

Предметная область и сфера применения. Правильное определение этих аспектов является основой для разработки UI в частности и всего приложения в целом.

Определение целевой аудитории, направлен на выделение из общей массы группы (или групп) потенциальных пользователей разрабатываемой программы. Естественно, что цели, задачи, способности и возможности групп пользователей будут существенно различаться.

Модель пользователя, или *профиль*, формируется в результате анализа целевых групп. Она отражает наиболее общие черты, характерные для представителей группы и может представлять следующую информацию о пользователе:

- Социальные и демографические характеристики (возраст, пол, основной язык, род занятий, потребности, привычки и т.п.).
- Уровень компьютерной грамотности.
- Цель и задачи, решаемые пользователем.
- Окружение (рабочее место, конфигурация оборудования, используемая операционная система и т.п.)
- Требования, специфичные для конкретной целевой группы.

После выделения одного или нескольких основных профилей пользователей и определения задач, стоящих перед ними, переходят к следующему этапу проектирования. Он связан с составлением пользовательских сценариев. Сценарий — это описание действий, выполняемых пользователем в рамках решения конкретной задачи на пути достижения его цели. Очевидно, что достигнуть некоторой цели можно, решая ряд задач. Каждую их них пользователь может решать несколькими способами, следовательно, должно быть сформировано несколько сценариев. Чем больше их будет, тем ниже вероятность того, что некоторые ключевые объекты и операции будут упущены.

1) Оформить отчет, определив в нем: предметную область и сферу применения, целевую аудиторию, модель пользователя, пользовательские сценарии.

Задание 3.

1) Для модуля проекта выбрать параметр оптимизации и определить его количественные характеристики.

2) Провести оптимизацию программы по выбранному параметру.

3) Сравнить характеристики исходного модуля и модуля, полученного в результате оптимизации.

4) Оформить отчет, содержащий описание, обоснование и результаты оптимизации программы.

Контрольные вопросы

1. Что такое интерфейс?
2. Какие типы пользовательских интерфейсов существуют?
3. Перечислите этапы разработки пользовательских интерфейсов?
4. К какому типу интерфейсов будет относиться интерфейс, разработанный в данной лабораторной работе?

5. Какие модели интерфейсов существуют?
6. Какая модель интерфейса будет использована в данной работе?
7. Почему необходимо проводить оптимизацию, а не минимизацию программы?
8. От чего зависит выбор метода оптимизации?
9. Почему большое внимание уделяется циклическим участкам?
10. К каким нежелательным последствиям может привести оптимизация?

Лабораторная работа №10
«ВЫПОЛНЕНИЕ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ, ТЕСТИРОВАНИЕ ИНТЕГРАЦИИ»

Цель работы: получить практические навыки выполнения функционального тестирования, тестирование интеграции.

Теоретические сведения

1. Основные сведение о функциональном тестировании

Процесс тестирования состоит из трёх этапов:
1. Проектирование тестов.
2. Исполнение тестов.
3. Анализ полученных результатов.

На первом этапе решается вопрос о выборе некоторого подмножества множества тестов, которое сможет найти наибольшее количество ошибок за наименьший промежуток времени. На этапе исполнения тестов проводят, запуск тестов и отлавливают ошибки в тестируемом программном продукте.

Виды тестов

- Функциональное тестирование (functional testing)
- Системное тестирование (system testing)
- Тестирование проиводительности (performance testing)
- Регрессионное тестирование (regression testing)
- Модульное тестирование (unit testing)
- Тестирование безопасности (security testing)

Функциональные тесты составляются на уровне спецификации, до решения задачи. Будущий алгоритм рассматривается как «черный ящик» - функция с неизвестной (или не рассматриваемой) структурой, преобразующая входы в выходы. Суть функциональных тестов: каким бы способом ни решалась задача, при заданных входных значениях должны получиться соответствующие выходные значения.

Структурные тесты составляются для проверки логики решения, или логики работы уже готового алгоритма. Логика определяется последовательностью операций, их условным выполнением или повторением (т.е. композицией базовых конструкций). Совокупность структурных тестов должна обеспечить проверку каждой из таких конструкций.

Чаще всего совокупность тщательно составленных функциональных тестов покрывает множество структурных тестов.

Приведенные понятия различаются тем, что первое рассматривает программу только с точки зрения входов и выходов, тогда как второе относится к ее структуре; но оба понятия не касаются процесса организации тестирования.

Общая последовательность разработки тестов

Наиболее рациональная процедура заключается в том, что сначала разрабатываются функциональные тесты, а затем – структурные.

Функциональное тестирование (тестирование «черного ящика»)

При функциональном тестировании выявляются следующие категории ошибок:

- некорректность или отсутствие функций;

- ошибки интерфейса;
- ошибки в структурах данных;
- ошибки машинных характеристик (нехватка памяти и др.);
- ошибки инициализации и завершения.

Техника тестирования ориентирована:

- на сокращение необходимого количества тестовых вариантов;
- на выявление классов ошибок, а не отдельных ошибок.

Способы функционального тестирования

Разбиение на классы эквивалентности

Это самый популярный способ. Его суть заключается в разделении области входных данных программы на классы эквивалентности и разработке для каждого класса одного тестового варианта.

Класс эквивалентности – набор данных с общими свойствами, в силу чего при обработке любого набора данных этого класса задействуется один и тот же набор операторов .

Классы эквивалентности определяются по спецификации программы. Тесты строятся в соответствии с классами эквивалентности, а именно: выбирается вариант исходных данных некоторого класса и определяются соответствующие выходные данные.

Самыми общими классами эквивалентности являются классы допустимых и недопусти-мых (аномальных) исходных данных. Описание класса строится как комбинация условий, описывающих каждое входное данное.

Условия допустимости или недопустимости данных задают возможные значения данных и могут описывать:

- некоторое конкретное значение; определяется один допустимый и два недопустимых класса эквивалентности: заданное значение, множество значений меньше заданного, множество значений больше заданного;
- диапазон значений; определяется один допустимый и два недопустимых класса эквивалентности: множество значений в границах диапазона; множество значений, выходящих за левую границу диапазона; множество значений, выходящих за правую границу диапазона;
- множество конкретных значений; определяется один допустимый и один недопустимый класс эквивалентности: заданное множество и множество значений, в него не входящих.

Такие классы можно описать языком логики, например, языком исчисления предикатов. Описания более сложных условий и соответствующих классов (например, элементы массива должны находиться в некотором диапазоне и при этом массив не должен содержать нулевых элементов) могут быть построены на основании приведенных выше условий.

Анализ граничных значений

Этот способ построения тестов дополняет предыдущий. Также он предполагает анализ значений, лежащих на границе допустимых и недопустимых данных. Построение таких тестов часто диктуется интуицией.

Основные правила построения тестов:

- если условие правильности данных задает диапазон, то строятся тесты для левой и правой границы диапазона; для значений чуть левее левой и чуть правее правой границы;
- если условие правильности данных задает дискретное множество значений, то строятся тесты для минимального и максимального значений; для значений чуть меньше минимума и чуть больше максимума;

- если используются структуры данных с переменными границами (массивы), то строятся тесты для минимального и максимального значения границ.
- Диаграммы причин-следствий
- Взаимосвязь классов эквивалентности и соответствующих им действий описывается формально в виде графа на основе автоматного подхода. Граф преобразуется в таблицу решений, столбцы которой в свою очередь преобразуются в тестовые варианты.

2. Тестирование интеграции

Интеграционное тестирование — это тестирование программного обеспечения на корректность взаимодействия нескольких модулей, объединенных в единое целое. Несмотря на то, что результатом тестирования и верификации отдельных модулей, составляющих программную систему, является заключение о том, что эти модули являются внутренне непротиворечивыми и соответствуют требованиям, это не гарантирует их корректную совместную работу. Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности. Тестирование этих проектируемых единиц — объединения, множества или группа модулей — выполняются через их интерфейс, используя тестирование «чёрного ящика».

Интеграционное тестирование называют еще тестированием архитектуры системы. Результаты выполнения интеграционных тестов – один из основных источников информации для процесса улучшения и уточнения архитектуры системы, межмодульных и межкомпонентных интерфейсов. Т.е. с интеграционные тесты проверяют корректность взаимодействия компонент системы.

Интеграционное тестирование, как правило, представляет собой итеративный процесс, при котором проверяется функциональность все более и более увеличивающейся в размерах совокупности модулей.

Модульное тестирование, или юнит-тестирование (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Интеграционные тесты должны проверять совместную работу отдельных модулей или наборов модулей, между которыми есть зависимости. При этом тесты должны проверить, правильно ли обрабатываются входные данные отдельных методов при вызове их при разных ситуациях и наборах исходных данных (функциональная сторона интеграционного тестирования) и проверить стабильность работы модуля при разных исходных данных (проверка на надежность работы модуля интеграционным тестированием).

Существует несколько подходов к интеграционному тестированию:

- Снизу вверх. Сначала собираются и тестируются модули самих нижних уровней, а затем по возрастанию к вершине иерархии. Данный подход требует готовности всех собираемых модулей на всех уровнях системы.
- Сверху вниз. Данный подход предусматривает движение с высокоуровневых модулей, а затем направляется вниз. При этом используются заглушки для тех модулей, которые находятся ниже по уровню, но включение которых в тест еще не произошло.
- Большой взрыв. Все модули всех уровней собираются воедино, а затем тестируется. Данный метод экономит время, но требует тщательной проработки тест кейсов.

Преимущество.

Интеграционное тестирование позволяет имитировать действия пользователей и быстро получать подтверждение, что программный продукт успешно взаимодействует с другими системами. Такой подход гарантирует сразу несколько преимуществ:

1. Предотвращение появления критичных ошибок в опытно-промышленной эксплуатации
2. Снижение влияния человеческого фактора
3. Экономия затрат на исправление дефектов

Главной задачей интеграционного тестирования является поиск ошибок, связанных с взаимодействием модулей системы или нескольких систем. В результате все смежные системы и модули одной системы должны работать согласованно. Способы проведения интеграционного тестирования подбираются в зависимости от интеграционных решений.

2.1. Как сделать интеграционное тестирование?

Алгоритм интеграционного тестирования:

1. Подготовка план интеграционных тестов
2. Разработка тестовых сценариев.
3. Выполнение тестовых сценариев и фиксирование багов.
4. Отслеживание и повторное тестирование дефектов.
5. Повторять шаги 3 и 4 до успешного завершения интеграции.

Атрибуты Интеграционного тестирования

Включает в себя следующие атрибуты:

- Методы / Подходы к тестированию (об этом говорили выше).
- Области применения и Тестирование интеграции.
- Роли и обязанности.
- Предварительные условия для Интеграционного тестирования.
- Тестовая среда.
- Планы по снижению рисков и автоматизации.

Критерии старта и окончания интеграционного тестирования

Критерии входа и выхода на этап Интеграционного тестирования, независимо от модели разработки программного обеспечения

Критерии старта:

- Модули и модульные компоненты
- Все ошибки с высоким приоритетом исправлены и закрыты
- Все модули должны быть заполнены и успешно интегрированы.
- Наличие плана Интеграционного тестирования, тестовый набор, сценарии, которые должны быть задокументированы.
- Наличие необходимой тестовой среды

Критерии окончания:

- Успешное тестирование интегрированного приложения.
- Выполненные тестовые случаи задокументированы
- Все ошибки с высоким приоритетом исправлены и закрыты
- Технические документы должны быть представлены после выпуска Примечания.

Лучшие практики / рекомендации по интеграционному тестированию

- Сначала определите интеграционную тестовую стратегию, которая не будет противоречить вашим принципам разработки, а затем подготовьте тестовые сценарии и, соответственно, протестируйте данные.
- Изучите архитектуру приложения и определите критические модули. Не забудьте проверить их на приоритет.
- Получите проекты интерфейсов от команды разработки и создайте контрольные примеры для проверки всех интерфейсов в деталях. Интерфейс к базе данных / внешнему оборудованию / программному обеспечению должен быть детально протестирован.
- После тестовых случаев именно тестовые данные играют решающую роль.
- Всегда имейте подготовленные данные перед выполнением. Не выбирайте тестовые данные во время выполнения тестовых случаев.

Задания на работу

В качестве основы для выполнения данной лабораторной работы предлагается использовать приложение-проект, разработанное в Лабораторной работе №5.

Задание 1. Ознакомьтесь с теоретическим материалом.

Задание 2.

1) Разработать тестовые наборы для функционального тестирования.
2) Провести тестирование программы и представить результаты.
3) Выработать рекомендации для корректировки тестируемой программы.

Задание 3.

1) Провести интеграционное тестирование приложения согласно алгоритму и представить результаты.
3) Выработать рекомендации для корректировки тестируемого приложения.

Задание 4. Оформите отчет.

Контрольные вопросы

1. Что такое тестирование ПС?
2. Дайте определение понятия интеграционное тестирование.
3. Цели интеграционного тестирования.
4. Чем тестирование отличается от отладки ПС?
5. Для чего проводится функциональное тестирование?
6. В чем сущность метода «белого ящика»?
7. В чем сущность метода «черного ящика».
8. Каковы правила тестирования программы «как черного ящика»?
9. Как проводится тестирования программы по принципу «белого ящика»?
10. Как осуществляется сборка программы при модульно тестировании?
11. Преимущества «раннего начала» тестирования.

Лабораторная работа №11
«ДОКУМЕНТИРОВАНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ»

Цель работы: получить практические навыки выполнения функционального тестирования, тестирование интеграции.

Теоретические сведения

Типы тестов по знанию коду

Черный ящик – тестирование системы, функциональное или нефункциональное, без знания внутренней структуры и компонентов системы. У тестировщика нет доступа к внутренней структуре и коду приложения либо в процессе тестирования он не обращается к ним.

Белый ящик – тестирование основанное на анализе внутренней структуры компонентов или системы. У тестировщика есть доступ к внутренней структуре и коду приложения.

Серый ящик – комбинация методов белого и черного ящика, состоящая в том, что к части кода архитектуры у тестировщика есть, а к части кода – нет.

Типы тестов по степени автоматизации

Ручное – тестирование, в котором тест-кейсы выполняются тестировщиком вручную без использования средств автоматизации.

*Автоматизированно*е – набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство.

Типы тестов по изолированности компонентов

Unit/component (модульное) – тестирование отдельных компонентов (модулей) программного обеспечения.

Integration (интеграционное) – тестируется взаимодействие между интегрированными компонентами или системами.

System (системное) – тестируется работоспособность системы в целом с целью проверки того, что она соответствуе установленным требованиям.

Типы тестов по подготовленности.

Интуитивное тестирование выполняется без подготовки к тестам, без определения ожидаемых результатов, проектирования тестовых сценариев.

Исследовательское тестирование – метод проектирования тестовых сценариев во время выполнения этих сценариев. Тестировщик совершает проверки, продумывает их, придумывает новые проверки, часто использует для этого полученную информацию.

Тестирование по документации – тестирование по подготовленным тестовым сценариям, руководству по осуществлению тестов.

Титы тестов по месту и времени проведения

User Acceptance Testing (UAT) (приемочное тестирование) – формальное тестирование по отношению к потребностям, требованиям и бизнес процессам пользователя, проводимое с целью определения соответствия системы критериям приёмки и дать возможность пользователям, заказчикам или иным авторизованным лицам определить, принимать систему.

Alpha Testing (альфа-тестирование) – моделируемое или действительное функциональное тестирование, выполняется в организации, разрабатывающей продукт, но

не проектной командой (это может быть независимая команда тестировщиков, потенциальные пользователи, заказчики). Альфа тестирование часто применяется к коробочному программному обеспечению в качестве внутреннего приемочного тестирования.

Beta Testing (бета-тестирование) – эксплуатационное тестирование потенциальными или существующими клиентами/заказчиками на внешней стороне (в среде, где продукт будет использоваться) никак связанными с разработчиками, с целью определения действительно ли компонент или система удовлетворяет требованиям клиента/заказчика и вписывается в бизнес-процессы. Бета-тестирование часто проводится как форма внешнего приемочного тестирования готового программного обеспечения для того, чтобы получить отзывы рынка.

Типы тестов по объекту тестирования

Functional testing (функциональное тестирование) – это тестирование, основанное на анализе спецификации, функциональности компонента или системы. Функциональным можно назвать любой вид тестирования, который согласно требованиям проверяет правильную работу.

Safety testing (тестирование безопасности) – тестирование программного продукта с целью определить его безопасность (безопасность – способность программного продукта при использовании оговоренным образом оставаться в рамках приемлемого риска причинения вреда здоровью, бизнесу, программам, собственности или окружающей среде.

Security testing (тестирование защищенности) – это тестирование с целью оценить защищенность программного продукта. Тестирование защищенности проверяет фактическую реакцию защитных механизмов, встроенных в систему, на проникновение.

Compatibility testing (тестирование совместимости) – процесс тестирования для определения возможности взаимодействия программного продукта, проверка работоспособности приложения в различных средах (браузеры и их версии, операционные системы, их типа, версии и разрядность)

Виды тестов:

- кросс-браузерное тестирование (различные браузеры или версии браузеров)
- кросс-платформенное тестирование (различные операционные системы или версии операционных систем)

Итоговый отчет можно разделить на части с соответствующей информацией:

1) Приветствие/введение.
2) Общая информация (Common Information).
3) Тестовое окружение (Test Platform).
4) Рекомендации QA (QA Recommendations).
5) Детализированная информация (Detailed Information).
6) Окончание содержимого.

Приветствие/введение

Свое письмо с отчетом необходимо начать с приветствия всех адресатов. Если по каким-либо причинам произошла задержка данных отчета, либо не весь запланированный функционал был проверен, то эту информацию необходимо предоставить в начале письма. Также в самом начале письма следует указывать, если были какие-то внешние факторы, препятствующие проверке какой-то части функционала. Если во время тестирования не произошло никаких форс-мажорных обстоятельств, то достаточно обычного вежливого приветствия и далее уже переход к следующим пунктам.

Общая информация (Common Information)

В данной части отчета описывается, какие виды тестов проводились. Зачастую указываются модули, которые тестировались или функционал. Стоит удостовериться, не забыта ли какая-то часть функционала, особенно это актуально, когда нужно собрать итоговый отчет, соединив в себе данные о разных видах тестов и функционале.

Тестовое окружение (Test Platform)

Как правило, в этой части указываются:

- Название проекта.
- Номер сборки.
- Ссылка на проект (сборку). Необходимо убедиться, что зайдя по этой
- ссылке вы действительно попадаете на проект или можете установить приложение.

При указании данных в этой части отчета нужно быть очень внимательным, т.к. неправильная ссылка на сборку или неверный номер сборки не дают достоверной информации всем заинтересованным людям, а также затрудняют работу человеку, собирающему финальный отчет.

Рекомендации QA (QA Recommendations)

Данная часть отчета является наиболее важной, т.к. здесь отражается общее состояние сборки. Здесь показывается аналитическая работа тестировщика, его рекомендации по улучшению функционала, наиболее слабые места и наиболее критичные дефекты, динамика изменения качества проекта.

В этом разделе должна быть информация о следующем:

- Указан функционал (часть функционала), который заблокирован для проверки. Даны пояснения почему этот функционал не проверен (указаны наиболее критичные дефекты).
- Произведен анализ качества проверенного функционала. Следует указать, улучшилось оно или ухудшилось по сравнению с предыдущей версией, какое качество на сегодняшний момент, какие факторы повлияли на выставление именно такого качества сборки.
- Если качество сборки ухудшилось, то обязательно должны быть указаны регрессионные места.
- Наиболее нестабильные части функционала следует выделить и указать причину, по которой они таковыми являются.
- Даны рекомендации по тому функционалу и дефектам, скорейшее исправление которых является наиболее приоритетным.
- Список наиболее критичных для сборки дефектов, с указанием названия и их критичности.
- Для отчета уровня Smoke обязательно указать весь нестабильный функционал. Если сборка является релизной или предрелизной, то любое ухудшение качества является критичным и важно об этом сообщить менеджеру как можно раньше.

Помимо всего вышеуказанного для релизных и предрелизных сборок в отчете о качестве продукта важно указывать следующее:

- Дана информация о всех проблемах, характерных сборке. Проведен анализ, насколько оставшиеся проблемы являются критичными для конечного пользователя.
- Указаны дефекты, которые следует исправить, чтобы качество конечной сборки было выше.

Детализированная информация (Detailed Information)

В данной части отчета описывается более подробная информация о проверенных частях функционала, устанавливается качество каждой проверенной части функционала(модуля) в отдельности.

В зависимости от типа проводимых тестов, эта часть отчета будет отличаться.

Smoke-тестирование – выполнение минимального набора тестов для выявления явных дефектов критичной функциональности.

При оценке качества функционала на уровне Smoke теста, оно может быть либо Приемлемым, либо Неприемлемым. Качество сборки зависит от нескольких факторов:

- Если это релизная или предрелизная сборка, то для выставления Приемлемого качества на уровне Smoke не должно быть найдено функциональных дефектов.
- Наличие нового функционала. Новый функционал, который впервые поставляется на тестирование, не должен содержать дефектов уровня Smoke для выставления Приемлемого качества всей сборки.
- Чтобы установить сборке Приемлемое качество, не должно быть дефектов уровня Smoke у того функционала, по которому планируется проводить полные тесты.
- Все наиболее важные части функционала отрабатывают корректно, тогда качество всего функционала на уровне Smoke может быть оценено, как Примлемое.

В части о детализированной информации качества сборки следует более подробно описать проблемы, которые были найдены во время теста.

Defect Validation (DV) – проверка результата исправления дефектов. Включает в себя проверку на воспроизводимость дефектов, которые были исправлены в новой сборке продукта, а также проверку того, что исправление не повлияло на ранее работавшую функциональность.

В этой части отчета указывается качество о проведении валидации дефектов.

Здесь должна быть следующая информация:

- Общее количество всех дефектов, поступивших на проверку.
- Количество неисправленных дефектов и их процент от общего количества.
- Список дефектов, которые не были проверены и причины, по которым этого не было сделано.
- Наглядная таблица с неисправленными дефектами.

По вышеуказанным результатам выставляется качество теста. Если процент неисправленных дефектов < 10%, то качество Приемлемое, если > 10%, то качество Неприемлемое.

New Feature Test (NFT, AT of NF) – определение качества поставленной на тестирование новой функциональности, которая ранее не тестировалась.

При проведении полного теста нового функционала качество отдельно проверенного функционала может быть: Высокое, Среднее, Низкое.

В отчете следует отдельно указывать информацию о качестве каждой части нового функционала. В этой части отчета должна быть следующая информация:

- Дана общая оценка реализации нового функционала (сгруппированная по качеству).
- Подробная (детальная) информация о качестве каждой из частей новой функциональности.
- Проведен анализ каждой из новых функций в отдельности.
- Даны ясные пояснения о выставлении соответствующего качества.
- Даны рекомендации по улучшению качества (какие проблемы следует исправить).

- Показана таблица с новыми функциями (название), их качеством, статусом фуннкции из CQ.

Minimal Acceptance Test (MAT, Positive test): тестирование системы или ее части только на валидных данных (валидные данные – это данные, которые необходимо использовать для корректной работы модуля/функции). При тестировании проверяется правильной работы всех функций и модулей с валидными данными.

Acceptance Test (AT): полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях. Вид теста, направленный на подтверждение того, что приложение может использоваться по назначению при любых условиях.

Regression testing (регрессионное тестирование) – проводится с целью оценки качества ранее реализованной функциональности. Включает в себя проверку стабильности ранее реализованной функциональности после внесения изменений, например добавления новой функциональности, исправление дефектов, оптимизация кода, разворачивание приложения на новом окружении. Регрессионное тестирование может быть проведено на уровне Smoke, MAT или AT.

Если проводились тесты указанных уровней, то в первую очередь при написании отчета нужно анализировать динамику изменения качества проверенной функциональности в сравнении с более ранними версиями сборки. Также как и у предыдущего вида тестов, качество этих может быть: Высокое, Среднее, Низкое.

Для указанных видов тестов в данной части отчета должна быть описана информация следующего характера:

- Дана сравнительная характеристика каждой из частей функционала в сравнении с предыдущими версиями сборки.
- Подробная (детальная) информация о качестве каждой из частей проверенной функциональности.
- Даны ясные пояснения о выставлении соответствующего качества каждой функции в отдельности.
- Даны рекомендации по улучшению качества (какие проблемы следует исправить).

Окончание содержимого

В завершении содержимое отчета должно включать в себя информацию следующего характера:

- Ссылка на тест-план.
- Ссылка на документ feature matrix (если таковой имеется).
- Ссылка на документ со статистикой (если таковой имеется).
- Общее количество всех новых дефектов.
- Подпись высылающего отчет.

Данные ссылки должны быть корректными, необходимо проверить достоверную ли информацию получает пользователь, открывший ссылку. Следует обращать особое внимание на подпись, удостоверьтесь, что указана именно ваша подпись либо какая-то универсальная для определенного проекта подпись.

Задания на работу

В качестве основы для выполнения данной лабораторной работы предлагается использовать приложение-проект, разработанное в Лабораторной работе №5.

Задание 1. Запустите ранее созданное приложение.

Задание 2. Составьте итоговый отчет по результатам тестирования приложения.

Задание 3. Оформите отчет.

Контрольные вопросы

1. Какая структура итогового отчета о результатах тестирования?
2. Что содержится в разделе Приветствие?
3. Что содержится в разделе Общая информация?
4. Что содержится в разделе Тестовое окружение?
5. Что содержится в разделе Рекомендации QA?
6. Что содержится в разделе Детализированная информация?
7. Что содержится в разделе Окончание содержимого?

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тульский государственный университет»
Технический колледж имени С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторно-практических работ

по междисциплинарному курсу
МДК 02.03 Математическое моделирование

профессионального модуля
ПМ.02. ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ

специальности СПО
09.02.07 Информационные системы и программирование

Тула 2023

УТВЕРЖДЕНЫ

Цикловой комиссией информационных технологий

Протокол от «13» января 2023 г. № 6

Председатель цикловой комиссии И.В. Миляева

Составитель Воронцова Н.В.

Занятие 1. Практическое занятие
Построение простейших математических моделей, решение задач линейного программирования симплекс–методом

1.Теоретическая часть

Симплексный метод решения задачи линейного программирования основан на переходе от одного опорного плана к другому, при котором значение левой функции возрастает (при условии, что данная задача имеет оптимальный план и каждый ее опорный план является невырожденным). Указанный переход возможен, если известен какой-нибудь исходный опорный план. Рассмотрим задачу, для которой этот план можно непосредственно записать.

Пусть требуется найти максимальное значение функции

$$F=c_1x_1+c_2x_2+\dots c_nx_n$$

При условиях

$$\begin{cases} x_1 + a_{1m+1}x_{m+1} + \dots + a_{1n}x_n = b_1, \\ x_2 + a_{2m+1}x_{m+1} + \dots + a_{2n}x_n = b_2, \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ x_m + a_{mm+1}x_{m+1} + \dots + a_{mn}x_n = b_m, \\ x_j \ge 0 \left(j = \overline{1,n}\right) \end{cases}$$

Здесь a_{ij}, bi и, C_j (i= 1,m;;.j .=. 1,n) заданное постоянные числа $(m < n;\ b_i>0)$

Векторная форма данной задачи имеет следующий вид:

найти максимум функции $F=\sum_{j=1}^{n} c_j x_j \rightarrow \max$

при условиях

$$x_1A_1 + x_2A_2 + \dots + x_mA_m + \dots + x_nA_n = B,$$
$$x_j \ge 0 \ \left(j = \overline{1,n}\right)$$

где

$$A_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}; \quad A_2 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}; \dots; \quad A_m = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}; \quad A_{m+1} = \begin{pmatrix} a_{1m+1} \\ a_{2m+1} \\ \vdots \\ a_{mm+1} \end{pmatrix}; \dots;$$

$$A_n = \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix}; \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

(13)- (15)

План $X = (x_1, x_2, \ldots x_n)$ называется **опорным планом задачи** линейного программирования, если система векторов $A_1, A_2, \ldots A_n$. входящих в разложение (14) с положительными величинами $Xj > 0$ (i = 1,n), линейно независима. Опорный план называется **невырожденным,** если он содержит ровно ***m*** положительных компонент, иначе -вырожденным.

Так как $b_1A_1+b_2A_2+\ldots b_nA_n=B$, то вектор $X=(bj, b_2, \ldots b_n,,,0,\ldots 0)$ является **опорным планом** данной задачи (последние $n-m$ компонент вектора равны 0).

Этот план определяется системой единичных векторов $A_1, A_2, \ldots, A_m$, которые образуют **базис m-мерного пространства. Поэтому** каждый из векторов $Ai, A_2, \ldots, A_m$ а также вектор B могут быть представлены в виде линейной комбинации векторов данного базиса

Пусть $A_j = \sum_{i=1}^{m} x_{ij} A_i \left(j = \overline{1,n}\right)$

Положим $z_j = \sum_{i=1}^{m} c_i x_{ij} \left(j = \overline{1,n}\right)$, $\Delta_j = z_j - c_j \left(j = \overline{1,n}\right)$

Так как векторы $A_1, A_2, \ldots, A_m$ - единичные, то $x_{ij} = a_{ij}$ и $z_j = \sum_{i=1}^{m} c_i a_{ij}$, а

$$\Delta_j = \sum_{i=1}^{m} c_i a_{ij} - c_j.$$

Опорный план $X=(x_1, x_2, \ldots x_n; 0,0,\ldots 0)$ задачи (13)-(15)

Если $\Delta_к < 0$ для некоторого.i=k и среди чисел a_{ik}(i= l,m) нет положительных $(a,_к < 0)$, то целевая функция (13) задачи (13)- (15) не ограничена на множестве ее планов.

Если опорный план X задачи (13) - (15) не вырожден и $\Delta_к < 0$, но среди чисел a_{lk} есть положительные (не все $a_{lk} < 0$), то существует опорный- план X такой, что $F(X) > F(X)$.

Сформулированные положения позволяют проверить, является ли найденный опорный план оптимальным, и выявить целесообразность перехода к новому опорному плану.

Исследование опорного плана на оптимальность, а также дальнейший вычислительный процесс удобнее вести,, если условия задачи и первоначальные данные, полученные после определения исходного опорного плана, записать в виде таблицы.

i	Базис	$c_б$	B	C_1	C_2	…	c_r	…	c_m	C_{m+1}	…	c_k	c_n
				X_1	X_2	…	x_r	…	x_m	X_{m+1}	…	x_k	x_n
1	X_1	C_1	B_1	1	0		0		0	A_{1m+1}		A_{1k}	A_{1n}
2	X_2	C_2	B_2	0	1		0		0	A_{2m+2}		A_{2k}	A_{2n}
…	…			0	0								
r	x_r	C_r	b_r	0	0		1		0	A_{rm+1}		a_{rk}	a_{rn}
…	…			0	0		0						
m	x_m	c_m	b_m	0	0		0		0	A_{mm+1}		A_{mk}	A_{mn}
M+1			F_0	0	0		0		1	Δ_{m+1}		$\Delta<0$	Δ_n

.

В столбце $C_б$ этой таблицы записывают коэффициенты при неизвестных (целевой функции, имеющие те же индексы, что и векторы данного базиса. В .столбце *B* записывают положительные компоненты исходного опорного плана, в нем же в результате вычислений получают положительные компоненты оптимального плана. Столбцы векторов *Aj* представляют собой коэффициенты разложения этих векторов по векторам данного базиса. В табл.8 первые *m* строк определяются исходными данными задачи, а показатели (т+1)-й строки вычисляют. В этой строке в столбце вектора В записывают значение целевой функции, которое она принимает при данном опорном плане, а в столбце вектора А,- значение A_i =Zj-Cj.

Значение величины z находится как скалярное произведение вектора

$C_б = (c_1; c_2; \ldots; c_m)$ на вектор $A_j (j = \overline{1,m})$: $z_j = \sum_{i=1}^{m} c_i a_{ij} \quad (j = \overline{1,n})$

Значение F_o равно произведению вектора $C_б$ на вектор В:

$$F_0 = \sum_{i=1}^{m} c_i a_{ij} \qquad (j = 1, n)$$

После заполнения таблицы исходный опорный план проверяют на оптимальность. Для этого просматривают элементы (m+1)-й строки таблицы. В результате может иметь место один из следующих трех случаев:

1. $\Delta j > 0$ для j =m+l, m+2, ..., *n* (при j =*l.m, Zj* = c_j. Поэтому в данном случае числа $\Delta j>0$ для всех j от 1 до n;
2. $\Delta j<0$ для некоторого *j,* и все соответствующие этому индексу величины $a_{ij}<0$ (i=l,m)
3. $\Delta j<0$ для некоторых индексов j, и для каждого такого j по крайней мере одно из чисел a_{ij} положительно.

В первом случае исходный опорный план является оптимальным.

Во втором случае целевая функция не ограничена сверху на множестве планов.

В третьем случае можно перейти от исходного плана к новому опорному плану, при котором значение целевой функции увеличится.

Этот переход от одного опорного плана к другому осуществляется исключением из исходного базиса какого-нибудь из векторов и введением в него нового вектора. В качестве вектора, вводимого в базис, можно взять любой из векторов *A,* имеющий индекс j, для которого $\Delta j < 0$. Пусть, например, $\Delta_к < 0$ и решено ввести в базис вектор $A_к$.

Для определения вектора, подлежащего исключению из базиса, находят min(bj/a_k) для всех a_{ik} >Q. Пусть этот минимум достигается при i = r. Тогда из базиса исключают вектор A_r а число a_r называют разрешающим элементом. Столбец и строку, на пересечении которых находится разрешающий элемент, называют направляющими.

После выделения направляющей строки и направляющего столбца находят новый опорный план и коэффициенты разложения векторов A_i,

через векторы нового базиса, соответствующего новому опорному плану. Это легко реализовать, если воспользоваться методом Жордана- Гаусса.

При этом можно показать, что положительные компоненты нового опорного плана вычисляются по формулам,

$$b_i' = \begin{cases} b_i - (b_r / a_{rk}) a_{ik} \text{ при } i \neq r \\ b_r / a_{rk} \text{ при } i = r \end{cases}$$

а коэффициенты разложения векторов а, пересчитываются через векторы нового базиса, соответствующего новому опорному плану

$$a_{ij}' = \begin{cases} a_{ij} - (a_{rj} / a_{rk}) a_{ik} & \text{при } i \neq r \\ a_{rj} / a_{rk} & \text{при } i = r \end{cases}$$

После вычисления b_i и a_{1j} их значения заносят в таблицу. Элементы (т+1)-й строки этой таблицы могут .быть вычислены либо по формулам

$$F_0' = F_0 - (b_r / a_{rk}) \Delta_k$$

$$\Delta_j' = \Delta_j - (a_{rj} / a_{rk}) \Delta_k$$

либо на основании их определения.

Наличие двух способов нахождения элементов (n+1)-й строки позволяет осуществлять контроль правильности проводимых вычислений.

Из формулы (18) следует, что при переходе от одного опорного плана к другому наиболее целесообразно ввести в базис вектор *Aj* (переменную $x_{,j}$ имеющий индекс j, при котором максимальным по абсолютной величине является число (b_r, a_{rj}) $\Delta j<0$, $a_n > 0$). Однако с целью упрощения вычислительного процесса в дальнейшем будем вектор, вводимый в базис, определять исходя из максимальной абсолютной величины отрицательных чисел Δ_{ij}. Если же таких чисел несколько, то в базис будем вводить вектор, имеющий такой; индекс, как и максимальное из чисел C_{ij}, определяемых данными числами Δ_j.

Итак, переход от одного опорного плана к другому сводится к переходу от одной симплекс-таблицы к другой. Элементы новой симплекс-таблицы можно вычислить как с помощью рекуррентных формул (16) - (19), так и по правилам, непосредственно вытекающим из них. Эти правила состоят в следующем.

В столбцах векторов, входящих в базис, на пересечении строк и столбцов одноименных векторов проставляются единицы, а все остальные элементы данных столбцов полагают равными нулю. Элементы векторов A_i и A_j в строке новой симплекс-таблицы, в которой записан вектор, вводимый в базис, получают из элементов этой же строки исходной таблицы делением их на величину разрешающего элемента. В столбце C_6 в строке вводимого вектора проставляют величину c_r, где *к* - индекс вводимого вектора.

Остальные элементы столбцов вектора A_i и A_j новой симплекс-таблицы вычисляют по правилу треугольника. Для вычисления какого-нибудь из элементов находят три числа:

1)число, стоящее в исходной симплекс-таблице на месте искомого элемента новой симплекс-таблицы;

2)число, стоящее в исходной симплекс-таблице на пересечении строки, в которой находится искомый элемент новой симплекс-таблицы, и столбца, соответствующего вектору, вводимому в базис;

3)число, стоящее в новой симплекс-таблице на пересечении столбца, в котором стоит Искомый элемент, и строки вновь вводимого в базис вектора (как отмечено выше, эта строка получается из строки исходной симплекс-таблицы делением ее элементов на разрешающий элемент).

Эти три .числа, образуют своеобразный треугольник, две вершины которого соответствуют числам, находящимся в исходной симплекс-таблице, *а* третья - числу, находящемуся в новой симплекс-таблице. Для определения искомого элемента новой симплекс-таблицы из первого числа вычитают произведение второго и третьего. После заполнения новой симплекс-таблицы просматривают элементы (m+1)-й строки. Если все $z'j — Cj > 0$, то новый опорный план является оптимальным. Если же среди указанных чисел имеются отрицательные, то, используя описанную выше последовательность действий, находят новый опорный план. Этот процесс продолжают до тех пор, пока либо не получают оптимальный план задачи, либо устанавливают ее неразрешимость.

Примечание. Если задача имеет вырожденные опорные планы, то на одной из итераций одна или несколько переменных опорного плана могут оказаться равными нулю. Таким образом, при переходе от одного опорного плана к другому значение функции может остаться прежним. Более того, возможен случай, когда функция сохраняет свое значение в течение нескольких итераций, а также возможен возврат к первоначальному базису. В последнем случае обычно говорят, что произошло зацикливание. Однако при решении практических задач этот случай встречается очень редко и здесь могут быть применены специальные меры

Итак, нахождение оптимального плана симплексным методом включает следующие **этапы:**

1.Находят опорный план.

2.Составляют симплекс-таблицу.

3.Выясняют, имеется ли хотя бы одно отрицательное число А, . Если нет, то найденный опорный план оптимален. Если же среди чисел имеются отрицательные, то либо устанавливают неразрешимость задачи, либо переходят к новому опорному плану.

4. Находят направляющие столбец и строку. Направляющий столбец определяется наибольшим по абсолютной величине отрицательным числом Δ_j а направляющая строка - минимальным из отношений компонент столбца вектора. В к положительным компонентам направляющего столбца.

5.По формулам (16) - (19) определяют положительные компоненты нового опорного плана, коэффициенты разложения векторов *A*> по векторам нового базиса и числа Fo', Δj. Все эти числа записываются в новой симплекс-таблице.

6. Проверяют найденный опорный план на оптимальность. Если план не оптимален и необходимо перейти к новому опорному плану, то возвращаются к этапу 4, а в случае получения оптимального плана или установления неразрешимости процесс решения задачи заканчивают.

2.Практическая часть

Пример. Для изготовления различных изделий А, В и С предприятие использует три различных вида сырья. Нормы расхода сырья на производство одного изделия каждого вида, цена одного изделия А, В, С а также общее количество сырья каждого вида, которое может быть использовано предприятием, приведены в таблице.

Исходные данные задачи

Вид сырья	Нормы затрат сырья (кг) на одно изделие			Общее количество сырья (кг)
	A	B	C	
I	18	15	12	360
II	6	4	8	192
III	5	3	3	180
Цена одного изделия (руб)	9	10	16	

Изделия А, В и С- могут производиться в любых соотношениях (сбыт обеспечен), но производство ограничено выделенным предприятию сырьем каждого вида. Составить план производства изделий, при котором общая стоимость всей произведенной предприятием продукции является максимальной.

Решение. Составим математическую модель задачи. Искомый выпуск изделий А обозначим через x_h изделий В - через x_2, изделий С -через x_3. Поскольку имеются ограничения на выделенный предприятию фонд сырья каждого вида, переменные $x_h x_:, x_3$ должны удовлетворять следующей системе неравенств:

$$\begin{cases} 18x_1 + 15x_2 + 12x_3 \le 360, \\ 6x_1 + 4x_2 + 8x_3 \le 192, \\ 5x_1 + 3x_2 + 3x_3 \le 180. \end{cases}$$

Общая стоимость произведенной предприятием продукции составляет критерий

$$F=9x_i + 10x_2+16x_3 \quad -> \quad max. \qquad (21)$$

По своему экономическому содержанию переменные могут принимать только лишь неотрицательные значения:

$$X_ьx_2,x_3>0. \qquad (22)$$

Приходим к следующей математической задаче: среди всех неотрицательных решений системы неравенств (20) требуется найти такое, при котором функция (21) принимает максимальное значение.

Запишем эту задачу в форме основной задачи линейного программирования. Для этого перейдем от ограничений-неравенств к ограничениям-равенствам. Введем три дополнительные переменные, в результате чего ограничения запишутся в виде системы уравнении

$$\begin{cases} 18x_1 + 15x_2 + 12x_3 + x_4 = 360, \\ 6x_1 + 4x_2 + 8x_3 + x_5 = 192, \\ 5x_1 + 3x_2 + 3x_3 + x_6 = 180. \end{cases}$$

Эти дополнительные переменные по экономическому смыслу означают не используемое при данном плане производства количество сырья того или иного вида. Например, x_4 - это не используемое количество сырья I вида.

Преобразованную систему уравнений запишем в векторной форме:

$$x_1A_1+x_2A_2+x_3A_3+x_4A_4+x_5A_5+x_6A_6=B$$

где $A_1 = \begin{pmatrix} 18 \\ 6 \\ 5 \end{pmatrix}$; $A_2 = \begin{pmatrix} 15 \\ 4 \\ 3 \end{pmatrix}$; $A_3 = \begin{pmatrix} 12 \\ 8 \\ 3 \end{pmatrix}$; $A_4 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$; $A_5 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$; $A_6 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$; $B = \begin{pmatrix} 360 \\ 192 \\ 180 \end{pmatrix}$.

Для данной опорной задачи можно непосредственно записать опорный план X=(0; 0; 0; 360; 192; 180), определяемый системой трехмерных единичных векторов A_4, A_5, A_6, которые образуют базис трехмерного векторного пространства.

Составляем симплексную таблицу для I итерации (табл.11), подсчитываем значения F_o, Aj $=Zj-Cj$ и проверяем исходный опорный план на оптимальность-

$F_0 = (C, B) = 0;\quad z_1 = (C, A_1) = 0;\quad z_2 = (C, A_2) = 0;\quad z_3 = (C, A_3) = 0;$

$\Delta_1 = z_1 - c_1 = 0 - 9 = -9;\quad \Delta_2 = z_2 - c_2 = 0 - 10 = -10;\quad \Delta_3 = z_3 - c_3 = -16.$

Для векторов базиса $\Delta_j = z_j - c_j = 0$.

Первая итерация решения задачи симплекс-методом

i	Базис	C_6	B	9	10	16	0	0	0
				x_1	x_2	x_3	x_4	x_5	x_6
1	x_4	0	360	18	15	12	1	0	0
2	x_5	0	192	6	4	8	0	1	0
3	x_6	0	180	5	3	3	0	0	1
4			F_0=0	-9	-10	-16	0	0	0

Из табл.11 видно, что значения всех основных переменных X_1, x_2, x_3 равны нулю, а дополнительные переменные принимают свои значения в соответствии с ограничениями задачи. Эти значения переменных отвечают такому плану, при котором ничего не производится, сырье не используется и значение целевой функции равно нулю (т.е. стоимость произведенной продукции отсутствует).

Этот план не является оптимальным. Это видно и из 4-й строки табл..] 1 так как в ней имеется три отрицательных числа: Zj — c_j =-9, $z_2 - c_2$,= -10 и Z_3-c_3=-16

Отрицательные числа не только свидетельствуют о возможности увеличения общей стоимости производимой продукции, но и показывают, насколько увеличится эта сумма при введении в план единицы того или другого вида продукции.

Так, число -9 означает, что при включении в план производства одного 1 изделия А обеспечивается увеличение выпуска продукции на 9 руб. Если включить в план производства по одному изделию В и С, то общая стоимость изготовляемой продукции возрастет соответственно на 10 и 16 руб. Поэтому с экономической точки зрения наиболее целесообразным является включение в план производства изделий С.

Это *же* необходимо сделать и на основании формального признака симплексного метода, поскольку максимальное по абсолютной величине отрицательное. число Δ_j, стоит в 4-й строке столбца вектора A_3. Следовательно, в базис введем вектор A_3.

Определяем вектор, подлежащий исключению из базиса. Для этого находим $\theta_o=min(b/_{13})$ для $a_{13}>0$, т.е. θ_0=min(360/12; 192/8; 180/3) = 192/8,; Найдя число 192/8=24, мы тем самым с экономической точки зрения определили, какое количество изделий С предприятие может изготовлять с учетом норм расхода и имеющихся объемов сырья каждого вида.

Так как сырья каждого вида соответственно имеется 360, 192 и 180 кг, а на одно изделие С требуется затратить сырья каждого вида соответственно 12, 8 и 3 кг, то максимальное число изделий С, которое может быть изготовлено предприятием, равно min(360/12; 192/8; 180/3) = 192/8 = 24, т.е. ограничивающим фактором для производства изделий С является имеющийся объем сырья 2 вида. С учетом его наличия предприятие может изготовить 24 изделия С. При. этом сырье 2 вида будет полностью использовано.

Следовательно, вектор A_5 подлежит исключению из базиса. Столбец вектора A_3 и 2-я строка являются направляющими. Составляем таблицу для II; итерации (табл.12).

Вторая итерация решения задачи симплекс-методом

i	Базис	C_6	B	9	10	16	0	0	0
				x_1	x_2	x_3	x_4	x_5	x_6
1	x_4	0	72	9	**9**	0	1	-3/2	0
2	x_3	16	24	3/4	1/2	1	0	1/8	0
3	x_6	0	108	11/4	3/2	0	0	-3/8	1
4			384	3	-2	0	0	2	0

Сначала заполняем строку вектора, вновь введенного в базис, т.е. строку, номер которой совпадает с номером направляющей строки. Здесь направляющей является 2-я строка. Элементы этой строки табл.. 12 получаются из соответствующих элементов табл.. 11 делением их на разрешающий элемент (т.е на 8). При этом в столбце C_6 записываем коэффициент C_3 =16, стоящий в столбце вводимого в базис вектора A_3.

Затем заполняем элементы столбцов для векторов, входящих в новый базис. В этих столбцах на пересечении строк и столбцов одноименных векторов проставляем единицы, а все остальные элементы полагаем равными нулю.

Для определения остальных элементов табл.12 применяем правило треугольника. Эти элементы могут быть вычислены и непосредственно по рекуррентным формулам.

Вычислим элементы табл.12, стоящие в столбце вектора *B*. Для вычисления первого элемента столбца находим три числа:

1)число, стоящее в табл.11 на пересечении столбца вектора *B* и 1-й строки (360);

2)число, стоящее в табл.11, на пересечении столбца вектора A_3 и 1-й строки (12);

3)число, стоящее в табл.12 на пересечении столбца вектора *B* и 2-йстроки (24).

Вычитая из первого числа произведение двух других, находим искомый элемент: 360-12*24 = 72; записываем его в 1-й строке столбца, вектора *B* табл.12. (Этот же результат получается и по данным табл.11: 360-12-192/8).

Второй элемент столбца вектора *B* табл.6.12 был уже вычислен ранее. Для вычисления третьего элемента столбца вектора *B* также находим три числа. Первое из них (180) находится на пересечении 3-й строки и столбца вектора *B* табл.11, второе (3) - на пересечении 3-й строки и столбца вектора A_3 , табл.11, третье (24) - на пересечении 2-й строки и столбца вектора *B* табл.12. Итак, указанный элемент есть^80-24*3 = 108. Число 108 записываем в 3-й строке столбца вектора *B* табл.12.

Значение F_o *в* 4-й строке столбца этого же вектора можно найти двумя способами:

1)по формуле F_o = (C,B),т.е. F_o =0-72 + 16-24+0*108-384;

2) по правилу треугольника; в данном случае треугольник образован числами 0, -16,24. Этот способ приводит к тому же результату; 0- (-16) • 24 = 384.

При определении по правилу треугольника элементов столбца вектора *B* третье число, стоящее в нижней вершине треугольника, все время оставалось неизменным и менялись лишь первые два числа. Учтем это при нахождении элементов столбца вектора A_1 табл.12. Для вычисления указанных

элементов первые два числа берем из столбцов векторов A_1 и A_3 табл.11, а третье число из табл.12.

... Это число стоит на пересечении 2-й строки и столбца вектора A_1 последней таблицы. В результате получаем значения искомых элементов:

18-12-(3/4) = 9; 5-3-(3/4) =11/4

Число z_i-c_i в 4-й строке столбца вектора A_1 табл.12 можно найти двумя способами:

1) по формуле Zi — Ci = *(C,A)-Cj* имеем 0*9-(-16)*(3/4)+0*(11/4)-9=3.

2) по правилу треугольника получим: - 9-(-16)•(3/4)=3.

Аналогично находим элементы столбца вектора A_2

Элементы столбца вектора A_5 вычисляем по правилу треугольника. Элемент 1-й строки указанного столбца равен 0-12•(1/8)=-3/2. Элемент, стоящий в 3-й строке данного столбца, равен $0-3\bullet(1/8)$ =-3/8.

По окончании расчета всех элементов табл.12 в ней получены опорный план и коэффициенты разложения векторов A_j (j = 1,6) через базисные векторы A_4,A_3,A_6 и значения Δ_j и F_o.

Новым опорным планом задачи является план F = (0; 0; 24; *11;* 0; 108) При данном плане производства изготовляется 24 изделия С и остается ней пользованным 72 кг сырья I вида и 108 кг сырья Ш вида. Стоимость всей изводимой при этом плане продукции равна 384 руб.

Изменились данные столбцов, а их экономическое содержание стало более сложным. Так, например, число 1/2 во 2-й строке столбца вектора *A?* показывает, на сколько следует уменьшить изготовление изделий С, если запланировать выпуск одного изделия В.

Числа 9 и 3/2 в 1-й и 3-й строках вектора A_2 показывают соответственно, сколько потребуется сырья I и II вида при включении в план производства одного изделия В, а число -2 в 4-й строке показывает, что если будет

запланирован выпуск одного изделия В, то это обеспечит увеличение выпуска продукции в стоимостном выражении на 2 руб.

Иными словами, если включить в план производства продукции одно изделие В, то это потребует уменьшения выпуска изделия С на 1/2 ед. и потребует дополнительных затрат 9 кг сырья I вида и 3/2 кг сырья Ш вида, а общая стоимость изготовляемой продукции в соответствии с новым оптимальным планом возрастет на 2 руб.

Таким образом, числа 9 и 3/2 выступают как бы новыми «нормами» затрат сырья 1 и Ш вида на изготовление одного изделия В (как видно из табл.11, ранее они были равны 15 и 3), что объяеняется уменьшением выпуска изделий С. Такой же экономический смысл имеют данные столбца вектора *А* / табл. 12.

Число 1/8 во 2-й строке столбца *А5,* показывает, что увеличение объемов сырья П вида на 1 кг позволили бы увеличить выпуск изделий С на 1/8 ед. Одновременно потребовалось бы дополнительно 3/2 кг сырья I вида и 3/8 кг сырья Ш вида. Увеличение выпуска изделий С на 1/8 ед. приведет к росту выпуска продукции на 2 руб. Из изложенного выше экономического содержания данных табл.12 следует, что найденный на II итерации план задачи не является оптимальным. Это видно и из 4-й строки табл.12, поскольку в столбце вектора A_2 этой строки стоит отрицательное число -2. Значит, в базис следует ввести вектор A_2, т.е. в новом плане следует предусмотреть выпуск изделий В

При определении возможного числа изготовления изделий В следует учитывать имеющееся количество сырья каждого вида, а именно: возможный выпуск изделий В определяется min (b *{) /a',)* для *a* ,2 > 0, т. е. находим

$$\theta_0 = \min\left(\frac{72}{9}; \frac{24\cdot 2}{1}; \frac{108\cdot 2}{3}\right) = \frac{72}{9} = 8.$$

Следовательно, исключению из базиса подлежит вектор Ад, иными словами, выпуск изделий В ограничен имеющимся в распоряжении предприятия сырьем I вида. С учетом имеющихся объемов этого сырья

предприятию следует изготовить 8 изделий В. Число 9 является разрешающим элементом, а столбец вектора A_2 и 1-я строка табл.12 являются направляющими. Составляем таблицу для III итерации (табл.. 13).

В табл.6.13 сначала заполняем элементы 1-й строки, которая представляет собой строку вновь вводимого в базис вектора A_2. Элементы этой строки получаем из элементов 1-й строки табл.12 делением последних на разрешающий элемент (т.е. на 9). При этом в столбце C_6 данной строки записываем Сг —10. Затем заполняем элементы столбцов векторов оазиса и по правилу; треугольника вычисляем элементы остальных столбцов.

Таблица 13

i	Базис	$C_б$	B	9	10	16	0	0	0
				x_1	x_2	x_3	x_4	x_5	x_6
1	x_2	10	8	1	1	0	1/9	-1/6	0
2	x_3	16	20	1/4	0	1	-1/18	5/24	0
3	x_6	0	96	5/4	0	0	-1/6	-1/8	1
4			400	5	0	0	2/9	5/3	0

В результате получаем новый опорный план X=(0;8; *20;0;0;96)* и коэффициенты разложения векторов А, (j=l,6) через базисные векторы A_2,$Л_3$,A_6 и соответствующие значения Aj и F_o".

В 4-й строке среди чисел А, нет отрицательных. Это означает, что найденный опорный план является оптимальным и F_{max} = 400

Следовательно, план выпуска продукции, включающий изготовление 8 изделий В и 20 изделий С, является оптимальным.

При данном плане выпуска изделий полностью используется сырье 1 и 2 видов и остается неиспользованным 96 кг сырья Ш вида, а стоимость произведенной продукции равна 400 руб.

Оптимальным планом производства продукции не предусматривается изготовление изделий А. Введение в план выпуска продукции изделий вида А привело бы к уменьшению указанной обшей стоимости.

Это видно из 4-й строки столбца вектора ,4/, где число 5 показывает, что при данном плане включение в него выпуска .единицы изделия А приводит лишь к уменьшению общей величины стоимости на 5 руб.

Решение данного примера симплексным методом можно было бы проводить, используя лишь одну таблицу (табл.14). В этой таблице последовательно записаны одна за другой все три итерации вычислительного процесса

16

i-	Базис	Ca	B	. 9	10	Ib	0	0	0
				X.	\2	*3	*4	*5	
Первая итерация									
I	\4	0	360	18	15	12	1	0	0 ¥
2	X_5	0	192	6	4	8	0	1	
3	X_6	0	180	5	3	3	0	0	
4			F_0=0	-9	-10	-16	0	0	0
Вторая итерация									
1	*4	0	72	9	9	0	1	-3/2	
2		16	24	3/4	1/2	1	0	1/8	
3	X_6	0	108	11/4	3/2	0	0	-3/8	
4			384	3	-2	0	0	2	0
Третья итерация									
1	*2	10	8	1	1	0	1/9	-1/6	0
2	x_3	16	20	1/4	0	1	-1/18	5/24	0
3	X_6	0	96	5/4	0	0	-1/6	-1/8	1
4			400	5	0	0	2/9	5/3	0

Составить программный модуль для решения задач симплекс-методом.

3 Задания для самостоятельной работы

1. **Задание 1** Решить задачу линейного программирования симплекс-методом.

$$f = 2X_1 + X_2 - 2X_3 \rightarrow \min$$

$$\begin{cases} X_1 + X_2 - X_3 \geq 8, \\ X_1 - X_2 + 2X_3 \geq 2, \\ -2X_1 - 8X_2 + 3X_3 \geq 1, \\ X_i \geq 0 (i = 1,2,3) \end{cases}$$

2. **Задание 2**

Решить задачу линейного программирования симплекс-методом

$Y = 12X_1 + X_2 - 2X_3 \rightarrow max$

$-2X_1 - X_2 \geq -10$

$2X_1 - X_2 - 2X_3 \leq 8$

$-X_1 + X_3 \leq -2$

3. **Задание 3**

Решить задачу линейного программирования симплекс-методом

$$z = 4x_1 + 6x_2 \rightarrow \max$$

$$\begin{cases} 2x_1 + x_2 \leq 64 \\ x_1 + 3x_2 \leq 72 \\ x_2 \leq 20 \end{cases}$$

$$x_i \geq 0, i = 1,2.$$

4. **Задание 4**

Решить задачу линейного программирования симплекс-методом

$$x_2 - 3x_3 + 2x_5 \Rightarrow \min$$

$$\begin{array}{llllll} x_1 & +3x_2 & -x_3 & & +2x_5 & = 7 \\ & -2x_2 & +4x_3 & +x_4 & & = 12 \\ & -4x_2 & +3x_3 & & +8x_5 & +x_6 = 10 \end{array}$$

$$x_i \geq 0, \quad i = \overline{1,6}$$

Занятие 2. Решение транспортной задачи

1. Теоретическая часть

Общая постановка транспортной задачи состоит в определении оптимального плана перевозок некоторого однородного груза из *m* пунктов отправления $A_1, A_2, A_3 \ldots A_n$ пунктов в n пунктов назначения $B_1, B_2 B_n$. При этом в качестве критерия оптимальности обычно берется либо минимальная стоимость перевозок всего груза, либо минимальное время его доставки .

Рассмотрим транспортную задачу, в качестве критерия оптимально которой взята минимальная стоимость перевозок всего груза. Обозначим через c_{ij} тарифы перевозки единицы груза из i-го пункта отправления в j-й пункт назначения, через a_i - запасы груза в i-м пункте отправления, через b_j -

потребности в грузе в j-м пункте назначения, а через x_{ij} - количество единиц груза,: ввозимого из i-го пункта отправления j-й пункт назначения.

Тогда математическая постановка задачи состоит в определении минимального значения функции

$$F = \sum_{i=1}^{m} \sum_{j=1} c_{ij} x_{ij}$$

при условиях

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = \overline{1,n}),$$

$$\sum_{j=1}^{n} x_{ij} = a_i \quad (i = \overline{1,m}),$$

$$x_{ij} \geq 0 \quad (i = \overline{1,m};\ j = \overline{1,n}). \qquad (1)\text{-}(4)$$

Поскольку переменные x_{ij} *(i* - l,m; *j* = 1,п) удовлетворяют системам линейных уравнений (2) и (3) и условию неотрицательности (4), то обеспечиваются доставка необходимого количества груза в каждый из пунктов назначения, вывоз имеющегося груза из всех пунктов отправления, а также исключаются обратные перевозки.

Всякое неотрицательное решение систем линейных уравнений (2) и (3), определяемое матрицей $X=(x_{ij});(i=1,m;\ j=1n)$, называется планом транспортной задачи.

План $X^* = (x^*_{ij})$ (i = 1, *m; j=1,n*), при котором функция (1.5) принимает свое минимальное значение, называется оптимальным планом транспортной .задачи.

Обычно исходные данные задачи записывают в виде табл.1

Исходные данные транспортной задачи

Пункты отправления	Пункты назначения					Запасы
	B_1	...	B_j	...	B_n	
A_1	c_{11} x_{11}	...	c_{1j} x_{1j}	...	c_{1n} x_{1n}	a_1
...	...	...	...	...	...	...
A_i	c_{i1} x_{i1}	...	c_{ij} x_{ij}	...	c_{in} x_{in}	a_i
...	...	...	...	...	...	...
A_m	c_{m1} x_{m1}	...	c_{mj} x_{mj}	...	c_{mn} x_{mn}	a_m
Потребности	b_1	...	b_j	...	b_n	

Общее наличие груза у поставщиков равно $\sum_{ш=1}^{b} a_i$ а общая потребность в грузе в пунктах назначения равна $\sum_{j=1}^{b} b_j$ единиц. Если общая потребность в грузе в пунктах назначения равна запасу груза в пунктах, отправления, т.е.

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, \qquad (5)$$

то модель такой транспортной задачи называется закрытой. Если же указанное условие не выполняется, то модель транспортной задачи называется открытой. Для разрешимости транспортной задачи необходимо и достаточно, чтобы запасы груза в пунктах отправления были равны потребностям в грузе в пунктах назначения, т.е. чтобы выполнялось равенство (5).

В случае превышения запаса над потребностью ($\sum_{ш=1}^{b} a_i > \sum_{j=1}^{b} b_j$) вводится фиктивный (n+1) пункт назначения с потребностью $b_{n+t} = \sum_{ш=1}^{b} a_i - \sum_{j=1}^{b} b_j$ и соответствующие тарифы считаются равными нулю: $c_{ij} = 0$ (i= 1,m). Полученная задача является транспортной задачей, для которой выполняется равенство (5).

Аналогично, при ($\sum_{ш=1}^{b} a_i < \sum_{j=1}^{b} b_j$) вводится фиктивный (m+1)-й пункт отправления с запасом груза $a_{m+1} = \sum_{j=1}^{b} b_j - \sum_{ш=1}^{b} a_i$, и тарифы полагаются равными нулю: $c_{m+1j} = 0$ (j = 1,n).

Этим задача сводится к обычной транспортной задаче, из оптимального плана которой получается оптимальный план исходной задачи.

В дальнейшем будем рассматривать закрытую модель транспортной задачи. Если же модель конкретной задачи является открытой, то, исходя из сказанного выше, перепишем таблицу условий задачи так, чтобы выполнялось равенство (5).

Число переменных x_v в транспортной задаче *с m* пунктами отправления и *n* пунктами назначения равно *nm,* а число уравнений в системах (2) и (3)' равно n + m. Так как мы предполагаем, что выполняется условие (5), то число линейно независимых уравнений равно *n* + *m* - 1. Следовательно, опорный план транспортной задачи может иметь не более n+ *m* - 1 отличных от нуля неизвестных.

Если в опорном плане число отличных от нуля компонент равно в точности n + *m* -1, то план является невырожденным, а если меньше - то вырожденным.

Для определения опорного плана существует несколько методов: метод северо-западного угла, метод минимального элемента, метод аппроксимации Фогеля и др. Как и для всякой задачи линейного программирования, оптимальный план транспортной задачи является и опорным планом., Для определения оптимального плана транспортной задачи можно использовать стандартные методы линейного программирования, например, симплекс-метод. Однако ввиду исключительной практической важности этой задачи и специфики ее ограничений (каждая неизвестная входит лишь в два уравнения систем (2) и (3) и коэффициенты при неизвестных равны единице) для определения оптимального плана транспортной задачи разработаны специальные методы: метод потенциалов, метод дифференциальных рент и др.

Определение опорного плана транспортной задачи

Поиск оптимального плана транспортной задачи начинают с нахождения какого-нибудь ее опорного плана. Опорный план находят последовательно за n+m-1 шагов, на каждом из которых в таблице условий задачи заполняют одну клетку, которую называют занятой.

Заполнение одной из клеток обеспечивает полностью либо удовлетворение потребности в грузе одного из пунктов назначения (того, в столбце которого находится заполненная клетка), либо вывоз груза из одного из пунктов отправления (из того, в строке которого находится заполняемая клетка).

В первом случае временно исключают из рассмотрения столбец, содержащий заполненную на данном шаге клетку, и рассматривают задачу, таблица условий которой содержит на один столбец меньше, чем было перед этим шагом, но то же количество строк и соответственно измененные запасы груза в одном из пунктов отправления (в том, за счет запаса которого была удовлетворена потребность в грузе пункта назначения на данном шаге).

Во втором случае временно исключают из рассмотрения строку, содержащую заполненную клетку, и считают, что таблица условий имеет на одну строку меньше при неизменном количестве столбцов и при соответствующем изменении потребности в грузе в пункте назначения, в столбце которого находится заполняемая клетка.

После того, как проделаны *n+m-2* описанных выше шагов, получают задачу с одним пунктом отправления и одним пунктом назначения. При этом останется свободной только одна клетка, а запасы оставшегося пункта отправления будут равны потребностям оставшегося пункта назначения. Заполнив эту клетку, тем самым делают (n+m-1.)-й шаг и получают искомый опорный план.

Следует заметить, что на некотором шаге (уо не на последнем) может оказаться, что потребности очередного пункта назначения равны запасам очередного пункта отправления.

В этом случае также временно исключают из рассмотрения либо столбец, либо строку (что-нибудь одно). В итоге, либо запасы соответствующего пункта отправления, либо потребности данного пункта назначения считают равными нулю. Этот нуль записывают в очередную заполняемую клетку.

Указанные выше условия гарантируют получение n+m-l занятых клеток, в которых стоят компоненты опорного плана, что является исходным условием для проверки последнего на оптимальность и нахождения оптимального плана.

При нахождении опорного плана транспортной задачи методом северо-западного угла на каждом шаге рассматривают первый из оставшихся пунктов отправления и первый из оставшихся пунктов назначения. Заполнение клеток таблицы условий начинается с левой верхней клетки для неизвестного x_{11}

(«северо-западный угол») и заканчивается клеткой для неизвестного x_{mn}, т.е. процесс заполнения клеток идет как бы по диагонали таблицы.

2. Практическая часть

Пример. Для транспортной задачи, исходные данные которой приведены в табл..4, найти оптимальный план.

Решение. Сначала, используя метод северо-западного угла, находим опорный план задачи. Этот план записан в табл.4.

Найденный опорный план проверяем на оптимальность. В связи с этим находим потенциалы пунктов отправления и назначения.

Найденные числа записываем в каждую из свободных клеток табл. .5.

Таблица.5

Исходные данные задачи

Пункты отправления	Пункты назначения				Запасы
	B_1	B_2	B_3	B_4	
A_1	1 30	2 20	4	1	50
A_2	2	3 10	1 10	5 10	30
A_3	3	2	4	4 10	10
Потребности	30	30	10	20	90

Для определения потенциалов получаем систему, содержащую шесть уравнений с семью неизвестными

$$\beta_1 - \alpha_1 = 1, \quad \beta_2 - \alpha_1 = 2, \quad \beta_2 - \alpha_2 = 3,$$
$$\beta_3 - \alpha_2 = 1, \quad \beta_4 - \alpha_2 = 5, \quad \beta_4 - \alpha_3 = 4.$$

Полагая $\alpha_1 = 0$, находим $\beta_1 = 1$, $\beta_2 = 2$, $\alpha_2 = -1$, $\beta_3 = 0$, $\beta_4 = 4$, $\alpha_3 = 0$.
Для каждой свободной клетки вычисляем число $\Delta_{ij} = \beta_j - \alpha_i - c_{ij}$:

$$\Delta_{13} = -4, \quad \Delta_{14} = 3, \quad \Delta_{21} = \Delta_{32} = 0, \quad \Delta_{31} = -2, \quad \Delta_{33} = -4.$$

Так как среди чисел Δ_{ij} имеются положительные, то построенный план перевозок не является оптимальным и надо перейти к новому опорному плану. Наибольшим среди положительных чисел Δ_{ij} являются $\Delta_{14} = 3$, поэтому для данной свободной клетки строим цикл пересчета (табл..5) и производим сдвиг по этому циклу. Наименьшее из чисел в минусовых клетках равно 10. Клетка, в которой находится это число, становится свободной в новой табл. .6.

Таблица .6

Промежуточные результаты решения транспортной задачи

Пункты отправления	Пункты назначения				Запасы
	B_1	B_2	B_3	B_4	
A_1	1 30	2 (-) 20	4 {-4}	1 (+) {+3}	50
A_2	2 {0}	3 (+) 10	1 10	5 (-) 10	30
A_3	3 {-2}	2 {0}	4 {-4}	4 10	10
Потребности	30	30	10	20	90

Другие числа в табл..5 получаются следующим способом. К числу 10 в плюсовой клетке A_2B_2, добавим 10 и вычтем 10 из числа 20, находящегося в минусовой клетке AiB_2. Клетка на пересечении строки A_2 и столбца B_4 становится свободной.

После этих преобразований получаем новый опорный план (табл. :7).

Таблица .7

Пункты отправления	Пункты назначения				Запасы
	B_1	B_2	B_3	B_4	
A_1	1 30	2 (-) 10	4 {-2}	1 (+) 10	50
A_2	2 {0}	3 20	1 10	5 {-3}	30
A_3	3 {+1}	2 (+) {+3}	4 {-1}	4 (-) 10	10
Потребности	30	30	10	20	90

Этот план проверяем на оптимальности. Снова находим потенциалы пунктов отправления и назначения. Для этого составляем следующую систему уравнений:

Полагаем $\alpha_1 = 0$, получаем $\beta_1 = \beta_4 = 1$, $\beta_2 = 2$, $\beta_3 = 0$, $\alpha_3 = -3$, $\alpha_2 = -1$. Для каждой свободной клетки вычисляем число Δ_{ij}.

Имеем: $\Delta_{13} = -2$, $\Delta_{21} = 0$, $\Delta_{24} = -3$, $\Delta_{31} = 1$, $\Delta_{32} = 3$, $\Delta_{33} = 2$.

Видим, что данный план перевозок не является оптимальным. Поэтому переходим к новому опорному плану (табл. .8)

Улучшенный опорный план решения транспортной задачи

Пункты отправления	Пункты назначения								Запасы
	B_1		B_2		B_3		B_4		
A_1	1	30	2	{0}	4	{-4}	1	20	50
A_2	2	{0}	3	20	1	10	5	{-3}	30
A_3	3	{-2}	2	10	4	{-4}	4	{-3}	10
Потребности		30		30		10		20	90

Сравнивая разности $\Delta_{ij}=\beta j - \alpha_i$ новых потенциалов, отвечающих свободным клеткам табл.7, с соответствующими числами c_{ij} видим, что указанные разности потенциалов для всех свободных клеток не превосходят чисел c_{ij}.

Следовательно, получен оптимальный план:

$$X^* = \begin{pmatrix} 30 & 0 & 0 & 20 \\ 0 & 20 & 10 & 0 \\ 0 & 10 & 0 & 0 \end{pmatrix}.$$

При данном плане стоимость перевозок составляет величину

S= 1*30 + 2*0+ 1*20+ 3*20+1*10 + 2*10=140.

Задания для самостоятельной работы

1

Пункты отправления	Пункты назначения				Запасы
	B_1	B_2	B_3	B_4	
A_1	4	8	4	5	420
A_2	6	2	6	8	230
A_3	3	11	5	12	560
Потребности	230	320	160	170	

2

Пункты отправления	Пункты назначения				Запасы
	B_1	B_2	B_3	B_4	
A_1	8	57	35	27	200
A_2	12	62	34	32	280
A_3	11	68	33	11	320
Потребности	300	220	130	200	

Занятие 3. Лабораторная работа
Решение оптимизационной задачи с помощью инструментальных средств

I.Теоретическая часть

Исходные данные транспортной задачи приведены схематически: внутри прямоугольника заданы удельные транспортные затраты на перевозку единицы груза *(c_{ij})*, слева указаны мощности поставщиков (a,), а сверху - мощности потребителей *(b_j)*. Найти оптимальный план закрепления поставщиков за потребителями *(x_{ij})*.

Мощности поставщиков	Мощности потребителей			
	250	100	150	50
80	6	6	1	4
320	8	30	6	5
100	5	4	3	30
50	9	9	9	9

В данной задаче суммарные запасы равны суммарным потребно-

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = 550.$$

Транспортная задача, в которой суммарные запасы и потребности совпадают, является закрытой.

Ввод условий задачи состоит из следующих основных шагов:

1. Создание формы для ввода условий задачи.
2. Ввод исходных данных.
3. Ввод зависимостей из математической модели.
4. Назначение целевой функции.
5. Ввод ограничений и граничных условий.

Матрица перевозок(изменямые ячейки)

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

исходные данные

	250	100	150	50
80	6	6	1	4
320	8	30	6	5
100	5	4	3	30
50	9	9	9	9

Рис 1 . Создание формы для ввода условий задачи.

Изменяемые ячейки **B3:B6.** В эти ячейки будет записан оптимальный план перевозки x_{ij}. Введены исходные данные задачи.

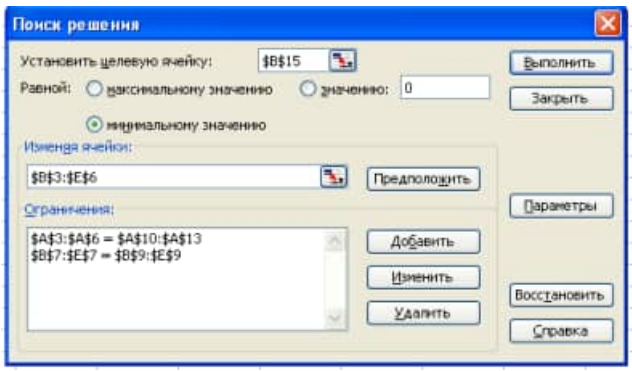


Рис.2 Введены зависимости из математической модели. Выражение для вычисления целевой функции получено с помощью функции СУММПРОИЗВ(B3:B6БВ10:E13)

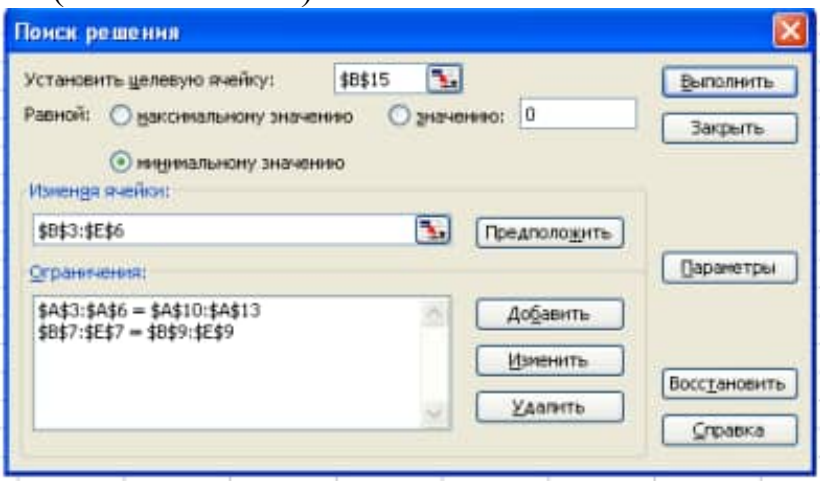


Рис.3 На экране диалоговое окно *Поиск решения*

После вызова *Поиска решения* курсор подвести в поле «Установить целевую ячейку» и ввести адрес: B15. Ввести направление целевой функции *«минимальному значению»*. Поместить курсор в поле «Изменяя ячейки». Ввести адреса изменяемых ячеек B3:E6. Далее следует *добавить ограничения.*

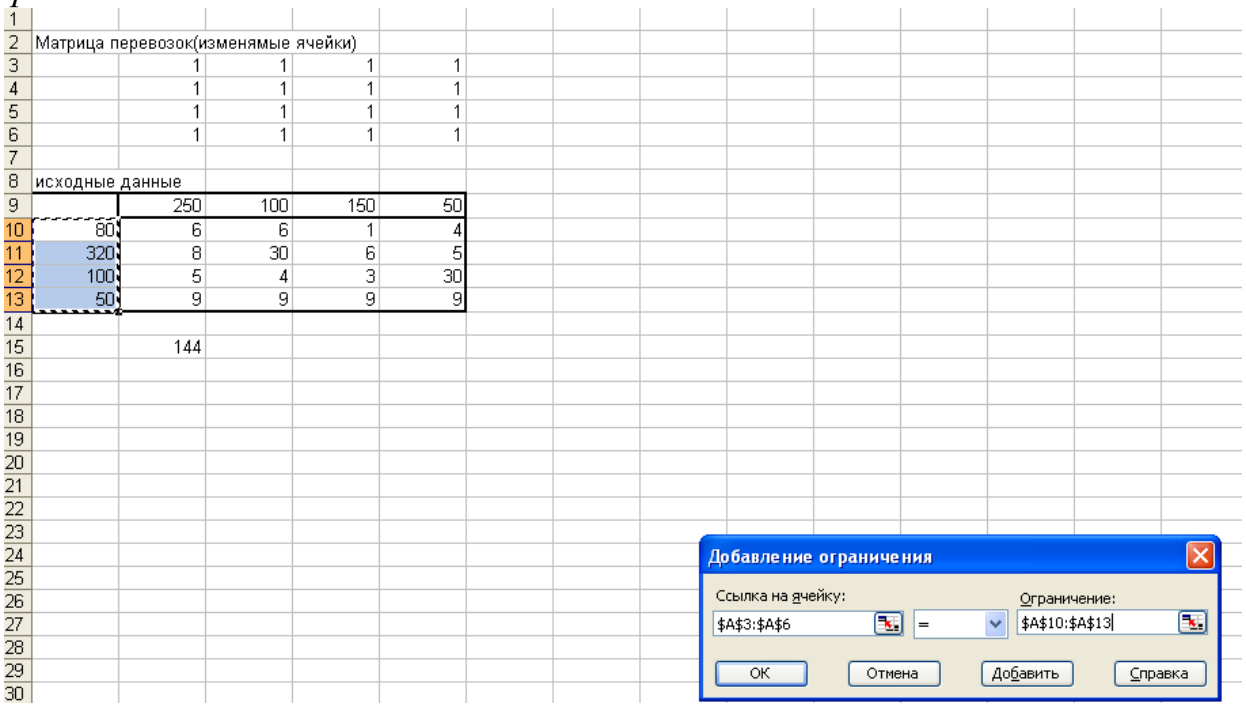


Рис 4. На экране диалоговое окно добавление ограничения

Все грузы должны быть перевезены, т.е.

$$\sum_{j=1}^{n} x_{ij} = a_i, \quad i = 1,\ldots,m - \text{A3:A6} = \text{A10:A13}.$$

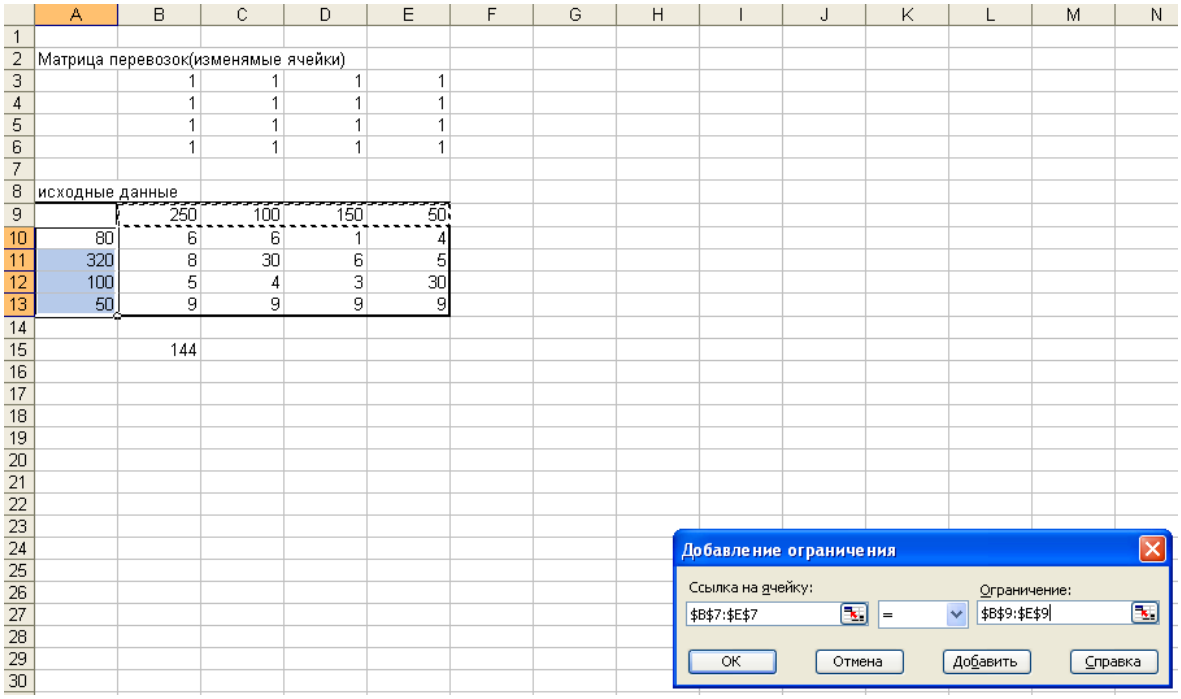


Рис. 5. На экране диалоговое окно *Добавление ограничения.*

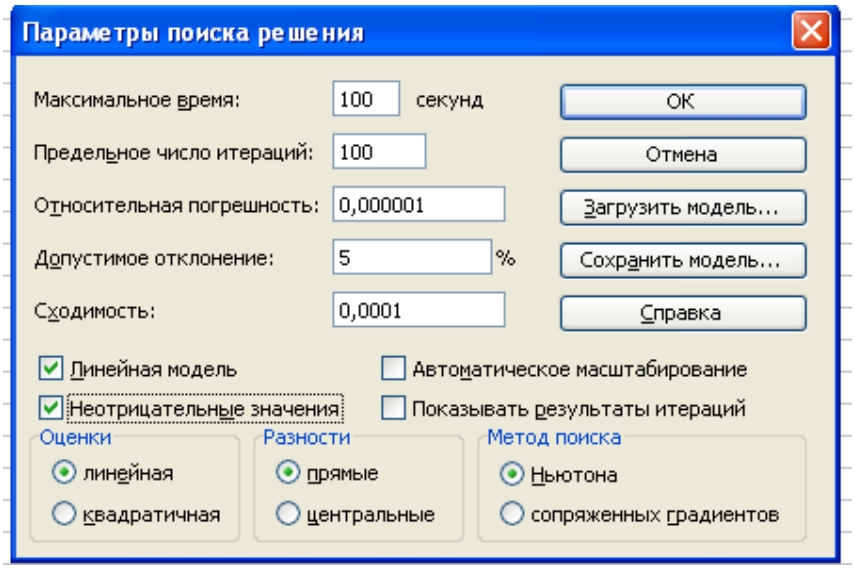
Все потребности должны быть удовлетворены, т.е.

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1,\ldots,n - \text{B7:E7} = \text{B9:E9}.$$

После ввода последнего ограничения вместо *добавить* ввести ОК. На экране появится окно *Поиск решения* с введенными ограничениями (см. рис. 2.).

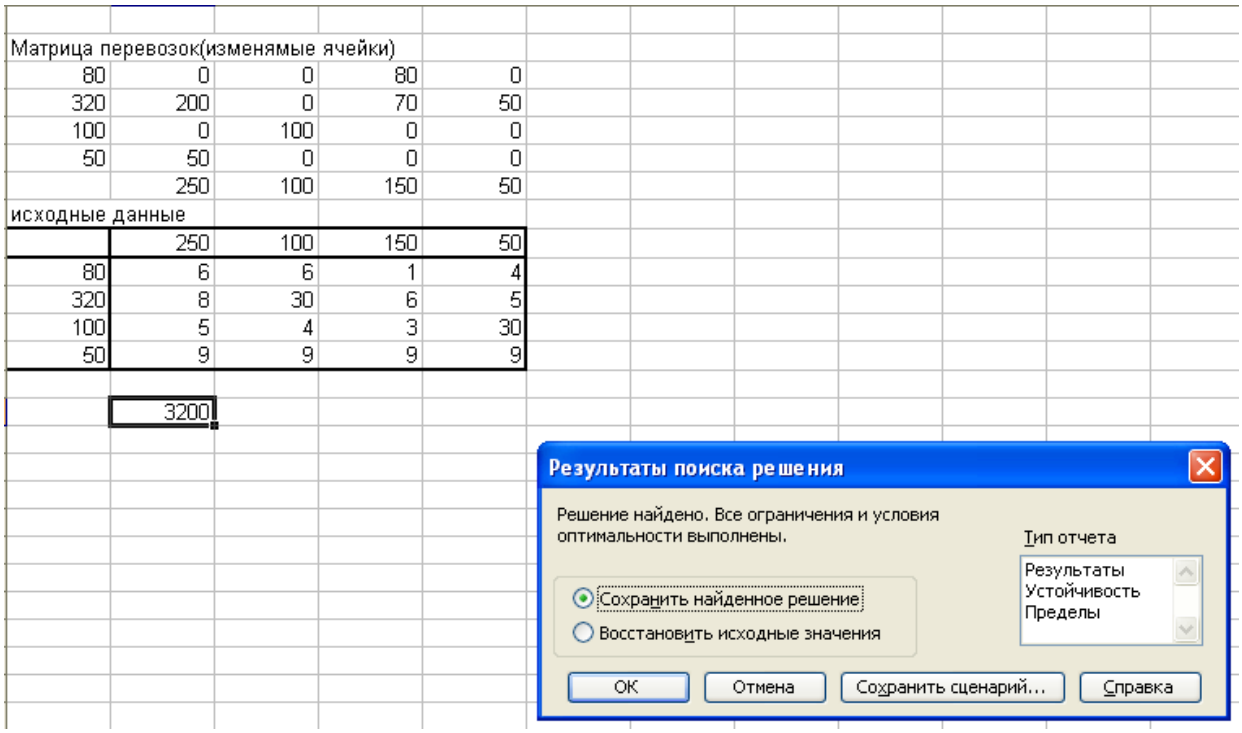
Решение задачи

Решение задачи производится сразу же после ввода данных, когда на экране находится диалоговое окно *Поиск решения* (рис. 2.4.3). С помощью окна *Параметры* можно вводить условия для решения оптимизационных задач. В нашей задаче следует установить флажок *«неотрицательные значения»* и флажок *«линейная модель»* (рис. 2.4.6). Нажать кнопку ОК. Опять появится диалоговое окно *Поиск решения.*



В результате решения получен оптимальный план перевозок:

Матрица		перевозок		(изменяемые
80	**0**	**0**	**so**	0
320	**200**	**0**	70	50
100	**0**	**100**	0	0
50	**50**	**2.13E-X4**	0	0
550	250	100	150	50



Хв = 80 ед. груза следует перевезти от 1-го поставщика 3-му потребителю;

Х21 = 200 ед. груза следует перевезти от 2-го поставщика 1-му потребителю;

X_2з = 70 ед. груза следует перевезти от 2-го поставщика 3-му потребителю;

Х24 = 50 ед. груза следует перевезти от 2-го поставщика 4-му потребителю;

Хзг = 100 ед. груза следует перевезти от 3-го поставщика 2-му потребителю;

$X_4$1 = 50 ед. груза следует перевезти от 4-го поставщика 1 -му потребителю;

Х42 = 0 ед. груза следует перевезти от 4-го поставщика 2-му потребителю.

Общая стоимость перевозок = 3200.

III. Контрольные вопросы

1. Какая модель транспортной задачи является открытой?
2. Какая модель транспортной задачи является закрытой?
3. . Какие ячейки называются зависимыми?
4. Как работает надстройка Поиск решения.
5. Как работает функция СУММПРОИЗВ?

IV. Оформление отчёта

Отчет по лабораторно-практической работе составляется по следующей структуре:

1. Наименование лабораторной работы.
2. Цель работы.
3. Описание назначения команд и их параметров.
4. Результаты работы - изображение структуры каталогов.
5. Ответы на контрольные вопросы.
6. Вывод по работе.

Занятие 4. Практическое занятие
Тема: Решение задач методом динамического программирования

1 Теоретическая часть

Уравнения Беллмана. Вместо исходной задачи ДП с фиксированным числом шагов и и начальным состоянием s_0 рассмотрим последовательность задач, полагая последовательно *n=1,2, ...* при различных *s*— одношаговую, двухшаговую и т.д., — используя принцип оптимальности.

Введем ряд новых обозначений. Обозначения в ДП несут большую информационную нагрузку, поэтому очень важно их четко усвоить.

На каждом шаге любого состояния системы S_{k-1} решение $X_к$ нужно выбирать "с оглядкой", так как этот выбор влияет на последующее состояние s_k и

дальнейший процесс управления, зависящий от s_k. Это следует из принципа оптимальности.

Но есть один шаг, последний, который можно для любого состояния s_k-$_1$ планировать локально-оптимально, исходя только из соображений этого шага. Рассмотрим n-й шаг: s_{n-1} — состояние системы к началу n-го шага, $s_n = s$ — конечное состояние, X_n — управление на n-м шаге, *а* $f_n(s_n\text{-}_1, X_n)$ — целевая функция (выигрыш) n-го шага.

Согласно принципу оптимальности, X_n нужно выбирать так, чтобы для любых состояний s_{n-1} получить максимум целевой функции на этом шаге. Обозначим через $Z^*_n(s_{n-1})$ максимум целевой функции — показателя эффективности n-го шага при условии, что к началу последнего шага система *S* была в произвольном состоянии s_{n-1}, а на последнем шаге управление было оптимальным.

$Z^*_n(s_{n-1})$ называется условным максимумом целевой функции на n-м шаге. Очевидно, что

$$Z_n^*(s_{n-1}) = \max_{\{X_n\}} f_n(s_{n-1}, X_n). \qquad (5)$$

Максимизация ведется по всем допустимым управлениям X_n.

Решение X_n при котором достигается $Z^*_n(s_{n-1})$ также зависит от s_n-$_1$ и называется *условным оптимальным управлением* на *n-м шаге.* Оно обозначается через X^*_n- Решив одномерную задачу локальной оптимизации по уравнению

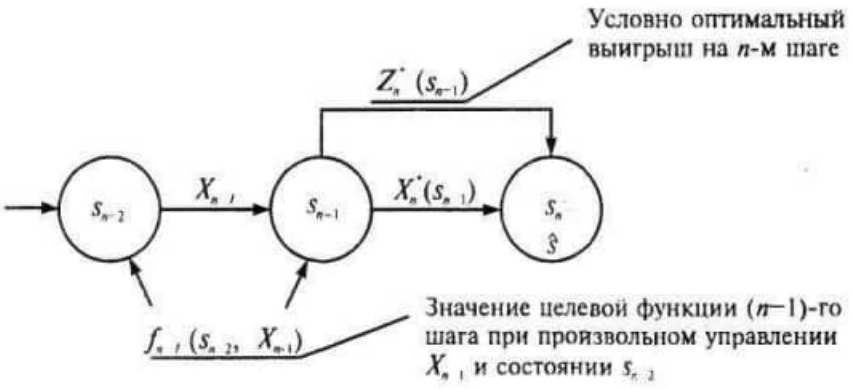


Рис. .2

(5), найдем для всех возможных состояний s_{n-1} две функции: $Z_N^*(s_{n-1})$ и $X_N^*(s_{n1})$.

Рассмотрим теперь двухшаговую задачу: присоединим к n-му шагу (n-1)-й (рис. 2).

Для любых состояний s_n-$_2$, произвольных управлений X_{n-1} и оптимальном управлении на n-м шаге значение целевой функции на двух последних шагах равно:

$$f_{n-1}(s_{n-2}, X_{n-1}) + Z_n^*(s_{n-1}). \qquad (6)$$

Согласно принципу оптимальности для любых s_n-i решение нужно выбирать так, чтобы оно вместе с оптимальным управлением на последнем (n-м) шаге приводило бы к максимуму целевой функции на двух последних шагах. Следовательно, нужно найти максимум выражения (6) по всем допустимым управлениям X_{n-1} Максимум этой суммы зависит от s_{n-2},

обозначается через $Z^*_n(s_{n-2})$ и называется *условным максимумом целевой функции при оптимальном управлении на двух последних шагах.* Соответствующее управление X_{n-1} на (n—1)-м шаге обозначается через

$X^*_{n-1}(s_n-2)$ и называется *условным оптимальным управлением* на (n-1)-м шаге.

$$Z^*_{n-1}(s_{n-2}) = \max_{\{X_{n-1}\}} \left\{ f_{n-1}(s_{n-2}, X_{n-1}) + Z^*_n(s_{n-1}) \right\}. \quad (7)$$

Следует обратить внимание на то, что выражение, стоящее в фигурных скобках (7), зависит только от S_{n-2} и X_{n-1}, так как можно найти из уравнения состояний (2) при $к=n—1$

$$s_{n-1} = \varphi_{n-1}(s_{n-2}, X_{n-1})$$

и подставить вместо s_{n-1} в функцию $Z_n^*(s_{n-1})$.

В результате максимизации только по одной переменной X_{n-1} согласно уравнению (7) вновь получаются две функции:

$$Z^*_{n-1}(s_{n-2}) \text{ и } X^*_{n-1}(s_{n-2}).$$

Далее рассматривается трехшаговая задача: к двум последним шагам присоединяется (n—2)-й и т. д.

Обозначим через $Z^*_k(s_{n-1})$ *условный максимум целевой функции, полученный при оптимальном управлении на n~к*+1 *шагах, начиная с*

k-го до конца, при условии, что к началу k-го шага система находилась в состоянии s_{k-1}. Фактически эта функция равна

$$Z_k^*(s_{k-1}) = \max_{\{(x_k,\dots,x_n)\}} \sum_{i=k}^{n} f_i(s_{i-1}, X_i).$$

Тогда

$$Z_{k+1}^*(s_k) = \max_{\{(x_{k+1},\dots,x_n)\}} \sum_{i=k+1}^{n} f_i(s_{i-1}, X_i).$$

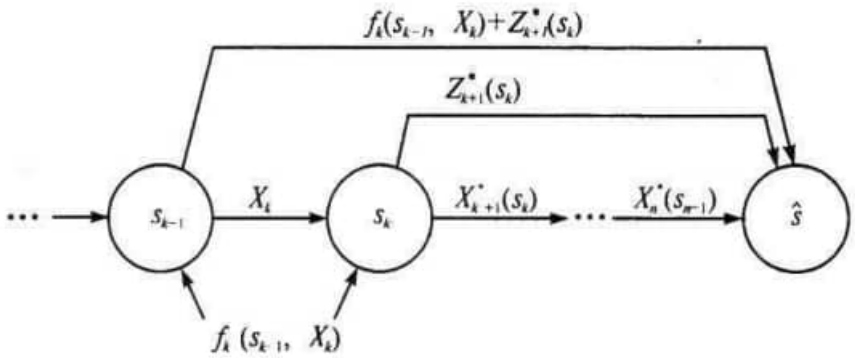


Рис. 3

Целевая функция на *n—к* последних шагах (рис. 3) при произвольном управлении $X_к$ на *к-м* шаге и оптимальном управлении на последующих *n—к* шагах равна

$$f_k(s_{k-1}, X_k) + Z_{k+1}^*(s_k).$$

Согласно принципу оптимальности, $X_к$ выбирается из условия максимума этой суммы, т.е.

$$Z_k^*(s_{k-1}) = \max_{\{X_k\}} \left\{ f_k(s_{k-1}, X_k) + Z_{k+1}^*(s_k) \right\}, \qquad \textbf{(8)}$$

$$k = n-1, n-2, \dots, 2, 1.$$

Управление $X_к$ на *к-м* шаге, при котором достигается максимум в (8), обозначается через $X^*_k(s_{k-1})$ и называется *условным оптимальным управлением на к-м шаге* (в правую часть уравнения(8) следует вместо s_k подставить выражение $s_k=\varphi_k(s_{k-1}, X_k)$, найденное из уравнений состояния).

Уравнения (8) называют *уравнениями Беллмана.* Это рекуррентные соотношения, позволяющие найти предыдущее значение функции, зная последующие. Если из (5) найти $Z^*_n(s_{n-1})$, то при *к=n—* 1 из (8) можно определить, решив задачу максимизации для всех возможных значений $s_{n\sim2}$, выражения для $Z^*_{n-1}(s_n-_2)$ и соответствующее $X^*_{n-1}(s_n-_2)$- Далее, зная $Z^*_{n-1}(s_n-_2)$, находим, используя (8) и (2), уравнения состояний.

Процесс решения уравнений (5) и (8) называется *условной оптимизацией.*

В результате условной оптимизации получаются две последовательности:

$$Z_n^*(s_{n-1}), Z_{n-1}^*(s_{n-2}), \dots, Z_2^*(s_1), Z_1^*(s_0) -$$

условные максимумы целевой функции на последнем, на двух последних, на ...n шагах и

$$X_n^*(s_{n-1}),\ X_{n-1}^*(s_{n-2}),\ \ldots,\ X_2^*(s_1),\ X_1^*(s_0) -$$

условные оптимальные управления на n-м, (n-1)-м, ..., 1-м шагах. Используя эти последовательности, можно найти решение задачи ДП при данных n и s0. По определению Z1*(s0) — условный максимум целевой функции за n шагов при условии, что к началу 1-го шага система была в состоянии s0, т.е.

$$Z_{\max} = Z_1^*(s_0). \qquad (9)$$

Далее следует использовать последовательность условных оптимальных управлений и уравнения состояний (2).

При фиксированном s0 получаем X1*= X1*(s0). Далее из уравнений (2) находим s1*=φ(s0, X1*) и подставляем это выражение в последовательность условных оптимальных управлений:

2. Практическая часть

. Задача о распределении средств между предприятиями

Планируется деятельность четырех промышленных предприятий (системы) на очередной год. Начальные средства: s0 =5 усл. ед.

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
1	8	6	3	4
2	10	9	4	6
3	11	11	7	8
4	12	13	11	13
5	18	15	18	16

Размеры вложения в каждое предприятие кратны 1 усл. ед. Средства x, выделенные k:-му предприятию (к=1, 2, 3, 4), приносят в конце года прибыль fk(x). Функции fk(x) заданы таблично (табл. 1). Принято считать, что:

а) прибыль fk(x) не зависит от вложения средств в другие предприятия;

б) прибыль от каждого предприятия выражается в одних условных единицах;

в) суммарная прибыль равна сумме прибылей, полученных от каждого предприятия.

Определить, какое количество средств нужно выделить каждому предприятию, чтобы суммарная прибыль была наибольшей.

Таблица 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
1	8	6	3	4
2	10	9	4	6
3	11	11	7	8
4	12	13	11	13
5	18	15	18	16

Решение. Обозначим через x* количество средств, выделенных к-му предприятию. (Нумерацию предприятий 1, 2, 3, 4 сохраняем в процессе решения неизменной.)

Суммарная прибыль равна

$$Z=\sum_{k=1}^{4} f_k(x_k). \qquad \textbf{(1)}$$

Переменные $x_к$ удовлетворяют ограничениям:

Требуется найти переменные x_1, x_2, x_3, x_4, удовлетворяющие системе

$$\sum_{k=1}^{4} x_k = 5,$$
$$x_k \geq 0,\ k = 1,2,3,4.$$

ограничений (2) и обращающие в максимум функцию 1).

Особенности модели. Ограничения линейные, но переменные целочисленные, а функции $f_r(x)$ заданы таблично] поэтому нельзя применить методы целочисленного линейного программирования .

Схема решения задачи методом ДП имеет следующий вид: процесс решения распределения средств s_0 =5 можно рассматривать как шаговый, номер шага совпадет с номером предприятия; выбор временных x_1, x_2, x_3, x_4,— управление соответственно на I, II, III, *V* шагах, *s* — конечное состояние процесса распределения — равно нулю, так как все средства должны быть вложены в производство, *s=0* Схема распределения показана на рис. 1.

Уравнения состояний (12.2) в данной задаче имеют вид:

$$s_k=s_{k-1}-x_k,\ \ k=1,\ 2,\ 3,\ 4, \qquad (3)$$

Сравнивая подчеркнутые числа, получим Z_1^* (5)=24 усл. ед. = Zmax при $x_1^*=x_1*(5)= 1$.

Используя уравнения (3), получим s_1^* =5-1=4, а по табл. 2 в столбце 9 находим $x_2^* = x_2*(4)$ - 2. Далее находим s_2*=4-2=2, а по табл. 2 в столбце 6 — $x_3^* = x_3^*\{2)$ = 1. Наконец, $s_3*=2-1=1$ и $x_4^* = x_4^*\{1) = 1$ т.е. $X^*(1;2;\ 1;\ 1)$.

Максимум суммарной прибыли равен 24 усл. ед. средств при условии, что 1-му предприятию выделено 1 усл. ед.; 2-му предприятию — 2 усл. ед.; 3-му предприятию — 1 усл. ед.; 4-му предприятию — 1 усл.ед. **(2)**

Замечание 1. Решение четырехмерной задачи на определение условного экстремума сведено фактически к решению четырех одномерных задач: на каждом шаге определялась одна переменная *x*.

Замечание 2. На разобранной задаче видно, что метод ДП безразличен к виду и способу задания функции: $f_k(x)$ были заданы таблично, поэтому и $Z^*_k(s)$ и $X^*_k(s)$ принимали дискретные значения, представленные в табл. 2.

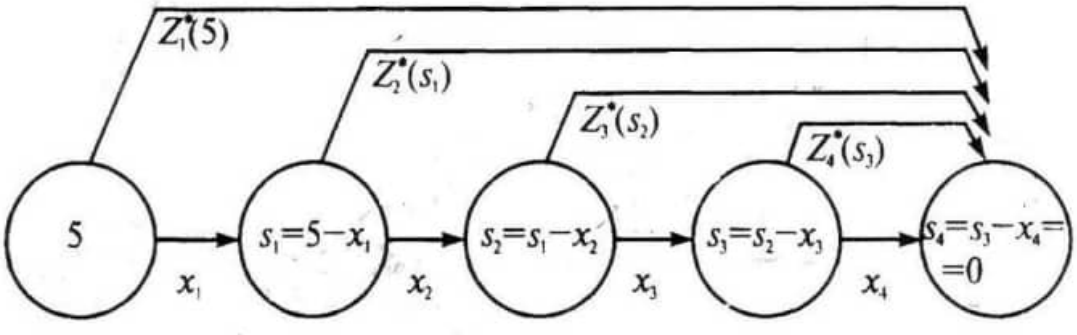


Рис.1

Таблица 2

s_{k-1}	x_k	s_k	$k=3$			$k=2$			$k=1$		
			$f_3(x_3)+Z_4^*(s_3)$	$Z_3^*(s_2)$	$x_3^*(s_2)$	$f_2(x_2)+Z_3^*(s_2)$	$Z_2^*(s_1)$	$x_2^*(s_1)$	$f_1(x_1)+Z_2^*(s_1)$	$Z_1^*(s_0)$	$x_1^*(s_0)$
1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0+1=1			0+4=4			0+6=6	8	1
	1	0	3+0=3	4	0	6+0=6	6	1	8+0=8		
2	0	2	0+6=6			0+7=7			0+10=10		
	1	1	3+4=7	7	1	6+4=10	10	1	8+6=14	14	1
	2	0	4+0=4			9+0=9			10+0=10		
3	0	3	0+8=8			0+9=9			0+13=13		
	1	2	3+6=9	9	1	6+7=13	13	1	8+10=18	18	1
	2	1	4+4=8			9+4=13		2	10+6=16		

s_{k-1}	x_k	s_k	$k=3$			$k=2$			$k=1$		
			$f_3(x_3)+Z_4^*(s_3)$	$Z_3^*(s_2)$	$x_3^*(s_2)$	$f_2(x_2)+Z_3^*(s_2)$	$Z_2^*(s_1)$	$x_2^*(s_1)$	$f_1(x_1)+Z_2^*(s_1)$	$Z_1^*(s_0)$	$x_1^*(s_0)$
1	2	3	4	5	6	7	8	9	10	11	12
	3	0	7+0=7			11+0=11			11+0=11		
4	0	4	0+13=13			0+13=13			0+16=16		
	1	3	3+8=11			6+9=15			8+13=21		
	2	2	4+6=10	13	0	9+7=16	16	2	10+10=20	21	1
	3	1	7+4=11			11+4=15			11+6=17		
	4	0	11+0=11			13+0=13			12+0=12		
5	0	5	0+16=16			0+18=18			0+19=19		
	1	4	3+13=16			6+13=19			8+16=24		
	2	3	4+8=12	18	5	9+9=18	19	1	10+13=23	24	1
	3	2	7+6=13			11+7=18			11+10=21		
	4	1	11+4=15			13+4=17			12+6=18		
	5	0	18+0=18			15+0=15			18+0=18		

Введем в рассмотрение функцию $Z_k^*(s_{k-1})$ — условную оптимальную прибыль, полученную от *k-го,* (k+1)-го, ..., 4-го предприятий, если между ними распределялись оптимальным образом средства $s_{k-1}(0< s_{k-1}<5)$. Допустимые управления на *к-м* шаге удовлетворяют условию: $0< X_k < s_{k-1}$ (либо *к-му* предприятию ничего не выделяем, $x_k=0$, либо не больше того, что имеем к *к-му* шагу,

Уравнения (5) и (8) имеют вид:

$$k=4,\ s_4 = 0 \Rightarrow Z_4^*(s_3) = \max_{0\le x_4 \le s_3} f_4(x_4), \qquad \text{(а)}$$

$$Z_3^*(s_2) = \max_{0\le x_3 \le s_2} \{f_3(x_3) + Z_4^*(s_3)\}, \qquad \text{(б)}$$

$$Z_2^*(s_1) = \max_{0\le x_2 \le s_1} \{f_2(x_2) + Z_3^*(s_2)\}, \qquad \text{(в)}$$

$$Z_1^*(5) = \max_{0\le x_1 \le 5} \{f_1(x_1) + Z_2^*(s_1)\}. \qquad \text{(г)}$$

Последовательно решаем записанные уравнения, проводя условную оптимизацию *(см.* рис. 1) каждого шага.

IV ш а г. В табл. 1 $f_4(x)$ прибыли монотонно возрастают, поэтому все средства, оставшиеся к IV шагу, следует вложить в 4-е предприятие. При этом для возможных значений $s_з=0, 1, \ldots, 5$ получим:

$$Z_4^*(s_3) = f_4(s_3) \text{ и } x_4^*(s_3) = s_3.$$

I l l шаг. Делаем все предположения относительно остатка средств s_2 к III шагу (т.е. после выбора X_1 и x_2). S_2 может принимать значения 0, 1, 2, 3, 4, 5 (например, $s_2 = 0$, если все средства отданы 1-му и 2-му предприятиям, $s_2=5$, если 1-е и 2-е предприятия ничего не получили, и т.д.). В зависимости от этого выбираем $0<x_з<s_2>$ находим $S_3=\sim S_2 — x_з$ и сравниваем для разных x_3, при фиксированном s_2 значения суммы $f_3(x_3) + Z_4^*(s_3)$. Для каждого s_2 наибольшее из этих значений есть $Z_3^*(s_2)$ — условная оптимальная прибыль, полученная при оптимальном распределении средств s_2 между 3-м и 4-м предприятиями. Оптимизация дана в табл. 2 при k=3. Для каждого значения s_2 $Z_3^*(s_2)$ и $X_3^*(s_2)$ помещены в графах 5 и 6 соответственно.

II шаг. Условная оптимизация, согласно уравнению (в), проведена в табл. 2 при *k=2.* Для всех возможных значений s_1 значения $Z_2^*(s_1)$ и $X_2^*(s_1)$ находятся в столбцах 8 и 9 соответственно; первые слагаемые в столбце 7 — значения $f_2(x_2)$, взяты из табл. 1 а вторые слагаемые взяты из столбца 5 табл. 2 при $s_2 = s_1 — x_1$.

I шаг. Условная оптимизация (уравнение (г) проведена в табл. 2 при *k=1* для $s_0=5$. Поясним решение подробно: если $x_1=0$, то $s_1=5$, прибыль, полученная от четырех предприятий при условии, что $s_1=5$ ед. средств между оставшимися тремя предприятиями будут распределены оптимально, равна $f_1(0)+Z_2^*(5)=0+19=19$ $(Z_2^*(5)$ взято из столбца 9 табл. 2 при $s_1=5)$. Если $x_1=1$, то $s_2=4$. Суммарная прибыль при условии, что $s_2=4$ ед. средств между оставшимися тремя предприятиями будут распределены оптимально, равна $f_1(l)+Z_2^*(4)=8+16=24$ $f_1(l)$ взято из табл. 1, а $Z_2^*(4)$ — из столбца 9 табл. 2.) Аналогично при $x_1,=2$, $s_2=3$ $F_1(2)+Z_2(3)=10+13=23$;

при $x_1=3$, $s_2=2$ и $f_1(3)+Z_2^*(3)=11+10=\underline{21}$;

при $x_1=4$, $s_2=1$ и $f_1(4)+Z_2^*(1)=12+16=\underline{28}$;

при $x_1=5$, $s_2=0$ и $f_1(5)+Z_2^*(0)=18+0=\underline{18}$.

В результате найдены два оптимальных решения: $X^{(1)}$*(1; 1; 1; 1) и $X^{(2)}$* (1; 2; 0; 1). Если начальные средства увеличились, например, на 1 усл. ед., s_o=6, а функции прибыли $f_k(x)$ остались прежними, то в табл. 2 достаточно добавить раздел для $s_o=6$ при k=3, 2, 1; этот фрагмент расчетов помещен в табл. 3.

Таблица 3

1	2	3	4	5	6	7	8	9	10	11	12
	0	6	0+16=16			0+22=22			0+24=24		
	1	5	3+16=19			6+18=24			8+19=27		
6	2	4	4+13=17	22	5	9+13=22	24	1	10+16=26	27	1
	3	3	7+8=15			11+9=20			11+13=24		
	4	2	11+6=17			13+7=20			12+10=22		
	5	1	18+4=22			15+4=19			18+6=24		

Получаем $Z_{max}=27$, $x_1^*=1 \rightarrow s_1^*=6-1=5 \Rightarrow x_2^*=1 \rightarrow s_2^*=5-1=$
$=4 \Rightarrow x_3^*=0 \rightarrow s_3^*=4-0=4 \Rightarrow x_4^*=4$.

Оптимальное решение X (1; 1; 0; 4).

Если принято решение распределить средства $s_o=5$ между 2-, 3- и 4-м предприятиями, то задача уже решена в табл.2. В разделе $k=2$ таблицы находим $Z_{max}= Z_2$*(5)=19 при условии, **что $x^*_2=1$, x_3*=0 $x^*_4=4$.**

Наконец, если увеличилось количество предприятий (число шагов), то схему можно дополнить, присоединяя шаги с номерами k=0, 1, ... и т. д. Например, пусть средства ξ,о = 6 распределяются между пятью предприятиями. Функция прибыли для пятого предприятия задана формулой f(x)=3x+1, если x≠0 и f(0)=0. Присвоим 5-му предприятию номер k=0, тогда x_0 — средства, выделенные этому предприятию. Обозначим через Z_Q^*(6) оптимальную прибыль, полученную от пяти предприятий:

$$Z_0^*(6) = \max_{0\le x_0 \le 6}\left\{f_0(x_0) + Z_1^*(s_1)\right\},$$

а $s_1=6—X_Q$.

Условная оптимизация 0-го шага дана в табл. .4

Таблица 4

x_0	0	1	2	3	4	5	6
$s_1=6-x_0$	6	5	4	3	2	1	0
$f(0)=0$	0	4	7	10	13	16	19
$Z_1^*(s_1)$ (взята из табл. 12.2 и 12.3 при $k=1$)	27	24	21	18	14	8	0
$f(x_0)+ Z_1(s_1)$	27	28	28	28	27	24	19

Следовательно, $Z_{max}=28$, а оптимальных решений четыре: X_1^*(1; 2; **1** 1), X_2^* (2\ **1;** 1; 1; 1), X_3^* (2; 1; 2; **0;** 1), X_4^* (3; 1; 1; 0; 1).

Замечание К недостаткам метода по-прежнему следует отнести возникновение технических трудностей при вычислениях в случае увеличения размерности. Если каждое управление X_k^* будет зависеть от r переменных, а состояние s_k^* — от p параметров, то на каждом шаге возникает р-мерная задача оптимизации

3 Задания для самостоятельной работы

Задача 1. Для двух предприятий выделено a единиц средств. Как распределить все средства в течение 4 лет, чтобы доход был наибольшим, если известно, что доход от x единиц средств, вложенных в первое предприятие, равен $f_1(x)$, а доход от y единиц средств, вложенных во второе предприятие, равен $f_2(y)$. Остаток средств к концу года составляет $g_1(x)$ для первого предприятия и $g_2(y)$ для второго предприятия. Задачу решить методом динамического программирования.

a	f_1	g_1	f_2	g_2
1000	3x	0,1x	2y	0,5y

Задача 2. Планируется распределение начальной суммы X_0 млн. р. Между четырьмя предприятиями некоторого объединения. Средства выделяются только в размерах кратных $a = 80$ млн. р. Функции прироста продукции от вложенных средств на каждом предприятии заданы таблично. Требуется так распределить вложения между предприятиями, чтобы общий прирост продукции (в млн. р.) был максимальным. Решить задачу на основе функционального уравнения Беллмана.

X_0	Вкладываемые средства X	Функции прироста продукции на предприятии			
		$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
400	0	10	15	13	14
	80	13	20	17	16
	160	16	22	21	23
	240	21	25	26	25
	320	25	30	28	27
	400	25	32	30	32

Занятие 5 Практическое занятие
Нахождение кратчайших путей в графе

1 Теоретическая часть

Алгоритм Флойда (поиск всех кратчайших путей в графе)

Алгоритм Флойда является одним из методов поиска кратчайших путей в графе. В отличии от алгоритма Дейкстры, который позволяет при доведении до конца построить ориентированное дерево кратчайших путей от некоторой вершины, метод Флойда позволяет найти длины всех кратчайших путей в графе. Конечно эта задача может быть решена и многократным применением алгоритма Дейкстры (каждый раз последовательно выбираем вершину от первой до N-ной, пока не получим кратчайшие пути от всех вершин графа), однако реализация подобной процедуры потребовала бы значительных вычислительных затрат.

Прежде чем представлять алгоритмы, необходимо ввести некоторые обозначения. Перенумеруем вершины исходного графа целыми числами от 1 до N. Обозначим через $d_{i,j}^m$ длину кратчайшего пути из вершинм i в вершину j, который в качестве промежуточных может содержать только первые m вершин графа. (Напомним, что промежуточной вершиной пути является любая принадлежащая ему вершина, не совпадающая с его начальной или конечной вершинами.) Если между вершинами i и j не существует ни одного пути указанного типа, то условно будем считать, что $d_{i,j}^m=\infty$. Из данного определения величин $d_{i,j}^m$ следует, что величина $d_{i,j}^0$, представляет длину кратчайшего пути из вершины i в вершину j, не имеющего промежуточных вершин, т. е. длину кратчайшей дуги, соединяющей i с j (если такие дуги присутствуют в графе). для любой вершины i положим $d_{i,i}^m= 0$. Отметим далее, что величина $d_{i,j}^m$представляет длину кратчайшего пути между вершинами i и j.

Обозначим через D^m матрицу размера NxN, элемент (i, j) которой совпадает с $d_{i,j}^m$. Если в исходном графе нам известна длина каждой дуги, то мы можем сформировать матрицу D^0. Наша цель состоит в определении матрицы D^N, представляющей кратчайшие пути между всеми вершинами рассматриваемого графа.

В алгоритме Флойда в качестве исходной выступает матрица D^0. Вначале из этой матрицы вычисляется матрица D^1. Затем по матрице D^1 вычисляется матрицав D^2 и т. д. Процесс повторяется до тех пор, пока по матрице D^{N-1} не будет вычислена матрица D^N.

Рассмотрим основную идею, лежащую в основе алгоритма Флойда. Суть алгоритма Флойда заключается в проверке того, не окажется ли путь из вершины i в вершину j короче, если он будет проходить через некоторую промежуточную вершину m. Предположим, что нам известны:

1. кратчайший путь из вершины i в вершину m, в котором в качестве промежуточных допускается использование только первых (m - 1) вершин;
2. кратчайший путь из вершины m в вершину j, в котором в качестве промежуточных допускается использование только первых (m - 1) вершин;
3. кратчайший путь из вершины i в вершину j, в котором в качестве промежуточных допускается использование только первых (m - 1) вершин.

Поскольку по предположению исходный граф не может содержать контуров отрицательной длины, один из двух путей — путь, совпадающий с представленным в пункте 3, или путь, являющийся объединением путей из пунктов 1 и 2 — должен быть кратчайшим путем из вершины i в вершину j, в котором в качестве промежуточных допускается использование только первых m вершин. Таким образом,

$$\mathbf{d_{i,j}^{m}=min\{ d_{i,m}^{m-1}+ d_{m,j}^{m-1}; d_{i,j}^{m-1}\}}$$

Из соотношения видно, что для вычисления элементов матрицы D^m необходимо располагать лишь элементами матрицы D^{m-1}. Более того, соответствующие вычисления могут быть проведены без обращения к исходному графу. Теперь мм в состоянии датьформальное описание алгоритма Флойда для нахождения на графе кратчайших путей между всеми парами вершин.

Алгоритм

1. Перенумеровать вершины графа от 1 до N целыми числами, определить матрицу D^0, каждый элемент $d_{i,j}$ которой есть длина кратчайшей дуги между вершинами i и j. Если такой дуги нет, положить значение элемента равным ∞. Кроме того, положить значения диагонального элемента $d_{i,i}$равным 0.
2. Для целого m, последовательно принимающего значения 1...N определить по элементам матрицы D^{m-1} элементы D^m
3. Алгоритм заканчивается получением матрицы всех кратчайших путей D^N, N – число вершин графа.

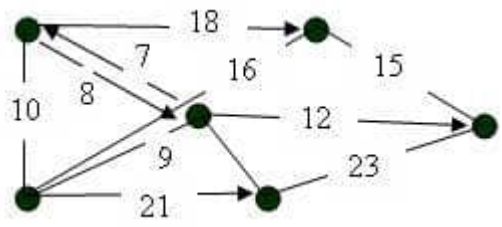
Напомним, для определения по известным элементам матрицы D^{m-1} элементов матрицы D^m в алгоритме Флойда применяется рекурсивное соотношение:

$$\mathbf{d_{i,j}^{m}=min\{ d_{i,m}^{m-1}+ d_{m,j}^{m-1}; d_{i,j}^{m-1}\}}$$

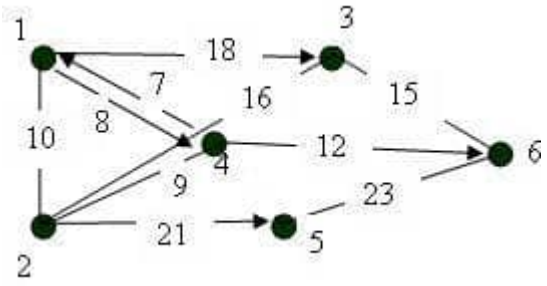
$d_{i,j}^{m}$ – элемент матрицы D^m, $d_{i,j}^{m-1}$ – элементы матрицы D^{m-1} найденой на предыдущем шаге алгоритма.

2 Практическая часть

Необходимо найти кратчайшие пути между каждой парой вершин в графе, представленом на рисунке:



Пронумеруем все вершины графа, и составим матрицу длин кратчайших дуг D^0, в случае, если дуги между вершиной i и j не существует, элементу $d_{i,j}$ матрицы присваивается значение ∞. Исходный граф с пронумерованными вершинами представлен на рисунке ниже.



Матрица D^0:

$D^0=$	0	10	18	8	∞	∞
	10	0	16	9	21	∞
	∞	16	0	∞	∞	15
	7	9	∞	0	∞	12
	∞	∞	∞	∞	0	23
	∞	∞	15	∞	23	0

На основании матрицы D^0, вычислим последовательно все элементы матрицы D^1. Для этого мы используем рекуррентное соотношение $d_{i,j}^{1}=\min\{ d_{i,1}^{0}+ d_{1,j}^{0}; d_{i,j}^{0}\}$.

$d_{1,1}^{1}=\min\{d_{1,1}^{0}+d_{1,1}^{0}d_{1,1}^{0}\}=\min\{0+0;0\}=0$
$d_{1,2}^{1}=\min\{d_{1,1}^{0}+d_{1,2}^{0}d_{1,2}^{0}\}=\min\{0+10;10\}=10$
$d_{1,3}^{1}=\min\{d_{1,1}^{0}+d_{1,3}^{0}d_{1,3}^{0}\}=\min\{0+18;18\}=18$
$d_{1,4}^{1}=\min\{d_{1,1}^{0}+d_{1,4}^{0}d_{1,4}^{0}\}=\min\{0+8;8\}=8$
$d_{1,5}^{1}=\min\{d_{1,1}^{0}+d_{1,5}^{0}d_{1,5}^{0}\}=\min\{0+\infty;\infty\}=\infty$
$d_{1,6}^{1}=\min\{d_{1,1}^{0}+d_{1,6}^{0}d_{1,6}^{0}\}=\min\{0+\infty;\infty\}=\infty$
$d_{2,1}^{1}=\min\{d_{2,1}^{0}+d_{1,1}^{0}d_{2,1}^{0}\}=\min\{10+0;10\}=10$
$d_{2,2}^{1}=\min\{d_{2,1}^{0}+d_{1,2}^{0}d_{2,2}^{0}\}=\min\{10+10;0\}=0$
$d_{2,3}^{1}=\min\{d_{2,1}^{0}+d_{1,3}^{0}d_{2,3}^{0}\}=\min\{10+18;16\}=16$
$d_{2,4}^{1}=\min\{d_{2,1}^{0}+d_{1,4}^{0}d_{2,4}^{0}\}=\min\{10+8;9\}=9$
$d_{2,5}^{1}=\min\{d_{2,1}^{0}+d_{1,5}^{0}d_{2,5}^{0}\}=\min\{10+\infty;21\}=21$
$d_{2,6}^{1}=\min\{d_{2,1}^{0}+d_{1,6}^{0}d_{2,6}^{0}\}=\min\{10+\infty;\infty\}=\infty$
$d_{3,1}^{1}=\min\{d_{3,1}^{0}+d_{1,1}^{0}d_{3,1}^{0}\}=\min\{\infty+0;\infty\}=\infty$
$d_{3,2}^{1}=\min\{d_{3,1}^{0}+d_{1,2}^{0}d_{3,2}^{0}\}=\min\{\infty+10;16\}=16$
$d_{3,3}^{1}=\min\{d_{3,1}^{0}+d_{1,3}^{0}d_{3,3}^{0}\}=\min\{\infty+18;0\}=0$
$d_{3,4}^{1}=\min\{d_{3,1}^{0}+d_{1,4}^{0}d_{3,4}^{0}\}=\min\{\infty+8;\infty\}=\infty$
$d_{3,5}^{1}=\min\{d_{3,1}^{0}+d_{1,5}^{0}d_{3,5}^{0}\}=\min\{\infty+\infty;\infty\}=\infty$
$d_{3,6}^{1}=\min\{d_{3,1}^{0}+d_{1,6}^{0}d_{3,6}^{0}\}=\min\{\infty+\infty;15\}=15$
$d_{4,1}^{1}=\min\{d_{4,1}^{0}+d_{1,1}^{0}d_{4,1}^{0}\}=\min\{7+0;7\}=7$
$d_{4,2}^{1}=\min\{d_{4,1}^{0}+d_{1,2}^{0}d_{4,2}^{0}\}=\min\{7+10;9\}=9$
$d_{4,3}^{1}=\min\{d_{4,1}^{0}+d_{1,3}^{0}d_{4,3}^{0}\}=\min\{7+18;\infty\}=25$
$d_{4,4}^{1}=\min\{d_{4,1}^{0}+d_{1,4}^{0}d_{4,4}^{0}\}=\min\{7+8;0\}=0$
$d_{4,5}^{1}=\min\{d_{4,1}^{0}+d_{1,5}^{0}d_{4,5}^{0}\}=\min\{7+\infty;\infty\}=\infty$
$d_{4,6}^{1}=\min\{d_{4,1}^{0}+d_{1,6}^{0}d_{4,6}^{0}\}=\min\{7+\infty;12\}=12$
$d_{5,1}^{1}=\min\{d_{5,1}^{0}+d_{1,1}^{0}d_{5,1}^{0}\}=\min\{\infty+0;\infty\}=\infty$
$d_{5,2}^{1}=\min\{d_{5,1}^{0}+d_{1,2}^{0}d_{5,2}^{0}\}=\min\{\infty+10;\infty\}=\infty$
$d_{5,3}^{1}=\min\{d_{5,1}^{0}+d_{1,3}^{0}d_{5,3}^{0}\}=\min\{\infty+18;\infty\}=\infty$
$d_{5,4}^{1}=\min\{d_{5,1}^{0}+d_{1,4}^{0}d_{5,4}^{0}\}=\min\{\infty+8;\infty\}=\infty$
$d_{5,5}^{1}=\min\{d_{5,1}^{0}+d_{1,5}^{0}d_{5,5}^{0}\}=\min\{\infty+\infty;0\}=0$
$d_{5,6}^{1}=\min\{d_{5,1}^{0}+d_{1,6}^{0}d_{5,6}^{0}\}=\min\{\infty+\infty;23\}=23$
$d_{6,1}^{1}=\min\{d_{6,1}^{0}+d_{1,1}^{0}d_{6,1}^{0}\}=\min\{\infty+0;\infty\}=\infty$
$d_{6,2}^{1}=\min\{d_{6,1}^{0}+d_{1,2}^{0}d_{6,2}^{0}\}=\min\{\infty+10;\infty\}=\infty$
$d_{6,3}^{1}=\min\{d_{6,1}^{0}+d_{1,3}^{0}d_{6,3}^{0}\}=\min\{\infty+18;15\}=15$
$d_{6,4}^{1}=\min\{d_{6,1}^{0}+d_{1,4}^{0}d_{6,4}^{0}\}=\min\{\infty+8;\infty\}=\infty$
$d_{6,5}^{1}=\min\{d_{6,1}^{0}+d_{1,5}^{0}d_{6,5}^{0}\}=\min\{\infty+\infty;23\}=23$
$d_{6,6}^{1}=\min\{d_{6,1}^{0}+d_{1,6}^{0}d_{6,6}^{0}\}=\min\{\infty+\infty;0\}=0$

Представим матрицу D^1, включив в нее рассчитанные элементы.

$D^1=$

0	10	18	8	∞	∞
10	0	16	9	21	∞
∞	16	0	∞	∞	15
7	9	25	0	∞	12
∞	∞	∞	∞	0	23
∞	∞	15	∞	23	0

На основании матрицы D^1, вычислим последовательно все элементы матрицы D^2. Для этого мы используем рекуррентное соотношение $d_{i,j}^2=\min\{ d_{i,2}^1+ d_{2,j}^1; d_{i,j}^1\}$.
$d_{1,1}^2=\min\{d_{1,2}^1+d_{2,1}^1 d_{1,1}^1\}=\min\{10+10;0\}=0$
$d_{1,2}^2=\min\{d_{1,2}^1+d_{2,2}^1 d_{1,2}^1\}=\min\{10+0;10\}=10$
$d_{1,3}^2=\min\{d_{1,2}^1+d_{2,3}^1 d_{1,3}^1\}=\min\{10+16;18\}=18$
$d_{1,4}^2=\min\{d_{1,2}^1+d_{2,4}^1 d_{1,4}^1\}=\min\{10+9;8\}=8$
$d_{1,5}^2=\min\{d_{1,2}^1+d_{2,5}^1 d_{1,5}^1\}=\min\{10+21;\infty\}=31$
$d_{1,6}^2=\min\{d_{1,2}^1+d_{2,6}^1 d_{1,6}^1\}=\min\{10+\infty;\infty\}=\infty$
$d_{2,1}^2=\min\{d_{2,2}^1+d_{2,1}^1 d_{2,1}^1\}=\min\{0+10;10\}=10$
$d_{2,2}^2=\min\{d_{2,2}^1+d_{2,2}^1 d_{2,2}^1\}=\min\{0+0;0\}=0$
$d_{2,3}^2=\min\{d_{2,2}^1+d_{2,3}^1 d_{2,3}^1\}=\min\{0+16;16\}=16$
$d_{2,4}^2=\min\{d_{2,2}^1+d_{2,4}^1 d_{2,4}^1\}=\min\{0+9;9\}=9$
$d_{2,5}^2=\min\{d_{2,2}^1+d_{2,5}^1 d_{2,5}^1\}=\min\{0+21;21\}=21$
$d_{2,6}^2=\min\{d_{2,2}^1+d_{2,6}^1 d_{2,6}^1\}=\min\{0+\infty;\infty\}=\infty$
$d_{3,1}^2=\min\{d_{3,2}^1+d_{2,1}^1 d_{3,1}^1\}=\min\{16+10;\infty\}=26$
$d_{3,2}^2=\min\{d_{3,2}^1+d_{2,2}^1 d_{3,2}^1\}=\min\{16+0;16\}=16$
$d_{3,3}^2=\min\{d_{3,2}^1+d_{2,3}^1 d_{3,3}^1\}=\min\{16+16;0\}=0$
$d_{3,4}^2=\min\{d_{3,2}^1+d_{2,4}^1 d_{3,4}^1\}=\min\{16+9;\infty\}=25$
$d_{3,5}^2=\min\{d_{3,2}^1+d_{2,5}^1 d_{3,5}^1\}=\min\{16+21;\infty\}=37$
$d_{3,6}^2=\min\{d_{3,2}^1+d_{2,6}^1 d_{3,6}^1\}=\min\{16+\infty;15\}=15$
$d_{4,1}^2=\min\{d_{4,2}^1+d_{2,1}^1 d_{4,1}^1\}=\min\{9+10;7\}=7$
$d_{4,2}^2=\min\{d_{4,2}^1+d_{2,2}^1 d_{4,2}^1\}=\min\{9+0;9\}=9$
$d_{4,3}^2=\min\{d_{4,2}^1+d_{2,3}^1 d_{4,3}^1\}=\min\{9+16;25\}=25$
$d_{4,4}^2=\min\{d_{4,2}^1+d_{2,4}^1 d_{4,4}^1\}=\min\{9+9;0\}=0$
$d_{4,5}^2=\min\{d_{4,2}^1+d_{2,5}^1 d_{4,5}^1\}=\min\{9+21;\infty\}=30$
$d_{4,6}^2=\min\{d_{4,2}^1+d_{2,6}^1 d_{4,6}^1\}=\min\{9+\infty;12\}=12$
$d_{5,1}^2=\min\{d_{5,2}^1+d_{2,1}^1 d_{5,1}^1\}=\min\{\infty+10;\infty\}=\infty$
$d_{5,2}^2=\min\{d_{5,2}^1+d_{2,2}^1 d_{5,2}^1\}=\min\{\infty+0;\infty\}=\infty$
$d_{5,3}^2=\min\{d_{5,2}^1+d_{2,3}^1 d_{5,3}^1\}=\min\{\infty+16;\infty\}=\infty$
$d_{5,4}^2=\min\{d_{5,2}^1+d_{2,4}^1 d_{5,4}^1\}=\min\{\infty+9;\infty\}=\infty$
$d_{5,5}^2=\min\{d_{5,2}^1+d_{2,5}^1 d_{5,5}^1\}=\min\{\infty+21;0\}=0$
$d_{5,6}^2=\min\{d_{5,2}^1+d_{2,6}^1 d_{5,6}^1\}=\min\{\infty+\infty;23\}=23$
$d_{6,1}^2=\min\{d_{6,2}^1+d_{2,1}^1 d_{6,1}^1\}=\min\{\infty+10;\infty\}=\infty$
$d_{6,2}^2=\min\{d_{6,2}^1+d_{2,2}^1 d_{6,2}^1\}=\min\{\infty+0;\infty\}=\infty$
$d_{6,3}^2=\min\{d_{6,2}^1+d_{2,3}^1 d_{6,3}^1\}=\min\{\infty+16;15\}=15$
$d_{6,4}^2=\min\{d_{6,2}^1+d_{2,4}^1 d_{6,4}^1\}=\min\{\infty+9;\infty\}=\infty$
$d_{6,5}^2=\min\{d_{6,2}^1+d_{2,5}^1 d_{6,5}^1\}=\min\{\infty+21;23\}=23$
$d_{6,6}^2=\min\{d_{6,2}^1+d_{2,6}^1 d_{6,6}^1\}=\min\{\infty+\infty;0\}=0$
Представим матрицу D^2, включив в нее рассчитанные элементы.

$D^2=$	0	10	18	8	31	∞
	10	0	16	9	21	∞
	26	16	0	25	37	15

	7	9	25	0	30	12
	∞	∞	∞	∞	0	23
	∞	∞	15	∞	23	0

На основании матрицы D^2, вычислим последовательно все элементы матрицы D^3. Для этого мы используем рекуррентное соотношение $d_{i,j}^3=\min\{ d_{i,3}^2+ d_{3,j}^2; d_{i,j}^2\}$.
$d_{1,1}^3=\min\{d_{1,3}^2+d_{3,1}^2 d_{1,1}^2\}=\min\{18+26;0\}=0$
$d_{1,2}^3=\min\{d_{1,3}^2+d_{3,2}^2 d_{1,2}^2\}=\min\{18+16;10\}=10$
$d_{1,3}^3=\min\{d_{1,3}^2+d_{3,3}^2 d_{1,3}^2\}=\min\{18+0;18\}=18$
$d_{1,4}^3=\min\{d_{1,3}^2+d_{3,4}^2 d_{1,4}^2\}=\min\{18+25;8\}=8$
$d_{1,5}^3=\min\{d_{1,3}^2+d_{3,5}^2 d_{1,5}^2\}=\min\{18+37;31\}=31$
$d_{1,6}^3=\min\{d_{1,3}^2+d_{3,6}^2 d_{1,6}^2\}=\min\{18+15;\infty\}=33$
$d_{2,1}^3=\min\{d_{2,3}^2+d_{3,1}^2 d_{2,1}^2\}=\min\{16+26;10\}=10$
$d_{2,2}^3=\min\{d_{2,3}^2+d_{3,2}^2 d_{2,2}^2\}=\min\{16+16;0\}=0$
$d_{2,3}^3=\min\{d_{2,3}^2+d_{3,3}^2 d_{2,3}^2\}=\min\{16+0;16\}=16$
$d_{2,4}^3=\min\{d_{2,3}^2+d_{3,4}^2 d_{2,4}^2\}=\min\{16+25;9\}=9$
$d_{2,5}^3=\min\{d_{2,3}^2+d_{3,5}^2 d_{2,5}^2\}=\min\{16+37;21\}=21$
$d_{2,6}^3=\min\{d_{2,3}^2+d_{3,6}^2 d_{2,6}^2\}=\min\{16+15;\infty\}=31$
$d_{3,1}^3=\min\{d_{3,3}^2+d_{3,1}^2 d_{3,1}^2\}=\min\{0+26;26\}=26$
$d_{3,2}^3=\min\{d_{3,3}^2+d_{3,2}^2 d_{3,2}^2\}=\min\{0+16;16\}=16$
$d_{3,3}^3=\min\{d_{3,3}^2+d_{3,3}^2 d_{3,3}^2\}=\min\{0+0;0\}=0$
$d_{3,4}^3=\min\{d_{3,3}^2+d_{3,4}^2 d_{3,4}^2\}=\min\{0+25;25\}=25$
$d_{3,5}^3=\min\{d_{3,3}^2+d_{3,5}^2 d_{3,5}^2\}=\min\{0+37;37\}=37$
$d_{3,6}^3=\min\{d_{3,3}^2+d_{3,6}^2 d_{3,6}^2\}=\min\{0+15;15\}=15$
$d_{4,1}^3=\min\{d_{4,3}^2+d_{3,1}^2 d_{4,1}^2\}=\min\{25+26;7\}=7$
$d_{4,2}^3=\min\{d_{4,3}^2+d_{3,2}^2 d_{4,2}^2\}=\min\{25+16;9\}=9$
$d_{4,3}^3=\min\{d_{4,3}^2+d_{3,3}^2 d_{4,3}^2\}=\min\{25+0;25\}=25$
$d_{4,4}^3=\min\{d_{4,3}^2+d_{3,4}^2 d_{4,4}^2\}=\min\{25+25;0\}=0$
$d_{4,5}^3=\min\{d_{4,3}^2+d_{3,5}^2 d_{4,5}^2\}=\min\{25+37;30\}=30$
$d_{4,6}^3=\min\{d_{4,3}^2+d_{3,6}^2 d_{4,6}^2\}=\min\{25+15;12\}=12$
$d_{5,1}^3=\min\{d_{5,3}^2+d_{3,1}^2 d_{5,1}^2\}=\min\{\infty+26;\infty\}=\infty$
$d_{5,2}^3=\min\{d_{5,3}^2+d_{3,2}^2 d_{5,2}^2\}=\min\{\infty+16;\infty\}=\infty$
$d_{5,3}^3=\min\{d_{5,3}^2+d_{3,3}^2 d_{5,3}^2\}=\min\{\infty+0;\infty\}=\infty$
$d_{5,4}^3=\min\{d_{5,3}^2+d_{3,4}^2 d_{5,4}^2\}=\min\{\infty+25;\infty\}=\infty$
$d_{5,5}^3=\min\{d_{5,3}^2+d_{3,5}^2 d_{5,5}^2\}=\min\{\infty+37;0\}=0$
$d_{5,6}^3=\min\{d_{5,3}^2+d_{3,6}^2 d_{5,6}^2\}=\min\{\infty+15;23\}=23$
$d_{6,1}^3=\min\{d_{6,3}^2+d_{3,1}^2 d_{6,1}^2\}=\min\{15+26;\infty\}=41$
$d_{6,2}^3=\min\{d_{6,3}^2+d_{3,2}^2 d_{6,2}^2\}=\min\{15+16;\infty\}=31$
$d_{6,3}^3=\min\{d_{6,3}^2+d_{3,3}^2 d_{6,3}^2\}=\min\{15+0;15\}=15$
$d_{6,4}^3=\min\{d_{6,3}^2+d_{3,4}^2 d_{6,4}^2\}=\min\{15+25;\infty\}=40$
$d_{6,5}^3=\min\{d_{6,3}^2+d_{3,5}^2 d_{6,5}^2\}=\min\{15+37;23\}=23$
$d_{6,6}^3=\min\{d_{6,3}^2+d_{3,6}^2 d_{6,6}^2\}=\min\{15+15;0\}=0$
Представим матрицу D^3, включив в нее рассчитанные элементы.

$D^3=$	0	10	18	8	31	33
	10	0	16	9	21	31
	26	16	0	25	37	15
	7	9	25	0	30	12
	∞	∞	∞	∞	0	23
	41	31	15	40	23	0

На основании матрицы D^3, вычислим последовательно все элементы матрицы D^4. Для этого мы используем рекуррентное соотношение $d_{i,j}^4=\min\{ d_{i,4}^3+ d_{4,j}^3; d_{i,j}^3\}$.

$d_{1,1}^4=\min\{d_{1,4}^3+d_{4,1}^3 d_{1,1}^3\}=\min\{8+7;0\}=0$
$d_{1,2}^4=\min\{d_{1,4}^3+d_{4,2}^3 d_{1,2}^3\}=\min\{8+9;10\}=10$
$d_{1,3}^4=\min\{d_{1,4}^3+d_{4,3}^3 d_{1,3}^3\}=\min\{8+25;18\}=18$
$d_{1,4}^4=\min\{d_{1,4}^3+d_{4,4}^3 d_{1,4}^3\}=\min\{8+0;8\}=8$
$d_{1,5}^4=\min\{d_{1,4}^3+d_{4,5}^3 d_{1,5}^3\}=\min\{8+30;31\}=31$
$d_{1,6}^4=\min\{d_{1,4}^3+d_{4,6}^3 d_{1,6}^3\}=\min\{8+12;33\}=20$
$d_{2,1}^4=\min\{d_{2,4}^3+d_{4,1}^3 d_{2,1}^3\}=\min\{9+7;10\}=10$
$d_{2,2}^4=\min\{d_{2,4}^3+d_{4,2}^3 d_{2,2}^3\}=\min\{9+9;0\}=0$
$d_{2,3}^4=\min\{d_{2,4}^3+d_{4,3}^3 d_{2,3}^3\}=\min\{9+25;16\}=16$
$d_{2,4}^4=\min\{d_{2,4}^3+d_{4,4}^3 d_{2,4}^3\}=\min\{9+0;9\}=9$
$d_{2,5}^4=\min\{d_{2,4}^3+d_{4,5}^3 d_{2,5}^3\}=\min\{9+30;21\}=21$
$d_{2,6}^4=\min\{d_{2,4}^3+d_{4,6}^3 d_{2,6}^3\}=\min\{9+12;31\}=21$
$d_{3,1}^4=\min\{d_{3,4}^3+d_{4,1}^3 d_{3,1}^3\}=\min\{25+7;26\}=26$
$d_{3,2}^4=\min\{d_{3,4}^3+d_{4,2}^3 d_{3,2}^3\}=\min\{25+9;16\}=16$
$d_{3,3}^4=\min\{d_{3,4}^3+d_{4,3}^3 d_{3,3}^3\}=\min\{25+25;0\}=0$
$d_{3,4}^4=\min\{d_{3,4}^3+d_{4,4}^3 d_{3,4}^3\}=\min\{25+0;25\}=25$
$d_{3,5}^4=\min\{d_{3,4}^3+d_{4,5}^3 d_{3,5}^3\}=\min\{25+30;37\}=37$
$d_{3,6}^4=\min\{d_{3,4}^3+d_{4,6}^3 d_{3,6}^3\}=\min\{25+12;15\}=15$
$d_{4,1}^4=\min\{d_{4,4}^3+d_{4,1}^3 d_{4,1}^3\}=\min\{0+7;7\}=7$
$d_{4,2}^4=\min\{d_{4,4}^3+d_{4,2}^3 d_{4,2}^3\}=\min\{0+9;9\}=9$
$d_{4,3}^4=\min\{d_{4,4}^3+d_{4,3}^3 d_{4,3}^3\}=\min\{0+25;25\}=25$
$d_{4,4}^4=\min\{d_{4,4}^3+d_{4,4}^3 d_{4,4}^3\}=\min\{0+0;0\}=0$
$d_{4,5}^4=\min\{d_{4,4}^3+d_{4,5}^3 d_{4,5}^3\}=\min\{0+30;30\}=30$
$d_{4,6}^4=\min\{d_{4,4}^3+d_{4,6}^3 d_{4,6}^3\}=\min\{0+12;12\}=12$
$d_{5,1}^4=\min\{d_{5,4}^3+d_{4,1}^3 d_{5,1}^3\}=\min\{\infty+7;\infty\}=\infty$
$d_{5,2}^4=\min\{d_{5,4}^3+d_{4,2}^3 d_{5,2}^3\}=\min\{\infty+9;\infty\}=\infty$
$d_{5,3}^4=\min\{d_{5,4}^3+d_{4,3}^3 d_{5,3}^3\}=\min\{\infty+25;\infty\}=\infty$
$d_{5,4}^4=\min\{d_{5,4}^3+d_{4,4}^3 d_{5,4}^3\}=\min\{\infty+0;\infty\}=\infty$
$d_{5,5}^4=\min\{d_{5,4}^3+d_{4,5}^3 d_{5,5}^3\}=\min\{\infty+30;0\}=0$
$d_{5,6}^4=\min\{d_{5,4}^3+d_{4,6}^3 d_{5,6}^3\}=\min\{\infty+12;23\}=23$
$d_{6,1}^4=\min\{d_{6,4}^3+d_{4,1}^3 d_{6,1}^3\}=\min\{40+7;41\}=41$
$d_{6,2}^4=\min\{d_{6,4}^3+d_{4,2}^3 d_{6,2}^3\}=\min\{40+9;31\}=31$
$d_{6,3}^4=\min\{d_{6,4}^3+d_{4,3}^3 d_{6,3}^3\}=\min\{40+25;15\}=15$
$d_{6,4}^4=\min\{d_{6,4}^3+d_{4,4}^3 d_{6,4}^3\}=\min\{40+0;40\}=40$
$d_{6,5}^4=\min\{d_{6,4}^3+d_{4,5}^3 d_{6,5}^3\}=\min\{40+30;23\}=23$

$d_{6,6}^{4}=\min\{d_{6,4}^{3}+d_{4,6}^{3}d_{6,6}^{3}\}=\min\{40+12;0\}=0$

Представим матрицу D^4, включив в нее рассчитанные элементы.

$D^4=$	0	10	18	8	31	20
	10	0	16	9	21	21
	26	16	0	25	37	15
	7	9	25	0	30	12
	∞	∞	∞	∞	0	23
	41	31	15	40	23	0

На основании матрицы D^4, вычислим последовательно все элементы матрицы D^5. Для этого мы используем рекуррентное соотношение $d_{i,j}^{5}=\min\{ d_{i,5}^{4}+ d_{5,j}^{4}; d_{i,j}^{4}\}$.

$d_{1,1}^{5}=\min\{d_{1,5}^{4}+d_{5,1}^{4}d_{1,1}^{4}\}=\min\{31+\infty;0\}=0$

$d_{1,2}^{5}=\min\{d_{1,5}^{4}+d_{5,2}^{4}d_{1,2}^{4}\}=\min\{31+\infty;10\}=10$

$d_{1,3}^{5}=\min\{d_{1,5}^{4}+d_{5,3}^{4}d_{1,3}^{4}\}=\min\{31+\infty;18\}=18$

$d_{1,4}^{5}=\min\{d_{1,5}^{4}+d_{5,4}^{4}d_{1,4}^{4}\}=\min\{31+\infty;8\}=8$

$d_{1,5}^{5}=\min\{d_{1,5}^{4}+d_{5,5}^{4}d_{1,5}^{4}\}=\min\{31+0;31\}=31$

$d_{1,6}^{5}=\min\{d_{1,5}^{4}+d_{5,6}^{4}d_{1,6}^{4}\}=\min\{31+23;20\}=20$

$d_{2,1}^{5}=\min\{d_{2,5}^{4}+d_{5,1}^{4}d_{2,1}^{4}\}=\min\{21+\infty;10\}=10$

$d_{2,2}^{5}=\min\{d_{2,5}^{4}+d_{5,2}^{4}d_{2,2}^{4}\}=\min\{21+\infty;0\}=0$

$d_{2,3}^{5}=\min\{d_{2,5}^{4}+d_{5,3}^{4}d_{2,3}^{4}\}=\min\{21+\infty;16\}=16$

$d_{2,4}^{5}=\min\{d_{2,5}^{4}+d_{5,4}^{4}d_{2,4}^{4}\}=\min\{21+\infty;9\}=9$

$d_{2,5}^{5}=\min\{d_{2,5}^{4}+d_{5,5}^{4}d_{2,5}^{4}\}=\min\{21+0;21\}=21$

$d_{2,6}^{5}=\min\{d_{2,5}^{4}+d_{5,6}^{4}d_{2,6}^{4}\}=\min\{21+23;21\}=21$

$d_{3,1}^{5}=\min\{d_{3,5}^{4}+d_{5,1}^{4}d_{3,1}^{4}\}=\min\{37+\infty;26\}=26$

$d_{3,2}^{5}=\min\{d_{3,5}^{4}+d_{5,2}^{4}d_{3,2}^{4}\}=\min\{37+\infty;16\}=16$

$d_{3,3}^{5}=\min\{d_{3,5}^{4}+d_{5,3}^{4}d_{3,3}^{4}\}=\min\{37+\infty;0\}=0$

$d_{3,4}^{5}=\min\{d_{3,5}^{4}+d_{5,4}^{4}d_{3,4}^{4}\}=\min\{37+\infty;25\}=25$

$d_{3,5}^{5}=\min\{d_{3,5}^{4}+d_{5,5}^{4}d_{3,5}^{4}\}=\min\{37+0;37\}=37$

$d_{3,6}^{5}=\min\{d_{3,5}^{4}+d_{5,6}^{4}d_{3,6}^{4}\}=\min\{37+23;15\}=15$

$d_{4,1}^{5}=\min\{d_{4,5}^{4}+d_{5,1}^{4}d_{4,1}^{4}\}=\min\{30+\infty;7\}=7$

$d_{4,2}^{5}=\min\{d_{4,5}^{4}+d_{5,2}^{4}d_{4,2}^{4}\}=\min\{30+\infty;9\}=9$

$d_{4,3}^{5}=\min\{d_{4,5}^{4}+d_{5,3}^{4}d_{4,3}^{4}\}=\min\{30+\infty;25\}=25$

$d_{4,4}^{5}=\min\{d_{4,5}^{4}+d_{5,4}^{4}d_{4,4}^{4}\}=\min\{30+\infty;0\}=0$

$d_{4,5}^{5}=\min\{d_{4,5}^{4}+d_{5,5}^{4}d_{4,5}^{4}\}=\min\{30+0;30\}=30$

$d_{4,6}^{5}=\min\{d_{4,5}^{4}+d_{5,6}^{4}d_{4,6}^{4}\}=\min\{30+23;12\}=12$

$d_{5,1}^{5}=\min\{d_{5,5}^{4}+d_{5,1}^{4}d_{5,1}^{4}\}=\min\{0+\infty;\infty\}=\infty$

$d_{5,2}^{5}=\min\{d_{5,5}^{4}+d_{5,2}^{4}d_{5,2}^{4}\}=\min\{0+\infty;\infty\}=\infty$

$d_{5,3}^{5}=\min\{d_{5,5}^{4}+d_{5,3}^{4}d_{5,3}^{4}\}=\min\{0+\infty;\infty\}=\infty$

$d_{5,4}^{5}=\min\{d_{5,5}^{4}+d_{5,4}^{4}d_{5,4}^{4}\}=\min\{0+\infty;\infty\}=\infty$

$d_{5,5}^{5}=\min\{d_{5,5}^{4}+d_{5,5}^{4}d_{5,5}^{4}\}=\min\{0+0;0\}=0$

$d_{5,6}^{5}=\min\{d_{5,5}^{4}+d_{5,6}^{4}d_{5,6}^{4}\}=\min\{0+23;23\}=23$

$d_{6,1}^{5}=\min\{d_{6,5}^{4}+d_{5,1}^{4}d_{6,1}^{4}\}=\min\{23+\infty;41\}=41$

$d_{6,2}^{5}=\min\{d_{6,5}^{4}+d_{5,2}^{4}d_{6,2}^{4}\}=\min\{23+\infty;31\}=31$

$d_{6,3}^5=\min\{d_{6,5}^4+d_{5,3}^4 d_{6,3}^4\}=\min\{23+\infty;15\}=15$
$d_{6,4}^5=\min\{d_{6,5}^4+d_{5,4}^4 d_{6,4}^4\}=\min\{23+\infty;40\}=40$
$d_{6,5}^5=\min\{d_{6,5}^4+d_{5,5}^4 d_{6,5}^4\}=\min\{23+0;23\}=23$
$d_{6,6}^5=\min\{d_{6,5}^4+d_{5,6}^4 d_{6,6}^4\}=\min\{23+23;0\}=0$
Представим матрицу D^5, включив в нее рассчитанные элементы.

$D^5=$	0	10	18	8	31	20
	10	0	16	9	21	21
	26	16	0	25	37	15
	7	9	25	0	30	12
	∞	∞	∞	∞	0	23
	41	31	15	40	23	0

На основании матрицы D^5, вычислим последовательно все элементы матрицы D^6. Для этого мы используем рекуррентное соотношение $d_{i,j}^6=\min\{ d_{i,6}^5+ d_{6,j}^5; d_{i,j}^5\}$.
$d_{1,1}^6=\min\{d_{1,6}^5+d_{6,1}^5 d_{1,1}^5\}=\min\{20+41;0\}=0$
$d_{1,2}^6=\min\{d_{1,6}^5+d_{6,2}^5 d_{1,2}^5\}=\min\{20+31;10\}=10$
$d_{1,3}^6=\min\{d_{1,6}^5+d_{6,3}^5 d_{1,3}^5\}=\min\{20+15;18\}=18$
$d_{1,4}^6=\min\{d_{1,6}^5+d_{6,4}^5 d_{1,4}^5\}=\min\{20+40;8\}=8$
$d_{1,5}^6=\min\{d_{1,6}^5+d_{6,5}^5 d_{1,5}^5\}=\min\{20+23;31\}=31$
$d_{1,6}^6=\min\{d_{1,6}^5+d_{6,6}^5 d_{1,6}^5\}=\min\{20+0;20\}=20$
$d_{2,1}^6=\min\{d_{2,6}^5+d_{6,1}^5 d_{2,1}^5\}=\min\{21+41;10\}=10$
$d_{2,2}^6=\min\{d_{2,6}^5+d_{6,2}^5 d_{2,2}^5\}=\min\{21+31;0\}=0$
$d_{2,3}^6=\min\{d_{2,6}^5+d_{6,3}^5 d_{2,3}^5\}=\min\{21+15;16\}=16$
$d_{2,4}^6=\min\{d_{2,6}^5+d_{6,4}^5 d_{2,4}^5\}=\min\{21+40;9\}=9$
$d_{2,5}^6=\min\{d_{2,6}^5+d_{6,5}^5 d_{2,5}^5\}=\min\{21+23;21\}=21$
$d_{2,6}^6=\min\{d_{2,6}^5+d_{6,6}^5 d_{2,6}^5\}=\min\{21+0;21\}=21$
$d_{3,1}^6=\min\{d_{3,6}^5+d_{6,1}^5 d_{3,1}^5\}=\min\{15+41;26\}=26$
$d_{3,2}^6=\min\{d_{3,6}^5+d_{6,2}^5 d_{3,2}^5\}=\min\{15+31;16\}=16$
$d_{3,3}^6=\min\{d_{3,6}^5+d_{6,3}^5 d_{3,3}^5\}=\min\{15+15;0\}=0$
$d_{3,4}^6=\min\{d_{3,6}^5+d_{6,4}^5 d_{3,4}^5\}=\min\{15+40;25\}=25$
$d_{3,5}^6=\min\{d_{3,6}^5+d_{6,5}^5 d_{3,5}^5\}=\min\{15+23;37\}=37$
$d_{3,6}^6=\min\{d_{3,6}^5+d_{6,6}^5 d_{3,6}^5\}=\min\{15+0;15\}=15$
$d_{4,1}^6=\min\{d_{4,6}^5+d_{6,1}^5 d_{4,1}^5\}=\min\{12+41;7\}=7$
$d_{4,2}^6=\min\{d_{4,6}^5+d_{6,2}^5 d_{4,2}^5\}=\min\{12+31;9\}=9$
$d_{4,3}^6=\min\{d_{4,6}^5+d_{6,3}^5 d_{4,3}^5\}=\min\{12+15;25\}=25$
$d_{4,4}^6=\min\{d_{4,6}^5+d_{6,4}^5 d_{4,4}^5\}=\min\{12+40;0\}=0$
$d_{4,5}^6=\min\{d_{4,6}^5+d_{6,5}^5 d_{4,5}^5\}=\min\{12+23;30\}=30$
$d_{4,6}^6=\min\{d_{4,6}^5+d_{6,6}^5 d_{4,6}^5\}=\min\{12+0;12\}=12$
$d_{5,1}^6=\min\{d_{5,6}^5+d_{6,1}^5 d_{5,1}^5\}=\min\{23+41;\infty\}=64$
$d_{5,2}^6=\min\{d_{5,6}^5+d_{6,2}^5 d_{5,2}^5\}=\min\{23+31;\infty\}=54$
$d_{5,3}^6=\min\{d_{5,6}^5+d_{6,3}^5 d_{5,3}^5\}=\min\{23+15;\infty\}=38$
$d_{5,4}^6=\min\{d_{5,6}^5+d_{6,4}^5 d_{5,4}^5\}=\min\{23+40;\infty\}=63$
$d_{5,5}^6=\min\{d_{5,6}^5+d_{6,5}^5 d_{5,5}^5\}=\min\{23+23;0\}=0$

$d_{5,6}^{6}=\min\{d_{5,6}^{5}+d_{6,6}^{5}d_{5,6}^{5}\}=\min\{23+0;23\}=23$
$d_{6,1}^{6}=\min\{d_{6,6}^{5}+d_{6,1}^{5}d_{6,1}^{5}\}=\min\{0+41;41\}=41$
$d_{6,2}^{6}=\min\{d_{6,6}^{5}+d_{6,2}^{5}d_{6,2}^{5}\}=\min\{0+31;31\}=31$
$d_{6,3}^{6}=\min\{d_{6,6}^{5}+d_{6,3}^{5}d_{6,3}^{5}\}=\min\{0+15;15\}=15$
$d_{6,4}^{6}=\min\{d_{6,6}^{5}+d_{6,4}^{5}d_{6,4}^{5}\}=\min\{0+40;40\}=40$
$d_{6,5}^{6}=\min\{d_{6,6}^{5}+d_{6,5}^{5}d_{6,5}^{5}\}=\min\{0+23;23\}=23$
$d_{6,6}^{6}=\min\{d_{6,6}^{5}+d_{6,6}^{5}d_{6,6}^{5}\}=\min\{0+0;0\}=0$
Представим матрицу D^6, включив в нее рассчитанные элементы.

$D^6=$	0	10	18	8	31	20
	10	0	16	9	21	21
	26	16	0	25	37	15
	7	9	25	0	30	12
	64	54	38	63	0	23
	41	31	15	40	23	0

В результате, нами получена матрица длин кратчайших путей между каждой парой вершин графа. Ниже представлена **таблица путей**. Каждый элемент Cij таблицы, это путь из вершины i в вершину j:

Таблица путей						
i \ j	**1**	**2**	**3**	**4**	**5**	**6**
1	-	d_{1-2}=1-2	d_{1-3}=1-3	d_{1-4}=1-4	d_{1-5}=1-2-5	d_{1-6}=1-4-6
2	d_{2-1}=2-1	-	d_{2-3}=2-3	d_{2-4}=2-4	d_{2-5}=2-5	d_{2-6}=2-4-6
3	d_{3-1}=3-2-1	d_{3-2}=3-2	-	d_{3-4}=3-2-4	d_{3-5}=3-2-5	d_{3-6}=3-6
4	d_{4-1}=4-1	d_{4-2}=4-2	d_{4-3}=4-1-3	-	d_{4-5}=4-2-5	d_{4-6}=4-6
5	d_{5-1}=5-6-3-2-1	d_{5-2}=5-6-3-2	d_{5-3}=5-6-3	d_{5-4}=5-6-3-2-4	-	d_{5-6}=5-6
6	d_{6-1}=6-3-2-1	d_{6-2}=6-3-2	d_{6-3}=6-3	d_{6-4}=6-3-2-4	d_{6-5}=6-5	-

3 Задания для самостоятельной работы

1.. Методом Флойда определить кратчайшие пути между всеми парами вершин в нагруженном орграфе, заданном матрицей длин дуг.

$$\begin{bmatrix} 0 & 7 & 2 & 3 & - \\ - & 0 & - & 2 & - \\ 3 & - & 0 & 5 & 4 \\ - & - & 2 & 0 & 1 \\ - & 3 & 4 & - & 0 \end{bmatrix}$$

2. Методом Флойда определить кратчайшие пути между всеми парами вершин в нагруженном орграфе, заданном матрицей длин дуг.

$$\begin{bmatrix} 0 & 4 & - & - & 5 & 1 & - \\ 3 & 0 & 2 & 1 & - & - & - \\ 1 & 1 & 0 & - & - & - & 3 \\ - & 3 & 1 & 0 & 1 & - & - \\ - & - & 2 & - & 0 & 1 & 5 \\ - & 3 & - & 2 & 2 & 0 & - \\ - & - & 2 & - & - & 2 & 0 \end{bmatrix}$$

3. Методом Флойда определить кратчайшие пути между всеми парами вершин в нагруженном орграфе, заданном матрицей длин дуг.

$$\begin{bmatrix} 0 & 5 & - & 3 & 4 \\ 5 & 0 & 2 & - & 3 \\ - & 2 & 0 & 4 & 1 \\ 3 & - & 4 & 0 & 2 \\ 4 & 3 & 1 & 2 & 0 \end{bmatrix}$$

4.Составить программный модуль для решения задач методом Флойда.

Занятие 6 Практическое занятие
Одноканальная СМО с отказами

1 Теоретическая часть

В качестве показателей эффективности СМО с отказами будем рассматривать:

А — абсолютную пропускную способность СМО, т.е. среднее число заявок, обслуживаемых в единицу времени;

Q — относительную пропускную способность, т.е. среднюю долю пришедших заявок, обслуживаемых системой;

$P_{отк}$ *— вероятность отказа,* т.е. того, что заявка покинет СМО необслуженной;

к — среднее число занятых каналов (для многоканальной системы).

Одноканальная система с отказами. Рассмотрим задачу.

Имеется один канал, на который поступает поток заявок с интенсивностью *X*. Поток обслуживании имеет интенсивность ц[1]. Найти предельные вероятности состояний системы и показатели ее эффективности.

Система *S* (СМО) имеет два состояния: S_0 — канал свободен, S_1— канал занят. Размеченный граф состояний представлен на рис. 6.

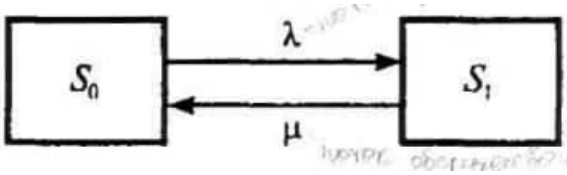


Рис. 6

В предельном, стационарном режиме система алгебраических уравнений для вероятностей состояний имеет вид

$$\begin{cases} \lambda p_0 = \mu p_1, \\ \mu p_1 = \lambda p_0, \end{cases} \qquad (1)$$

т.е. система вырождается в одно уравнение. Учитывая нормировочное условие $p_o + p_1 = 1$ найдем из (1) предельные вероятности состояний

$$p_0 = \frac{\mu}{\lambda + \mu}, \; p_1 = \frac{\lambda}{\lambda + \mu}, \qquad (2)$$

которые выражают среднее относительное время пребывания сие темы в состоянии S_0 (когда канал свободен) и S_1 (когда канал занят), т.е. определяют соответственно относительную пропускную способность *Q* системы и вероятность отказа $P_{тк}$:

$$Q = \frac{\mu}{\lambda + \mu}, \qquad (3)$$

$$P_{\text{отк.}} = \frac{\lambda}{\lambda + \mu}.$$

(4) Абсолютную пропускную способность найдем, умножив относительную пропускную способность Q на интенсивность потока отказов

$$A = \frac{\lambda\mu}{\lambda + \mu}. \qquad (5)$$

2 Практическая часть

Задача 1Известно, что заявки на телефонные переговоры в телевизионном ателье поступают с интенсивностью λ, равной 90 заявок в час, а средняя продолжительность разговора по телефону $t_{\text{об.}}$=2 мин. Определить показатели эффективности работы СМО (телефонной связи) при наличии одного телефонного номера.

Решение. Имеем λ=90 (1/ч), $t_{ОБС}$=*2* мин. Интенсивность потока обслуживании $\mu=1/t_{\text{обс}}==1/2=0{,}5$ (1/мин)=30 (1/ч). По (20) относительная пропускная способность СМО Q=30/(90+30)=0,25, т.е. в среднем только 25% поступающих заявок осуществят переговоры по телефону. Соответственно вероятность отказа в обслуживании составит $P_{отк}$=*0,75 (см.* (21)). Абсолютная пропускная способность СМО по (20) А=90*0,25=22,5, т.е. в среднем в час будут обслужены 22,5 заявки на переговоры. Очевидно, что при наличии только одного телефонного номера СМО будет плохо справляться с потоком заявок.

3 Задания для самостоятельной работы

1. Рассматривается круглосуточная работа пункта проведения профилактического осмотра автомашин с одним каналом (одной группой проведения осмотра). На осмотр и выявление дефектов каждой машины затрачивается в среднем 0,5 ч. На осмотр поступает в среднем 36 машин в сутки. Потоки заявок и обслуживании — простейшие. Если машина, прибывшая в пункт осмотра, не застает ни одного канала свободным, она покидает пункт осмотра необслуженной. Определить вероятности состояний и характеристики обслуживания профилактического пункта осмотра.

2. Анализируется работа междугородного переговорного пункта в небольшом городке. Пункт имеет один телефонный аппарат для переговоров. В среднем за сутки поступает 240 заявок на переговоры. Средняя длительность переговоров (с учетом вызова абонентов в другом городе) составляет 5 мин. Никаких ограничений на длину очереди нет. Потоки заявок и обслуживания простейшие. Определить предельные вероятности состояний и характеристики обслуживания переговорного пункта в стационарном режиме.

Занятие 7 Практическое занятие
Одноканальная СМО с ожиданиями

1 Теоретическая часть

СМО с ограниченным временем ожидания. На практике часто встречаются СМО с так называемыми "нетерпеливыми" заявками. Такие заявки могут уйти из очереди, если время ожидания превышает некоторую величину. В частности, такого рода заявки возникают в различных технологических системах, в которых задержка с началом обслуживания может привести к потере качества продукции, в системах оперативного управления, когда срочные сообщения теряют ценность (или даже смысл), если они не поступают на обслуживание в течение определенного времени.

2 Практическая часть

Задание 1. 2 В порту имеется один причал для разгрузки судов. Интенсивность потока судов равна 0,4 (судов в сутки). Среднее время разгрузки одного судна составляет 2 суток. Предполагается, что очередь может быть неограниченной длины. Найти показатели эффективности работы причала, а также вероятность того, что ожидают разгрузки не более чем 2 судна.

Решение. Имеем $\rho = \lambda/\mu = \lambda t_{обс} = 0{,}4*2 = 0{,}8$. Так как $\rho = 0{,}8 < 1$, то очередь на разгрузку не может бесконечно возрастать и предельные вероятности существуют. Найдем их.

Вероятность того, что причал свободен, по (33) p_o=1—0,8=0,2, а вероятность того, что он занят, $P_{зан}$=1—0,2=0,8. По формуле (34) вероятности того, что у причала находятся 1, 2, 3 судна (т.е. ожидают разгрузки 0, 1, 2 судна), равны p_1= 0,8(1-0,8) = 0,16; $p_2=0{,}8^2*(1-0{,}8)=0{,}128$; $p_3= 0{,}8^3(1-0{,}8) = 0{,}1024$.

Вероятность того, что ожидают разгрузку не более чем 2 судна, равна По формуле (40) среднее число судов, ожидающих разгрузки,

$$L_{оч.} = 0{,}8^2/(1-0{,}8) = 3{,}2,$$

а среднее время ожидания разгрузки по формуле

$$T_{оч.} = \frac{1}{\lambda} L_{оч.}$$

Среднее число судов, находящихся у причала, $L_{сист.}$=0,8/(1-0,8)= 4 (сутки) ($L_{сист}$ = 3,2+0,8 = 4 (сутки), а среднее время пребывания судна у причала по формуле (41) $L_{сист}$ = 4/0,8 = 5 (сутки).

Очевидно, что эффективность разгрузки судов невысокая. Для ее повышения необходимо уменьшение среднего времени, разгрузки судна $t_{обс}$ либо увеличение числа причалов n.

Найти наибольший делитель $d=(a,b)$ и наименьшее общее кратное $m=[a,b]$ целых чисел a,b с помощью алгоритма Евклида, а также целые числа u,v, удовлетворяющие условию $au + bv = d$.

3 Задания для самостоятельной работы

1. В торговом павильоне покупателей обслуживает один продавец. Площадь павильона составляет 24 м^2, причем 10 м^2 приходится на торговый зал, вместимость которого ограничена. Поэтому если очередь на обслуживание составляет 10 человек, то потенциальный покупатель туда не входит, что свидетельствует об отказе в обслуживании и как следствие, снижении товарооборота и ухудшению "показателей работы коммерческого предприятия. Дать оценку системы массового обслуживания и определить рекомендации по созданию оптимального режима работы, если интенсивность прихода покупателей составляет 120 человек в час. а среднее время обслуживания одного покупателя 3 мин.
2. Мини-маркет с одним контролером-кассиром обслуживает покупателей, входящий поток которых подчиняется закону Пуассона с параметром 20 покупателей в час. Время обслуживания подчиняется показательному закону с параметром 25 покупателей в час. Определить: вероятность простоя контролера-кассира, среднюю длину очереди. среднее число покупателей в мини-маркете. среднее время ожидания обслуживания, среднее время пребывания покупателей в мини-маркете и дать оценку работы системы.

Занятие 8. Практическое занятие
Многоканальная СМО с ожиданиями

1 Теоретическая часть

Рассмотрим задачу. Имеется n-канальная СМО с неограниченной очередью. Поток заявок, поступающих в СМО, имеет интенсивностьλ, а поток обслуживании — интенсивность μ. Необходимо найти предельные вероятности состояний СМО и показатели ее эффективности.

Система может находиться в одном из состояний $S_0, S_1, S_2, ..., S_k ..., S_n, ...$, нумеруемых по числу заявок, находящихся в СМО: S_o — в системе нет заявок (все каналы свободны); S_1— занят один канал, остальные свободны; S_2 — заняты два канала, остальные свободны; ..., S_k — занято k каналов, остальные свободны; ..., S_n — заняты все n каналов (очереди нет); S_{n+1} — заняты все n каналов, в очереди одна заявка; ..., S_{n+r} — заняты все n каналов, r заявок стоит в очереди,....

Граф состояний системы показан на рис. 9. Обратим внимание на то, что в отличие от предыдущей СМО, интенсивность потока обслуживании (переводящего систему из одного состояния в другое справа налево) не остается постоянной, а по мере увеличения числа заявок в СМО от 0 до n увеличивается от величины μ до nμ, так как соответственно увеличивается число каналов обслуживания. При числе заявок в СМО большем, чем я, интенсивность потока обслуживании сохраняется равной $n\mu$

Можно показать, что при p/n < 1 предельные вероятности существуют.

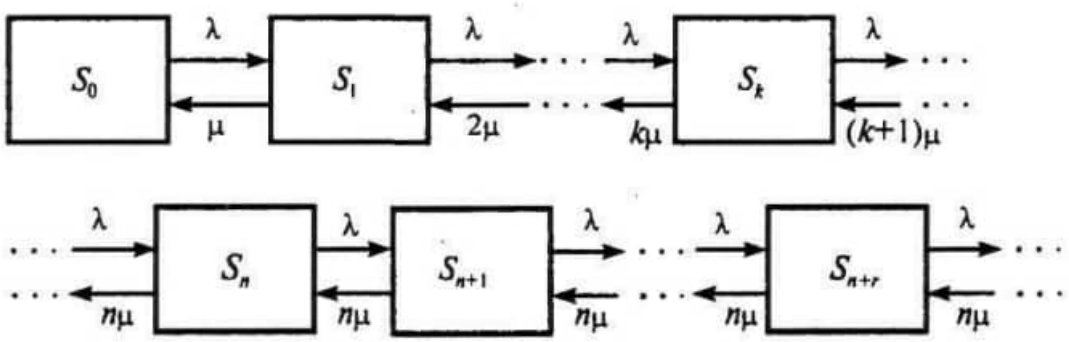


Рис.1

Если p/n≥1, очередь растет до бесконечности. Используя формулы (2) и (3) для процесса гибели и размножения, можно получить следующие формулы для предельных вероятностей состояний n-канальной СМО с неограниченной очередью

$$p_0 = \left(1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \ldots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n!(n-\rho)}\right)^{-1}, \qquad \textbf{(6)}$$

$$p_1 = \frac{\rho}{1!}p_0, \ldots, p_k = \frac{\rho^k}{k!}p_0, \ldots, p_n = \frac{\rho^n}{n!}\rho_0, \qquad \textbf{(7)}$$

$$p_{n+1} = \frac{\rho^{n+1}}{n \cdot n!}p_0, \ldots, p_{n+r} = \frac{\rho^{n+r}}{n^r \cdot n!}p_0, \ldots . \qquad \textbf{(8)}$$

Вероятность того, что заявка окажется в очереди,

$$P_{оч.} = \frac{\rho^{n+1}}{n!(n-\rho)}p_0 . \qquad \textbf{(9)}$$

Для n-канальной СМО с неограниченной очередью, используя прежние приемы, можно найти: среднее число занятых каналов

$$\bar{k} = \frac{\lambda}{\mu} = \rho, \qquad \textbf{(10)}$$

среднее число заявок в очереди

$$L_{оч.} = \frac{\rho^{n+1} p_0}{n \cdot n! \left(1 - \frac{\rho}{n}\right)^2}, \qquad \textbf{(11)}$$

среднее число заявок в системе

$$L_{сист.} = L_{оч.} + \rho.$$

2 Практическая часть

Задача 1В универсаме к узлу расчета поступает поток покупателей с интенсивностью λ=81чел. в час. Средняя продолжительность обслуживания контролером-кассиром одного покупателя $t_{об.}$=2 мин. Определить:

а. Минимальное количество контролеров-кассиров n_{min}, при котором очередь не будет расти до бесконечности, и соответствующие характеристики обслуживания при $n=n_{min}$

б. Оптимальное количество $n_{опт}$ контролеров-кассиров, при котором относительная величина затрат $C_{оти}$, связанная с издержками на содержание каналов обслуживания и с пребыванием в очереди покупателей, задаваемая, например, как $C_{оти} = \frac{1}{\lambda} n + 3t_{оч.}$ будет минимальна, и сравнить характеристики обслуживания при $n=n_{min}$ и $n=n_{опт}$.

в. Вероятность того, что в очереди будет не более трех покупателей.

Решение. а. По условию λ=81(1/ч)=81/60=1,35(1/мин.). $\rho = \lambda/\mu = \lambda t_{об}$=1,35*2 =2,7. Очередь не будет возрастать до бесконечности при условии $\rho/n < 1$, т.е. при $n > \rho = 2{,}7$. Таким образом, минимальное количество контролеров-кассиров $n_{min} = 3$.

Найдем характеристики обслуживания СМО при $n = 3$.

Вероятность того, что в узле расчета отсутствуют покупатели, по формуле (6) $p_0=(1+2{,}7+2{,}7^2/2!+2{,}7^3/3!+2{,}7^4/3!(3-2{,}7)^{-1}$ =0,025, т.е. в среднем 2,5% времени контролеры-кассиры будут простаивать.

по (9)

$$P_{оч.} = (2{,}7^4/3!(3-2{,}7))0{,}025 = 0{,}735.$$

Среднее число покупателей, находящихся в очереди, по (11)

$$L_{оч.} = (2{,}7^4/3 \cdot 3!(1-2{,}7/3)^2)0{,}025 = 7{,}35.$$

Среднее время ожидания в очереди

$$T_{оч.} = 7{,}35/1{,}35 = 5{,}44 \text{ (мин)}.$$

и

Среднее число покупателей в узле расчета

$$L_{сист.} = 7{,}35+2{,}7 = 10{,}05.$$

Среднее время нахождения покупателей в узле расчета

$$T_{сист.} = 10{,}05/1{,}35 = 7{,}44 \text{ (мин)}.$$

Среднее число контролеров-кассиров, занятых обслуживанием покупателей, по (10) $к = 2{,}7$.

Коэффициент (доля) занятых обслуживанием контролеров-кассиров

$$\bar{k} = \rho/n = 2{,}7/3 = 0{,}9.$$

Абсолютная пропускная способность узла расчета $A = 1{,}35$ (1/мин), или 81 (1/ч), т.е. 81 покупатель в час.

Анализ характеристик обслуживания свидетельствует о значительной перегрузке узла расчета при наличии трех контролеров-кассиров.

б. Относительная величина затрат при $n = 3$

$$C_{отн.} = \frac{1}{\lambda} n + 3T_{оч.} = 3/1{,}35+3\cdot 5{,}44 = 18{,}54.$$

Рассчитаем относительную величину затрат при других значениях n (табл. 2).

Таблица 2

Характеристика обслуживания	Число контролеров-кассиров				
	3	4	5	6	7
Вероятность простоя контролеров-кассиров p_0	0,025	0,057	0,065	0,067	0,067
Среднее число покупателей в очереди $T_{оч.}$	5,44	0,60	0,15	0,03	0,01
Относительная величина затрат $C_{отн.}$	18,54	4,77	4,14	4,53	5,22

Как видно из табл. 2, минимальные затраты получены при $n = n_{опт}=5$ контролерах-кассирах.

Определим характеристики обслуживания узла расчета при $n= n_{опт}=5$. Получим $P_{оч} = 0.091$; $L_{оч}= 0{,}198$; $T_{оч}.=0{,}146$ (мин); $L_{сист.} = 2{,}90$; $T_{сист.} = 2{,}15$ (мин); $к = 2{,}7$; $k_З = 0{,}54$.

Как видим, при $n =5$ по сравнению с n= 3 существенно уменьшились вероятность возникновения очереди $P_{оч}$, длина очереди $L_{оч}$ и среднее время пребывания в очереди $T_{оч}$ и соответственно среднее число покупателей $L_{сцст}$ и среднее время нахождения в узле расчета $T_{сист.}$, а также доля занятых обслуживанием контролеров $k_З$ Но среднее число занятых

обслуживанием контролеров-кассиров $к$ и абсолютная пропускная способность узла расчета A естественно не изменились.

в. Вероятность того, что в очереди будет не более 3 покупателей, определится как

(Заметим, что в случае n=3 контролеров-кассиров та же вероятность существенно меньше: $P(r\leq 3)=0{,}464$)

Задача 2. Железнодорожная касса с двумя окошками продает билеты в два пункта A и B. Интенсивность потока пассажиров, желающих купить билеты, для обоих пунктов одинакова: $\lambda_A = \lambda_B = 0{,}45$ (пассажиров в минуту). На обслуживание пассажиров кассир тратит в среднем 2 мин. Рассматриваются два варианта продажи билетов: первый — билеты продаются в одной кассе с двумя окошками одновременно в оба пункта A и B второй — билеты продаются в двух специализированных кассах (по одному окошку в каждой), одна только в пункт A, другая — только в пункт B. Необходимо:

а. Сравнить два варианта продажи билетов по основным характеристикам обслуживания.

б. Определить, как надо изменить среднее время обслуживания одного пассажира, чтобы по второму варианту продажи пассажиры затрачивали на приобретение билетов в среднем меньше времени, чем по первому варианту.

Решение, а**. По первому варианту** имеем двухканальную СМО, на которую поступает поток заявок интенсивностью $\lambda = 0{,}45+0{,}45 = 0{,}9$; интенсивность потока обслуживании $\mu = 1/2 = 0{,}5$; $p/n = 1.8/2 = 1{,}8$. Так как $p/n = 1{,}8/2 = 0{,}9 < 1$, то предельные вероятности существуют.

Вероятность простоя двух кассиров по (6)

$$p_0 = \left(1 + \frac{1{,}8}{1!} + \frac{1{,}8^2}{2!} + \frac{1{,}8^3}{2!(2-1{,}8)}\right)^{-1} = 0{,}0526.$$

Среднее число пассажиров в очереди по (15.50)

$$L_{оч.} = 1{,}8^3/2\cdot 2!(1-1{,}8/2)\cdot 0{,}0526 = 7{,}67.$$

Среднее число пассажиров у кассы по (15.51)

$$L_{сист.} = 7{,}67+1{,}8 = 9{,}47.$$

Среднее время на ожидание в очереди и покупку билетов равно соответственно: $T_{оч.} = 7{,}67/0{,}9 = 8{,}52$ (мин) и $T_{сист.} = 9{,}47/0{,}9 = 10{,}5$ (мин).

По **второму варианту** имеем две одноканальные СМО (два специализированных окошка); на каждую поступает поток заявок с интенсивностью $\lambda = 0{,}45$. По-прежнему $\mu = 0{,}5$; $p = \lambda/\mu = 0{,}9 < 1$, предельные вероятности существуют.

$$L_{оч.} = 0{,}9^2/(1-0{,}9) = 8{,}1;\ L_{сист.} = 0{,}9/(1-0{,}9) = 9{,}0;$$

$$T_{оч.} = 8{,}1/0{,}45 = 18{,}0\ (мин),\ T_{сист.} = 9{,}0/0{,}45 = 20{,}0\ (мин).$$

Итак, по второму варианту увеличились и длина очереди, и среднее время ожидания в ней и в целом на покупку билетов. Такое различие объясняется тем, что в первом варианте (двухканальная СМО) меньше средняя доля времени, которую простаивает каждый из двух кассиров: если он не занят обслуживанием пассажира, покупающего билет в пункт *A,* он может заняться обслуживанием пассажира, покупающего билет в пункт *B,* и наоборот. Во втором варианте такой взаимозаменяемости нет.

Можно заметить, что среднее время на покупку билетов по второму варианту увеличилось более чем в 2 раза. Такое значительное увеличение связано с тем, что СМО работает на пределе своих возможностей (р = 0,9): достаточно незначительно увеличить среднее время обслуживания $T_{об}$, т.е. уменьшить μ, и р превзойдет 1, т.е. очередь начнет неограниченно возрастать.

б. Выше было получено, что по первому варианту продажи билетов при среднем времени обслуживания одного пассажира $t_{об.} = 2$ (мин) среднее время на покупку билетов составит $T_{сист1} = 10{,}5$ (мин). По условию для второго варианта продажи $T_{сист2} < T_{сист1}$ или $\frac{1}{\lambda}\frac{\rho}{1-\rho} < T_{сист1.}$ Полагая $\rho=\lambda/\mu=\lambda_{tоб.}$ Получим $-\frac{t_{об.}}{1-\lambda t_{об.}} < T_{сист1,}$ откуда найдем

$$t_{об.} < \frac{T_{сист\ 1}}{1+\lambda T_{сист\ 1}} \text{ или } t_{об.} < \frac{10{,}5}{1+0{,}45*10{,}5} = 1{,}83\ ((мин))$$

Итак, средние затраты времени на покупку билетов по второму варианту продажи уменьшатся, если среднее время обслуживания одного пассажира уменьшится более чем на 0,17 мин, или более чем на 8,5%

Занятие 9. Практическое занятие
Применение метода Монте-Карло
1 Теоретическая часть

Во многих задачах исходные данные носят случайный характер, поэтому для их решения должен применяться статистико-вероятностный подход. На основе таких подходов построен ряд численных методов, которые учитывают случайный характер вычисляемых или измеряемых величин. К ним принадлежит метод статистических испытаний, называемый также методом Монте -Карло, который применяется к решению некоторых задач вычислительной математики, в том числе вычисления интегралов.

Рассмотрим идею применения метода Монте-Карло на примере вычисления определённого интеграла от функции, зависящей от одной переменной. Пусть необходимо вычислить интеграл от некоторой заданной функции *f(x)* на интервале [*a*,*b*]. Рассмотрим прямоугольник высотой *H* и длинной (*b*-*a*), такой, что функция *f(x)* целиком лежит внутри данного прямоугольника (1).

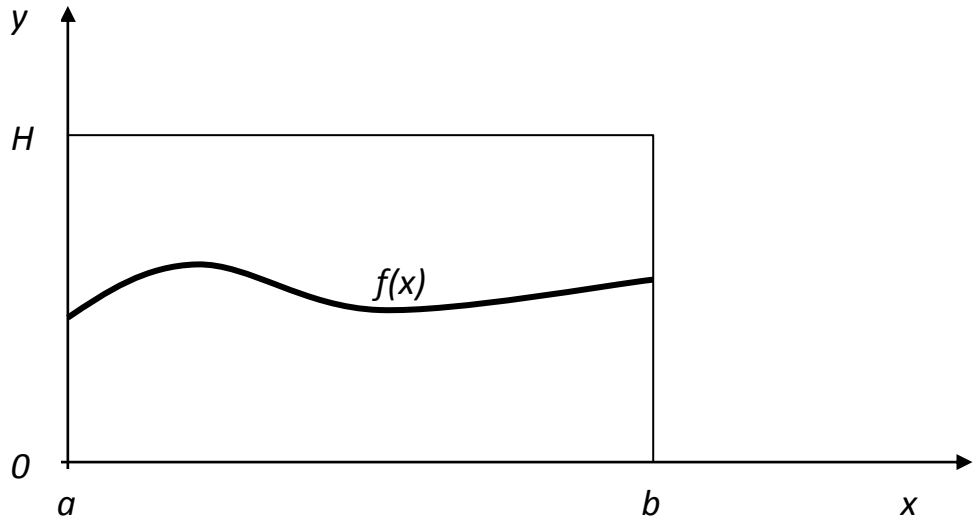


Рис. 1 График функции *f(x)*.

Сгенерируем *N* пар случайных чисел (x_i, y_i), $i=1,2,\ldots,N$, равномерно распределённых в данном прямоугольнике: $a \le x_i \le b$, $0 \le y_i \le H$. Тогда доля точек (x_i, y_i), удовлетворяющих условию $y_i \le f(x_i)$, является оценкой отношения интеграла от функции *f(x)* к площади *S* рассматриваемого прямоугольника. Следовательно, оценка интеграла в данном методе может быть получена по формуле:

$$I_N = S \frac{n_S}{N}, \qquad (1)$$

где

S - площадь прямоугольника;

n_S - количество точек, удовлетворяющих условию $y_i \le f(x_i)$;

N - полное количество точек.

Можно использовать другой способ вычисления определённого интеграла, рассматривая его как среднее значение функции $f(x)$ на отрезке $[a,b]$:

$$I_N = (b - a)\frac{1}{N}\sum_{i=1}^{N} f(x_i), \qquad (2)$$

где

x_i - последовательность случайных чисел с равномерным законом распределения на отрезке $[a,b]$.

Метод Монте-Карло также может быть использован и для вычисления кратных интегралов.

2 Практическая часть

Задача: Методом Монте-Карло вычислить интеграл

$$I = \iint_G (x^2 + y^2)dxdy$$

где область G определятся следующими неравенствами

$1/2 \le x \le 1 \quad 0 \le y \le 1$

Решение

Область интегрирования принадлежит единичному квадрату $0 \le x \le 1 \quad 0 \le y \le 1$

Для вычисления интеграла воспользуемся таблицей случайных чисел при этом каждые два последовательных числа из этой таблицы примем за координаты случайной точки M(x,y)

Таблица - Случайные числа, равномерно распределенные на отрезке [0.1]

0,57705	0,35483	0,11578	0,65339
0,71618	0,09393	0,93045	0,93382
0,73710	0,30304	0,93011	0,05758
0,70131	0,55186	0,42844	0,00336
0,16961	0,64003	0,52906	0,88222
0,53324	0,20514	0,09461	0,98585
0,43166	0,00188	0,99602	0,52103
0,26275	0,55709	0,69962	0,91827
0,05926	0,86977	0,31311	0,07069
0,66289	0,31303	0,27004	0,13928

Записываем координаты x и y случайных точек в таблицу 2 округляя до трех знаков после запятой, и выбираем те из них, которые принадлежат области интегрирования.

П о р я д о к з а п о л н е н и я таблицы 2

1) Среди всех значений x выделяем те, которые заключены между $x = 0.5$ è $x = 1$. Для этих значений полагаем $\varepsilon_1=1$, для всех остальных $\varepsilon_1=0$. Например, для x=0,577 имеем $\varepsilon_1=1$, для x=0,170 имеем $\varepsilon_1=0$.

2) Среди всех значений y, соответствующих выделенным x, выбираем те, которые заключены между $\underline{y(x)} = 0$ è $\overline{y(x)} = 2x - 1$. Для этих значений полагаем

$\varepsilon_2=1$, для всех остальных $\varepsilon_2=0$. Например, для н=0,701 имеем $\varepsilon_2=0$, для x=0,205 имеем $\varepsilon_2=1$

x	$\underline{x}$	$\overline{x}$	ε_1	y	$\underline{y(x)}$	$\overline{y(x)}$	ε_2	ε	f(x,y)
0.577	0.500	1.000	1	0.716	0	0.154	0	0	
0.737	0.500	1.000	1	0.701	0	0.474	0	0	
0.170	0.500	1.000	0	0.533			0	0	
0.432	0.500	1.000	0	0.263			0	0	
0.059	0.500	1.000	0	0.663			0	0	
0.355	0.500	1.000	0	0.094			0	0	
0.303	0.500	1.000	0	0.552			0	0	
0.640	0.500	1.000	1	0.205	0	0.280	1	1	0.452
0.002	0.500	1.000	0	0.557			0	0	
0.870	0.500	1.000	1	0.323	0	0.740	1	1	0.855
0.116	0.500	1.000	0	0.930			0	0	
0.930	0.500	1.000	1	0.428	0	0.860	1	1	1.048
0.529	0.500	1.000	1	0.095	0	0.058	0	0	
0.996	0.500	1.000	1	0.700	0	0.992	1	1	1.482
0.313	0.500	1.000	0	0.270			0	0	
0.653	0.500	1.000	1	0.934	0	0.306	0	0	
0.058	0.500	1.000	0	0.003			0	0	
0.882	0.500	1.000	1	0.986	0	0.764	0	0	
0.521	0.500	1.000	1	0.918	0	0.042	0	0	
0.071	0.500	1.000	0	0.139			0	0	
							Суммы	4	3,837

.

1) Вычисляем $\varepsilon=\varepsilon_1*\varepsilon_2$. Области интегрирования принадлежат только те точки, для которых ε=1. Например для точки M(0,640б0,205) имеем ε=1.
2) Вычисляем значение подынтегральной функции в полученных точках. После заполнения табл.2 вычисляем области интегрирования и по формуле (3) находим

$$I \approx \frac{1}{4} * \frac{1}{4}(0.452 + 0.855 + 1.048 + 1.482) = \frac{1}{16} * 3.837 = 0.240$$

Точное значение интеграла I=0.21875. Результат имеет сравнительно небольшую точность потому, что N=20 не достаточно велико.

$$V_G = \frac{1}{2} * 1 * \frac{1}{2} = \frac{1}{4}$$

3 Задания для самостоятельной работы

Вычислить с указанной точностью ε несобственные интегралы методом усечения

1. $\int_1^{\infty} \frac{arctgx}{1+x^2}dx$ $\varepsilon=10^{-2}$ 2. $\int_1^{\infty} \frac{xe^{-x^2}}{2+\sin x}dx$ $\varepsilon=10^{-2}$

Занятие 10. Практическое занятие
Использование алгоритмов псевдогенераторов случайных чисел

1 Теоретическая часть

Генераторы ПСЧ могут работать по разным алгоритмам. Одним из простейших генераторов является так называемый *линейный конгруэнтный генератор*, который для вычисления очередного числа k_i использует формулу:

$$k_i = (a \times k_{i-1} + b) \bmod c,$$

где *a*, *b*, *c* — некоторые константы, а k_{i-1} — предыдущее псевдослучайное число. Для получения k_1 задается начальное значение k_0.

Достоинством линейных конгруэнтных генераторов ПСЧ является их простота и высокая скорость получения псевдослучайных значений. Линейные конгруэнтные генераторы находят применение при решении задач моделирования и математической статистики, однако в криптографических целях их нельзя рекомендовать к использованию, так как специалисты по криптоанализу научились восстанавливать всю последовательность ПСЧ по нескольким значениям. Например, предположим, что противник может определить значения k_0, k_1 , k_2, k_3. Тогда:

$$k_1 = (a \times k_0 + b) \bmod c,$$

$$k_2 = (a \times k_1 + b) \bmod c$$

$$k_3 = (a \times k_2 + b) \bmod c$$

Решив систему из этих трех уравнений, можно найти *a*, *b* и *c* .

Для получения псевдослучайных чисел предлагалось использовать также квадратичные и кубические генераторы:

$$k_i = (a_1^2 \times k_{i-1} + a_2 \times k_{i-1} + b) \bmod c$$

$$k_i = (a_1^3 \times k_{i-1} + a_2^2 \times k_{i-1} + a_3 \times k_{i-1} + b) \bmod c$$

Однако такие генераторы тоже оказались непригодными для целей криптографии по той же самой причине «предсказуемости».

2 Практическая часть

Возьмем в качестве примера *a*=5,*b*=3,*c*=11 и пусть k_0 = 1. В этом случае мы сможем по приведенной выше формуле получать значения от 0 до 10 (так как *c* = 11). Вычислим несколько элементов последовательности:

k_1 = (5 * 1 + 3) mod 11 = 8;
k_2 = (5 * 8 + 3) mod 11 = 10;
k_3 = (5 * 10 + 3) mod 11 = 9;

$k_4 = (5 * 9 + 3) \bmod 11 = 4;$

$k_5 = (5 * 4 + 3) \bmod 11 = 1.$

Полученные значения (8, 10, 9, 4, 1) выглядят похожими на случайные числа. Однако следующее значение k_6 будет снова равно 8:

$k_6 = (5 * 1 + 3) \bmod 11 = 8.$

А значения k_7 и k_8 будут равны 10 и 9 соответственно:

$k_7 = (5 * 8 + 3) \bmod 11 = 10;$

$k_8 = (5 * 10 + 3) \bmod 11 = 9.$

Выходит, наш генератор псевдослучайных чисел повторяется, порождая периодически числа 8, 10, 9, 4, 1. К сожалению, это свойство характерно для всех линейных конгруэнтных генераторов. Изменяя значения основных параметров *a*, *b* и *c*, можно влиять на длину периода и на сами порождаемые значения k_i. Так, например, увеличение числа *c* в общем случае ведет к увеличению периода. Если параметры *a*, *b* и *c* выбраны правильно, то генератор будет порождать случайные числа с максимальным периодом, равным *c*. При программной реализации значение *c* обычно устанавливается равным 2^{b-1} или 2^b, где *b* — длина слова ЭВМ в битах.

3 Задания для самостоятельной работы

1 Определите последовательность из первых десяти чисел и период линейного конгруэнтного генератора ПСЧ для различных параметров *a*, *b* и *c*:

а) *a* =11, *b* =7 и *c* = 16; б) *a* =21, *b* =19 и *c* = 32;

в) *a* =19, *b* =7 и *c* = 32; г) *a* = 11, *b* =3 и *c* = 32.

2 Составить программный модуль для линейного конгруэнтного генератора ПСЧ .

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.02. ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ

основной профессиональной образовательной программы среднего профессионального образования – программы подготовки специалистов среднего звена

по специальности
09.02.07 Информационные системы и программирование

квалификация
Программист

Тула 2023

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от «13» января 2023 г. № 6
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 десятиление 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).

4 Методические указания по выполнению внеаудиторных самостоятельных работ по МДК 02.02 Инструментальные средства разработки программного обеспечения

Методические указания по выполнению внеаудиторных самостоятельных работ по МДК 02.02 Инструментальные средства разработки программного обеспечения разработаны в соответствии с рабочей программой профессионального модуля и предназначены для приобретения необходимых практических навыков и закрепления теоретических знаний, полученных обучающимися при изучении профессионального модуля, обобщения и систематизации знаний перед промежуточной аттестацией.

Методические указания предназначены для обучающихся специальности 09.02.07 «Информационные системы и программирование».

Рабочая программа профессионального модуля предусматривает проведение практических работ в объеме 14 часов.

Порядок выполнения работы

- записать название работы, ее цель в тетрадь;
- выполнить основные задания в соответствии с ходом работы;
- выполнить индивидуальные задания.

Самостоятельная работа №1
«СОВРЕМЕННЫЕ СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Цель работы: изучить современные средства разработки программного обеспечения.

Теоретические сведения

Средства разработки программного обеспечения – совокупность приемов, методов, методик, а также набор инструментальных программ (компиляторы, прикладные/системные библиотеки и т.д.), используемых разработчиком для создания программного кода Программы, отвечающего заданным требованиям.

Разработка программ – сложный процесс, основной целью которого является создание, сопровождение программного кода, обеспечивающего необходимый уровень надежности и качества. Для достижения основной цели разработки программ используются средства разработки программного обеспечения.

ОСНОВНЫЕ СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ НА РАЗНЫХ ЭТАПАХ РАЗРАБОТКИ ПРОГРАММ

В зависимости от предметной области и задач, поставленных перед разработчиками, разработка программ может представлять собой достаточно сложный, поэтапный процесс, в котором задействовано большое количество участников и разнообразных средств. Для того, чтобы определить, когда и в каких случаях какие средства применяются, выделяют следующие основные этапы разработки программного обеспечения:

1. Проектирование приложения.
2. Реализация программного кода приложения.
3. Тестирование приложения.

Средства проектирования приложений

На этапе проектирования приложения в зависимости от сложности разрабатываемого

программного продукта, напрямую зависящего от предъявляемых требований, выполняются следующие задачи проектирования:

1. Анализ требований.
2. Разработка архитектуры будущего программного обеспечения.
3. Разработка устройств основных компонент программного обеспечения.
4. Разработка макетов Пользовательских интерфейсов.

Результатом проектирования обычно является «Эскизный проект» (Software Design Document) или «Технический проект» (Software Architecture Document).

Задача «Анализ требований» обычно выполняется с использованием методов системологии (анализа и синтеза) с учетом экспертного опыта проектировщика. Результатом анализа обычно является содержательная или формализованная модель процесса функционирования программы. В зависимости от сложности процесса для построения данных моделей могут быть применены различные методы и вспомогательные средства. В общем случае для описания моделей обычно применяются следующие нотации (в скобках приведены программные средства, которые могут быть использованы для получения моделей):

- BPMN (Vision + BPMN, AcuaLogic BPMN, Eclipse, Sybase Power Designer).
- Блок-схемы (Vision и многие другие).
- ER-диаграмы (Visio, ERWin, Sybase Power Designer и многие другие).
- UML-диаграмы (Sybase Power Designer, Rational Rose и многие другие).
- макеты, мат-модели и т.д.

Результаты анализа позволяют сформировать обоснованные требования к той или иной функциональности разрабатываемой программы и просчитать реальную выгоду от внедрения разрабатываемого продукта. Более того, иного получается так, что по результатам анализа первоначальные цели и задачи автоматизации кардинально меняются или по результатам оценки эффективности разработки и внедрения принимается решение продукт не разрабатывать.

Целью второй и третьей задачи из приведенного списка задач является разработка модели (описания) будущей системы, понятной для кодировщика – человека, который пишет код программы. Здесь огромное значение имеет то, какую парадигму программирования необходимо использовать при написании программы. В качестве примера основных парадигм необходимо привести следующее:

- Функциональное программирование;
- Структурное программирование;
- Императивное программирование;
- Логическое программирование;
- Объектно-ориентированное программирование (прототипирование; использование классов; субъективно-ориентированное программирование).

Выбор её во многом зависит от сложившихся привычек, опыта, традиций, инструментальных средств, которыми располагает коллектив разработчиков. Иногда разрабатываемый программный продукт настолько сложен, что для решения ряда задач в разных компонентах системы используются разные парадигмы. Выбор того или иного подхода накладывает ограничения на средства, которые будут применены на этапе реализации программного кода. Результатом решения данной задачи в зависимости от подхода могут быть (в скобках приведены программные средства, которые могут быть использованы для их получения):

- диаграмма классов и т.д (Ration Rose, Sybase PowerDisigner и многие другие);
- описание модулей структур и их программного интерфейса (например, Sybase PowerDisigner и многие другие).

Разработка макетов пользовательских интерфейсов подразумевает создание наглядного представления того, как будут выглядеть те или иные видеоформы, окна в разрабатываемом приложении. Решение данной задачи основывается на применение средств дизайнера.

Средства реализации программного кода

На этапе реализации программного кода выполняется кодирование отдельных компонент программы в соответствии с разработанным техническим проектом. Средства, которые могут быть применены, в значительной степени зависит от того, какие подходы были использованы во время проектирования и, кроме этого, от степени проработанности технического проекта. Тем не менее, среди средств разработки программного кода необходимо выделить следующие основные виды средств:

- методы и методики алгоритмирования.
- языки программирования (C++,Си, Java, C#, php и многие другие);
- средства создания пользовательского интерфейса (MFC, WPF, QT, GTK+ и т.д.)
- средства управления версиями программного кода (cvs, svn, VSS).
- средства получения исполняемого кода (MS Visual Studio, gcc и многие другие).
- средства управления базами данных (Oracle, MS SQL, FireBird, MySQL и многие
- другие).
- отладчики (MS Visual Studio, gdb и т.д.).

ЗАДАНИЕ НА РАБОТУ

Изучить интерфейс, функционал средств проектирования и разработки программного обеспечения: Microsoft Visual Studio 2019 Community, NetBeans 12.0 сборка Java SE (для Java), Anaconda 5.3 For Windows Python, PyCharm Community Edition 2020 (для Python), MS Visio, Git (GitHub) и другие.

Самостоятельная работа №2
«СОВРЕМЕННЫЕ СРЕДСТВА ПОДДЕРЖКИ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЧЕНИЯ»

Цель работы: изучить современные средства поддержки тестирования программного обеспечения.

Теоретические сведения

Основными задачами тестирования является проверка соответствия функциональности разработанной программы первоначальным требованиям, а также выявление ошибок, которые в явном или неявном виде проявляются во время работы программы. Среди основных работ по тестированию можно выделить следующее:

- Тестирование на отказ и восстановление.
- Функциональное тестирование.
- Тестирование безопасности.
- Тестирование взаимодействия.
- Тестирование процесса установки.
- Тестирование удобства пользования.
- Конфигурационное тестирование.
- Нагрузочное тестирование.

Наличие правильных процессов, инструментов и технических решений для автоматических тестирований API становится важным, как никогда ранее. И с помощью тенденции shift-left, тестирование API становится больше, чем просто решение по контролю за качеством, теперь это критически важный компонент успешной непрерывной интеграции и развёртывания программного обеспечения.

Среди основных видов средств, которые могут быть применены для выполнения поставленных работ можно привести следующие:

- средства анализа кода, профилирования (Code Wizard – ParaSoft, Purify – Rational Softawre. Test Coverage – Semantic и т.д.);
- средства для тестирования функциональности (TEST – Parasoft, QACenter –Compuware, Borland SilkTestи т.д.);
- средства тестирования API (SoapUI, Postman, Katalon Studio, Tricentis Tosca, Apigee, JMeter, Rest-Assured, Assertible, Karate DSL)
- средства для тестирования производительности (QACenter Performance –Compuware и т.д).

Некоторые могут посчитать, что свойств коммерческих продуктов (Postman, Tricentis Tosca,…) будет достаточно, однако цена вопроса будет служить серьезным сдерживающим фактором. Бесплатные и общедоступные решения (Rest-Assured, Karate DSL,…) являются довольно-таки приемлемыми, но требуют квалифицированных умений и много усилий для имплементации правильной платформы. Инструменты, которые удерживают баланс между ценой и другими факторами (Katalon Studio, Postman), могут иметь недостатки для некоторых типов проектов, и эти недостатки требуют пристальной оценки.

Тестирование API создало свой собственный тренд в области автоматического тестирования, и чем дальше, тем больше инструментов будет создаваться для удовлетворения растущих запросов от команд разработки программного обеспечения.

ЗАДАНИЕ НА РАБОТУ

Изучить интерфейс, функционал средств поддержки тестирования программного обеспечения: Karate DSL, Katalon Studio, JMeter, Rest-Assured, Postman и др.

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__