МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ДЛЯ РЕАЛИЗАЦИИ ПРОГРАММНОГО МОДУЛЯ
ПМ.02 ОСУЩЕСТВЛЕНИЕ РЕАЛИЗАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ
ДЛЯ СПЕЦИАЛЬНОСТИ
09.02.03 ПРОГРАММИРОВАНИЕ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

Базовая подготовка
среднего профессионального образования

2024

СОДЕРЖАНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические рекомендации по выполнению практических работ по программному модулю ПМ.02 Осуществление реализации программных модулей составлены в соответствии с требованиями ФГОС СПО к минимуму содержания и уровню подготовки выпускников СПО по специальности 09.02.07 «Информационные системы и программирование» и на основе рабочей программы модуля.

Методические рекомендации включают практические занятия по четырем модулям:

МДК.02.01 Технология разработки программного обеспечения

МДК.02.02 Средства разработки программного обеспечения

МДК.02.03 Математическое моделирование

МДК.02.04 Web-программирование/Основы интеллектуального труда

Требования к оформлению практических работ

Студент должен выполнить практическую работу в соответствии с полученным заданием. Каждый студент после выполнения работы должен представить отчет о проделанной работе с анализом полученных результатов и выводом по работе.

Отчет о проделанной работе следует выполнять на отдельных листах в клетку формата А4, которые хранятся в отдельных папках. Содержание отчета указано в описании практической работы.

Если студент не выполнил практическую работу или часть работы, то он может выполнить работу или оставшуюся часть во внеурочное время, согласованное с преподавателем.

Оценку по практической работе студент получает, с учетом срока выполнения работы, если:

- работа выполнена правильно и в полном объеме;
- сделан анализ проделанной работы и вывод по результатам работы;
- студент может пояснить выполнение любого этапа работы;
- отчет выполнен в соответствии с требованиями к выполнению работы.

Зачет по практическим работам студент получает при условии выполнения всех предусмотренных программой работ, после сдачи отчетов по работам при получении удовлетворительных отметок.

Критерии оценки

Ответ оценивается отметкой «5», если:

работа выполнена полностью;

в логических рассуждениях и обосновании решения нет пробелов и ошибок;

в решении нет математических ошибок (возможны некоторые неточности, описки, которая не является следствием незнания или непонимания учебного материала).

Отметка «4» ставится в следующих случаях:

работа выполнена полностью, но обоснования шагов решения недостаточны (если умение обосновывать рассуждения не являлось специальным объектом проверки);

допущены одна ошибка, или есть два – три недочёта в выкладках, рисунках, чертежах или графиках (если эти виды работ не являлись специальным объектом проверки).

Отметка «3» ставится, если:

допущено не более двух ошибок или более двух – трех недочетов в выкладках, чертежах или графиках, но обучающийся обладает обязательными умениями по проверяемой теме.

Отметка «2» ставится, если:

допущены существенные ошибки, показавшие, что обучающийся не обладает обязательными умениями по данной теме в полной мере.

Преподаватель может повысить отметку за оригинальный ответ на вопрос или оригинальное решение задачи, которые свидетельствуют о высоком математическом развитии обучающегося; за решение более сложной задачи или ответ на более сложный вопрос, предложенные обучающемуся дополнительно после выполнения им каких-либо других заданий.

Практические занятия по модулям

МДК 01.01 Технология разработки программного обеспечения

Практическое занятие 1

Тема: «Анализ предметной области. Разработка и оформление технического задания»

Цель: Освоить процесс анализа предметной области при проектировании программного обеспечения. Ознакомление с процедурой разработки технического задания на создание программного продукта с применением ГОСТ 19.102-77 «Стадии разработки программ и программной документации».

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Предметная область - это целенаправленная первичная трансформация картины внешнего мира в некоторую умозрительную картину, определенная часть которой фиксируется в информационной системе (ИС) в качестве алгоритмической модели фрагмента действительности.

Совокупность реалий (объектов) внешнего мира - объектов, о которых можно задавать вопросы, - образует объектное ядро предметной области, которое имеет онтологический статус. Нельзя получить в ИС ответ на вопрос о том, что ей неизвестно. Термин объект является первичным, неопределяемым понятием. Синонимами термина "объект" являются "реалия, сущность, вещь". Однако термин сущность понимается нами несколько уже, как компонент модели предметной области, т.е. как уже выделенный на концептуальном уровне объект. Таким образом, выделяемые в предметной области объекты превращаются аналитиками в сущности. Сущность предметной области является результатом абстрагирования реального объекта путем выделения и фиксации набора его свойств. Хотя сущность нередко отождествляется с объектом.

Одним из ключевых моментов создания ПО является всестороннее изучение объектов предметной области, их свойств, взаимоотношений между этими объектами и представление полученной информации в виде информационной модели данных.

Информационная модель данных предназначена для представления семантики предметной области в терминах субъективных средств описания - сущностей, атрибутов, идентификаторов сущностей, супертипов, подтипов и т.д.

Информационная модель предметной области содержит следующие основные конструкции:

- ✓ диаграммы "сущность-связь" (Entity - Relationship Diagrams);
- ✓ определения сущностей;
- ✓ уникальные идентификаторы сущностей;
- ✓ определения атрибутов сущностей;
- ✓ отношения между сущностями;
- ✓ супертипы и подтипы.

Элементы информационной модели данных предметной области являются входными данными для решения задачи проектирования ПО - создания логической модели данных.

Вторым ключевым моментом создания ПО является анализ функционального взаимодействия объектов предметной области. Результатом такого анализа является функциональная модель предметной области. Функциональная модель предметной области является собирательным понятием. Состав функциональной модели существенно зависит от контекста конкретного проекта и может быть представлен посредством довольно широкого спектра документов в виде текстовой и графической информации.

Функциональная модель в виде иерархии функций способствует пониманию поведения субъекта моделирования.

В соответствии с методологией структурного анализа в первую очередь строится контекстная диаграмма – самое общее описание главной функции системы в целом и ее

взаимодействия с внешней средой. Последующая функциональная декомпозиция сопровождается построением диаграмм декомпозиции, которые описывают каждый фрагмент декомпозиции и их взаимодействие. Детализация функциональной модели продолжается до достижения необходимой степени подробности.

Информационная и функциональная модели предметной области могут быть представлены с помощью диаграмм UML:

- вариантов использования (use case diagram);
- классов (class diagram);
- кооперации (collaboration diagram);
- последовательности (sequence diagram);
- состояний (statechart diagram);
- деятельности (activity diagram);
- компонентов (component diagram);
- развертывания (deployment diagram).

В качестве примера можно привести диаграмму классов предметной области компьютерной обучающей системы:

Рис. 1 Диаграмма классов обучающей системы.

На данной стадии выполняются следующие работы:

1. Обоснование необходимости разработки программ:

- постановка задачи;
- сбор исходных материалов;
- выбор и обоснование критериев эффективности и качества;
- обоснование необходимости проведения научно-исследовательских работ.

2. Выполнение научно-исследовательских работ:

- определение структуры входных и выходных данных;
- предварительный выбор методов решения задач;
- обоснование целесообразности применения ранее разработанных программ;
- определение требований к техническим средствам;
- обоснование принципиальной возможности решения поставленных задач.

3. Разработка и утверждение технического задания:

- определение требований к программе;
- разработка технико-экономического обоснования разработки программы;
- определение стадий, этапов и сроков разработки программы и документации на нее;
- выбор языков программирования;
- определение необходимости проведения научно-исследовательской работы на последующих стадиях.

Ход работы

Задание.

1. Построить диаграмму классов предметной области задачи.
2. Построить диаграмму вариантов использования предметной области задачи.
3. Составить отчет по практической работе.

Задача: составить список учебной группы, включающей 25 человек. Для каждого учащегося указать дату рождения, год поступления в колледж, курс, группу, оценки каждого года обучения.

Назначение задачи: получить значение определённого критерия и упорядочить список студентов по нему.

Достигаемая цель: упорядочить список студентов по среднему баллу и получить его.

Задание.

1. Ознакомиться с ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению».
2. Ознакомиться с примерами оформления технического задания.
3. Проанализировать предложенные примеры технических заданий на соответствие ГОСТ 19.201-78.
4. Результаты анализа оформить в виде таблицы.

Отчет по практической работе должен включать:

1. Анализ предметной области задачи.
2. Таблицу анализа предметной области

Список используемой литературы

1. Рудаков А. Технология разработки программных продуктов: учебник. изд.Academia. Среднее профессиональное образование. 2020 г.

Практическое занятие 2

Тема: «Построение диаграммы Вариантов использования и диаграммы Последовательности»

Цель: ознакомиться с методологией моделирования информационных систем на основе языка UML.

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Задание № 1. Ознакомиться с методологией построения диаграммы вариантов использования основе языка UML.

Задание № 2. Проанализируйте пример построения диаграммы вариантов использования. Пример. Магазин видеопродукции Магазин продает видеокассеты, DVD-диски, аудиокассеты, CD-диски и т.д., а также предлагает широкой публике прокат видеокассет и DVD-дисков. Товары поставляются несколькими поставщиками. Каждая партия товара предварительно заказывается магазином у некоторого поставщика и доставляется после оплаты счета. Вновь поступивший товар маркируется, заносится в базу данных и затем распределяется в торговый зал или прокат. Видеоносители выдаются в прокат на срок от 1 до 7 дней. При прокате с клиента взимается залоговая стоимость видеоносителя. При возврате видеоносителя возвращается залоговая стоимость минус сумма за прокат. Если возврат задержан менее чем на 2 дня, взимается штраф в размере суммы за прокат за 1 день* кол-во дней задержки. При задержке возврата более чем на 2 дня – залоговая сумма не возвращается. Клиент может взять одновременно до 4 видеоносителей (прокат-заказ). На каждый видеоноситель оформляется квитанция. Клиенты могут стать членами видео-клуба и получить пластиковые карточки. С членов клуба не берется залог (за исключением случая описанного ниже), устанавливается скидка на ставку проката и покупку товаров. Члены клуба могут делать предварительные заказы на подбор видеоматериалов для проката или покупки. Каждый член клуба имеет некоторый стутус. Первоначально – "новичок". При возврате всрок 5 прокат-заказов, статус меняется на "надежный". При задержке хотя бы одного видеоносителя более чем на 2 дня, статус "новичок" или "надежный" меняется на "ненадежный" и клиенту высылается предупреждение. При повторном нарушении правил статус меняется на "нарушитель". Члены клуба со статусом "надежный" могут брать до 8 видеоносителей единовременно, все остальные

– 4. С членов клуба со статусом "нарушитель" берется залоговая сумма. Клиенты при покупке

товара или получении видеоносителя в прокат могут расплачиваться наличными или кредитной картой. Прокатные видеоносители через определенное количество дней проката списываются и утилизируются по акту. Списываются также товары и прокатные видеоносители, у которых обнаружился брак. На рисунке 1 приведена диаграмма прецедентов для рассматриваемого примера. В этом примере можно выделить следующие субъекты и соответствующие им прецеденты:

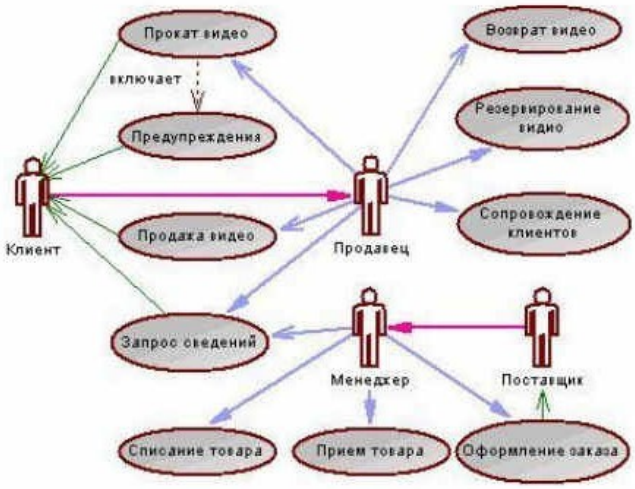


Рисунок 1

Менеджер изучает рынок видеопродукции, анализирует продажи (прецедент "Запрос сведений"), работает с поставщиками: составляет заявки на поставки товара (прецедент "Оформление заказа"), оплачивает и принимает товар (прецедент "Прием товара"), списывает товар (прецедент "Списание товара"). Продавец – работает с клиентами: продает товар (прецедент "Продажа видео"), оформляет членство в клубе (прецедент "Сопровождение клиентов"), резервирует (прецедент "Резервирование видио"), выдает в прокат (прецедент "Прокат видео") и принимает назад видеоносители (прецедент "Возврат видео"), отвечает на вопросы клиента (прецедент "Запрос сведений"). Поставщик – оформляет документы для оплаты товара (прецедент "Оформление заказа"), поставляет товар (прецедент "Прием товара")) Клиент – покупает(прецедент "Продажа видео"), берет на прокат и возвращает видеоносители (прецеденты "Прокат видео" и "Возврат видео"), вступает в клуб (прецедент "Сопровождение клиентов"), задает вопросы (прецедент "Запрос сведений"). Последние два субъекта Поставщик и Клиент не будут иметь непосредственного доступа к разрабатываемой системе (второстепенные субъекты), однако именно они являются основным источником событий, инициализирующих прецеденты, и получателями результата работы прецедентов От прецедента "Прокат видео" к прецеденту "Предупреждения" установлено отношение включения на том основании, что каждый выданный видеоноситель должен быть проверен на своевременный возврат и, в случае необходимости, выдано предупреждение клиенту. Дальнейшее развитие модели поведения системы предполагает спецификацию прецедентов. Для этого традиционно используют два способа. Первый – описание с помощью текстового документа. Такой документ описывает, что должна делать система, когда субъект инициировал прецедент. Типичное описание содержит следующие разделы: – краткое описание;

– участвующие субъекты; – предусловия, необходимые для инициирования прецедента; – поток событий (основной и, возможно, подпотоки, альтернативный); – постусловия, определяющие состояние системы, по достижении которого прецедент завершается. Описательная спецификация прецедента "Прокат видео"

Раздел	Описание
Краткое описание	Клиент желает взять на прокат видеокассету или диск, которые снимаются с полки магазина или были предварительно зарезервированы клиентом. При условии, что у клиента нет невозвращенных в срок видеоносителей, сразу после внесения платы фильм выдается напрокат. Если невозвращенные в срок видеоносители есть, клиенту выдается напоминание о просроченном возврате
Субъекты	Продавец, Клиент
Предусловия	В наличие имеются видеокассеты или диски, которые можно взять напрокат. У клиентов есть клубные карточки. Устройство сканирования работает правильно. Работники за прилавком знают, как обращаться с системой
Основной поток	Клиент может назвать номер заказа или взять видеоноситель с полки. Видеоноситель и членская карточка сканируются, и продавцу не сообщается никаких сведений о задержках, так, что он не задает клиенту соответствующих вопросов. Если клиент имеет статус <надежный>, он может взять до 8 видеоносителей, во всех остальных случаях – до 4-х. Если статус клиента определен как <нарушитель>, его просят внести задаток. Клиент расплачивается наличными или кредитной картой. После получения суммы, информация о наличии фильмов обновляется и видеоносители передаются клиенту вместе с квитанциями на прокат. О прокате каждого видеоносителя делается отдельная запись с указанием идентификационного номера клиента, даты проката, даты возврата, идентификационного номера продавца, полученной суммы. Прецедент генерирует предупреждения о просроченном возврате клиенту, если видеофильм не был возвращен в течение двух дней по истечении даты возврата и изменяет статус клиента на <ненадежный> (первое нарушение) или <нарушитель> (повторное нарушение)
Альтернативный поток	У клиента нет членской карточки. В этом случае прецедент <Сопровождение клиента> может быть активизирован для выдачи новой карточки. Видеофильмы не выдаются, поскольку у клиента есть невозвращенные в срок видеоносители. Попытка взять напрокат слишком много видеоносителей. Видеоноситель или кредитная карта не могут быть отсканированы из-за их повреждения У клиента не хватило наличных или платеж по кредитной карте отклонен
Постусловия	Видеофильмы сданы напрокат, и база данных соответствующим образом обновлена

Список используемой литературы

1. Рудаков А. Технология разработки программных продуктов: учебник. изд.Academia. Среднее профессиональное образование. 2020 г.

Практическое занятие 3

Тема: «Построение диаграммы Кооперации и диаграммы Развертывания»

Цель: Закрепление теоретических сведений о диаграмме кооперации и диаграмме Развертывания; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме кооперации и диаграмме развертывания.

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Диаграмма кооперации (collaboration diagram) предназначена для описания поведения системы на уровне отдельных объектов, которые обмениваются между собой сообщениями, чтобы достичь нужной цели или реализовать некоторый вариант использования.

Кооперация (collaboration) — спецификация множества объектов отдельных классов, совместно взаимодействующих с целью реализации отдельных вариантов использования в общем контексте моделируемой системы.

Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

На диаграмме кооперации размещаются объекты, представляющие собой экземпляры классов, связи между ними, которые в свою очередь являются экземплярами ассоциаций, и сообщения. Связи дополняются стрелками сообщений, а также именами ролей, которые играют объекты в данной взаимосвязи. На диаграмме кооперации показываются структурные отношения между объектами в виде различных соединительных линий и изображаются динамические взаимосвязи — потоки сообщений в форме стрелок с указанием направления рядом с соединительными линиями между объектами, при этом задаются имена сообщений и их порядковые номера в общей последовательности сообщений.

Одна и та же совокупность объектов может участвовать в реализации различных кооперациий. В зависимости от рассматриваемой кооперации, могут изменяться как связи между отдельными объектами, так и поток сообщений между ними. Именно это отличает диаграмму кооперации от диаграммы классов, на которой должны быть указаны все без исключения классы, их атрибуты и операции, а также все ассоциации и другие структурные отношения между элементами модели.

Объект (object) — сущность с хорошо определенными границами и индивидуальностью, которая инкапсулирует состояние и поведение. Объект создается на этапе реализации модели или выполнения программы. Он имеет собственное имя и конкретные значения атрибутов.

Для диаграмм кооперации имя объекта – строка текста, разделенная двоеточием: <собственное имя объекта>'/'<Имя роли класса>:<Имя класса>.

Имя роли класса указывается в том случае, когда соответствующий класс отсутствует в модели. Имя класса – это имя одного из классов, представленного на диаграмме классов.

Если указано собственное имя объекта, то оно должно начинаться со строчной буквы. Имя объекта, имя роли с символом "/" или имя класса могут отсутствовать, но ":" всегда должно стоять перед именем класса, а "/" – перед именем роли.

Следующие варианты возможных записей полного имени объекта:

- o : C – объект с собственным именем o, экземпляр класса C.
- : C – анонимный объект, экземпляр класса C.
- o : (или o) – объект-сирота с собственным именем o.
- o / R : C – объект с собственным именем o, экземпляр класса C, играющий роль R.
- / R : C – анонимный объект, экземпляр класса C, играющий роль R.
- o / R – объект-сирота с собственным именем o, играющий роль R.
- / R –анонимный объект и одновременно объект-сирота, играющий роль R.

Составной объект (composite object) или объект-композит предназначен для представления объекта, имеющего собственную структуру и внутренние потоки управления.

Составной объект является экземпляром класса-композита, который связан отношением композиции со своими частями. На диаграммах кооперации составной объект изображается как обычный объект, состоящий из двух секций: верхней и нижней. В верхней секции записывается имя составного объекта, а в нижней – его объекты-части вместо списка атрибутов.

При изображении диаграммы кооперации отношения между объектами описываются с помощью связей, которые являются экземплярами соответствующих ассоциаций.

При изображении диаграммы кооперации отношения между объектами описываются с помощью связей, которые являются экземплярами соответствующих ассоциаций.

Связь (link) — любое семантическое отношение между некоторой совокупностью объектов.

Бинарная связь изображается отрезком сплошной линии, соединяющей два прямоугольника объектов, на концах линии можно указать имена ролей.

Связи на диаграмме кооперации могут быть только анонимными и при необходимости записываются без двоеточия перед именем ассоциации. Имена связей и кратность концевых точек, как правило, на диаграммах кооперации не указываются. Обозначения агрегации и композиции могут присутствовать на отдельных концах связей.

Пример: обобщенная схема компании с именем «с», которая состоит из департаментов (анонимный мультиобъект класса «Департамент»). В последние входят «Сотрудники». Рефлексивная связь указывает на то, что руководитель департамента является одновременно и его сотрудником.

Связь может иметь некоторые стереотипы:

- «association» – ассоциация (предполагается по умолчанию, поэтому может не указываться);
- «parameter» – соответствующий объект может быть только параметром метода;
- «local» – область видимости переменной ограничена только соседним объектом;
- «global» – область видимости переменной распространяется на всю диаграмму кооперации;
- «self» – рефлексивная связь объекта с самим собой, которая допускает передачу объектом сообщения самому себе.

Каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой.

Сообщение (message) — спецификация передачи информации от одного элемента модели к другому с ожиданием выполнения определенных действий со стороны принимающего элемента.

При этом первый объект предполагает, что после получения сообщения вторым объектом последует выполнение некоторого действия. На диаграмме кооперации сообщение является причиной или стимулом начала выполнения операций, отправки сигналов, создания и уничтожения отдельных объектов. Связь обеспечивает канал для направленной передачи сообщений между объектами от объекта-источника к объекту-получателю.

Иногда отправителя сообщения называют клиентом, а получателя – сервером. При этом сообщение от клиента имеет форму запроса некоторого сервиса, а реакция сервера на запрос после получения сообщения может быть связана с выполнением определенных действий или передачи клиенту необходимой информации тоже в форме сообщения.

Сообщения в языке UML специфицируют роли, которые играют объекты – отправитель и получатель сообщения. Сообщения на диаграмме кооперации изображаются дополнительными стрелками рядом с соответствующей связью или ролью ассоциации. Направление стрелки указывает на получателя сообщения. На диаграммах кооперации может использоваться один из трех типов стрелок для обозначения сообщений.

Диаграмма кооперации, с одной стороны, обеспечивает концептуально согласованный переход от статической модели диаграммы классов к динамическим моделям поведения, представляемым диаграммами последовательности, состояний и деятельности. С другой стороны, диаграмма этого типа предопределяет особенности реализации модели на диаграммах компонентов и развертывания.

Пример: Программное средство представляет среду для формирования отчетов по лекционному материалу. Пользователь может вводить свою информацию в отчеты, изменять параметры. После оформления отчетов пользователю предоставлена возможность сохранение отчета.

Для построения диаграммы развертывания необходимо:

- изучить на какой платформе и на каких вычислительных средствах реализована ИС;
- рассмотреть возможность отображения физических устройств, которые будут участвовать в работе проектируемой ИС;
- Научиться выявлять узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Ход работы

Задание.

Изучить теоретические сведения по теме “Диаграмма кооперации” и «Диаграмма развертывания».

Разработать диаграмму кооперации и диаграмму развертывания для произвольной системы индивидуального задания.

Оформить отчет, включив в него описание всех компонентов диаграммы кооперации и диаграммы развертывания согласно индивидуальному варианту задания.

Список используемой литературы

1. Рудаков А. Технология разработки программных продуктов: учебник. изд.Academia. Среднее профессиональное образование. 2020 г.

Практическое занятие 4

Тема: «Построение диаграммы Деятельности, диаграммы Состояний и диаграммы Классов»

Цель: закрепление теоретических сведений о диаграмме состояний и диаграмме классов; овладение практическими навыками моделирования процессов, описывающих взаимодействие объектов в диаграмме состояний и диаграмме классов. Ознакомление с методологией и инструментальными средствами моделирования классов на основе языка UML.

Оснащение: ПК, учебная и справочная литература.

а. Теоретические сведения

При моделировании поведения системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Компонентами диаграммы деятельности являются:

-состояния действия,
-переходы,
-дорожки,
-объекты

Состояние действия. Состояние действия (action state) является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Этот переход неявно предполагает, что входное действие уже завершилось. Состояние действия не может иметь внутренних переходов, поскольку оно является элементарным. Обычное использование состояния действия заключается в моделировании одного шага выполнения алгоритма (процедуры) или потока управления.

Внутри фигуры записывается выражение действия (action-expression), которое должно быть уникальным в пределах одной диаграммы деятельности.

Переходы. При построении диаграммы деятельности используются только нетриггерные переходы, т. е. такие, которые срабатывают сразу после завершения деятельности или

выполнения соответствующего действия. На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действия выходит не единственный переход, то сработать может только один из них. Тогда для каждого из таких переходов должно быть явно записано сторожевое условие в форме булевского выражения в прямых скобках.

Дорожки. Диаграммы деятельности могут быть т.к. использованы для моделирования бизнес-процессов. Применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение несет ответственность за реализацию отдельных действий, а сам бизнес-процесс представляется в виде переходов действий из одного подразделения к другому.

Для моделирования этих особенностей в языке UML используется специальная конструкция, получившее название дорожки (swimlanes). При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии и образуют дорожку, а группа состояний между этими линиями выполняется отдельным подразделением.

Названия подразделений явно указываются в верхней части дорожки. Пересекать линию дорожки могут только переходы, которые в этом случае обозначают выход или вход потока управления в соответствующее подразделение компании.

Объекты. В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий.

Для графического представления объектов используются прямоугольник класса, с тем отличием, что имя объекта подчеркивается. Далее после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой.

Диаграмма классов представляет собой логическую модель статического представления моделируемой системы. Характеристика состояний системы не зависит (или слабо зависит) от логической структуры, зафиксированной в диаграмме классов. Поэтому при рассмотрении состояний системы приходится на время отвлечься от особенностей ее объектной структуры и мыслить совершенно другими категориями, образующими динамический контекст поведения моделируемой системы.

Каждая прикладная система характеризуется не только структурой составляющих ее элементов, но и некоторым поведением или функциональностью. Для общего представления функциональности моделируемой системы предназначены диаграммы вариантов использования, которые на концептуальном уровне описывают поведение системы в целом. Сейчас наша задача заключается в том, чтобы представить поведение более детально на логическом уровне.

Для моделирования поведения на логическом уровне в языке UML могут использоваться сразу несколько канонических диаграмм: состояний, деятельности, последовательности и кооперации, каждая из которых фиксирует внимание на отдельном аспекте функционирования системы. Диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее - одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Именно для описания реакции объекта на подобные внешние воздействия и используются диаграммы состояний.

Главное предназначение этой диаграммы - описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла.

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга, образуая вложенные диаграммы более детального представления отдельных элементов модели. Компонентами диаграммы состояний являются:

-состояния и подсостояния,

-переходы.

Автомат (state machine) в языке UML представляет собой некоторый формализм для моделирования поведения элементов модели и системы в целом. Автомат является пакетом, в котором определено множество понятий, необходимых для представления поведения моделируемой сущности в виде дискретного пространства с конечным числом состояний и переходов. С другой стороны, автомат описывает поведение отдельного объекта в форме последовательности состояний, которые охватывают все этапы его жизненного цикла, начиная от создания объекта и заканчивая его уничтожением. Каждая диаграмма состояний представляет некоторый автомат.

Простейшим примером визуального представления состояний и переходов на основе формализма автоматов может служить ситуация с исправностью технического устройства, такого как компьютер. В этом случае вводятся в рассмотрение два самых общих состояния: "исправен" и "неисправен" и два перехода: "выход из строя" и "ремонт".

Основными понятиями, входящими в формализм автомата, являются состояние и переход. Главное различие между ними заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что переход объекта из состояния в состояние происходит мгновенно.

Поведение моделируется как последовательное перемещение по графу состояний от вершины к вершине по связывающим их дугам с учетом их ориентации. Для графа состояний системы можно ввести в рассмотрение специальные свойства.

Одним из таких свойств является выделение из всей совокупности состояний двух специальных: начального и конечного. Предполагается, что последовательность изменения состояний упорядочена во времени. Другими словами, каждое последующее состояние всегда наступает позже предшествующего ему состояния.

Еще одним свойством графа состояний может служить достижимость состояний. Для достижимости состояний необходимо наличие связывающего их ориентированного пути в графе состояний.

Формализм обычного автомата основан на выполнении следующих обязательных условий:

1. Автомат не запоминает историю перемещения из состояния в состояние.

В каждый момент времени автомат может находиться в одном и только в одном из своих состояний.

Длительность нахождения автомата в том или ином состоянии, а также время достижения того или иного состояния никак не специфицируются, т.е. время на диаграмме состояний присутствует в неявном виде, хотя для отдельных событий может быть указан интервал времени и в явном виде.

Количество состояний автомата должно быть обязательно конечным (начальное и конечное состояния).

Каждый переход должен обязательно соединять два состояния автомата. Допускается переход из состояния в себя, такой переход еще называют "петлей".

Автомат не должен содержать конфликтующих переходов, т. е. таких переходов из одного и того же состояния, когда объект одновременно может перейти в два и более последующих состояния (кроме случая параллельных подавтоматов).

Состояние. В языке UML под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

Имя состояния представляет собой строку текста, которая раскрывает содержательный смысл данного состояния. Имя всегда записывается с заглавной буквы. Для идентификации имени состояния рекомендуется использовать глаголы в настоящем времени (звенит, печатает, ожидает) или соответствующие причастия (занят, свободен, передано, получено). Имя у состояния может отсутствовать. В этом случае состояние является анонимным, и если на диаграмме состояний их несколько, то все они должны различаться между собой.

Список внутренних действий. Рассмотрим перечень внутренних действий или деятельностей, которые выполняются в процессе нахождения моделируемого элемента в данном состоянии. Каждое из действий записывается в виде отдельной строки и имеет следующий формат:

<метка-действия '/' выражение-действия>

Метка действия указывает на обстоятельства или условия, при которых будет выполняться деятельность, определенная выражением действия. Если список выражений действия пустой, то разделитель в виде наклонной черты '/' может не указываться.

Перечень меток действия имеет фиксированные значения в языке UML, которые не могут быть использованы в качестве имен событий. Эти значения следующие:

entry - эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент входа в данное состояние (входное действие);

exit - эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент выхода из данного состояния (выходное действие);

do - эта метка специфицирует выполняющуюся деятельность ("do activity"), которая выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится вычисление, специфицированное следующим за ней выражением действия.

В последнем случае при завершении события генерируется соответствующий результат;

include - эта метка используется для обращения к подавтомату, при этом следующее за ней выражение действия содержит имя этого подавтомата.

Во всех остальных случаях метка действия идентифицирует событие, которое запускает соответствующее выражение действия. Эти события называются внутренними переходами и семантически эквивалентны переходам в само это состояние.

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий. В этом состоянии находится объект по умолчанию в начальный момент времени. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, из которого может только выходить стрелка, соответствующая переходу.

Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий. В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени. Графически конечное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность, в которую может только входить стрелка, соответствующая переходу.

Простой переход (simple transition) представляет собой отношение между двумя последовательными состояниями (исходном и целевом состоянии), которые указывают на факт смены одного состояния другим.

Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности (do activity), получении объектом сообщения или приемом сигнала. На переходе указывается имя события, и могут указываться действия, производимые объектом в ответ на внешние события при переходе из одного состояния в другое. Срабатывание перехода может зависеть не только от наступления некоторого события, но и от выполнения определенного условия, называемого сторожевым условием. Объект перейдет из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение "истина".

Переход может быть направлен в то же состояние, из которого он выходит. В этом случае его называют переходом в себя. Этот переход изображается петлей со стрелкой и отличается от внутреннего перехода. При переходе в себя объект покидает исходное состояние, а затем снова входит в него. При этом всякий раз выполняются внутренние действия, специфицированные метками entry и exit.

На диаграмме состояний переход изображается сплошной линией со стрелкой, которая направлена в целевое состояние. Каждый переход может быть помечен строкой текста, которая имеет следующий общий формат:

<сигнатура события>'['<сторожевое условие>']' <выражение действия>.

При этом сигнатура события описывает некоторое событие с необходимыми аргументами:

<имя события>'('<список параметров, разделенных запятыми>')'.

Событие. Формально, событие представляет собой спецификацию некоторого факта, имеющего место в пространстве и во времени. Про события говорят, что они "происходят", при этом отдельные события должны быть упорядочены во времени. После наступления некоторого события нельзя уже вернуться к предыдущим событиям, если такая возможность не предусмотрена явно в модели.

В языке UML события играют роль стимулов, которые инициируют переходы из одних состояний в другие. В качестве событий можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий. Имя события идентифицирует каждый отдельный переход на диаграмме состояний и

может содержать строку текста, начинающуюся со строчной буквы. В этом случае принято считать переход триггерным, т. е. таким, который специфицирует событие-триггер. Например, переходы являются триггерными, поскольку с каждым из них связано некоторое событие-триггер, происходящее асинхронно в момент выхода из строя технического устройства или в момент окончания его ремонта.

Если рядом со стрелкой перехода не указана никакая строка текста, то соответствующий переход является нетриггерным, и в этом случае из контекста диаграммы состояний должно быть ясно, после окончания какой деятельности он срабатывает.

Сторожевое условие (guard condition), если оно есть, всегда записывается в прямых скобках после события-триггера и представляет собой некоторое булевское выражение. Если сторожевое условие принимает значение "истина", то соответствующий переход может сработать. Если же сторожевое условие принимает значение "ложь", то переход не может сработать, и при отсутствии других переходов объект не может перейти в целевое состояние по этому переходу

Примером события-триггера может служить разрыв телефонного соединения с провайдером Интернет-услуг после окончания загрузки электронной почты клиентской почтовой программой (при удаленном доступе к Интернету). В этом случае сторожевое условие есть не что иное, как ответ на вопрос: "Пуст ли почтовый ящик клиента на сервере провайдера?". В случае положительного ответа "истина", следует отключить соединение с провайдером, что и делает автоматически почтовая программа-клиент. В случае отрицательного ответа "ложь", следует оставаться в состоянии загрузки почты и не разрывать телефонное соединение.

Выражение действия (action expression) выполняется в том и только в том случае, когда переход срабатывает. Представляет собой атомарную операцию (достаточно простое вычисление), выполняемую сразу после срабатывания соответствующего перехода до начала каких бы то ни было действий в целевом состоянии. В качестве примера выражения действия может служить "разорвать телефонное соединение (телефонный номер), которое должно быть выполнено сразу после установления истинности ("истина") сторожевого условия "почтовый ящик на сервере пуст".

Составное состояние (composite state) - такое сложное состояние, которое состоит из других вложенных в него состояний. Последние будут выступать по отношению к первому как подсостояния (substate). Хотя между ними имеет место отношение композиции, графически все вершины диаграммы, которые соответствуют вложенным состояниям, изображаются внутри символа составного состояния. В этом случае размеры графического символа составного состояния увеличиваются, так чтобы вместить в себя все подсостояния. Составное состояние может содержать два или более параллельных подавтомата или несколько последовательных подсостояний. Каждое сложное состояние может уточняться только одним из указанных способов.

Краткое описание методологии моделирования классов в языке UML.

Объект представляет собой экземпляр класса – особую сущность, которая имеет заданные значения атрибутов и операций. Ваша стиральная машина, например, может иметь атрибуты: компания-производитель – Laundatorium, наименование модели – Washmeister, серийный номер изделия – GL57774 и емкость – 16 фунтов.

Атрибуты

Атрибут – это свойство класса. Атрибуты описывают перечень значений, в рамках которых указываются свойства объектов (т.е. экземпляров) этого класса. Класс может не иметь атрибутов или содержать любое их количество. Имена атрибутов, состоящие из одного слова, принято обозначать строчными буквами. Если имя состоит из нескольких слов, то эти слова объединяются, и каждое слово, за исключением первого, начинается с прописной буквы. Список имен атрибутов начинается ниже линии, отделяющей их от имени класса.

UML позволяет отображать дополнительную информацию об атрибутах. В изобра-жении класса можно указать тип для каждого значения атрибута. Перечень возможных типов включает строку, число с плавающей точкой, целое число, логическое зна-чение и другие

перечислимые типы. Для отображения типа используется двоеточие, которое отделяет имя атрибута от его типа. Здесь же можно указать значение атрибута по умолчанию.

Операции

Операция – это то, что может выполнять класс, либо то, что вы (или другой класс) можете выполнять над данным классом. Подобно имени атрибута, имя операции записывается строчными буквами, если это одно слово. Если имя состоит из нескольких слов, они соединяются, и все слова, кроме первого, пишутся с прописной буквы. Список операций начинается ниже линии, отделяющей операции от атрибутов.

Помимо дополнительной информации об атрибутах, можно отобразить дополнительную информацию об операциях. В скобках, следующих за именем операции, можно указать параметр операции и его тип. Один из типов операций, функция, по окончании работы возвращает значение. В этом случае можно указать возвращаемое значение и его тип.

Ход работы

Задание.

Построить диаграмму Деятельности.

Изучить теоретические сведения по теме “Диаграмма деятельности».

Разработать диаграмму деятельности для произвольной системы индивидуального задания.

Оформить отчет, включив в него описание всех компонентов диаграммы деятельности согласно индивидуальному варианту задания.

Список используемой литературы

1. Рудаков А. Технология разработки программных продуктов: учебник. изд.Academia. Среднее профессиональное образование. 2020 г.

Практическое занятие 5

Тема: «Построение диаграммы компонентов и диаграммы потоков данных»

Цель: ознакомиться с методологией моделирования информационных систем на основе языка UML. разработать диаграммы потоков данных по предложенной информационной системе. Ознакомиться с теоретическим материалом по составлению диаграмм потоков данных.

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Проанализируйте пример построения диаграммы компонентов

Выделяем компоненты, отображаем зависимости между ними. Фрагмент диаграммы компонентов с отношениями зависимости и реализации показан на рисунке 8. Графическое изображение отношения зависимости между компонентами приведено на рисунке 9. На рисунке 10 показано графическое изображение зависимости между компонентом и классами.

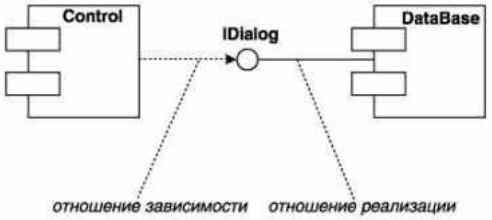


Рисунок 8

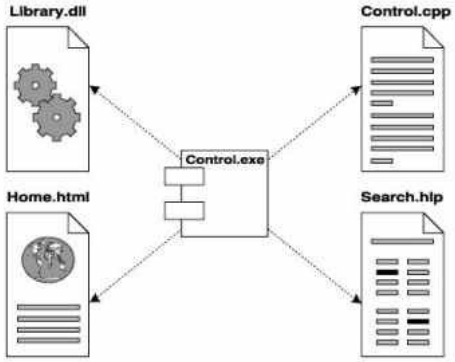


Рисунок 9

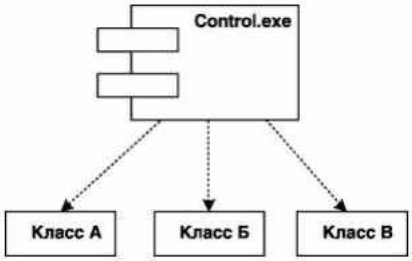


Рисунок 10

Ход работы

Задание. Постройте диаграмму компонентов для выбранной информационной системы

Задание. Сформировать диаграммы потоков данных по предложенной информационной системе.

Список используемой литературы

1. Рудаков А. Технология разработки программных продуктов: учебник. изд.Academia. Среднее профессиональное образование. 2020 г.

Практическое занятие 6

Тема: Разработка тестового сценария

Цели: усвоить знание о видах тестирования; освоить способы обнаружения и фиксирования ошибок.

Теоретические сведения

Общепринятая практика состоит в том, что после завершения продукта и до передачи его заказчику независимой группой тестировщиков проводится тестирование ПО.

Уровни тестирования:

Модульное тестирование. Тестируется минимально возможный для тестирования компонент, например отдельный класс или функция;

Интеграционное тестирование. Проверяется, есть ли какие-либо проблемы в интерфейсах и взаимодействии между интегрируемыми компонентами, например, не передается информация, передается некорректная информация;

Системное тестирование**.** Тестируется интегрированная система на ее соответствие исходным требованиям.

Таблица 1. Виды некоторых ошибок и способы их обнаружения

Виды программных ошибок	Способы их обнаружения
Ошибки выполнения, выявляемые автоматически: а) переполнение, защита памяти; б) несоответствие типов; в) зацикливание	Динамический контроль: аппаратурой процессора; run-time системы программирования; операционной системой – по превышению лимита времени

Тест – это набор контрольных входных данных совместно с ожидаемыми результатами.

Тесты должны обладать определенными свойствами.

Детективность: тест должен с большой вероятностью обнаруживать возможные ошибки.
Покрывающая способность: один тест должен выявлять как можно больше ошибок.
Воспроизводимость: ошибка должна выявляться независимо от изменяющихся условий.

С помощью тестирования разных видов обнаруживаются ошибки в разрабатываемом программном обеспечении. После обнаружения ошибок проводится их устранение.

Задание

Создать приложение Простой калькулятор, в котором реализовать выполнение простых операций с вводимыми двумя операндами. Выполнить тестирование приложения на различных данных, отличающихся по типу и значению.

Программа работы

1. Разработать интерфейс приложения и написать программные коды для событий кнопок.
2. Сохранить проект в отдельной папке, скопировать исполняемый файл на рабочий стол.
3. Составить тесты для проверки работы приложения.
4. Провести тестирование исполняемого файла

Составить отчет по итогам тестирования и рекомендации по устранению выявленных ошибок

Практическое занятие 7

Тема: Оценка необходимого количества тестов»

Цели: получить навыки разработки тестовых сценариев.

Теоретические вопросы

- Оценка стоимости и причины ошибок в программном обеспечении.
- Виды и методы тестирования.
- Понятие теста.
- Требования к разработке тестовых сценариев.
- Правила разработки тестовых сценариев.

Задание № 1

Написать программу решения квадратного уравнения ax2 + bx + c = 0.

Задание № 2

Найти минимальный набор тестов для программы нахождения веще-ственных корней квадратного уравнения ax2 + bx + c = 0. Решение представлено в таблице.

Номер теста	a	b	c	Ожидаемый результат	Что проверяется
1	2	-5	2	x_1=2, x_2=0,5	Случай вещественных корней
2	3	2	5	Сообщение	Случай комплексных корней
3	3	-12	0	x_1=4, x_2=0	Нулевой корень
4	0	0	10	Сообщение	Неразрешимое уравнение
5	0	0	0	Сообщение	Неразрешимое уравнение
6	0	5	17	Сообщение	Неквадратное уравнение
7	9	0	0	x_1=x_2=0	Нулевые корни

Таким образом, для этой программы предлагается минимальный набор функциональных тестов, исходя из 7 классов выходных данных.

Заповеди по отладки программного средства, предложенные Г. Майерсом.

Заповедь 1. Считайте тестирование ключевой задачей разработки ПС, поручайте его самым квалифицированным и одаренным программистам, нежелательно тестировать свою собственную программу.

Заповедь 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Заповедь 3. Готовьте тесты как для правильных, так и для неправильных данных.

Заповедь 4. Документируйте пропуск тестов через компьютер, детально изучайте результаты каждого теста, избегайте тестов, пропуск которых нельзя повторить. Заповедь 5. Каждый модуль подключайте к программе только один раз, никогда не изменяйте программу, чтобы облегчить ее тестирование.

Заповедь 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).

Задание № *3*

Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы:

Имеется консольное приложение (разработайте самостоятельно). Ему на вход подается 2 строки. На выходе приложение выдает число вхождений второй строки в первую. Например:

Строка 1	Строка 2	Вывод
абвгабвг	аб	2
стстсап	стс	2

Набор тестовых сценариев запишите в виде таблицы, приведенной выше.

Задание № 4

Оформить отчет.

Практическое занятие 8

Тема: Разработка тестовых пакетов. Оценка программных средств с помощью метрик

Цели: Получить навыки разработки тестовых пакетов. Знакомство с ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения»; определить способы получения информации о ПС; формирование информационно - правовых компетенции обучающихся.

Теоретические вопросы

- Системные основы разработки требований к сложным комплексам программ.
- Формализация эталонов требований и характеристик комплекса программ.
- Формирование требований компонентов и модулей путем декомпозиции функций комплексов программ.
- Тестирование по принципу «белого ящика».
- Необходимая документация**:** ГОСТ 28.195-89

Одной из важнейших проблем обеспечения качества программных средств является формализация характеристик качества и методология их оценки. Для определения адекватности качества функционирования, наличия технических возможностей программных средств к взаимодействию, совершенствованию и развитию необходимо использовать стандарты в области оценки характеристик их качества.

Показатели качества программного обеспечения устанавливают ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» и ГОСТ Р ИСО/МЭК 9126 «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению». Одновременное существование двух действующих стандартов, нормирующих одни и те же показатели, ставит вопрос об их гармонизации. Ниже рассмотрим каждый из перечисленных стандартов.

ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» устанавливает общие положения по оценке качества программных средств, номенклатуру и применяемость показателей качества.

Оценка качества ПС представляет собой совокупность операций, включающих выбор номенклатуры показателей качества оцениваемого ПС, определение значений этих показателей и сравнение их с базовыми значениями.

Методы определения показателей качества ПС различаются: по способам получения информации о ПС – измерительный, регистрационный, органолептический, расчетный; по источникам получения информации – экспертный, социологический.

Измерительный метод основан на получении информации о свойствах и характеристиках ПС с использованием инструментальных средств. Например, с использованием этого метода определяется объем ПС - число строк исходного текста программ и число строк - комментариев, число операторов и операндов, число исполненных операторов, число ветвей в программе, число точек входа (выхода), время выполнения ветви программы, время реакции и другие показатели.

Регистрационный метод основан на получении информации во время испытаний или функционирования ПС, когда регистрируются и подсчитываются определенные события, например, время и число сбоев и отказов, время передачи управления другим модулям, время начала и окончания работы.

Органолептический метод основан на использовании информации, получаемой в результате анализа восприятия органов чувств (зрения, слуха), и применяется для определения таких показателей как удобство применения, эффективность и т. п.

Расчетный метод основан на использовании теоретических и эмпирических зависимостей (на ранних этапах разработки), статистических данных, накапливаемых при

испытаниях, эксплуатации и сопровождении ПС. При помощи расчетного метода определяются длительность и точность вычислений, время реакции, необходимые ресурсы.

Определение значений показателей качества ПС *экспертным методом* осуществляется группой экспертов-специалистов, компетентных в решении данной задачи, на базе их опыта и интуиции. Экспертный метод применяется в случаях, когда задача не может быть решена никаким другим из существующих способов или другие способы являются значительно более трудоемкими. Экспертный метод рекомендуется применять при определении показателей наглядности, полноты и доступности программной документации, легкости освоения, структурности.

Социологические методы основаны на обработке специальных анкет-вопросников.

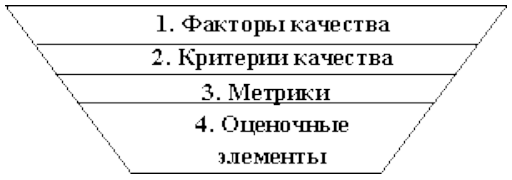


Рис. 1 – Уровни системы показателей качества

Показатели качества объединены в систему из четырех уровней. Каждый вышестоящий уровень содержит в качестве составляющих показатели нижестоящих уровней (рисунок 1).

Стандарт ИСО 9126 (ГОСТ Р ИСО/МЭК 9126) «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению».

Определенные настоящим стандартом характеристики дополнены рядом требований по выбору метрик и их измерению для различных проектов ПС. Они применимы к любому типу ПС, включая компьютерные программы и данные, содержащиеся в программируемом оборудовании. Эти характеристики обеспечивают согласованную терминологию для анализа качества ПС. Кроме того, они определяют схему для выбора и специфицирования требований к качеству ПС, а также для сопоставления возможностей различных программных продуктов, таких как функциональные возможности, надежность, практичность и эффективность.

Все множество атрибутов качества ПС может быть классифицировано в структуру иерархического дерева характеристик и субхарактеристик. Самый высший уровень этой структуры состоит из характеристик качества, а самый нижний уровень – из их атрибутов. Эта иерархия не строгая, поскольку некоторые атрибуты могут быть связаны с более чем одной субхарактеристикой. Таким же образом, внешние свойства (такие, как пригодность, корректность, устойчивость к ошибкам или временная эффективность) влияют на наблюдаемое качество. Недостаток качества в использовании (например, пользователь не может закончить задачу) может быть прослежен к внешнему качеству (например, функциональная пригодность или простота использования) и связанным с ним внутренним атрибутам, которые необходимо изменить.

Внутренние метрики могут применяться в ходе проектирования и программирования к неисполняемым компонентам ПС (таким, как спецификация или исходный программный текст). При разработке ПС промежуточные продукты следует оценивать с использованием внутренних метрик, которые измеряют свойства программ, и могут быть выведены из моделируемого поведения. Основная цель внутренних метрик – обеспечивать, чтобы было достигнуто требуемое внешнее качество. Внутренние метрики дают возможность пользователям, испытателям и разработчикам оценивать качество ЖЦ программ и заниматься вопросами технологического обеспечения качества задолго до того, как ПС становится готовым исполняемым продуктом.

Внутренние метрики позволяют измерять внутренние атрибуты или формировать признаки внешних атрибутов путем анализа статических свойств промежуточных или поставляемых программных компонентов. Измерения внутренних метрик используют категории, числа или характеристики элементов из состава ПС, которые, например, имеются в процедурах исходного программного текста, в графе потока управления, в потоке данных и в представлениях изменения состояний памяти. Документация также может оцениваться с использованием внутренних метрик.

Внешние метрики используют меры ПС, выведенные из поведения системы, частью которых они являются, путем испытаний, эксплуатации или наблюдения исполняемого ПС или системы. Перед приобретением или использованием ПС его следует оценить с использованием метрик, основанных на деловых и профессиональных целях, связанных с использованием, эксплуатацией и управлением продуктом в определенной организационной и технической среде. Внешние метрики обеспечивают заказчикам, пользователям, испытателям и разработчикам возможность определять качество ПС в ходе испытаний или эксплуатации.

Когда требования к качеству ПС определены, в них должны быть перечислены характеристики и субхарактеристики, которые составляют полный набор показателей качества. Затем определяются подходящие внешние метрики и их приемлемые диапазоны значений, устанавливающие количественные и качественные критерии, которые подтверждают, что ПС удовлетворяет потребностям заказчика и пользователя. Далее определяются и специфицируются внутренние атрибуты качества, чтобы спланировать удовлетворение требуемых внешних характеристик качества в конечном продукте и обеспечивать их в промежуточных продуктах в ходе разработки. Подходящие внутренние

метрики и приемлемые диапазоны специфицируются для получения числовых значений или категорий внутренних характеристик качества, чтобы их можно было использовать для проверки того, что промежуточные продукты в процессе разработки удовлетворяют внутренним спецификациям качества. Рекомендуется использовать внутренние метрики, которые имеют наиболее сильные связи с целевыми внешними метриками, чтобы они могли помогать при прогнозировании значений внешних метрик.

Метрики качества в использовании измеряют, в какой степени продукт удовлетворяет потребности конкретных пользователей в достижении заданных целей с результативностью,

продуктивностью и удовлетворением в заданном контексте использования. При этом результативность подразумевает точность и полноту достижения определенных целей пользователями при применении ПС; продуктивность соответствует соотношению израсходованных ресурсов и результативности при эксплуатации ПС, а удовлетворенность – психологическое отношение к качеству использования продукта. Эта метрика не входит в число шести базовых характеристик ПС, регламентируемых стандартом ИСО 9126, однако рекомендуется для интегральной оценки результатов функционирования комплексов программ.

Оценивание качества в использовании должно подтверждать его для определенных сценариев и задач, оно составляет полный объединенный эффект характеристик качества ПС для пользователя. *Качество в использовании* – это восприятие пользователем качества системы, содержащей ПС, и оно измеряется скорее в терминах результатов использования комплекса программ, чем собственных внутренних свойств ПС. Связь качества в использовании с другими характеристиками качества ПС зависит от типа пользователя, так, например, для конечного пользователя качество в использовании обусловливают, в основном, характеристики функциональных возможностей, надежности, практичности и эффективности, а для персонала сопровождения ПС качество в использовании определяет сопровождаемость. На качество в использовании могут влиять любые характеристики качества, и это понятие шире, чем практичность, которая связана с простотой использования и привлекательностью. Качество в использовании, в той или иной степени, характеризуется сложностью применения комплекса программ, которую можно описать трудоемкостью использования с требуемой результативностью. Многие характеристики и субхарактеристики ПС обобщенно отражаются неявными технико- экономическими показателями, которые поддерживают функциональную пригодность конкретного ПС. Однако их измерение и оценка влияния на показатели качества, представляет сложную проблему.

Задание № 1

В Древней Греции (II в. до н.э.) был известен шифр, называемый "квадрат Полибия". Шифровальная таблица представляла собой квадрат с пятью столбцами и пятью строками, которые нумеровались цифрами от 1 до 5. В каждую клетку такого квадрата записывалась одна буква. В результате каждой букве соответствовала пара чисел, и шифрование сводилось к замене буквы парой чисел. Для латинского алфавита квадрат Полибия имеет вид:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I, J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Пользуясь изложенным способом создать программу, которая: а) зашифрует введенный текст и сохранит его в файл;
б) считает зашифрованный текст из файла и расшифрует данный текст.

Задание № 2

Спроектировать тесты по принципу «белого ящика» для программы, разработанной в задании № 1. 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	Ожида-емый результат	Фактиче-ский результат	Результат тестирова-ния
…	…	…	…

Задание № 3

Проверить все виды тестов и сделать выводы об их эффективности

Задание №4. Провести сравнение понятий «качество» государственным и международным стандартами. Выписать документы, в которых даны данные определения.

Задание №5. Опишите методы получения информации о ПС по ГОСТу. Для каждого метода выделите источник информации.

Задание №6. Выберите стандарты для оценки качества ПС. Перечислите критерии надежности ПС по ГОСТу.

Задание №7. Методика оценки качественных показателей ПП основана на составлении метрики ПП. В лабораторной работе необходимо выполнить следующее:

1. Выбрать показатели качества (не менее 5) и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы (таблица 1).

Показатели качества	Сущность показателя	Экспертная оценка (вес) wi	Оценка, установленная экспериментом ri

2. Установить веса показателей wi (?wi =1).

3. Для каждого показателя установить конкретную численную оценку ri от 0 до 1, исходя из следующего:

§ 0 – свойство в ПП присутствует, но качество его неприемлемо;

§ 0.5 — 1 – свойство в ПП присутствует и обладает приемлемым качеством;

§ 1 – свойство в ПП присутствует и обладает очень высоким качеством.

§ Возможно, присвоение промежуточных значений в соответствии с мнением оценивающего лица относительно полезности того или иного свойства ПП.

$$K = \frac{\sum w_i \cdot r_i}{общее\_количество\_показателей}$$

Результатом выполнения данной работы является отчет

Практическое занятие 10

Тема: «Инспекция программного кода на предмет соответствия стандартам кодирования»

Цель: научиться выполнять реорганизацию программного кода на основании шаблонов рефакторинга.

Задание №1

Выполнить анализ программного кода для разрабатываемого ПО и модульных тестов с целью плохо организованного кода.

Задание №2

Используя шаблоны рефакторинга, выполнить реорганизацию программного кода разрабатываемого ПО.

Задание №3

Выполнить описание произведенных операций рефакторинга (было-стало-шаблон рефакторинга).

Задание №4

В случае необходимости скорректировать проектную документацию.

Задание №5

Сделать выводы по результатам выполнения работ.

Список литературы

Электронные издания (электронные ресурсы)

1. Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2019. — 235 с. Текст : электронный // ЭБС Юрайт [сайт]. — Режим доступа: https://biblio- online.ru/bcode/438444
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности [Электронный ресурс]: учеб. пособие / Г.Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - Режим доступа: http://znanium.com/catalog/product/989682
3. Программное обеспечение компьютерных сетей и web-серверов [Электронный ресурс]: учеб. пособие / Г.А. Лисьев, П.Ю. Романов, Ю.И. Аскерко. — М. : ИНФРА-М, 2019. — 145 с. - Режим доступа: http://znanium.com/catalog/product/988332
4. Гагарина, Л. Г. Разработка и эксплуатация автоматизированных информационных систем [Электронный ресурс]: учеб. пособие / Л.Г. Гагарина. — М. : ФОРУМ : ИНФРА- М, 2019. — 384 с.- Режим доступа: http://znanium.com/catalog/product/1003025 (ЭБСZnanium)
5. Технология разработки программного обеспечения [Электронный ресурс]: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРА-М, 2019. — 400 с. - Режим доступа: http://znanium.com/catalog/product/1011120 (ЭБСZnanium)
6. Программное обеспечение компьютерных сетей [Электронный ресурс]: учеб. пособие / О.В. Исаченко. — М. : ИНФРА-М, 2019. — 117 с. — (Среднее профессиональное образование). - Режим доступа: http://znanium.com/catalog/product/989894 (ЭБСZnanium)
7. Плохотников, К.Э. Метод и искусство математического моделирования : курс лекций / К.Э. Плохотников. — 2-е изд., стер. — Москва : ФЛИНТА, 2017. — 519 с. - Текст : электронный. - Режим доступа: https://new.znanium.com/catalog/product/1034329
8. Математическое моделирование технических систем [Электронный ресурс]: учебник / В.П. Тарасик. — Минск : Новое знание ; М. : ИНФРА-М, 2019. — 592 с. - Режим доступа: http://znanium.com/catalog/product/1019246

МДК.02.01 Средства разработки программного обеспечения

ЛАБОРАТОРНАЯ РАБОТА №1 (4 часа)

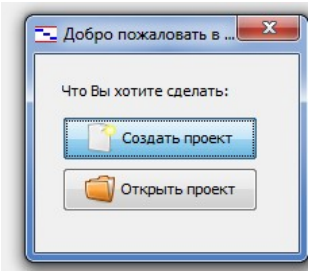
Тема: Разработка структуры проекта

Цель занятия: создание структуры проекта и заполнение базовой информации о проекте.

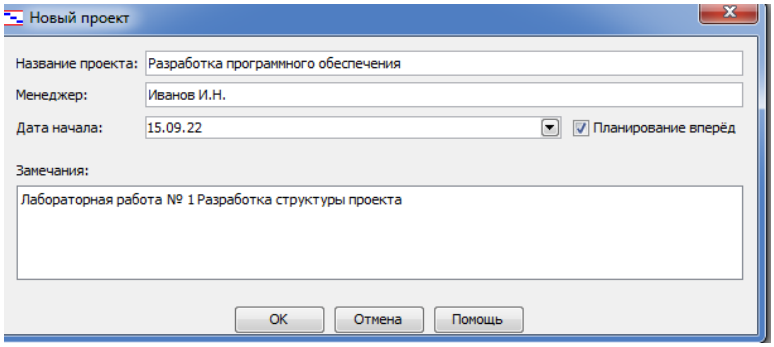
Технология работы:

Задание 1

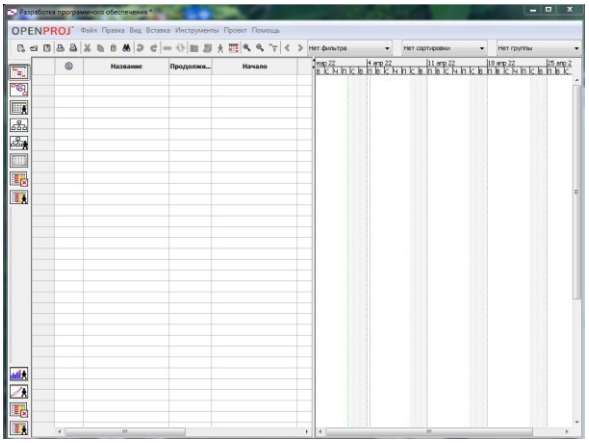
1. Запустите программу Open Project.
2. Изучите интерфейс программы.
3. Создайте новый проект.



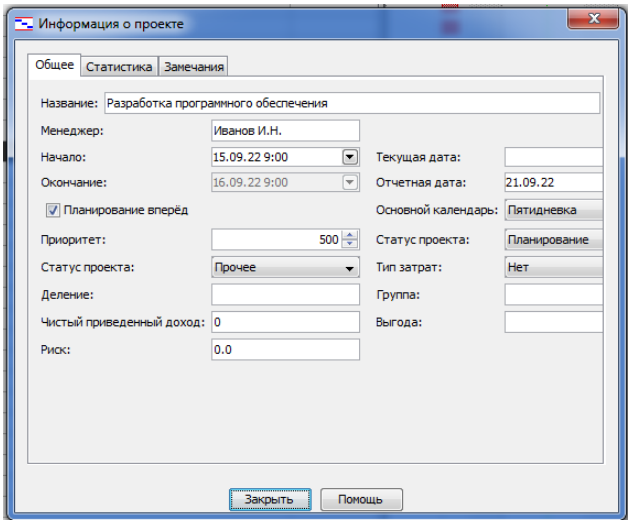
4. В появившемся окне следует заполнить данные о новом проекте. Необходимо указать название проекта, автора, дату начала проекта и краткие сведения.



5. После нажатия на кнопку «Ок» появится пустой проект.



6. После создания проекта необходимо настроить его основные параметры. Для настройки планирования от начальной даты выберите в меню **Проект** пункт **Сведения о проекте**. В появившемся окне ставим *Дату начала*. Да окончания будет рассчитана далее автоматически.

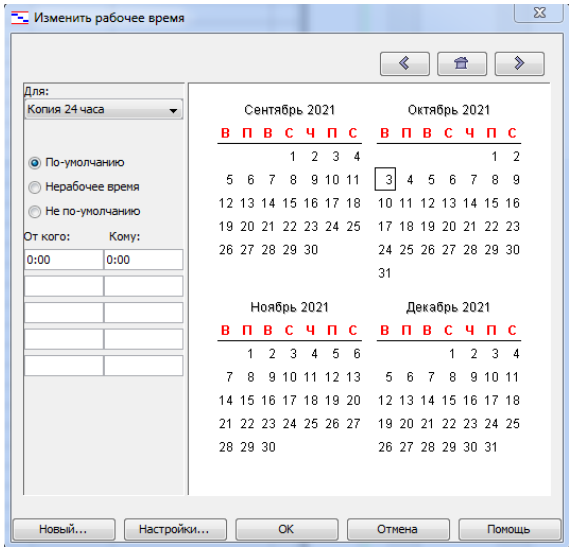


7. Выберите календарь для проекта.

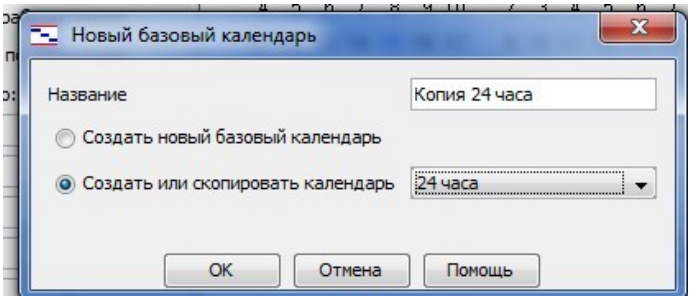
В состав пакета MS Project входит три базовых календаря – стандартный, ночная смена и 24 часа.

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.

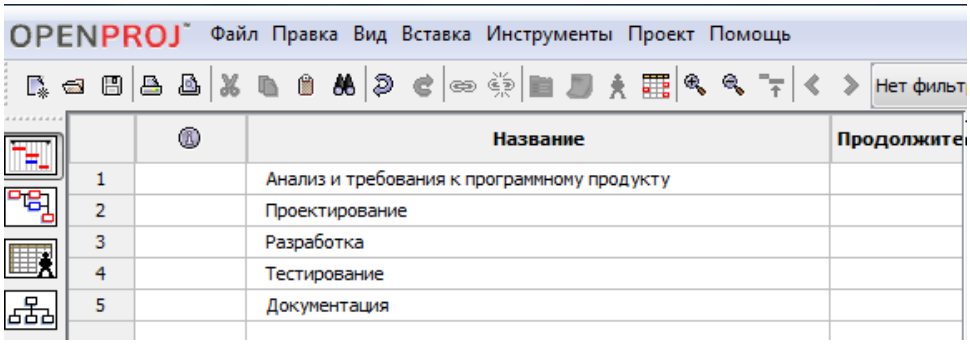
Базовые календари можно редактировать для этого в меню *Инструменты* необходимо выбрать пункт *Изменение рабочего времени*.



Можно создать новое базовое расписание. Для этого в окне *Изменение рабочего времени* нажимаем кнопку *Создать*. В появившемся окне 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*.

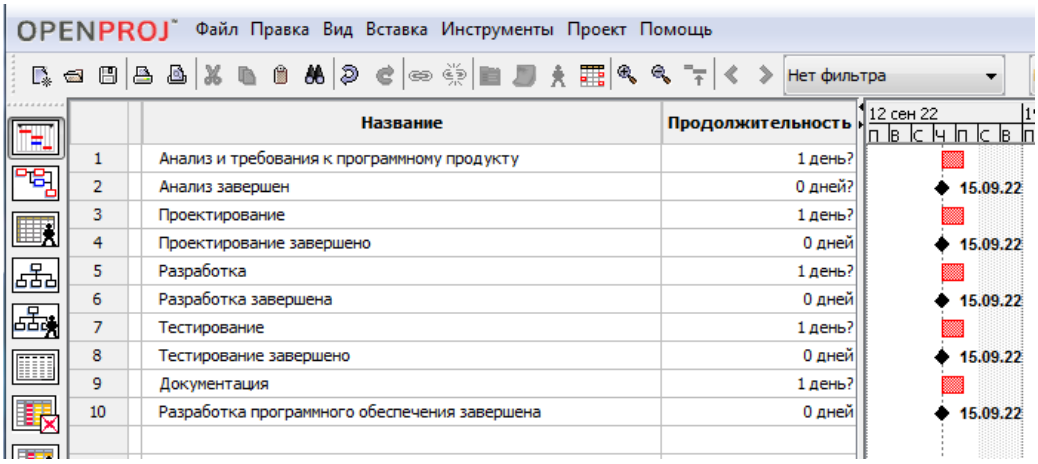


8. Добавьте в план проекта следующие фазы жизненного цикла ПО.
   - Анализ и требования к программному обеспечению
   - Проектирование
   - Разработка
   - Тестирование
   - Документация



9. Для каждой фазы укажите в плане проекта завершающие задачи (вехи проекта):
   - Анализ завершен
   - Проектирование завершено
   - Разработка завершена
   - Тестирование завершено
   - Документация завершена
   - Разработка программного обеспечения завершена

Для вставки пустых строк между названиями фаз проекта используйте клавишу Ins или команду Вставка / Новая задача. Для того, чтобы задача стала завершающей (вехой) следует установить ее длительность равной 0 дней.

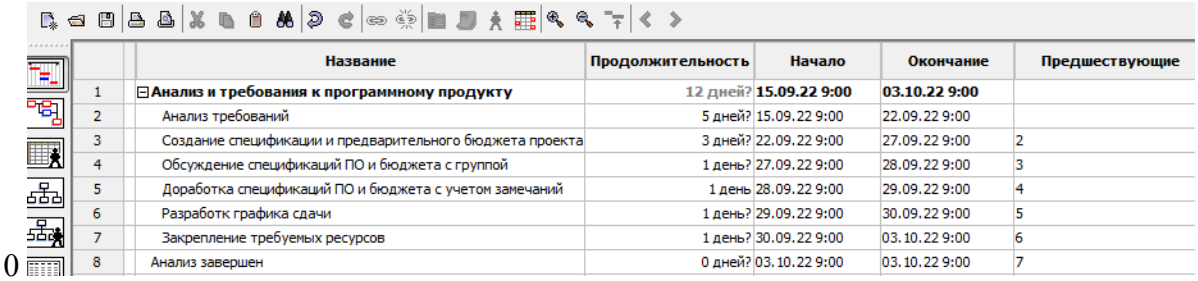


10. Декомпозируйте план проекта, добавив в него следующие задачи для фазы *Анализ и требования к программному обеспечению*:
   - Анализ требований
   - Создание спецификации и предварительного бюджета проекта
   - Обсуждение спецификаций программного обеспечения и бюджета с группой
   - Доработка спецификаций программного обеспечения и бюджета с учетом замечаний
   - Разработка графика сдачи
   - Закрепление требуемых ресурсов

11. Установите длительность задач фазы *Анализ и требования к программному обеспечению* в соответствие со следующей таблицей:

Задача	Длительность
Анализ требований	*5 дней*
Создание спецификации и предварительного бюджета проекта	*3 дня*
Обсуждение спецификаций программного обеспечения и бюджета с группой	*4 часа*
Доработка спецификаций программного обеспечения и бюджета с учетом замечаний	*1 день*
Разработка графика сдачи	*1 день*
Закрепление требуемых ресурсов	*1 день*

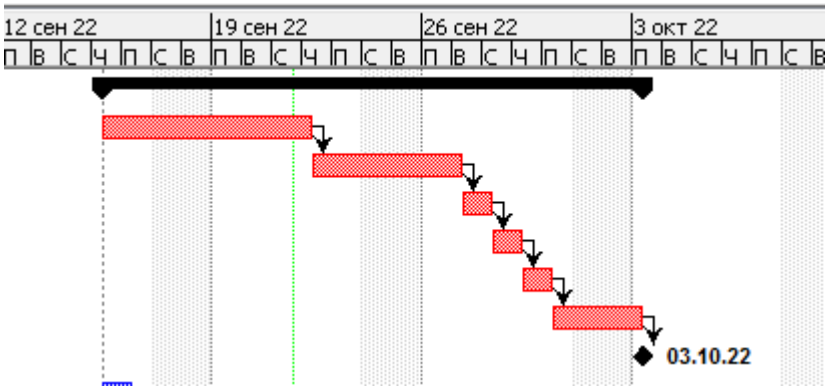
12. Установите связи между задачами.



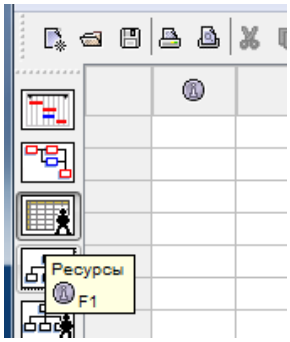
	Название	Продолжительность	Начало	Окончание	Предшествующие
1	**Анализ и требования к программному продукту**	12 дней?	**15.09.22 9:00**	**03.10.22 9:00**	
2	Анализ требований	5 дней?	15.09.22 9:00	22.09.22 9:00	
3	Создание спецификации и предварительного бюджета проекта	3 дней?	22.09.22 9:00	27.09.22 9:00	2
4	Обсуждение спецификаций ПО и бюджета с группой	1 день?	27.09.22 9:00	28.09.22 9:00	3
5	Доработка спецификаций ПО и бюджета с учетом замечаний	1 день	28.09.22 9:00	29.09.22 9:00	4
6	Разработк графика сдачи	1 день?	29.09.22 9:00	30.09.22 9:00	5
7	Закрепление требуемых ресурсов	1 день?	30.09.22 9:00	03.10.22 9:00	6
8	Анализ завершен	0 дней?	03.10.22 9:00	03.10.22 9:00	7

0

13. Созданные задачи и их связи отобразились на Диаграмме Ганта.



14. Добавьте к задачам исполнителей. Исполнители являются ресурсами проекта. Перейдем к вкладке «Ресурсы», расположенной на боковой панели.



15. В диалоговом окне появится таблица с ресурсами проекта. Добавить в проект 10 исполнителей.

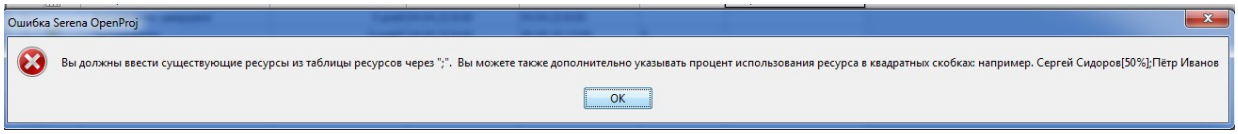
		Название	RBS	Тип	E-mail адрес	Ед.изм. материалов	Инициалы	Группировать
1		Иванов		Работа			И	Руководство
2		Петров		Работа			П	Руководитель проекта
3		Сидоров		Работа			С	Руководитель проекта
4		Жуков		Работа			Ж	Разработчик

16. Назначить исполнителей проекта, вписав их имена в соответствующее поле.

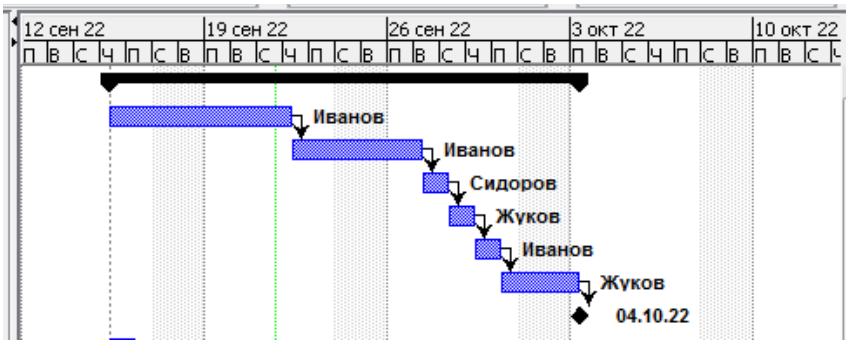
		Название	Продолжительность	Начало	Окончание	Предшествующие	Название
1		**Анализ и требования к программному продукту**	12 дней?	**15.09.22 9:00**	**03.10.22 9:00**		
2		Анализ требований	5 дней?	15.09.22 9:00	22.09.22 9:00		Иванов
3		Создание спецификации и предварительного бюджета проекта	3 дней?	22.09.22 9:00	27.09.22 9:00	2	Иванов
4		Обсуждение спецификаций ПО и бюджета с группой	1 день?	27.09.22 9:00	28.09.22 9:00	3	Сидоров
5		Доработка спецификаций ПО и бюджета с учетом замечаний	1 день	28.09.22 9:00	29.09.22 9:00	4	Жуков
6		Разработк графика сдачи	1 день?	29.09.22 9:00	30.09.22 9:00	5	Иванов
7		Закрепление требуемых ресурсов	1 день?	30.09.22 9:00	03.10.22 9:00	6	Жуков
8		Анализ завершен	0 дней?	03.10.22 9:00	03.10.22 9:00	7	

(

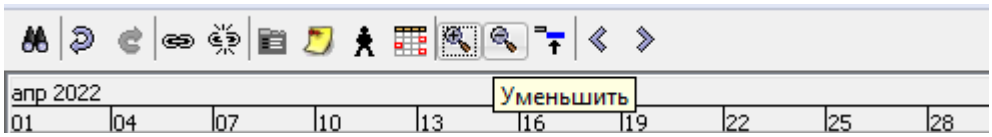
17. В случае если попытаться ввести имя несуществующего ресурса, программа выведет сообщение об ошибке.



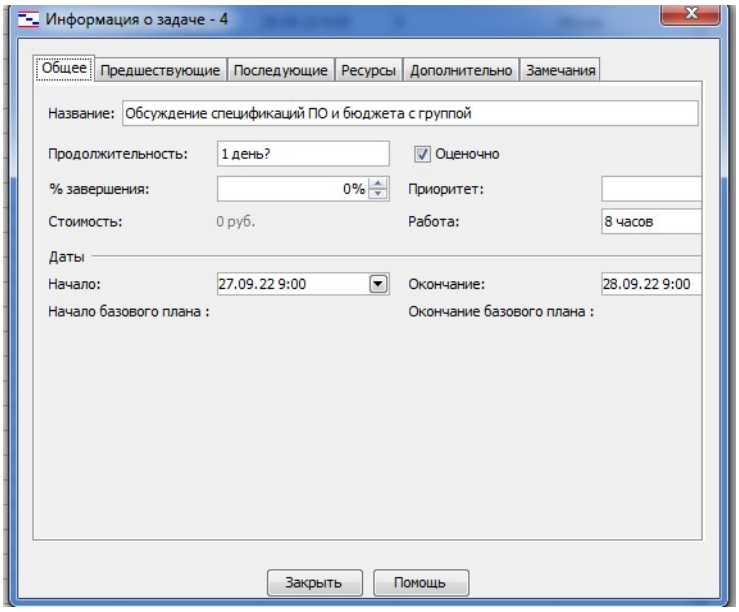
18. Назначенные исполнители отобразятся на Диаграмме Ганта.



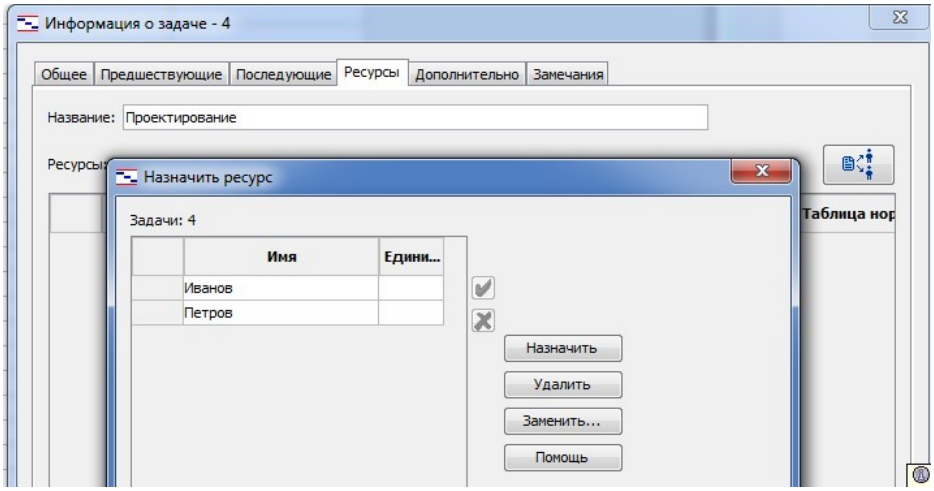
19. В программе присутствует возможность фильтрации задач на Диаграмме Ганта, а также изменение масштаба отображения временной шкалы.



20. Двойной клик мышью по задаче вызывает окно свойств этой задачи. Вкладка общие позволяет получить информацию о названии, продолжительности и проценте выполнения. Откройте любую задачу проекта.



21. Вкладки «Предыдущие» и «Последующие» позволяют получить информацию о связанных задачах. Интерес может представлять вкладка «Ресурсы», на которой будут отображаться исполнители задачи.



22. Аналогично декомпозируйте план проекта, добавив в него следующие задачи и подфазы:

- фаза Проектирование:
  - Разработка функциональных спецификаций
  - Разработка прототипа на основе функциональной спецификации
  - Ревизия функциональных спецификаций
  - Доработка функциональных спецификаций с учетом замечаний
- фаза Разработка:
  - Определение параметров модульной и уровневой архитектуры
  - Назначение персонала для разработки
  - Разработка кода
- фаза Тестирование:
  подфаза Тестирование модулей:
  - Тестирование модулей компонента в соответствии со спецификацией продукта
  - Выявление недостатков в спецификациях продукта
  - Изменение кода
  - Повторное тестирование измененного кода
  - Тестирование модулей завершено (веха подфазы); подфаза Тестирование интеграции:
  - Тестирование интеграции модулей
  - Выявление недостатков в спецификациях
  - Изменение кода
  - Повторное тестирование измененного кода
  - Тестирование интеграции завершено (веха подфазы);
- фаза Документация:
  - Разработка справки
  - Ревизия справки
  - Доработка справки с учетом замечаний
  - Разработка руководства пользователя
  - Ревизия всей документации для пользователей
  - Доработка документации для пользователей с учетом замечаний

23. Установите длительность задач плана проекта в соответствие со следующей таблицей:

Задача	Длительность
Разработка функциональных спецификаций	*5 дней*
Разработка прототипа на основе функциональной спецификации	*4 дня*
Ревизия функциональных спецификаций	*2 дня*
Доработка функциональных спецификаций с учетом замечаний	*4 часа*
Определение параметров модульной и уровневой архитектуры	*4 часа*
Назначение персонала для разработки	*1 день*
Разработка кода	*15 дней*
Тестирование модулей компонента в соответствии со спецификацией продукта	*2 дня*
Выявление недостатков в спецификациях продукта	*3 дня*
Изменение кода	*3 дня*
Повторное тестирование измененного кода	*2 дня*
Тестирование интеграции модулей	*5 дней*
Выявление недостатков в спецификациях	*2 дня*
Изменение кода	*3 дня*
Повторное тестирование измененного кода	*2 дня*
Разработка справки	*3 недели*
Ревизия справки	*3 дня*
Доработка справки с учетом замечаний	*2 дня*
Разработка руководства пользователя	*3 недели*
Ревизия всей документации для пользователей	*2 дня*
Доработка документации для пользователей с учетом замечаний	*2 дня*

24. Установите связи между задачами
25. Назначьте исполнителей проекта, вписав их имена в соответствующее поле.
26. Все изменения отобразятся на Диаграмме Ганта.
27. Создайте сетевой график с помощью пиктограммы 
28. После завершения планирования проекта сохраните результаты работы.

Контрольные вопросы:

1. Что такое Open Project?
2. Что представляет собой проект в Open Project?
3. Что представляет собой работа в Open Project?
4. Что обычно подразумеваются под ресурсами?
5. Какие существуют базовые календари в программе MS Project?
6. Что такое Диаграмма Ганта?
7. Какие этапы присутствуют в календарном плане проекта?

ЛАБОРАТОРНАЯ РАБОТА № 2 (2 часа)

Тема: Разработка модульной структуры проекта (диаграммы модулей)

Цель: Изучить процесс разработки модульной структуры программного обеспечения, осуществляемого с помощью структурных карт Константайна.

Задания - Разработать модульную структуру программного модуля.

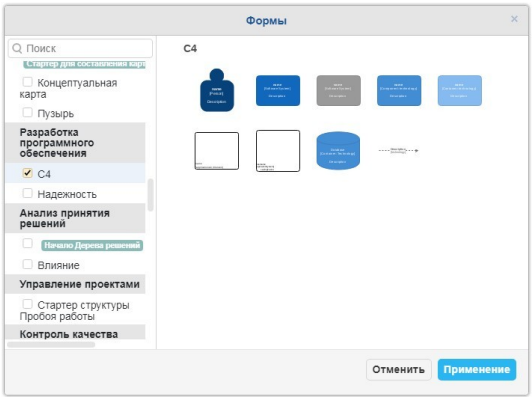
Технология выполнения:

1 Запустите программу Visual Paradigm Online

(https://online.visual-paradigm.com/drive/#diagramlist:proj=0&dashboard)

2 Выберите шаблоны группы Диаграммы: SDL Diagram, ArchiMate Tutorial Starter, Functional Decomposition Diagram

3 Добавьте необходимы формы



4 Составьте структурную карту Константайна

5 Составьте спецификацию программного модуля.

Варианты:

1. Разработать программный модуль «Учет успеваемости студентов». Программный модуль предназначен для оперативного учета успеваемости студентов в сессию деканом, заместителями декана и сотрудниками деканата. Сведения об успеваемости студентов должны храниться в течение всего срока их обучения и использоваться при составлении справок о прослушанных курсах и приложений к диплому.
2. Разработать программный модуль «Личные дела студентов». Программный модуль предназначен для получения сведений о студентах сотрудниками деканата, профкома и отдела кадров. Сведения должны храниться в течение всего срока обучения студентов и использоваться при составлении справок и отчетов.
3. Разработать программный модуль «Решение комбинаторно-оптимизационных задач». Модуль должен содержать алгоритмы поиска цикла минимальной длины (задача коммивояжера), поиска кратчайшего пути и поиска минимального связывающего дерева.
4. Разработать приложение Windows «Органайзер». Приложение предназначено для записи, хранения и поиска адресов и телефонов физических лиц, и организаций, а также расписания, встреч и др. Приложение предназначено для любых пользователей компьютера.
5. Разработать приложение Windows «Калькулятор». Приложение предназначено для любых пользователей и должно содержать все арифметические операции (с соблюдением приоритетов) и желательно (но не обязательно) несколько математических функций.
6. Разработать программный модуль «Кафедра», содержащий сведения о сотрудниках кафедры (ФИО, должность, ученая степень, дисциплины, нагрузка, общественная работа, совместительство и др.). Модуль предназначен для использования сотрудниками отдела кадров и деканата.

7. Разработать программный модуль «Лаборатория», содержащий сведения о сотрудниках лаборатории (ФИО, пол, возраст, семейное положение, наличие детей, должность, ученая степень). Модуль предназначен для использования сотрудниками профкома и отдела кадров.

8. Разработать программный модуль «Автосервис». При записи на обслуживание заполняется заявка, в которой указываются ФИО владельца, марка автомобиля, вид работы, дата приема заказа и стоимость ремонта. После выполнения работ распечатывается квитанция.

9. Разработать программный модуль «Учет нарушений правил дорожного движения». Для каждой автомашины (и ее владельца) в базе хранится список нарушений. Для каждого нарушения фиксируется дата, время, вид нарушения и размер штрафа. При оплате всех штрафов машина удаляется из базы.

11. Разработать программный модуль «Картотека агентства недвижимости», предназначенный для использования работниками агентства. В базе содержатся сведения о квартирах (количество комнат, этаж, метраж и др.). При поступлении заявки на обмен (куплю, продажу) производится поиск подходящего варианта. Если такого нет, клиент заносится в клиентскую базу и оповещается, когда вариант появляется.

12. Разработать программный модуль «Картотека абонентов АТС». Картотека содержит сведения о телефонах и их владельцах. Фиксирует задолженности по оплате (абонентской и повременной). Считается, что повременная оплата местных телефонных разговоров уже введена.

13. Разработать программный модуль «Авиакасса», содержащий сведения о наличии свободных мест на авиамаршруты. В базе должны содержаться сведения о номере рейса, экипаже, типе самолета, дате и времени вылета, а также стоимости авиабилетов (разного класса). При поступлении заявки на билеты программа производит поиск подходящего рейса.

14. Разработать программный модуль «Книжный магазин», содержащий сведения о книгах (автор, название, издательство, год издания, цена). Покупатель оформляет заявку на нужные ему книги, если таковых нет, он заносится в базу и оповещается, когда нужные книги поступают в магазин.

15. Разработать программный модуль «Автостоянка». В программе содержится информация о марке автомобиля, его владельце, дате и времени въезда, стоимости стоянки, скидках, задолженности по оплате и др.

16. Разработать программный модуль «Кадровое агентство», содержащий сведения о вакансиях и резюме. Программный модуль предназначен как для поиска сотрудника, отвечающего требованиям руководителей фирмы, так и для поиска подходящей работы.

Примечание. При разработке программы не ограничиваться функциями, приведенными в варианте, добавить несколько своих функций

Контрольные вопросы:

1. Из чего состоят структурная схема?
2. Что такое спецификация программного модуля?
3. В чем заключается методика Константайна?

ЛАБОРАТОРНАЯ РАБОТА № 3 (2 часа)

Тема: Разработка перечня артефактов

Цель: освоение интерфейса программы и навыков построения диаграммы прецедентов, разработка перечня артефактов.

Задание:

1. Определите внешних исполнителей (контрагентов компании)
2. Создайте «физическую» диаграмму

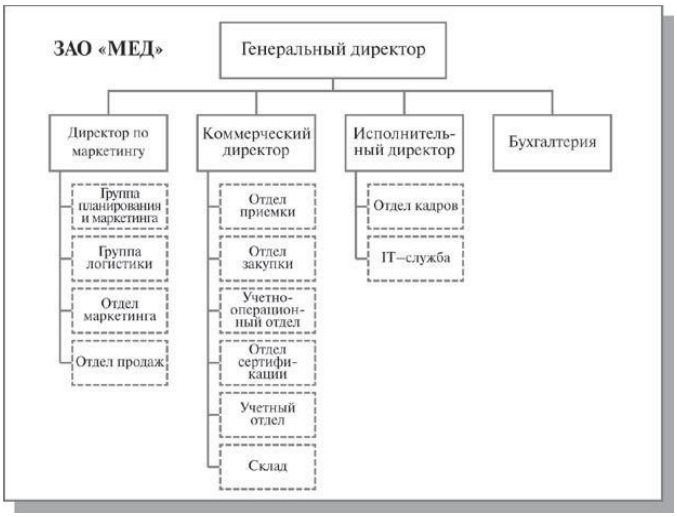
3. Постройте диаграмму прецедентов

Технология выполнения:

Краткое описание предметной области

Компания - дистрибьютор ЗАО "МЕД" закупает медицинские препараты отечественных и зарубежных 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 предприятия оптовой торговли ЗАО "МЕД" имеет следующий вид:



Основные цели автоматизации компании "МЕД":

- Разработка и внедрение комплексной автоматизированной системы поддержки логистических процессов компании.
- Повышение эффективности работы всех подразделений компании и обеспечение ведения учета в единой информационной системе.

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 закупками. Планирование закупок в разрезе поставщиков.
3. Управление продажами. Контроль лимита задолженности с возможностью блокировки формирования отгрузочных документов.
4. Полный контроль взаиморасчетов с поставщиками и клиентами.
5. Получение управленческих отчетов в необходимых аналитических срезах - как детальных для менеджеров, так и агрегированных для руководителей подразделений, и директоров фирмы.

Ограничения предметной области

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 ЗАО "МЕД":

- 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 состава автоматизируемых бизнес-процессов

1) Компания "МЕД» 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) Компания реализует медикаменты через дистрибьюторскую сеть и сеть аптек. Следовательно, контрагентами компании являются покупатели (дистрибьюторы, аптеки).

Внешними контрагентами компании "МЕД" являются поставщики (отечественные, зарубежные), покупатели (дистрибьюторы, аптеки) и транспортные компании.

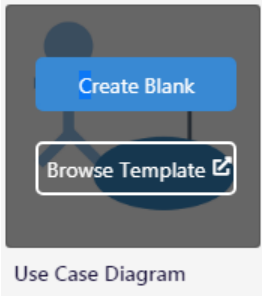
На физической диаграмме компанию изобразим прямоугольником.

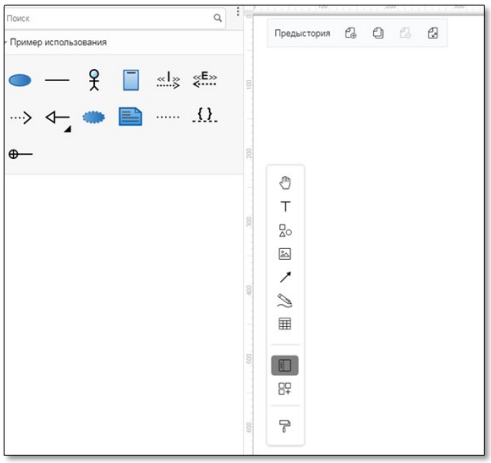
Для отображения контрагентов используются графический символ Actor (фигурка человечка).

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:

1. Запустите Visual Paradigm Online (https://online.visual-paradigm.com/drive/#diagramlist:proj=0&dashboard)
2. Появится окно, в котором выберите (категорию шаблонов) Вкладку Diargam
3. В открывшемся списке выберите диаграмму Use Case, активизируйте команду Create Blank.





6. Для изображения границ компании «МЕД» выберите из набора графических элементов, представленных в левой части окна, пиктограмму прямоугольника с надписью **«Система»** и переносите ее на рабочее поле мышкой при нажатой правой клавише, Отрегулируйте размеры прямоугольника.
7. Задайте цвет прямоугольника **«Система»** и введите подпись «ЗАО «МЕД»».
8. Для изображения на диаграмме контрагентов следует воспользоваться графическим символом с изображением человечка с надписью **«Актер»** и перенести его на рабочее поле при нажатой правой клавише мышки.

9. Соедините линиями изображение каждого контрагента с прямоугольником используя линии «**Ассоциация/Сообщение**» и при нажатой левой клавише мышки осуществите соединение фигур.

10. 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.

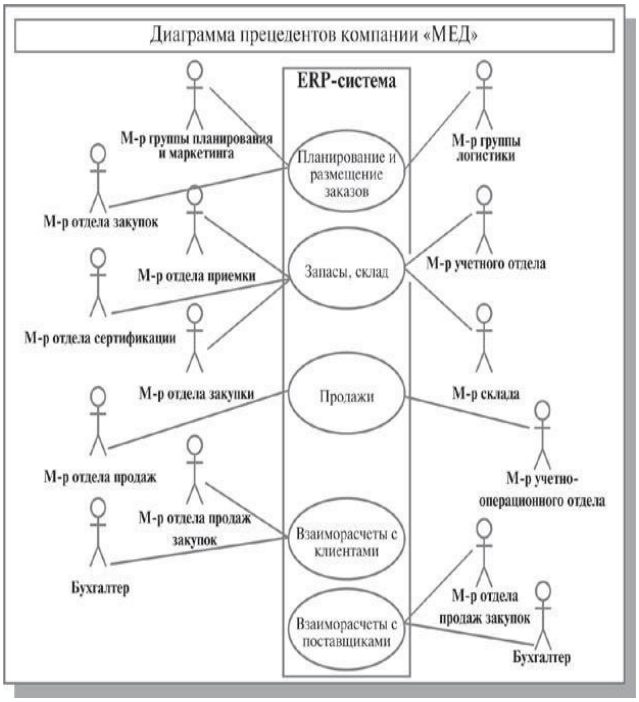
11. Введите наименование компании "МЕД" в нарисованный прямоугольник, щелкнув мышкой по прямоугольнику. Обратите внимание на то, что при этом должна быть активна пиктограмма «**Текст**» (символ буквы "А").

12. Аналогичным образом внесите надписи к линиям соединения фирмы и контрагентов.

Физическая диаграмма ЗАО "МЕД" представлена на рисунке ниже.

2. Построение диаграммы прецедентов

Используя навыки, полученные при выполнении задания 1, постройте диаграмму прецедентов, отображающую прецеденты (варианты использования) компании «Мед» и внутренних исполнителей, обеспечивающих реализацию этих прецедентов внутри системы.



Вариант	Предметная область	Сущность задачи
1	Страховая медицинская компания	Страховая медицинская компания (СМК) заключает договоры добровольного медицинского страхования с населением и договоры с лечебными учреждениями на лечение застрахованных клиентов. При возникновении страхового случая клиент подает заявку на оказание медицинских услуг по условиям договора инспектору, который работает с данным клиентом. Инспектор направляет данного клиента в лечебное учреждение. Отчеты о своей деятельности инспектор предоставляет в бухгалтерию. Бухгалтерия проверяет оплату договоров, перечисляет денежные средства за оказанные услуги лечебным учреждениям, производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. СМК не только оплачивает лечение застрахованного лица при возникновении с ним страхового случая, но и, при возникновении каких-либо осложнений после лечения, оплачивает лечение этих осложнений.
2	Банк	Банк – это предприятие, осуществляющее регулирование платежного оборота в наличной и безналичной формах. Банк привлекает денежные средства физических и юридических лиц во вклады; размещает привлеченные средства от своего имени и за свой счет; открывает и ведет банковские счета физических и юридических лиц; инкассирует денежные средства, векселя, платежные и расчетные документы; производит кассовое обслуживание физических и юридических лиц; производит куплю-продажу иностранной валюты в наличной и безналичной формах; предоставляет услугу хранения ценных бумаг и драгоценных металлов;
3	Кадровое агентство	Кадровое агентство способствует трудоустройству безработных граждан. Агентство ведет учет и классификацию данных о безработных на основании резюме от них. От предприятий города поступают данные о свободных вакансиях, на основании которых агентство предлагает различные варианты трудоустройства соискателям. В случае положительного исхода поиска вакансия считается заполненной, а безработный становится трудоустроенным. По результатам своей деятельности кадровое агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
4	Компания по разработке программных продуктов	Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение – куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
5	Туроператор	Туроператор предоставляет возможность своим клиентам осуществить туристическую или деловую поездку в различные города России и мира. При разработке нового тура сначала анализируется текущая ситуация на рынке туризма и выбирается направление тура. После этого определяется статус тура, бронируются места в гостиницах и билеты на переезд к месту тура, разрабатывается культурная/деловая/ развлекательная программа, утверждаются сроки тура. На каждый тур назначается ответственное лицо от туроператора, которое будет вести данный тур для улаживания проблем в случае возникновения каких-нибудь чрезвычайных или форс- мажорных ситуаций. Клиент приходит в офис туроператора, где вместе с менеджером выбирает уже разработанный тур и оформляет путевку. После возвращения из тура клиент может высказать свои замечания или пожелания, которые будут учтены при доработке существующих туров или при разработке новых. Также, для дальнейшего улучшения тура, туроператор проводит анализ отчетов от посредников (гостиница, гиды и т.д.). По результатам своей деятельности туроператор производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
6	Строительная организация	Строительная организация занимается строительством объектов по заказам клиентов. Сначала заказ проходит предварительную стадию: сбор различных разрешений на строительство, составление эскиза объекта, расчет объема и закупка строительных материалов. Сами строительные материалы доставляются на объект партиями. По мере поступления очередной партии стройматериалов закладывается фундамент объекта, строится каркас здания.

		По результатам данной работы происходит согласование с заказчиком, после чего утепляется контур, вставляются окна, устанавливается крыша. Дальше идет обсуждение с клиентом внутренней отделки здания, закупаются отделочные материалы. После того, как объект проходит технический контроль, он передается заказчику. В дополнительные услуги строительной организации входят: услуги дизайнера по интерьеру, закупка и доставка мебели, сотрудничество с охранным предприятием по установке сигнализации. По результатам своей деятельности строительная организация производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
7	Компьютерная компания	Компьютерная компания занимается продажей, ремонтом, сборкой, тестированием компьютерной техники. Также, специалисты компании предоставляют услуги по разработке и монтажу локальных вычислительных сетей. Вся техника и комплектующие закупаются оптом у дилеров и хранятся на складе. Клиент, который хочет приобрести товар, оформляет заказ в торговом зале, а забирает технику со склада или оставляет заявку на ее доставку. Клиент, который хочет отремонтировать технику, приносит ее в сервисный отдел, откуда, по прошествии некоторого времени, забирает как отремонтированную или как технику, не подлежащую ремонту. По желанию клиента, специалисты компании могут выехать к клиенту для общей диагностики возникшей проблемы с техникой. По результатам своей деятельности компьютерная компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
8	Компания по предоставлению телекомму- никационных услуг	Компания занимается оказанием телекоммуникационных услуг абонентам. Клиент делает заявку на подключение к телекоммуникационным услугам и ему, по необходимости, устанавливают соответствующее оборудование. Оплата за услуги вносится путем авансовых платежей. Каждый факт предоставления услуги фиксируется соответствующим оборудованием и является основанием для списания соответствующей суммы с личного счета абонента. Клиент в любое время суток может получить отчет об оказанных ему услугах, их стоимости и остатку на личном счете абонента. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
9	Управляющая компания ЖКХ.	Управляющая компания (УК) ЖКХ занимается обслуживанием жилого фонда города. УК получает финансовые средства от населения и бюджета города в виде компенсаций и субсидий на коммунальные услуги. На основании поступивших средств УК осуществляет текущий ремонт жилого фонда, а также капитальный ремонт согласно плану. Для непосредственного выполнения работ УК нанимает соответствующую рабочую силу (сантехников, дворников, электриков и т.д.). По результатам своей деятельности УКЖКХ производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
10	Автобаза	Автобаза предоставляет услуги по перевозке пассажиров, различных грузов как в черте города, так и между соседними городами. Для регулярных рейсов оплата клиентами услуги происходит в моментах оказания. В остальных случаях клиент должен сделать заявку, которая может быть отклонена. Для междугородных перевозок в диспетчерские автобазы фиксируется маршрут следования рейса. По результатам своей деятельности автобаза производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.
11	Спортивный комплекс	Спортивный комплекс предоставляет услуги по проведению спортивных тренировок. Тренировки, относящиеся к одному виду спорта, объединяются в спортивные секции. Клиент обращается в спортивный комплекс, где получает абонемент на посещение спортивной секции. На основе купленных абонементов составляется расписание тренировок на следующий месяц. Также, в зависимости от загруженности спортивного комплекса, распределяются тренера спортивных секций. По результатам своей деятельности спортивный комплекс производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

Контрольные вопросы:

1. Что такое артефакт?
2. Назовите сходства и различия диаграмм прецедентов и «физических» диаграмм?
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. Какие типы отношений (связей) между экторами и прецедентами используются на диаграммах прецедентов?
7. На какие 3 типа можно подразделять экторов?
8. Что представляет (описывает, отображает) прецедент?
9. Какие типы связей (отношений) допускаются между экторами?

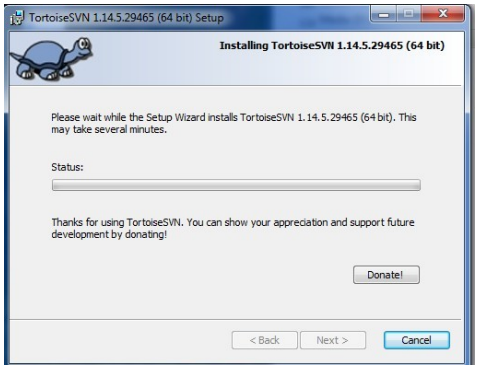
ЛАБОРАТОРНАЯ РАБОТА № 4 (2 часа)

Тема: Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)

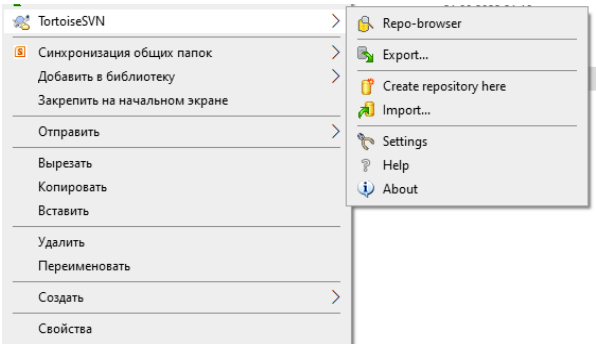
Цель: Изучить на практике понятия и компоненты систем контроля версий (СКВ), приемы работы с ними.

Технология выполнение работы:

1. Установите TortoiseSVN на компьютере.



Сделайте правый клик на папке в проводнике, и вы увидите новые пункты в контекстном меню, такие как эти:



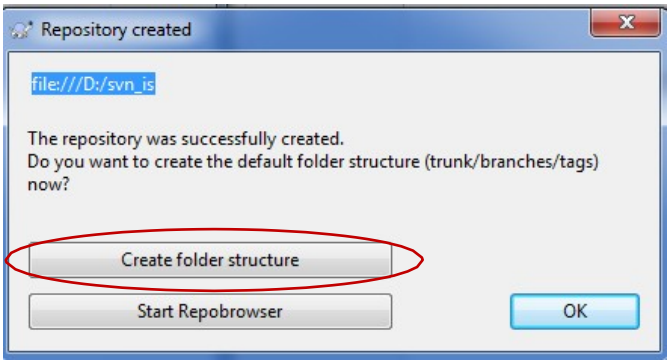
Создание хранилища

Сначала создайте новую пустую директорию на вашем ПК.

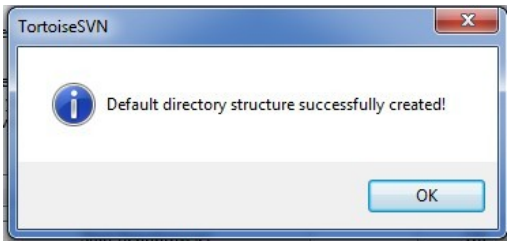
Например, C:\svn_is.

Теперь сделайте правый клик на новой папке и в контекстном меню выберите

b. ***TortoiseSVN → Создать здесь хранилище...***



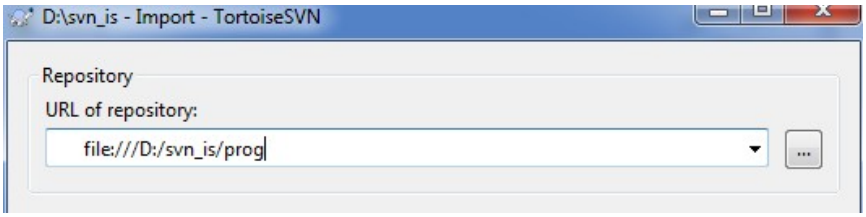
Хранилище, созданное внутри папки, готово к использованию. Создайте внутреннюю структуру папок нажав кнопку ОК.



Импорт проекта

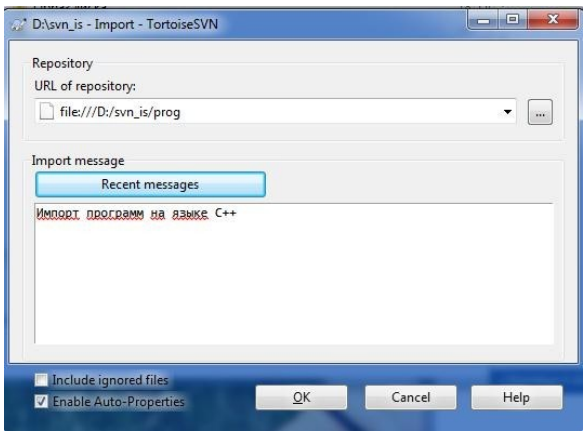
Сейчас у нас есть хранилище, но оно совершенно пустое в данный момент. Добавьте свои файлы в папку **prog** (путь C:\Projects\prog).

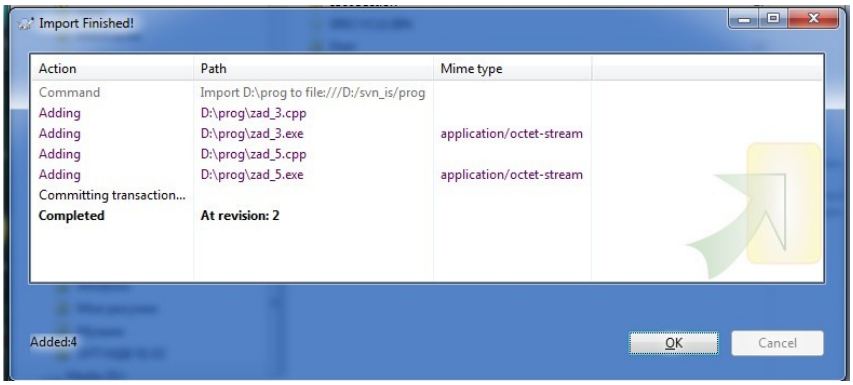
Перейдите к папке **prog** в Проводнике и сделайте правый клик на ней. Теперь выберите пункт **TortoiseSVN → Импорт... (Import)**, который вызовет диалог



Другая важная функция данного диалога - это окно **Сообщение импорта (Import messages)**, в которое вы можете добавить сообщение о том, что вы делаете. Когда вам понадобится просмотреть историю проекта, эти сообщения будут ценным подспорьем для просмотра какие изменения и когда были сделаны.

Введите и импортируйте сообщение, например, «Импорт программ на языке C++», затем нажмите ОК.



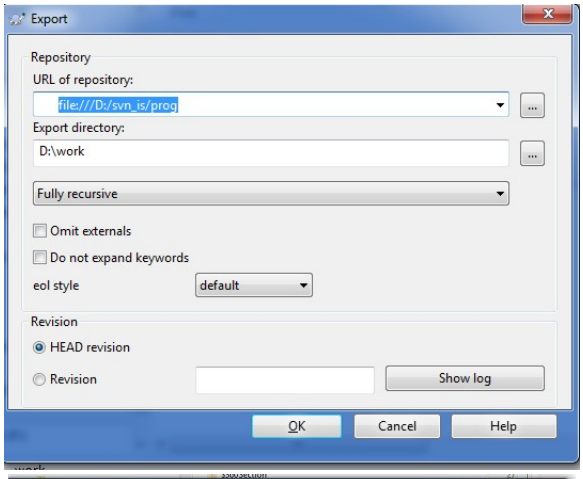


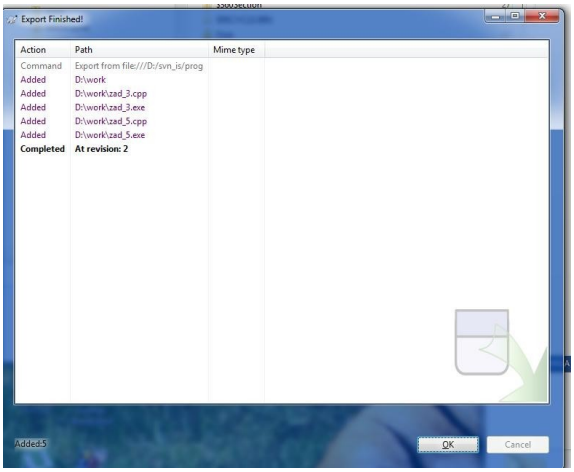
Извлечение рабочей копии

В хранилище создайте рабочую копию для повседневной работы. Для создания свежей рабочей копии в Subversion используется термин **Извлечь**.

Извлеките папку **work** из нашего хранилища в папку для разработки называемую C:\Projects\work. Создайте эту папку, затем сделайте правый клик на ней и выберите пункт **TortoiseSVN → Извлечь... (Export)**.

Введите URL-адрес для извлечения, в данном случае **file:///c:/SVN_IS/work** и кликните на OK. Наша папка для разработки заполнится файлами из хранилища.

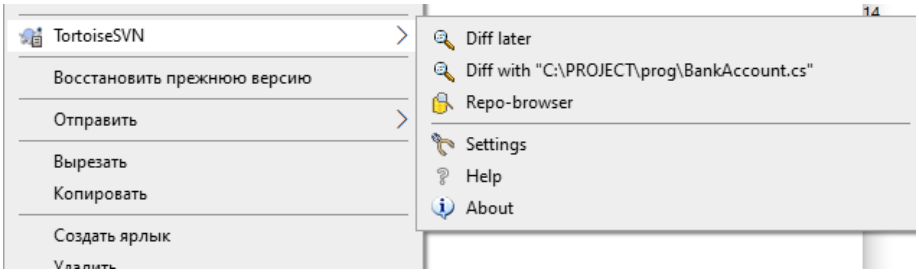


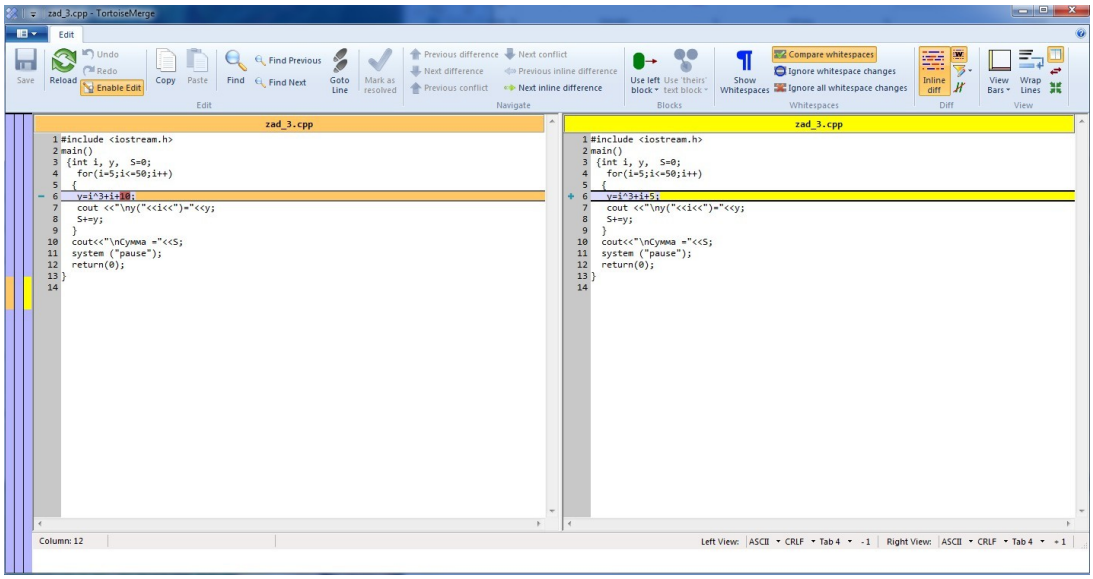


Внесение изменений

Можно приступать к работе. В папке work измените файлы, например, внесите изменения в файл zad3.c.

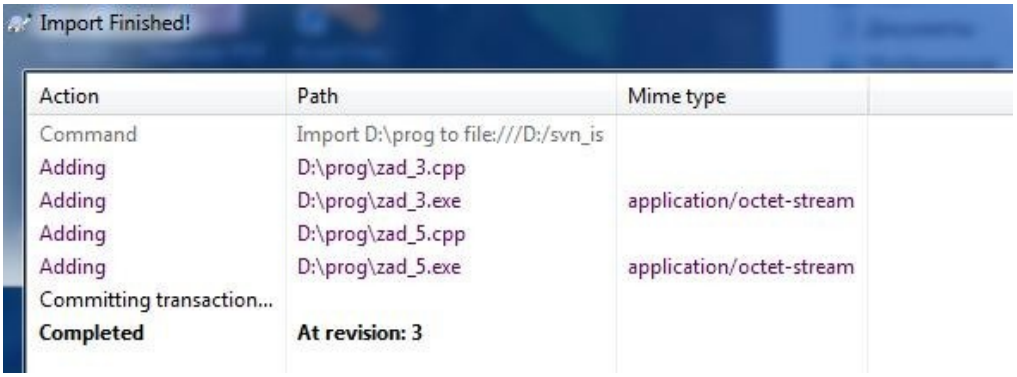
Нажмите правой кнопкой на одном из изменѐнных файлов и выберите команду **TortoiseSVN → Различия**. Запустится инструмент TortoiseSVN для сравнения файлов и покажет какие точно строки в файлах были изменены.





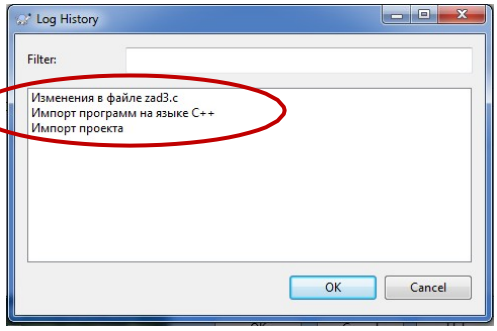
Добавление новых файлов

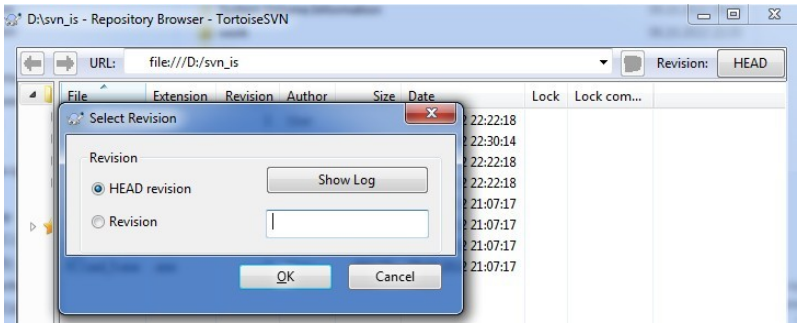
Во время работы над проектом, вам понадобится добавлять новые файлы. Нажмите правой кнопкой на папке и выберите команду **TortoiseSVN → Добавить**. Диалог добавления показывает все неверсированные файлы и вы можете выбрать те файлы, которые вы хотите добавить.

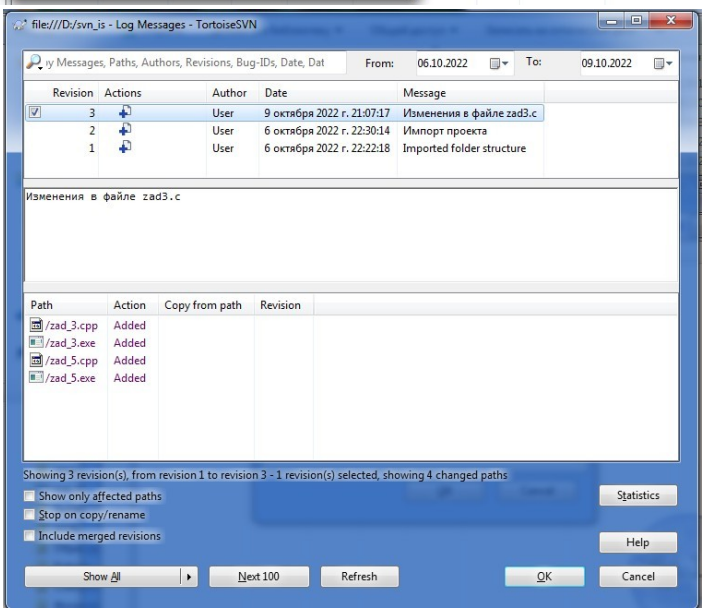


Просмотр истории проекта

Одной из самых полезных функций TortoiseSVN является диалоговое окно журнала. Оно показывает список всех фиксаций изменений в файле или папке, а также все те детальные сообщения, которые вы вводили при фиксации изменений.







Верхняя панель показывает список всех фиксированных ревизий вместе с началом сообщения фиксации. Если вы выберите одну из этих ревизий, то средняя панель отобразит полное сообщение журнала для той ревизии и нижняя панель покажет список измененных файлов и папок.

Отмена изменений

Одной общей функцией всех систем управления ревизиями является функция, которая позволяет вам отменить изменения, которые вы внесли ранее. Если вы хотите избавиться от изменений, которые вы еще не успели фиксировать и восстановить нужный файл в том виде, в котором он был перед началом изменений, то выберите команду **TortoiseSVN → Убрать изменения**. Это действие отменит ваши изменения (в Корзину) и вернет фиксированную версию файла, с которой вы начинали. Если же вы хотите убрать лишь некоторых изменения, то вы можете использовать инструмент TortoiseMerge для просмотра изменений и выборочного удаления измененных строк.

Если вы хотите отменить действия определѐнной ревизии, то начните с диалогового окна журнала и найдите проблемную ревизию. Выберите команду **Контекстное меню → Отменить изменения из этой ревизии** и те изменения будут отменены.

Контрольные вопросы:

1. Что такое Subversion?
2. Перечислите возможности TortoiseSVN.
3. Что такое системы контроля версий (СКВ) и для решения каких задач они предназначаются?
4. Объясните следующие понятия СКВ и их отношения: хранилище, commit, история, рабочая копия.
5. Что представляют собой и чем отличаются централизованные и децентрализованные СКВ? Приведите примеры СКВ каждого вида.

ЛАБОРАТОРНАЯ РАБОТА № 5 (2 часа)

Тема: Разработка и интеграция модулей проекта (командная работа)

Цель: изучить принципы разработки и интеграции модулей проекта и принципы формирования и управления командной работы.

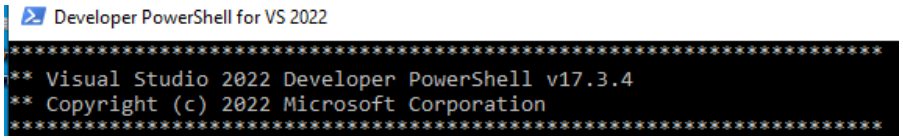
Технология выполнения работы

1. Выполнить поиск командной строки на компьютере двумя способами:

1. Способ - В Windows 10:

1. В поле поиска на панели задач начните вводить имя средства, например dev или developer command prompt. Откроется список установленных приложений, которые соответствуют вашему шаблону поиска.

2. *Выберите* ***Командная строка разработчика для VS 2022*** *или* ***PowerShell для разработчиков для VS 2022****.*



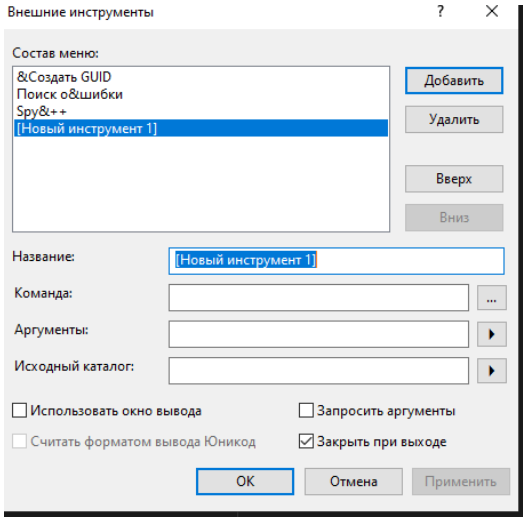
2. Запуск командной строки из Visual Studio.

1. Запустите Visual Studio.

2. Выберите опцию *Продолжить без кода*

3. Выберите меню *Средства* и затем выберите в нем пункт *Внешние инструменты*.

4. В диалоговом окне *Внешние инструменты* нажмите кнопку *Добавить*. Появится новый элемент.



5. Введите название для нового пункта меню, например, Command Prompt.

6. В поле Команда укажите файл, который должен запускаться, например, C:\Windows\System32\cmd.exe.

7. В поле Аргументы укажите, где найти конкретную командную строку, которую вы хотите использовать, например:

/k "C:\Program Files (x86)\ Microsoft VisualStudio\2017\Enterprise\Common7\Tools\ VsDevCmd.bat"

эта команда запускает командную строку разработчика, установленную с Visual Studio 2017 Enterprise). Измените это значение в соответствии с вашей версией, выпуском и расположением установки Visual Studio.

8. Выберите значение для поля Исходный каталог, например, Каталог проекта.

9. Нажмите кнопку ОК. Новый элемент добавится в меню.

10. Вызовите командную строку из меню *Средства* или в меню *Пуск*

11. Создайте папки project\myProj на диске C:

12. Сделайте текущим папку myProj

Выполните команды:

```
C:\>mkdir project

C:\>cd project

C:\project>mkdir myProj

C:\project>cd myProj
```

13. Создайте файл с помощью команды **Copy con prog.cpp, введите текст**

```
#include <iostream>
using namespace std;
int main()
{
   cout <<" Hell, word, from C++! ";
    cout << endl;
}
```

14. Нажмите Ctrl + Z, чтобы сохранить файл, и Ctrl + C, чтобы выйти из редактирования.

15. Введите команду dir, чтобы получить список содержимого каталога c:\project\myProj

```
 Содержимое папки C:\project\myProj

23.10.2022  19:11    <DIR>          .
23.10.2022  19:11    <DIR>          ..
23.10.2022  18:53               298 prog.cpp
               1 файлов            298 байт
               2 папок  210 774 020 096 байт свободно
```

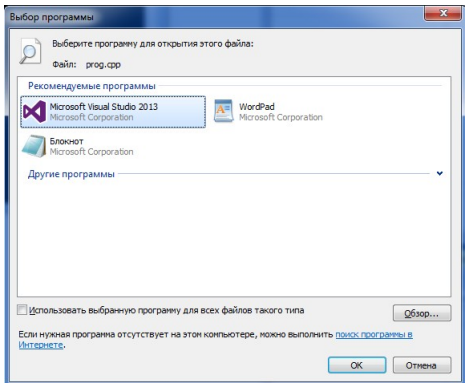
16. Скомпилируйте программу с помощью команды

cl /EHsc prog.cpp

Компилятор cl.exe создаст OBJ-файл, содержащий скомпилированный код, а затем запустит компоновщик для создания исполняемой программы с именем prog.exe. Это имя отображается в строках информации, выводимой компилятором. Если вы получаете сообщение об ошибке, например, "cl не распознается как внутренняя или внешняя команда, исполняемая программа или пакетный файл", ошибке C1034 или LNK1104, командная строка разработчика настроена неправильно.

17. Запустите программу prog.exe, если компиляция прошла успешна, набрав в командной строке **prog**

18. Откройте файл выполнив команду `C:\project\myProj>prog.cpp` Выберите в окне программу для открытия файла.



Контрольные вопросы

1. Как происходит управление проектом по стандарту PMBoK?

2. Как с целью структуризации управления проектом распределяются процессы управления проектом?

3. Понятие интеграции процессов управления.

4. Какие главные задачи решаются при организации работы над проектом?

5. Что понимается под командой проекта, как и на какой период она формируется?

6. Дайте характеристику двум основным принципам формирования команды для управления проектом.

ЛАБОРАТОРНАЯ РАБОТА № 6 (4 часа)

Тема: Отладка отдельных модулей программного проекта

Цель: усвоить знание основ модульного программирования; освоить способы создания и применения модулей.

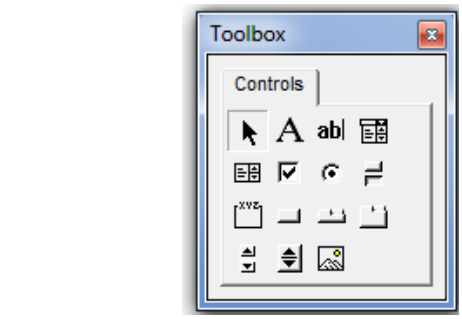
Задание:

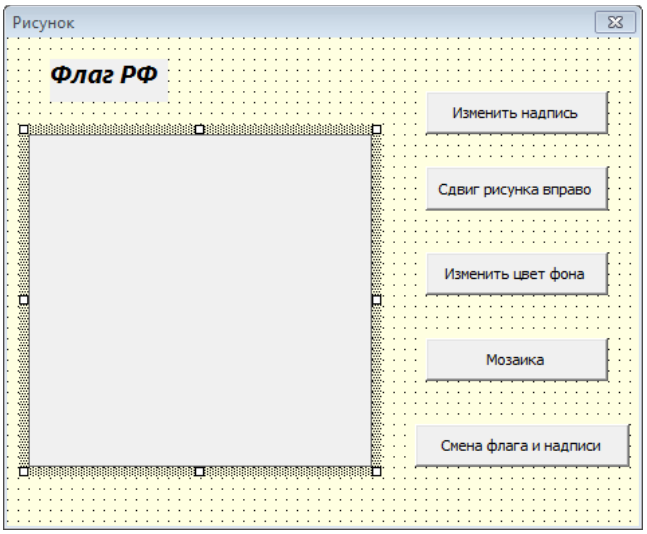
1. Провести отладку отдельных модулей программного проекта

Технология выполнение работы:

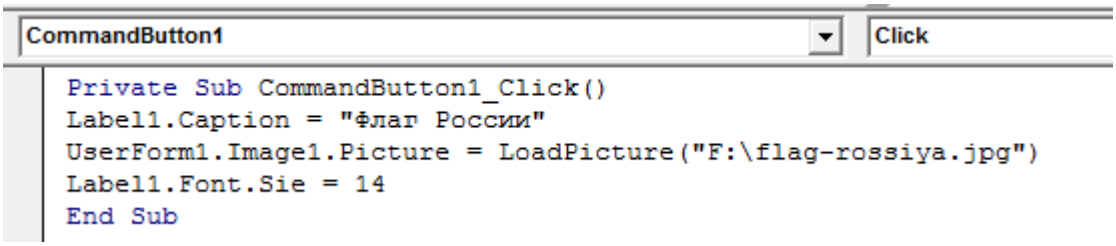
Задание 1.

1. Откройте новую рабочую книгу
2. Запустите редактор VB
3. Создайте форму пользователя с помощью командой Insert-FormUser, используя инструменты ToolBox и свойства Properties. Добавьте на форму элементы управления: 4 CommandButton, Image и Label.





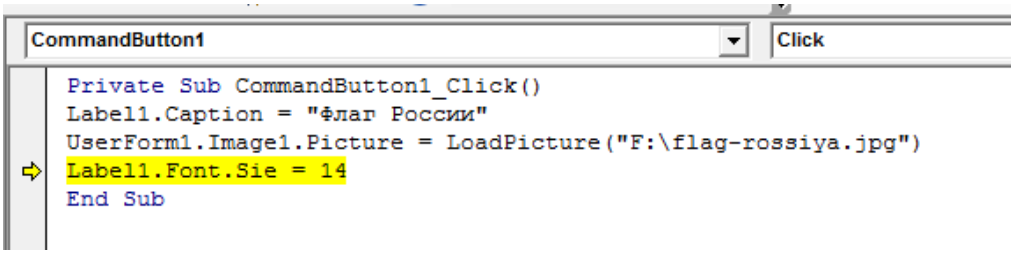
4. Создайте новую процедуру для кнопки «Измени надпись».
5. Введите текст процедуры. Правильно укажите адрес графического файла. В тексте намеренно сделаем ошибку в свойстве Size (напишем Sie):



6. Выполните компиляцию и выведите форму для работы.

7. После появления формы на экране нажмем на кнопку «Измени надпись». Так как в программе заложена ошибка, появится окно сообщения об ошибке, и открывается редактор VBA.

8. Нажмите на кнопку «Debug» (отладка), и отладчик укажет, в какой строке у вас ошибка



9. Исправьте ошибку и нажмите на стандартной панели инструментов на кнопку  («Продолжение»).

Тексты программ для кнопок CommandButton2, CommandButton3, CommandButton4, CommandButton5 представлены ниже:

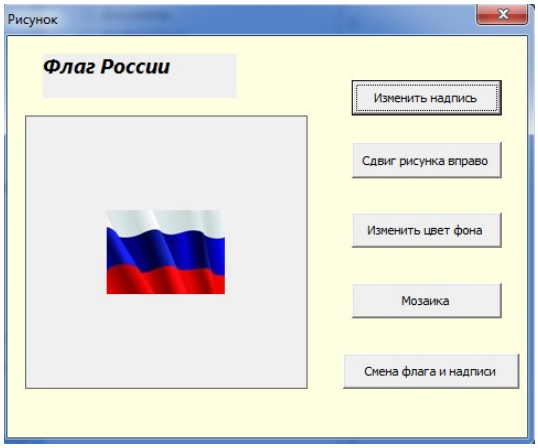
```
Private Sub CommandButton2_Click()
Image1.PictureAlignment = 4
End Sub
```

```
Private Sub CommandButton3_Click()
Image1.BackColor = &HFF80FF
UserForm1.BackColor = RGB(64, 0, 0)

End Sub
```

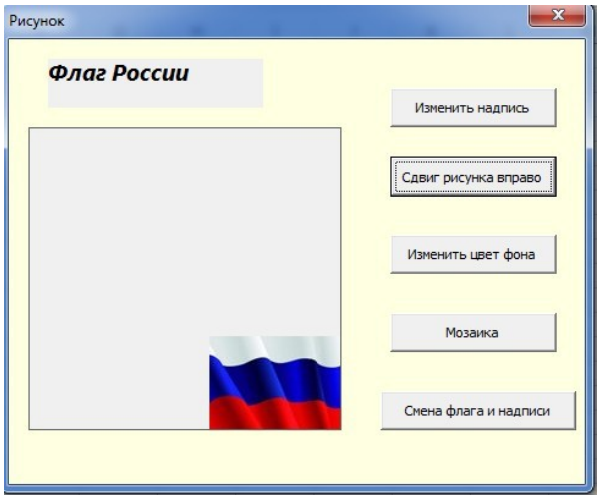
```
Private Sub CommandButton4_Click()
Image1.PictureTiling = True
End Sub
```

```
Private Sub CommandButton5_Click()
Label1.Caption = "Флаг Белоруссии"
Label1.Font.Size = 14
Label1.Font.Name = "Arial Black"
UserForm1.Image1.Picture = LoadPicture("F:\флагбел.gif")
End Sub
```

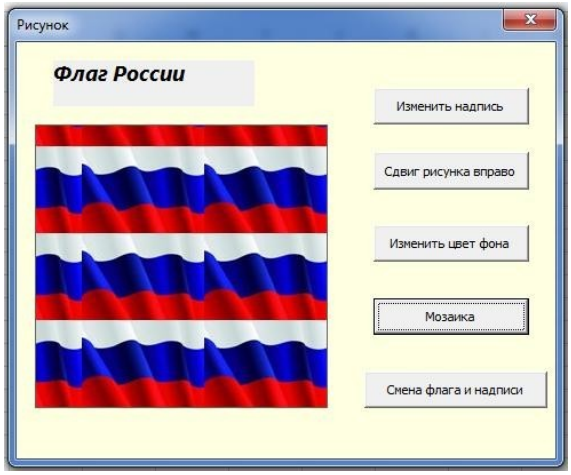
10. После щелчка по кнопке «Измени надпись» форма приобретет вид, представленный на рисунке ниже.



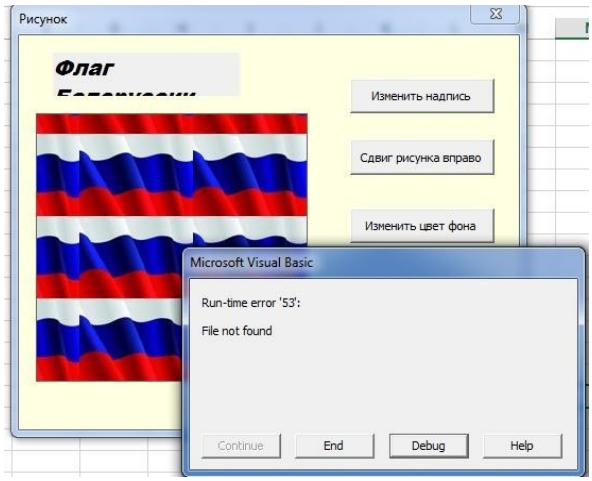
11. После щелчка по кнопке «Сдвинь рисунок вправо» форма приобретет вид, представленный на рисунке.



12. После щелчка по кнопке «Мозаика» форма приобретет вид



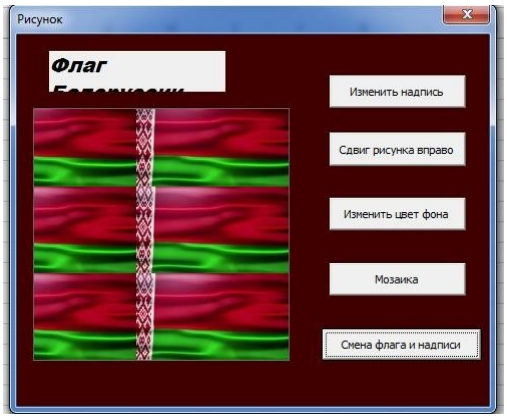
13. После щелчка по кнопке «Смена флага и надписи» появится сообщение об ошибке.



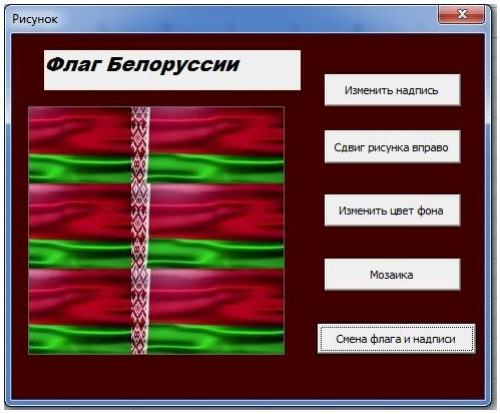
```
Private Sub CommandButton5_Click()
Label1.Caption = "Флаг Белоруссии"
Label1.Font.Name = "Arial Black"
UserForm1.Image1.Picture = LoadPicture("F:\флагбел.gif")
End Sub
```

14. Исправьте ошибку в имени файла (расширение jpg)/

```
Private Sub CommandButton5_Click()
Label1.Caption = "Флаг Белоруссии"
Label1.Font.Name = "Arial Black"
UserForm1.Image1.Picture = LoadPicture("F:\флагбел.jpg")
End Sub
```



15. Измените размер надписи. Проверьте работу программы.



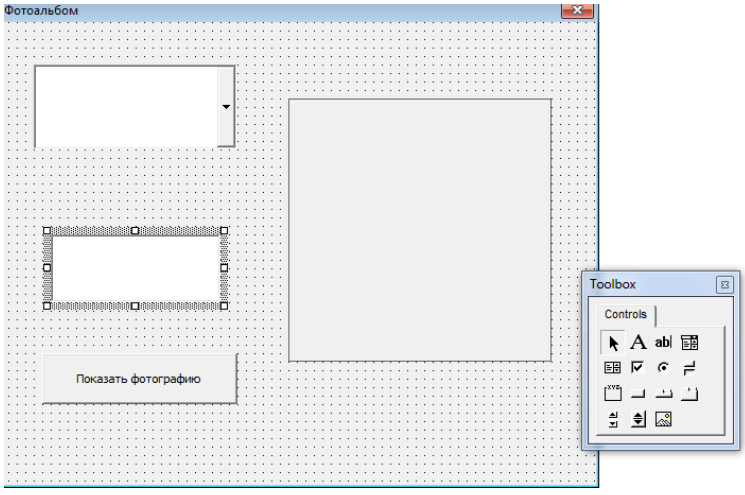
16. Сохраните свою работу.

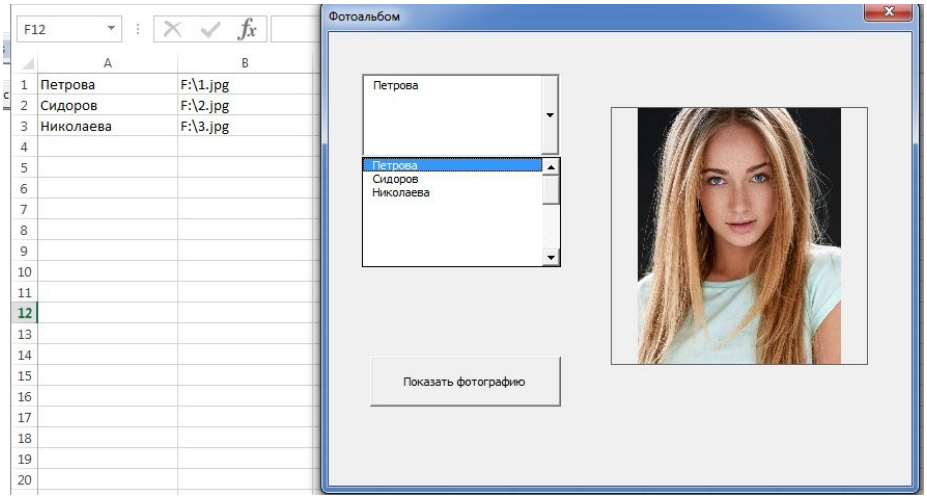
Задание 2.

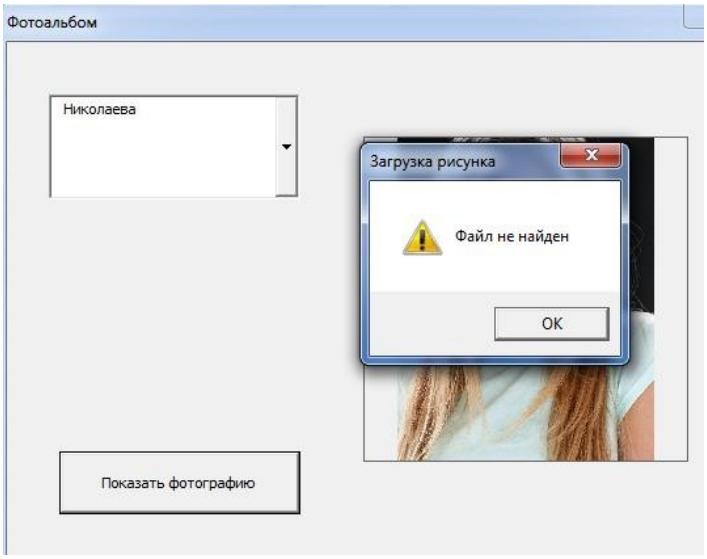
1. Напишите код на программный продукт с использованием редактора кода VBA.
2. Проведите отладку программного продукта.

Постановка задачи: Создать программу, которая показывает фотографии однокурсников. Фамилии студентов выбираются из раскрывающегося списка ComboBox и при нажатии кнопки «Показать фотографию» фотография выбранного студента отображается в области рисунка.

Для Image1 в свойство PictureSizeMode установить значение fmPictureSizeModeZoom. Фамилии студентов записаны на рабочем листе в столбце А, откуда они заносятся в раскрывающийся список ComboBox. Полные имена файлов, содержащих фотографии записаны в столбце В, откуда они заносятся в список ListBox. Список ListBox сделать невидимым (Enabled=False). При запуске программы раскрывающемся списке должна отображаться фамилия первого в списке студента, а в области рисунка – его фотография. Предусмотреть обработку ошибочных ситуаций. Если имя файла или путь к файлу заданы неверно, вывести об этом сообщение.







Порядок выполнения программы.

Модуль формы состоит из трех процедур:

- процедура обработки ошибочной ситуации;
- процедура обработки события Initialize объекта Userform;
- процедура обработки события CommandButton1_ Click.

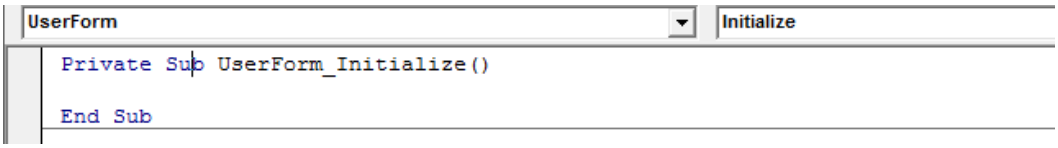
Процедура Обработка_ошибки с помощью оператора выбора анализирует код ошибки. Если имя файла задано неверно, при загрузке рисунка возникает ошибка с кодом 53, если путь к файлу задан неверно - ошибка с кодом 76. Обработчик ошибок выводит соответствующее сообщение. При появлении других выполнение программы прерывается, пользователь информируется об ошибке. Осуществляется выход из процедуры.

Процедура обработки ошибочной ситуации

```
Sub Обработка_ошибки()
Select Case Err.Number
Case 53
MsgBox "Файл не найден", 48, "Загрузка рисунка"
Case 76
MsgBox "Путь не найден", 48, "Загрузка рисунка"
Case Else
MsgBox "Произошла ошибка: " & Err.Description
End Select
End Sub
```

Процедура обработки события Initialize объекта заполняет список ComboBox из диапазона A1:A20, в который предварительно введены фамилии студентов, список ListBox – из диапазона B1:B20 полными именами файлов. В раскрывающемся списке отображается фамилия первого в списке студента, а в области рисунка - его фотография. При возникновении ошибки

при загрузке рисунка вызывается процедура Обработка_ошибки.




```
Private Sub UserForm_Initialize()
' передача управления на обработчик ошибок,
' помеченный меткой строка1
On Error GoTo строка1
' Список ListBox делаем невидимым
ListBox1.Visible = False
With ListBox1
.ColumnCount = 1
.RowSource = ("B1:B20")
End With
With ComboBox1
.ColumnCount = 1
.RowSource = ("A1:A20")
' Отображение фамилии первого в списке студента
.ListIndex = 0
End With
' Отображение фотографии первого в списке студента
Image1.Picture = LoadPicture(ListBox1.List(0, 0))
Exit Sub
' Обработчик ошибок
строка1:
' вызов процедуры обработки ошибок
Call Обработка_ошибки
End Sub
```

Процедура обработки события **CommandButton1** фотографию выбранного из раскрывающегося списка отображает в области рисунка. Если при загрузке рисунка возникает ошибка, вызывается процедура Обработка_ошибки.

```
Private Sub CommandButton1_Click()
Dim i As Integer
' передача управления на обработчик ошибок,
' помеченный меткой Строка2
On Error GoTo Строка2
i = ComboBox1.ListIndex ' индекс выбранного элемента
If ListBox1.List(i, 0) = "" Then
MsgBox "Не задано имя файла", 48, "Загрузка рисунка"
End If
Image1.Picture = LoadPicture(ListBox1.List(i, 0))
'Выход из процедуры при успешном выполнении загрузки рисунка
Exit Sub
' Обработчик ошибок
Строка2:
' вызов процедуры обработки ошибок
Call Обработка_ошибки
End Sub
```

Контрольные вопросы:

1. Какие ошибки в программах существуют?
2. Что понимают под отладкой программы?
3. Чем отладка отличается от тестирования?

ЛАБОРАТОРНАЯ РАБОТА № 7 (2 часа)

Тема: Организация обработки исключений

Цель: изучить операторы, используемые при обработке исключительных ситуаций, возникающих во время выполнения вычислительных процессов, получить практические навыки в составлении программ.

Технология выполнения работы:

Задание - Оптимизировать программу, производящую простые арифметические операции над числами (сложение, вычитание, умножение и деление), используя обработку исключительных ситуаций (обработка ввода чисел и операции деления).

Задание 1. Реализовать программу простейшего калькулятора

Технология выполнения задания:

1 Запустите программу Microsoft Studio

2 Создайте консольное приложение на языке C#

```
using System;
namespace ConsoleApplication
{
    class OurClass
    {
        static void Main(string[] args)
        {
            float num1 = 1, num2 = 2, summarize, multiply, sub, divide = 0;
            Console.Write("Введите первое число:");
            try { num1 = float.Parse(Console.ReadLine()); }
            catch
            {
                Console.WriteLine("Неправильный формат числа!\n" +
                    "В качестве значения первого числа будет 1");
            }
            Console.Write("Введите второе число:");
            try { num2 = float.Parse(Console.ReadLine()); }
            catch
            {
                Console.WriteLine("Неправильный формат числа!\n" +
                    "В качестве значения второго числа будет 2");
            }
            summarize = num1 + num2; multiply = num1 * num2; sub = num1 - num2;
            try { divide = num1 / num2; }
            catch (DivideByZeroException)
            {
                Console.WriteLine("Нельзя делить на нуль!");
            }
            Console.WriteLine(
            "\n" + num1 + " + " + num2 + " = " + summarize +
          "\n" + num1 + " * " + num2 + " = " + multiply +
          "\n" + num1 + " - " + num2 + " = " + sub +
          "\n" + num1 + " / " + num2 + " = " + divide);
            Console.Write("\nДля выхода из программы нажмите [Enter]:");
            string anykey = Console.ReadLine();
        }
    }
}
```

```
Введите первое число:5
Введите второе число:6

5 + 6 = 11
5 * 6 = 30
5 - 6 = -1
5 / 6 = 0,8333333

Для выхода из программы нажмите [Enter]:
```

```
Введите первое число:0.45
Неправильный формат числа!
В качестве значения первого числа будет 1
Введите второе число:0.12
Неправильный формат числа!
В качестве значения второго числа будет 2

1 + 2 = 3
1 * 2 = 2
1 - 2 = -1
1 / 2 = 0,5

Для выхода из программы нажмите [Enter]:
```

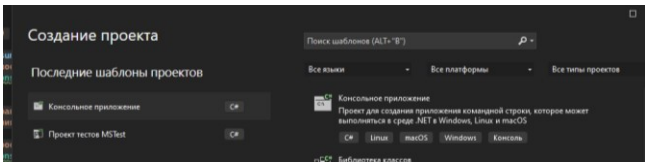
```
Введите первое число:5
Введите второе число:0

5 + 0 = 5
5 * 0 = 0
5 - 0 = 5
5 / 0 = ?

Для выхода из программы нажмите [Enter]:
```

Задание 2. Реализовать программу вычисления периметра четырехугольника

Технология выполнения задания:

1 Запустить программу Microsoft Studio

2 Создать консольное приложение на языке C#




```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            double x1, y1, x2, y2, x3, y3;
            x1 = GetDouble("x1");
            x2 = GetDouble("x2");
            x3 = GetDouble("x3");
            y1 = GetDouble("y1");
            y2 = GetDouble("y2");
            y3 = GetDouble("y3");
             //здесь обработка исключений не требуется, т.к. даже если при вычитании получаются
            //отрицательные числа, то после возведения в квадрат, они будут положительными,
            //т.е. функция корня Sqrt будет всегда обрабатывать только положительное или нулевое значение
            double AB = Math.Sqrt(Math.Pow(x2 - x1, 2) + Math.Pow(y2 - y1, 2));
            double BC = Math.Sqrt(Math.Pow(x3 - x2, 2) + Math.Pow(y3 - y2, 2));
            double AC = Math.Sqrt(Math.Pow(x1 - x3, 2) + Math.Pow(y1 - y3, 2));
            double P = AB + BC + AC;
            Console.Write(" периметр = " + P.ToString());
            Console.Write("\nДля выхода из программы нажмите [Enter]:");
            string ankey = Console.ReadLine();
        }
            static double GetDouble(string varname)
        {   bool IsDouble = false;
            double ReturnValue = 0;

            Console.Write("Введите переменую {0}:", varname);
            while (!IsDouble)
            {
                try
                {
                    ReturnValue = double.Parse(Console.ReadLine());
                    IsDouble = true;
                }
                catch
                {
                    Console.WriteLine("Введеное число не является типом double, попробуйте еще раз!");
                }
            }
            return ReturnValue;
        }
    }
}
```

3 Протестировать программу

```
Введите переменую x1:45
Введите переменую x2:64
Введите переменую x3:67.8
Введеное число не является типом double, попробуйте еще раз!
70
Введите переменую y1:52
Введите переменую y2:58.5
Введеное число не является типом double, попробуйте еще раз!
58
Введите переменую y3:102
 периметр = 120,23376490501505
Для выхода из программы нажмите [Enter]:
```

Контрольные вопросы:

1. Как создается защищенный блок кода?
2. Как описывается процедура обработки конкретного исключения?
3. Как генерируется исключение?
4. Как можно ограничить список исключений, которые могут генерироваться в функции?

ПРАКТИЧЕСКАЯ РАБОТА № 1 (2 часа)

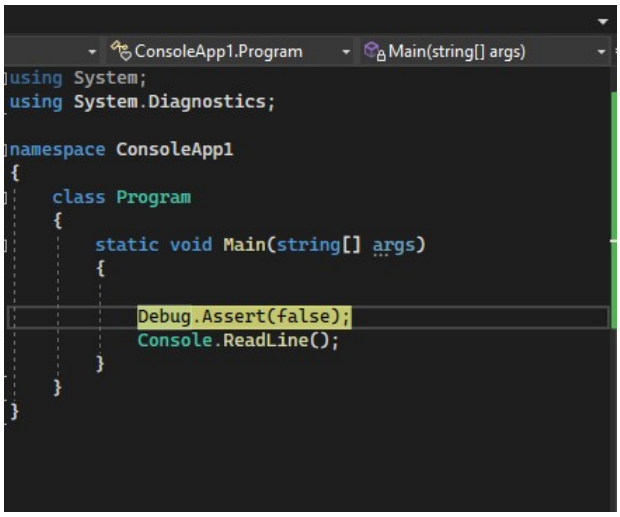
Тема: Применение отладочных классов в проекте

Цель: изучение свойств и методов встроенных классов отладки проектов и применения этих классов для отладки программ

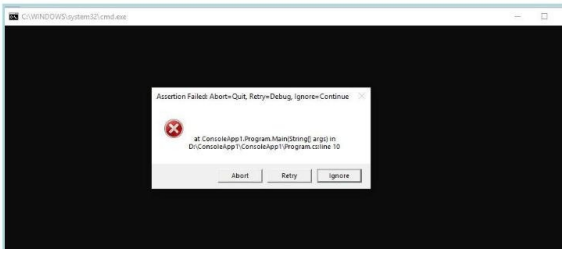
Задание 1 - Создать приложение на C# для отладки кода с использованием Debug.Assert

Технология выполнения работы

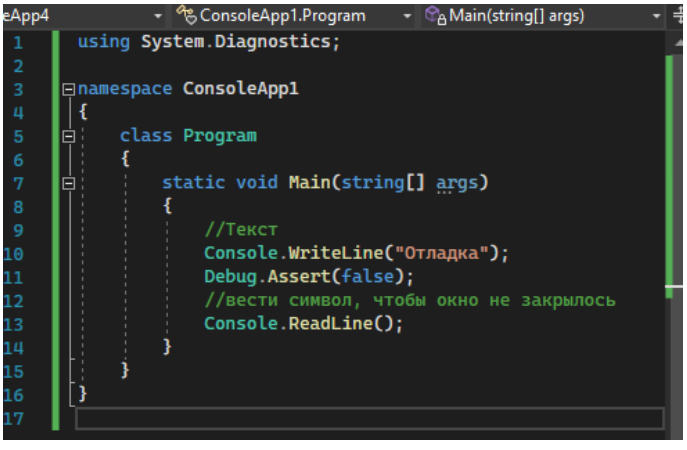
1. Создайте новое консольное приложение C#
2. Введите код программы



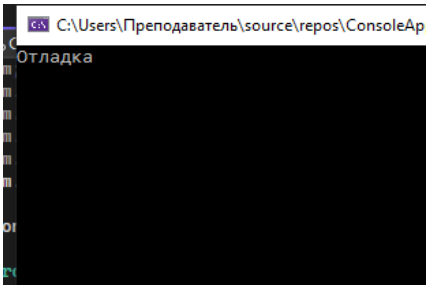
3. Выполните программу
4. Объясните полученный результат



5. Добавьте текст "Отладка"



6. Выполните программу.



7. Объясните полученный результат
8. Сделайте выводы о работе Debug.Assert

Задание 2 - Рассмотреть пример работы, в котором отладочная информация направляется в разные каналы – окно вывода, консоль, файл

Постановка задачи: Требуется реализовать экземпляр **TextWriterTraceListener** класса, который использует **StreamWriter** вызов **myOutputWriter** для записи в файл с именем TestFile.txt. Класс **TextWriterTraceListener** предоставляет **Writer** свойство для получения или задания модуля записи текста, получающего выходные данные трассировки или отладки. Сначала в программе создается файл для вывода. Затем он создает StreamWriter первый модуль записи текста, присваивает ему выходной файл и добавляет его в Listeners. Затем код выводит одну строку текста в файл. После этого очищается выходной буфер.

c. Технология выполнения задания:

1. Запустите VS22
2. Создайте консольное приложение C#
3. Введите код программы

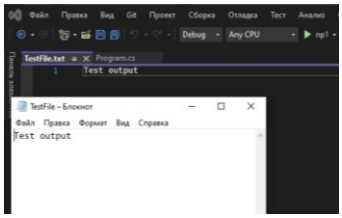
```
using System.Diagnostics;
public class Sample
{

   public static int Main(string[] args)
   {
      // Create a file for output named TestFile.txt.
      Stream myFile = File.Create("TestFile.txt");

      /* Create a new text writer using the output stream, and add it to
       * the trace listeners. */
      TextWriterTraceListener myTextListener = new
         TextWriterTraceListener(myFile);
      Trace.Listeners.Add(myTextListener);

      // Write output to the file.
      Trace.Write("Test output ");

      // Flush the output.
      Trace.Flush();

      return 0;
   }
```

4. Выполните программу

```
Консоль отладки Microsoft Visual Studio
C:\Users\Преподаватель\source\repos\np1\np1\bin\Debug\net6.0\np1.exe (процесс 15264) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "А
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Код 0 – Успешное завершение работы программы

5. Откройте папку и проверьте создание файла TestFile.txt
6. Откройте файл TestFile.txt, чтобы просмотреть выходные данные.



Контрольные вопросы

1. В чем заключается сущность объектно-ориентированного подхода при разработке программного продукта?
2. Какие техники обеспечения качества используются в процессе конструирования?
3. Что такое отладчик?
4. Дайте характеристику свойств и методов классов Debug и Trace.
5. В чем сходство и различие классов Debug и Trace?
6. Основные принципы метода доказательства правильности программ – метода Флойда.
7. Дайте характеристику свойств и методов классов StackTrace и BooleanSwitch.

ЛАБОРАТОРНАЯ РАБОТА № 8 (4 часа)

Тема: Отладка проекта

Цель: изучить действия необходимые для отладки проекта.

Технология выполнения:

Рассмотрим действия для отладки проекта.

1. Установка точки останова. В файле с исходным кодом программы нужно установить курсор на строчку, где требуется остановка. После этого выполнить одно из действий:

– ***Отладка◊Точка останова;***

– нажать горячую клавишу F9;

– ПКМ – Вставить точку останова.

```
#include "stdafx.h"
#include <SDKDDKVer.h>
#include <stdio.h>
int _tmain(int argc, _TCHAR* argv[])
{
    int a, b, s;
    printf("a=");
    scanf_s("%d", &a);
    printf("b=");
    scanf_s("%d", &b);
    s = a + b;
    printf("S=a+b=%d\n", s);
    return 0;

}
```

2. Запуск программы до ближайшей точки останова. После указания точки останова можно запустить программу: команда **Отладка◊Начать отладку** или нажать горячую клавишу F5.

Программа начнет выполняться до тех пор, пока не будет достигнута точка останова. О достижении этой точки будет свидетельствовать стрелочка, отображаемая поверх точки останова. В этой точке выполнение программы будет остановлено до указания дальнейших действий.

```
#include "stdafx.h"
#include <SDKDDKVer.h>
#include <stdio.h>
int _tmain(int argc, _TCHAR* argv[])
{
    int a, b, s;
    printf("a=");
    scanf_s("%d", &a);
    printf("b=");
    scanf_s("%d", &b);
    s = a + b;
    printf("S=a+b=%d\n", s);
    return 0;

}
```

100 %

Локальные

Имя	Значение	Тип
argc	1	int
argv	0x0053afe8 {0x0053aff0 L"C:\\Users\\User\\Docume	wchar_t
b	20	int
s	-858993460	int
a	10	int

3. Выполнение программы по шагам. Выполнение программы по шагам можно осуществить следующим образом. Команда **Отладка◊Шаг с обходом** выполняет одну строчку программы. Если очередным шагом программы является вызов функции, то программист может проанализировать работу этой функции. Для этого в режиме отладки используется команда **Отладка◊Шаг с заходом** или горячая клавиша F11. После этой команды отладчик заходит «внутрь» функции и программист может продолжить пошаговое выполнение команд.

Если отладка функции закончена и необходимо вернуться в вызывающую программу, то можно использовать команду **Отладка ◊ Шаг с выходом** или сочетание клавиш Shift+F11.

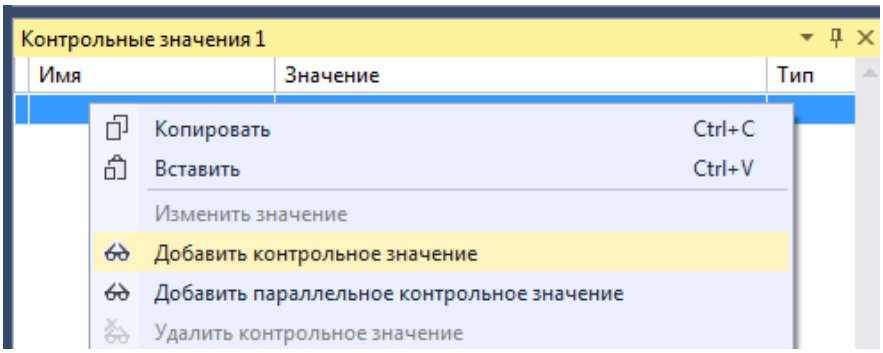
Видимые

Имя	Значение	Тип
&b	0x002afb9c {20}	int *
a	10	int
b	20	int
s	30	int

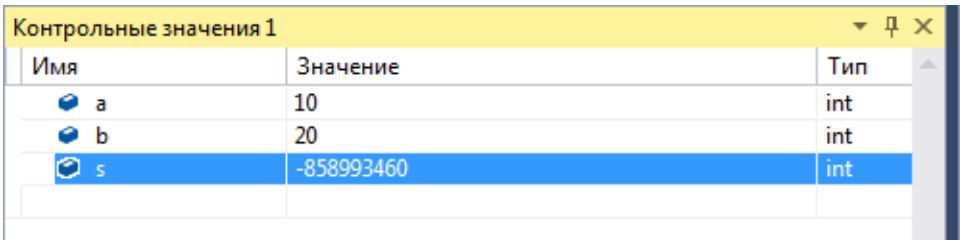
```
C:\Windows\system32\cmd.exe
a=10
b=20
S=a+b=30
Для продолжения нажмите любую клавишу . . .
```

4. Просмотр значения переменных в процессе выполнения программы. В любой момент отладки программист может просмотреть значения переменных, используемых в программе. Для этого используется нижняя часть окна среды Microsoft Visual Studio.

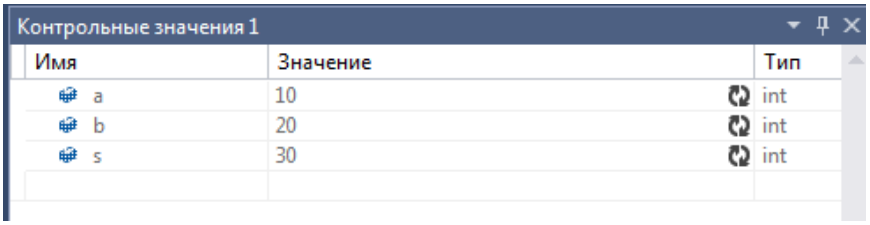
Откройте окно Контрольные значения, выбрав **Отладка◊Окна ◊Контрольные значения◊Контрольные значения 1.** Вы можете открыть дополнительные **окна контрольных значений**, выбрав окна **2**, **3** или **4**.



Для добавления переменной необходимо сделать ее активной и ввести с клавиатуры имя переменной. Если указанная переменная имеется в программе, то ее значение будет автоматически отражено в поле Значение.



Продолжите отладку, выбрав **Отладка ◊ Шаг с заходом** или нажав клавишу **F11** по мере необходимости для перехода. В процессе выполнения значения переменных в окне **Контрольные значения 1** должны меняться.



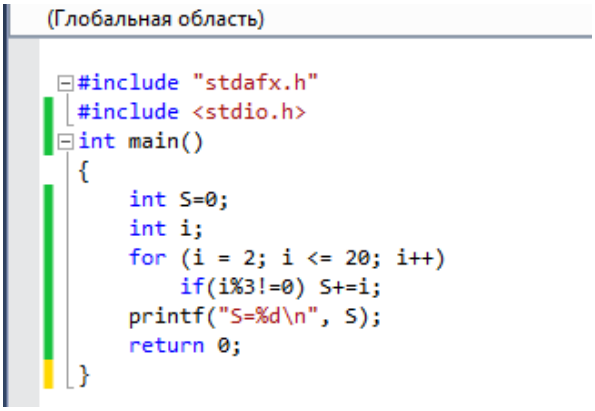
Таким образом, используя описанные приемы, программист может тщательно проанализировать выполнение написанной программы, проследить изменения значений используемых переменных, устранить некорректную работу реализованных алгоритмов и добиться правильного выполнения программы.

Задание 1 – Отладка программы

Постановка задачи: Вычислить факториал n!=1·2·3…n

Технология выполнения:

1. Создайте консольное приложение на C++
2. Введите текст программы



3. Установите точки останова

```
#include "stdafx.h"
#include <stdio.h>
int main()
{
    long int F;
    int i, N;
    printf("N=");
    scanf_s("%d", &N);
    F = 1;
    for (i = 1; i <= N; i++) F = F*i;
    printf("N!=%d\n", F);
    return 0;

}
```

4. Запустите программу до ближайшей точки останова
5. Выполните программу по шагам
6. Добавьте переменные для наблюдения их значений на вкладку **Отладка◊Окна**

Контрольные значения 4

Имя	Значение	Тип
i	-858993460	int
F	1	long

◊Контрольные значения

7. Проанализируйте алгоритм программы и исправьте ошибки.

Задание 2

1 Напишите программу на С++

2 Выполните отладку программы

Варианты заданий:

1. Вычислить значение арифметического выражения, заданного формулой $y = \frac{I}{J}\sin x + I * J$
2. Написать программу вычисления объема цилиндра V=π r2 h.
3. Найдите площадь равнобедренного треугольника по формуле $S = 0{,}5ab\sqrt{b^2 - \frac{a}{4}}$, где a - основание, b-боковая сторона.
4. Найдите объем конуса по формуле $V = \frac{\pi R^2 H}{3}$, если известны его радиус основания R и высота H.
5. Найдите периметр квадрата по указанному значению его площади, если P=4a, $S=a^2$.
6. Написать программу вычисления значения функции $y=-2{,}7x^3+0{,}23x^2-1{,}4$
7. Составить программу вычисления объема и площади поверхности куба по данной длине ребра.
8. Найти объем шара по формуле $V = \frac{4\pi R^3}{3}$, если известен его радиус.
9. Найдите длину окружности и площади круга по указанному радиусу.
10. Найдите площадь трапеций по формуле $S = \frac{a+b}{2}h$, если известны его стороны a, b и высота h.
11. Вычислить выражение $c=a^2+b^2-(2ab/(a-b))$, если известно, что a, b - вещественные числа.
12. Найдите силу тока I, если известны сопротивление R и напряжение U по закону Ома: I=U/R.

1. Составить программу вычисления выражения:

$$x = \beta \frac{\sqrt{z+1}}{tz+1}\ ;\ \zeta = \sqrt{|\sin b|}$$

$$t = \begin{cases} |\beta^2, & \beta > 1 \\ e^b, & \beta \le 1 \end{cases}$$

. Значение ‘b’ вводится с клавиатуры.

2. Составить программу нахождения значения функции:

$$y \begin{cases} x^3, \text{если } x < 5; \\ 5, \text{если } \le x \le 10; \\ -x + 2, \text{если } x > 10. \end{cases}$$

3. Составить программу нахождения значения функции:

$$y = \begin{cases} x^2 - 1, \text{ если } x > 1 \\ x + 5, \text{ если } x \le 1 \end{cases}$$

4. Составить программу нахождения значения функции:

$$y = \begin{cases} x - 1, & \text{если } x \ge 2 \\ x^2 - 4, & \text{если } x < 2 \end{cases}$$

5. Составить программу нахождения значения функции:

$$y \begin{cases} x^3, \text{если } x < 0; \\ 10, \text{если } 0 \le x \le 10; \\ -x + x^2, \text{если } x > 10. \end{cases}$$

Типовой пример 2. Программа вычисления кусочно-непрерывной функции:

$$y = \begin{cases} x^3 - 3, \text{ если } x > 1 \\ 0{,}5, \text{ если } x = 1 \\ \sqrt{1-x}, \text{ если } x < 1 \end{cases}$$

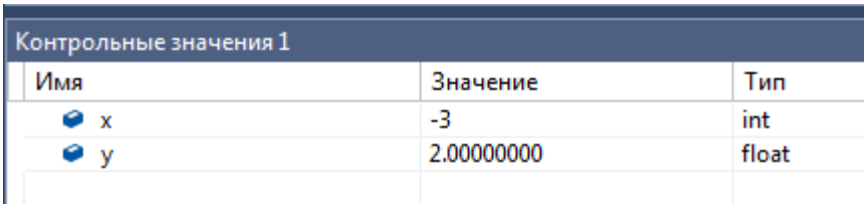
```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <math.h>
int main()
{
    int x;
    float y;
    printf("\nx=");
    scanf_s("%d", &x);
    if (x > 1) y = pow(x,3)-3;
    else
    {
        if (x == 1) y = 0.5;
        else y = sqrtf(1 - x);
    }
    printf("y=%f", y);
    return 0;
}
```

Контрольные значения 1

Имя	Значение	Тип
x	5	int
y	122.000000	float

Контрольные значения 1

Имя	Значение	Тип
x	1	int
y	0.500000000	float



Задание 4

1 Напишите циклическую программу на C++

2 Выполните отладку программы

Варианты заданий:

1. Написать программу для подсчета суммы чисел, кратных 3 в диапазоне от 30 до 60.
2. Написать программу для подсчета произведения четных чисел в диапазоне от 1 до 20.
3. Написать программу для подсчета суммы нечетных чисел в диапазоне от 0 до 30.
4. Написать программу для подсчета произведения, некратных 5 чисел в диапазоне от 3 до 30.
5. Написать программу для подсчета суммы чисел, кратных 7 в диапазоне от 10 до 50.
6. Написать программу для подсчета суммы чисел, некратных 4 в диапазоне от 0 до 40.
7. Написать программу для подсчета произведения, кратных 4 чисел в диапазоне от 30 до 50.
8. Написать программу для подсчета произведения, кратных 5 чисел в диапазоне от 20 до 40.
9. Написать программу для подсчета суммы N чисел.
10. Написать программу для подсчета произведения N чисел.
11. Написать программу для подсчета произведения четных чисел в диапазоне от 20 до 45.
12. Написать программу для подсчета суммы чисел, кратных 6 в диапазоне от 15 до 35.
13. Написать программу для подсчета суммы чисел, некратных 6 в диапазоне от 12 до 30.
14. Написать программу для подсчета произведения нечетных чисел в диапазоне от 25 до 40.
15. Написать программу для подсчета суммы чисел, кратных 2 в диапазоне от 10 до 50.
16. Написать программу для подсчета произведения, некратных 2 чисел в диапазоне от 20 до 30.

Замечания:

1. Проверка кратности N для переменной i **if (i% N == 0) …;**
2. Проверка не кратности N для переменной I **if (i% N != 0) …;**

Типовой пример 3. Программа для подсчета суммы чисел, некратных 3 в диапазоне от 2 до 20.

```
#include "stdafx.h"
#include <stdio.h>
int main()
{
    int S=0;
    int i;
    for (i = 2; i <= 20; i++)
        if(i%3!=0) S+=i;
    printf("S=%d\n", S);
    return 0;
}
```

Контрольные вопросы

1. Что включает в себя компиляция проекта?
2. Как найти ошибку в тексте программы, если она была обнаружена компилятором?
3. Что такое отладка?
4. Какие возможности отладки проекта предоставляет среда Microsoft Visual Studio?
5. Каким образом можно просмотреть значения переменных при отладке проекта?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2 (2 часа)

Тема: Инспекция кода модулей проекта

Цель: изучение формальных инспекций программного кода.

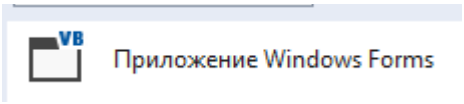
Задание. Написать программу, которая запрашивает у нового сотрудника имя, фамилию и дату рождения. Требуется создать новый класс с именем **Person** для хранения этой информации в свойствах, и создать метод класса, который будет вычислять текущий возраст нового сотрудника.

d. Добавление в ваш проект нового класса

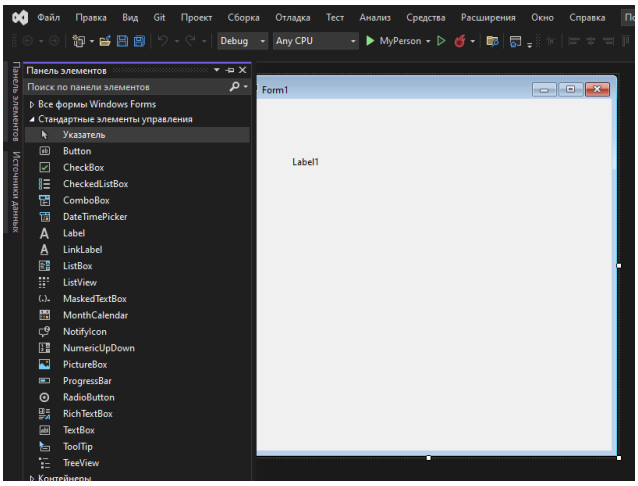
Класс, определенный пользователем, позволяет определить в программе ваши собственные объекты, которые имеют свойства, методы и события, точно так же, как объекты, создаваемые на формах Windows с помощью элементов управления из Области элементов. Чтобы добавить в ваш проект новый класс, щелкните в меню *Проект* на команде *Добавить класс*, а затем определите этот класс с помощью кода программы и нескольких новых ключевых слов Visual Basic.

Создание проекта

1. Запустите Visual Studio, затем создайте в своей папке новый проект с именем ***Сотрудники***.



2. Используйте элемент управления *Label* и добавьте в верхней части формы Form1 метку.



3. Используйте элемент управления *TextBox* и нарисуйте под меткой три текстового поля.

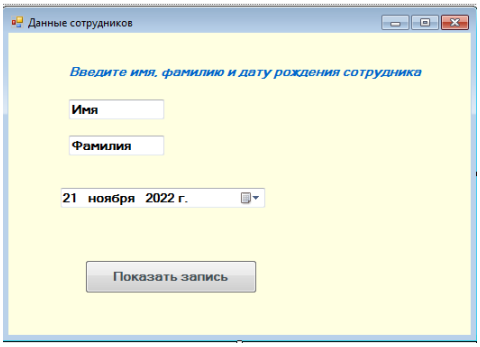
4. Используйте элемент управления *DateTimePicker* и создайте под текстовыми полями объект выбора даты и времени.

5. Используйте элемент управления *Button* и вставьте под объектом выбора даты и времени кнопку.

6. Установите для объектов формы следующие свойства:

Объект	Свойство	Установка
Label1	Text	Введите имя, фамилию и дату рождения сотрудника.
TextBox1	Text	Имя
TextBox2	Text	Фамилия
Button1	Text	Показать запись
Form1	Text	Данные сотрудников

7. Ваша форма должна выглядеть примерно так.

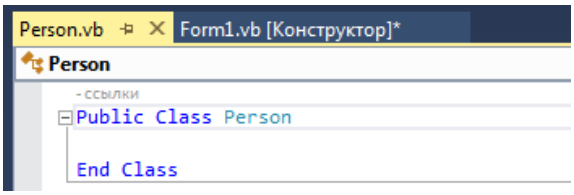


Это базовый интерфейс пользователя для формы, которая определяет запись нового сотрудника фирмы. (Эта форма не подключена к базе данных, так что храниться может только одна запись.) Теперь вы должны добавить в проект класс для хранения информации из этой записи.

8. Щелкните на команде *Добавить класс* в меню *Проект*.

Диалоговое окно Добавление нового элемента дает возможность задать имя вашего класса. Когда вы присвоите имя, обратите внимание, что вы можете сохранить в новом модуле класса несколько классов и указать имя, которое будет для них общим.

9. Введите в текстовом поле *Имя* имя *Person.vb*, а затем щелкните *Добавить*. *Visual Studio* откроет в Редакторе кода пустой модуль класса и добавит имя файла *Person.vb* в ваш проект в *Обозревателе решений*, как показано на рисунке.



Объявление переменных класса

Под оператором программы Public ClassPerson введите следующие объявления переменных:

Private Name1 As
String Private Name2
As String

Здесь вы объявляете две переменные, которые будут использованы исключительно в модуле класса для хранения значений двух строковых свойств. Переменные объявлены с помощью ключевого слова Private, так как по соглашению Visual Basic программисты должны держать внутренние переменные класса закрытыми - другими словами, недоступными для просмотра извне самого модуля класса.

Создание свойств

1. Под объявлением переменных введите следующий оператор программы и нажмите клавишу (Enter):

Public Property FirstName() As String

Этот оператор создает свойство вашего класса с именем FirstName, которое имеет тип String. Когда вы нажмете (Enter), VisualStudio немедленно создаст структуру кода для остальных элементов объявления свойства. Требуемыми элементами являются: блок Get, который определяет, что программисты увидят, когда будут проверять свойство FirstName, блок Set, который определяет, что произойдет, когда свойство FirstName будет установлено или изменено, и оператор EndProperty, который отмечает конец процедуры свойства.

2. Заполните структуру процедуры свойства FirstName

```
Public Property FirstName() As String
  Get
    Return Name1
  End Get
  Set (ByVal Value As String)
     Name1 = Value
  End Set
End Property
```

Ключевое слово Return указывает, что при обращении к свойству FirstName будет возвращена строковая переменная Name1. При установке значения свойства блок Set присваивает переменной Name1 строковое значение. Обратите особое внимание на переменную Value, используемую в процедурах свойств для обозначения значения, которое присваивается свойству класса при его установке. Хотя этот синтаксис может выглядеть странно, просто поверьте мне - именно так создаются свойства в элементах управления, хотя более сложные свойства будут иметь здесь дополнительную программную логику, которая будет проверять значения и производить вычисления.

3. Под оператором End Property введите для свойства LastName вашего класса вторую процедуру свойства.

```
Public Property LastName() As String
  Get
    Return Name2
  End Get
  Set (ByVal Value As String)
          Name2 = Value
  End Set
End Property
```

Эта процедура свойства аналогична первой, за исключением того, что она использует вторую строковую переменную (Name2), которую вы объявили в верхней части кода класса. Вы закончили определять два свойства вашего класса. Теперь перейдем к методу с именем Age, который будет определять текущий возраст нового сотрудника на основе даты рождения.

Создание метода

Под процедурой свойства LastName введите следующее определение функции:

```
Public Function Age (ByVal Birthday As Date) As
    Integer Return Int (Now.Subtract(Birthday).Days
    / 365.25)
End Function
```

Чтобы создать метод класса, который выполняет некое действие, добавьте в ваш класс процедуру Sub. Хотя многие методы не требуют для выполнения своей работы аргументов, метод Age, определенный мной, требует для своих вычислений аргумент Birthday типа Date. Это метод использует для вычитания даты рождения нового сотрудника из текущей системной даты метод Subtract, и возвращает значение, выраженное в днях, деленных на 365.25 - примерную длину одного года в днях. Функция Int преобразует это значение в целое, и это число с помощью оператора Return возвращается в вызывающую процедуру - как и в случае с обычной функцией.

Определение класса закончено! Вернитесь к форме *Form1* и используйте новый класс в процедуре события.

```
Public Class Person
    Private Name1 As String
    Private Name2 As String
    ссылок 2
    Public Property FirstName() As String
        Get
            Return Name1
        End Get
        Set(ByVal Value As String)
            Name1 = Value
        End Set
    End Property
    ссылка 1
    Public Property LastName() As String
        Get
            Return Name2
        End Get
        Set(ByVal Value As String)
            Name2 = Value
        End Set
    End Property
    ссылок 0
    Public Function Age(ByVal Birthday As Date) As Integer
        Return Int(Now.Subtract(Birthday).Days / 365.25)
    End Function
```

Создание объекта с помощью нового класса

1. Щелкните в Обозревателе решений на значке Form1.vb. Появится интерфейс пользователя Form1.

2. Чтобы открыть в Редакторе кода процедуру события Button1_Click, сделайте двойной щелчок мышью на кнопке **Показать запись**.

```
Form1.vb*  Person.vb*  Form1.vb [Конструктор]*
Button1                                  Click
    ссылок 2
Public Class Form1

    ссылок 0
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

    End Sub
End Class
```

3. Введите следующие операторы программы:

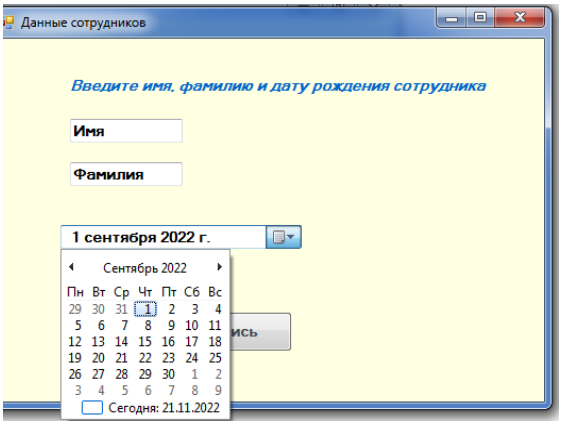
```
Public Class Form1

    ссылок 0
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim Employee As New Person
        Dim DOB As Date
        Employee.FirstName = TextBox1.Text
        Employee.LastName = TextBox2.Text
        DOB = DateTimePicker1.Value.Date
        MsgBox(Employee.FirstName & " " & Employee.LastName _
               & "в возрасте" & Employee.Age(DOB) & "лет.")
    End Sub
End Class
```

Эта процедура сохраняет в объекте с именем **Employee**, который имеет тип **Person**, значения, введенные пользователем. Ключевое слово **New** указывает, что вы хотите немедленно создать новый экземпляр объекта Employee. Теперь нужно объявить переменную с помощью класса, созданного вами самими! Затем процедура объявляет переменную с именем DOB типа Date. Она будет хранить дату, введенную пользователем, и устанавливает свойства FirstName и LastName объекта Employee равными имени и фамилии, введенным в два объекта текстовых полей формы. Значение, возвращаемое объектом выбора даты и времени, сохраняется в переменной DOB, а последний оператор программы отображает окно сообщения, содержащее свойства FirstName и LastName, а также возраст нового сотрудника, определенный методом Age, который при передаче в него переменной DOB возвращает целое значение. Как только вы определили класс в модуле класса, его легко можно использовать в процедуре события.

4. Чтобы запустить программу, щелкните на кнопке **Начать отладку** (F5). В среде разработки появится интерфейс пользователя, готовый к приему ваших данных.

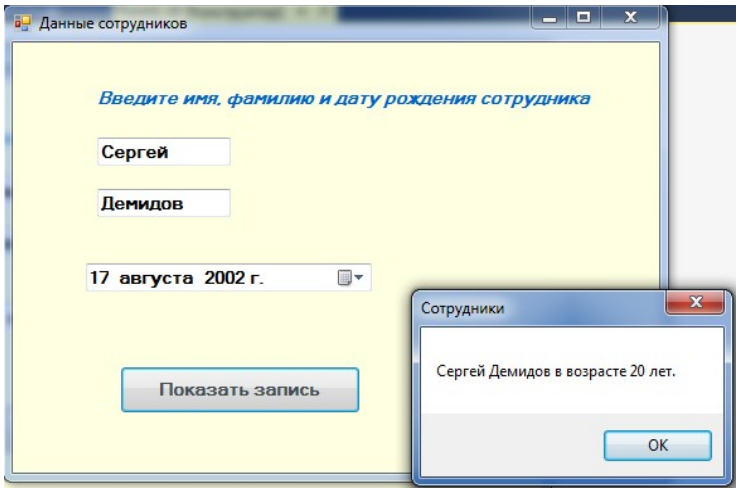
5. Введите в текстовое поле FirstName ваше имя, а в текстовое поле LastName - фамилию.

6. Щелкните на раскрывающемся списке объекта выбора даты и времени, и прокрутите его до вашей даты рождения.



7. Щелкните на кнопке *Показать запись*. Ваша программа сохраняет значения имени и фамилии в свойствах и использует метод *Age* для вычисления текущего возраста нового сотрудника. Появится диалоговое окно с результатом.

8. Чтобы закрыть это окно сообщения, щелкните на OK, а затем поэкспериментируйте с несколькими различными значениями дат, щелкая на *Показать записи*



е. Контрольные вопросы:

1. Что такое инспекция кода?
2. Какие виды инспекции кода вы знаете?
3. Определите понятия класс, экземпляр класса, объект.
4. Краткая характеристика метода экспертных исследований программного кода или документации на корректность или непротиворечивость.
5. Дайте характеристику этапов формальной инспекции и ролей еѐ участников.
6. В чѐм заключается сущность неформальной инспекции?

ЛАБОРАТОРНАЯ РАБОТА 9 (2 часа)

Тема: Тестирование интерфейса пользователя средствами инструментальной среды разработки
Цель: изучения процесса тестирования интерфейса пользователя различными методами.

Технология выполнения работы:

1. Определите предметную область и сферу применения программного продукта.
2. Определите целевую аудиторию.
3. Постройте описательную модель пользователя (профиль). При необходимости —

выделите группы пользователей.

4. Сформируйте сценарий поведения пользователей для ручного тестирования пользовательского интерфейса

5. Проведите тестирование интерфейса пользователя.

Контрольные вопросы

1. Что такое интерфейс?
2. Назовите цели тестирования и особенности тестирования при тестировании графического интерфейса пользователя (GUI).
3. Назовите типы требований к интерфейсу пользователя.
4. Назовите этапы функционального тестирования пользовательского интерфейса.
5. Назовите виды покрытий интерфейса пользователя (UI).
6. Достоинства и недостатки ручного тестирования.
7. Достоинства и недостатки автоматического тестирования.
8. Метод поиска элементов UI с использованием распознавания образов и (или) сравнение с образцом.
9. В чѐм сущность координатного метода тестирования UI?
10. Дайте характеристику Accessibility-метода.

ЛАБОРАТОРНАЯ РАБОТА № 10 (2 часа)

Тема: Разработка тестовых модулей проекта для тестирования отдельных модулей

Цель: научиться выполнят модульное тестирование программного продукта средствами Visual Studio.

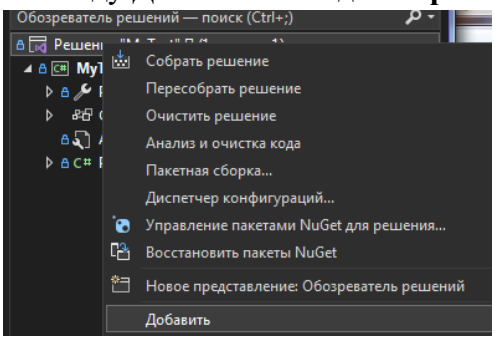
Задание 1 - Создание модульных тестов

Технология выполнения работы:

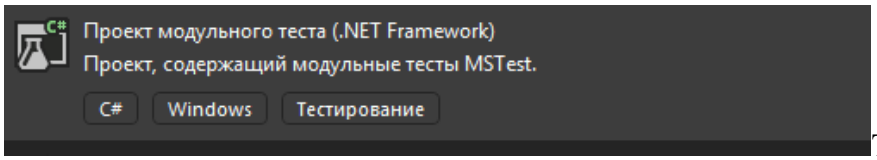
1. Запустите Visual Studio
2. Создайте простой консольный проект C# с именем **MyTest**.

```
using System;

namespace MyTest
{
    Ссылок: 0
    public class Program
    {
        Ссылок: 0
        public static void Main()
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

3. Выберите узел решения в **Обозревателе решений**. Затем с помощью щелкнув правой кнопкой мыши активизируйте команду **Добавить◊Создать проект**…

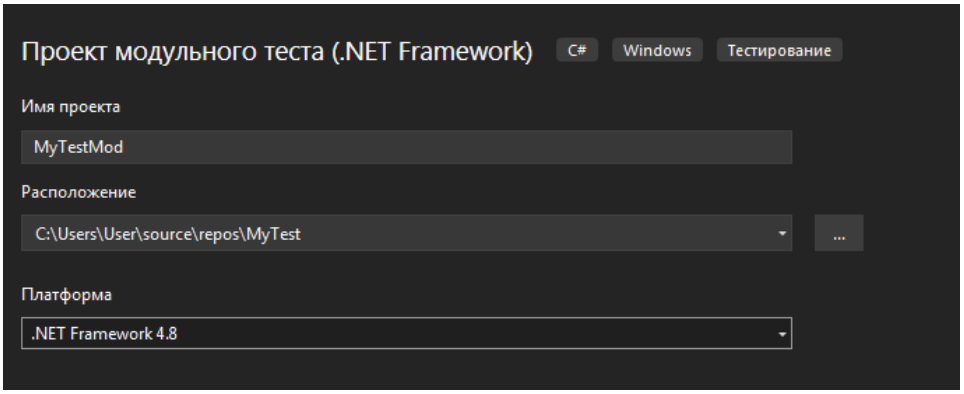


4. В диалоговом окне нового проекта найдите проект модульного теста, который хотите использовать. Введите **Тестирование** в поле поиска, чтобы найти шаблон проекта модульного теста для тестовой среды C#.



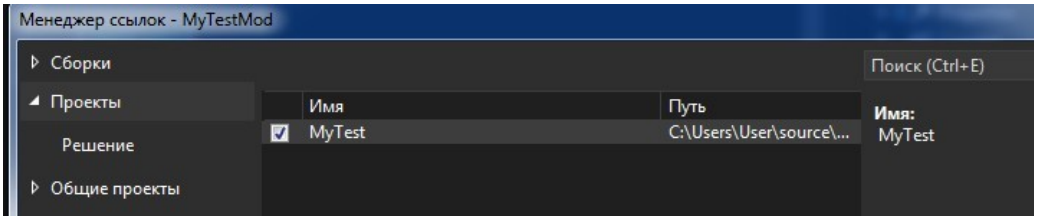
T

5. Нажмите **Далее**, выберите имя для тестового проекта и нажмите **Создать**.



6. В проекте модульного тестирования добавьте ссылку на проект, который вы хотите протестировать, щелкнув правой кнопкой мыши **Ссылки** или **Зависимости**, после чего выбрав **Добавить ссылку** или **Добавить ссылку на проект**.

7. Выберите проект, содержащий код, который будет тестироваться, и нажмите **ОК**.



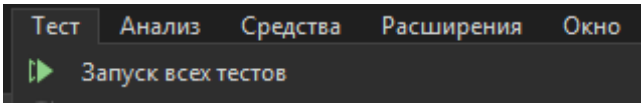
8. Добавьте код в метод модульных тестов.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.IO;

namespace MyTestMod
{
    [TestClass]
    Ссылок: 0
    public class UnitTest1
    {
        public const string Expected = "Hello World!";

        [TestMethod]
        Ссылок: 0
        public void TestMethod1()
        {
            using (var sw = new StringWriter())
            {
                Console.SetOut(sw);
                MyTest.program.Main();

                var result = sw.ToString().Trim();
                Assert.AreEqual(Expected, result);
            }

        }
    }
}
```

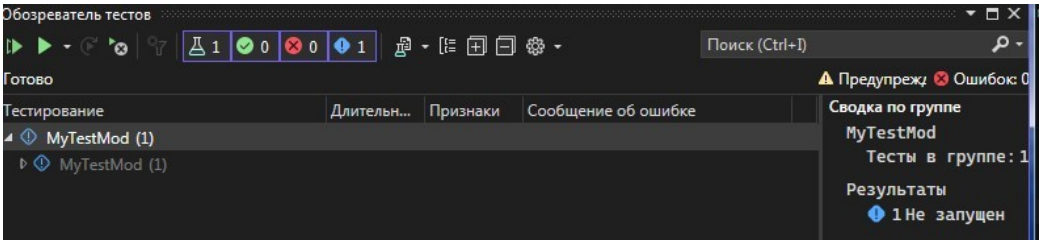
Задание 2 - Запуск модульных тестов

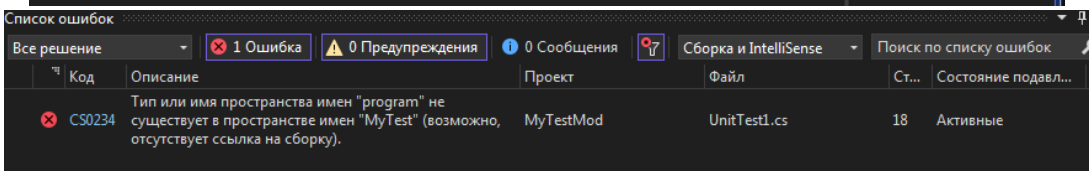
Технология выполнения работы:

1. Запустите модульные тесты, выбрав **Тест◊Запуск всех тестов**



Иконка  показывает, что тест не запускался Красный значок  указывает на сбой теста. В нижней части обозревателя теста должны выводиться ошибки в проекте.

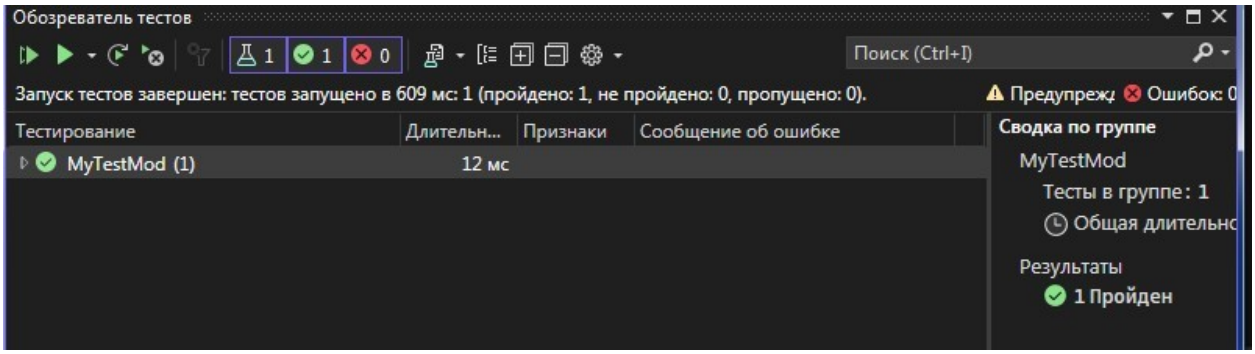




2. Исправьте ошибки и снова запустите тест.

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.IO;

namespace MyTestMod
{
    [TestClass]
    Ссылок: 0
    public class UnitTest1
    {
        public const string Expected = "Hello World!";

        [TestMethod]
        | Ссылок: 0
        public void TestMethod1()
        {
            using (var sw = new StringWriter())
            {
                Console.SetOut(sw);
                MyTest.Program.Main();

                var result = sw.ToString().Trim();
                Assert.AreEqual(Expected, result);
            }
        }
    }
}
```

3. После завершения зеленый флажок указывает, что тест пройден. В нижней части обозревателя теста не должно выводиться не каких ошибок



4. После успешного завершения теста посмотрите время его исполнения (в нашем случае это 12 мс).

Контрольные вопросы

1. Каковы цель и основные задачи модульного тестирования?
2. Назовите две основные проблемы, возникающие при модульном тестировании.
3. Дайте понятие модуля и его границ.
4. Характеристика процесса тестирования классов как модулей.
5. Определение степени полноты тестирования класса.
6. Протоколирование состояний объектов и их изменений.
7. Тестирование изменений, внесѐнных в исходный код.
8. Подходы к проектированию тестового окружения.

ЛАБОРАТОРНАЯ РАБОТА 11 (2 часа)

Тема: Выполнение функционального тестирования

Цель: изучение вопросов, связанных с функциональным тестированием. 13.2

Технология выполнения работы:

Задание 1 – Реализуйте функциональное тестирование способом разбиения по эквивалентности

Технология выполнения:

1. Определите входные данные
2. Определите выходные данные
3. Сформулируйте классы эквивалентности исходных данных алгоритма задачи. Результат занести в таблицу

Входное условие	Правильные классы эквивалентности	Неправильные классы эквивалентности

4. Разработайте тестовый вариант для каждого класса эквивалентности

Варианты задания 1

1. Определить размер аванса, выплачиваемого каждому работнику подразделения с табельными номерами 11 по 51.
2. Протестировать поле даты в форме.
3. Программа проверяет пароль пользователя. Он должен быть 8 символов и содержать заглавные и прописные латинские буквы, цифры.
4. В приложении Microsoft Paint есть опция «Изменить размер» — «Наклон», которая принимает значения -89… 89. Составьте классы эквивалентности.
5. Протестировать поля логин и пароль при регистрации пользователя, если граница 8.
6. Протестировать поля логин и пароль при авторизации пользователя, если логин – test, пароль – 1234.
7. Протестировать поле для ввода денежных средств.
8. Протестировать отправку письма в смартфоне.
9. Определить период отпусков работников подразделения с табельными номерами 25..100.
10. Протестировать текстовое поле в форме.

Задание 2 – Реализуйте функциональное тестирование способом анализа граничных значений

Технология выполнения:

Реализуйте анализ граничных значений исходных данных поставленной задачи:

- определите минимальное граничное значение;
- определите максимальное граничное значение;
- определите допустимые входные данные;
- определите недопустимые входные данные;
- составьте тестовые варианты.

Варианты задания 2

1. Система просит пользователя ввести возраст. В зависимости от того, является ли пользователь совершеннолетним или нет, отображается различный контент.

2. Протестировать текстовое поле «Имя», которое принимает длину от 6 до 12 символов.
3. Протестировать поле «Электронная почта» в форме «Гостевая книга» веб-сайта.
4. Онлайн-продажа билетов в кинотеатр на фильмы с рейтингом «18+».
5. Даны числа a, b, c, d. Требуется упорядочить их по возрастанию.
6. Даны числа a, b, c, d. Требуется упорядочить их по убыванию.
7. Протабулировать функцию на интервале [-10; 15] с шагом 0,2.
8. Проверить дату как дату формата дд.мм.гггг.
9. Проверить услуги страховой компании: клиент может застраховать жизнь при возрасте 18 до 60 лет.
10. При нажатии на кнопку «Загрузить», система должна проверять размер загружаемого файла до 100 Мб и в соответствии с этим отправлять на определенный ftp-сервер.

Задание 3 – Реализуйте функциональное тестирование способом таблицы решений

Технология выполнения:

1. Сформулируйте классы эквивалентности исходных данных алгоритма задачи
2. Разработайте тестовый вариант для каждого класса эквивалентности
3. Составьте программу и протестируйте ее
4. Выработайте рекомендации для корректировки тестируемой программы

Варианты задания 3

1. Найдите произведение нечетных чисел от –18 до 18.
2. Вычислите функцию y=ln(x+1), для каждого x из [0, 10] изменяющегося с шагом 1.
3. Вычислите функцию $y = \sqrt{x+2}$ для каждогох из [-2, 14] изменяющегося с шагом 1.
4. Вычислить Y=1+1/x+1/x^2. Область: $x^2+y^2=1$ для x∈ [0;1]; y∈ [0;1].
5. Найти сумму целых чисел из [20, 100], кратных 5.
6. Найти произведение чиселиз [20, 50], некратных 3.
7. Вычислить $y = \frac{2x}{4-x^2}$. Областьопределения: x∈R, x≠2, x≠ -2.
8. Вычислите функцию $Y = \sqrt{1-x}$, если $x^2<105$ выражение не имеет действительных решений
9. Найдите произведение чисел от -25 до 30, кратных 2. Исключить 0.
10. Вычислить $Y = \frac{\sin x}{x^2 - 9}$. Область определения: x∈R, x≠3, x≠ -3.

Типовой пример: Вычислите y = x·cos(x) + sin(3x). Область: $y^2+x^2 = 1$ для x, y∈[-1;1].

Технология выполнения:

1. Выделили классы эквивалентности
2. Определили граничные значения этих классов
3. Выполнили тесты по проверке значения до границы, на границе и сразу после границы.
4. Вычислили количество тестов для проверки граничных значений: [количество границ] * 3=2*3=6

Входные условия	**Правильные классы эквивалентности**	**Неправильные классы эквивалентности**
Переменная X	-1...1	x < -1 x>1
Тип переменной X - вещественный	2.9E-39..1.7E+38	Буква, символ, запятая в числе

5. Составили тестовые варианты (табл. 1)
6. Написали программу на языке C++.

```cpp
#include <iostream>
#include <math.h>
using namespace std;
int main() {
    setlocale(LC_ALL, "Russian");
    float x, y;
    std::cout << "Введите число в отрезке [-1; 1]" << endl;
    std::cout << "x = "; cin >> x;
    if ((x >= -1) && (x <= 1)) {
        y = x * cos(x) + sin(3 * x);
    std::cout << "x = " << x << ";\t";
    std::cout << "y = " << y << endl;
    }
    else {
    std::cout << "Функция не существует!" << endl;
    }

    return 0;
}
```

1. Протестировали программу

Таблица 1 - Название теста - Вычисление функции

№ теста	X	Ожидаемый результат Y	Фактический результат	Результат тестирования	Предложения по исправлению найденных ошибок
1	-1	-0,7	Консоль отладки Microsoft Visual Studio Введите число в отрезке [-1; 1] x = -1 x = -1; y = -0.681422	успешный	-
2	1	0,7	Консоль отладки Microsoft Visual Studio Введите число в отрезке [-1; 1] x = 1 x = 1; y = 0.681422	успешный	-
3	-1,001	Функция не определена	Консоль отладки Microsoft Visual Studio Введите число в отрезке [-1; 1] x = -1.001 Функция не существует!	успешный	-
4	1,001	Функция не определена	Консоль отладки Microsoft Visual Studio Введите число в отрезке [-1; 1] x = 1.001 Функция не существует!	успешный	-
5	0,5	1,4	Консоль отладки Microsoft Visual Studio Введите число в отрезке [-1; 1] x = 0.5 x = 0.5; y = 1.43629	успешный	-
6	&	Ошибка ввода	Консоль отладки Microsoft Visual Studio Введите число в отрезке [-1; 1] x = & x = 0; y = 0	неудачное	требуется ввести условие вычисления функции в случае символьного ввода для числовой переменной x

Контрольные вопросы

1. Что понимается под функциональным тестированием?
2. В чѐм сущность тестирования «черного ящика» (blackbox)?
3. В чѐм сущность тестирования «белого ящика» (whitebox)?
4. В чѐм сущность тестирования «серого ящика» (gray-box)?
5. Методы отбора тестов для black-box тестирования.
6. Тестирование сценариев использования – юз-кейсов (usecases).
7. Тестирование классов эквивалентности.
8. В чѐм сущность попарного тестирования?

ЛАБОРАТОРНАЯ РАБОТА № 12 (2 часа)

Тема: Тестирование интеграции

Цель: изучение вопросов, связанных с интеграционным тестированием.

Задание - Реализовать интеграционное тестирования любого Интернет-магазина

Постановка задачи: требуется проверить, что совместимость web-сайта со сторонними сервисами:

- работу сторонних модулей: оплата, шаринг (совместное, коллективное использование и потребление различных вещей и услуг), карты.
- рекламу (просмотр, переходы по рекламе, аналитика).
- метрики (переходы по страницам, показы элементов, клики).

Технология выполнения:

1. Разработайте тест-план:

- описание приложения;
- список функций и описание тестируемой системы и еѐ компонент;
- объекты тестирования.

2. Составьте тест-кейс
3. Протестируйте web-сайт

Тест-кейс «Название сайта»

Номер теста	Цель теста	Описание теста	Ожидаемый результат
1			

4. Сделайте выводы и рекомендации.

Контрольные вопросы

1. Что понимается под интеграционным тестированием?
2. Дайте характеристику подходам к интеграционному тестированию: снизу-вверх; сверху вниз; большой взрыв.
3. В чѐм заключается принцип действия системы непрерывной интеграции?
4. В чѐм заключается интеграционное тестирование ASP.NET Core?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3 (2 часа)

Тема: Документирование результатов тестирования

Цель: изучить правила и последовательность составления итогового отчета о результатах тестирования web-приложения.

Задание - Провести документирование результатов ручного тестирования web-приложения.

Технология выполнения работы

1. Запустить ранее созданное или тестированное приложение.
2. *Составить* ***итоговый отчет по результатам тестирования приложения:***

– Указать общую информацию о тестируемом продукте:

- название проекта;
- модули, которые подверглись тестированию;
- количество обнаруженных дефектов;

– Указать, кто и когда тестировал программный продукт.

– Описать тестовое окружение: ссылку на проект, браузер, операционную систему и другую информацию, конкретизирующую особенности конфигурации.

– Указать общую оценку качества протестированного приложения и ее обосновать. Уровни качества: высокое (High), среднее (Medium), низкое (Low).

– Определить показатель успешного прохождения тест-кейсов с помощью метрики по формуле

$$T^{SP} = \frac{T^{Success}}{T^{Total}} \cdot 100\ \%,$$

– Графически (в виде круговой диаграммы) отразить процентное соотношение дефектов GUI и функциональных дефектов:

1) ***Реализовать GUI – тестирование пользовательского*** интерфейса

№	Тест-кейсы	Результаты теста
1	Проверить все элементы графического интерфейса на размер, положение, ширину, длину и прием символов или цифр.	успешно/дефект
2	Проверить, можете ли вы выполнить намеченную функциональность приложения, используя графический интерфейс	
3	Проверить сообщения об ошибках отображаются правильно	
4	Проверить на четкое разграничение различных разделов на экране	
5	Проверить, что используемый в приложении шрифт читабелен	
6	Проверить правильность выравнивания текста	
7	Цвет шрифта и предупреждающих сообщений эстетичны	
8	Убедиться, что изображения имеют хорошую четкость	
9	Убедиться, что изображения правильно выровнены	
10	Проверить расположение элементов графического интерфейса для разных разрешений экрана.	

- Вычислить процент дефектов GUI по формуле:

$$\text{Процент дефектов} = \frac{\text{Количества дефектов}}{\text{количество тестов}} * \mathbf{100}\ \%$$

2) Реализовать **Functional** - тестирование, основанное на анализе спецификации функциональ-

ности

Функциональное тестирование интернет-магазина

При функциональном тестировании проверяется реализация функциональных требований, т.е. возможность программного продукта выполнять те функции, которые были описаны в спецификациях на разработку. Каждый блок в отдельности необходимо тестировать на корректность реализации присущих ему функций.

№	Тест-кейсы	Результаты теста
1	Проверка баннера	успешно/дефект
2	Поиск товара	
3	…..	

- Вычислить процент функциональных дефектов
- Графически (в виде столбчатой диаграммы) отразить распределение дефектов по модулям.
- Произвести детальный анализ качества всех модулей протестированного приложения с аргументацией выставленных уровней качества.

Модули	Уровень качества	Комментарии
Главная		
Каталог товаров		
Корзина		

- Привести список 3-5 наиболее критичных дефектов.
- Сформулировать рекомендации по улучшению качества программного продукта.

Контрольные вопросы:

1. Какая структура итогового отчета о результатах тестирования?
2. Что содержится в разделе Общая информация?
3. Что содержится в разделе Тестовое окружение?
4. Что содержится в разделе Рекомендации QA?
5. Что содержится в разделе Детализированная информация?
6. Что содержится в окончании отчета?

МДК.01.03 Математическое моделирование

Практическое занятие 1

Тема: «Построение простейших математических моделей. Построение простейших статистических моделей»

Цель: закрепить практические навыки по построению простейших математических и простейших статистических моделей.

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Построение математической модели процесса, явления или объекта начинается с построения упрощенного варианта модели, в котором учитываются только основные черты. В результате прослеживаются основные связи между входными параметрами, ограничениями и показателем

эффективности. Общего подхода к построению модели нет. В каждом конкретном случае при построении математической модели учитывается большое количество факторов: цель построения модели, круг решаемых задач, точность описания модели и точность выполнения вычислений. Математическая модель должна отражать все существенные факторы, определяющие ее поведение, и при этом быть простой и удобной для восприятия результатов. Каждая математическая модель процесса, явления или объекта в своей основе имеет математический количественный метод.

Применение математических количественных методов для обоснования выбора того или иного управляющего решения во всех областях человеческой деятельности называется *исследованием операций.* Целью исследования операций является нахождение с использованием специального математического аппарата решения, удовлетворяющего заданным условиям. На самом деле при решении практически любой задачи имеется неограниченное количество решений. Множество решений, удовлетворяющих заданным усло- виям (ограничениям), называется допустимым множеством решением. Выбор из множества допустимых решений одного решения, наилучшего в каком-либо смысле, называемого *оптимальным* решением, и есть задача исследования операций.

Модель — это материальный или идеальный объект, заменяющий оригинал, наделенный основными характеристиками (чертами) оригинала и предназначенный для проведения некоторых действий над ним с целью получения новых сведений об оригинале.

Рис. 1. Классификация моделей

Рис. 2. Классификация математических моделей

При построении математической модели необходимо обеспечить *достаточную* точность вычислений (точность решения) и *необходимую* подробность модели. Любая математическая модель включает в себя описание основных, т. е. *необходимых* для исследования свойств и законов функционирования исследуемого объекта, процесса или явления. В своей основе каждая математическая модель имеет целевую функцию, которая описывает функционирование реального объекта, процесса или явления. В зависимости от исследуемого (моделируемого) объекта, явления или процесса *целевая функция* может быть представлена одной функциональной зависимостью, системой уравнений (линейных, нелинейных, дифференциальных и т. д.), набором статистических данных и т. д. При работе с целевой функцией исследователь воздействует на нее через ***набор входных параметров*** (рис. 3).

Входной параметр 1		Выходной параметр 1
Входной параметр 2		Выходной параметр *2*
Входной параметр 3	Модель системы	Выходной параметр 3
Входной параметр *n*- 1	(объекта или процесса)	Выходной параметр *m* - 1
Входной параметр *n*		Выходной параметр *m*

Рис. 3. Обобщенная схема математической модели

По способу реализации математические модели можно разделить следующим образом.

1. Линейное программирование.

Математическая модель целиком (целевая функция и ограничения) описывается уравнениями первого порядка. Линейное программирование включает в себя несколько методов решения (задач):

- симплексный;
- графический;
- транспортная задача;
- целочисленное программирование.

2. Нелинейное программирование.

Целевая функция и ограничения, составляющие математическую модель, содержат хотя бы одно нелинейное уравнение (уравнение второго порядка и выше). Нелинейное программирование содержит несколько методов решения (задач):

- графический;
- регулярного симплекса;
- деформируемого многогранника (Нелдера - Мида);
- градиентный.

3. Динамическое программирование.

Ориентировано на решение задач прокладки магистралей кратчайшим путем и перераспределения различных видов ресурсов.

4. Сетевое планирование.

Решает проблему построения графика выполнения работ, распределения производственных, финансовых и людских ресурсов.

5. Принятие решений и элементы планирования.

В этом случае и качестве целевой функции выступает набор статистических данных или некоторые данные прогноза. Решением задачи являются рекомендации о способах поведения (стратегии). Решение носит рекомендательный характер (приблизительное решение). Выбор стратегии целиком остается за человеком — ответственным лицом, принимающим решение. Для принятия решения разработаны следующие теории:

- теория игр;
- системы массового обслуживания.

Ход работы

Задание. Составить математическую модель следующей задачи. На складе имеется 300 кг сырья. Надо изготовить два вида продукции. На изготовление первого изделия требуется 2 кг сырья, а на изготовление второго изделия — 5 кг. Определить план выпуска двух изделий.

Список используемой литературы

1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

Практическое занятие 2

Тема: «Решение простейших однокритериальных задач. Задача Коши для уравнения теплопроводности. Сведение произвольной задачи линейного программирования к основной задаче линейного программирования. »

Цель: закрепить практические навыки по решению простейших однокритериальных задач.

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Уравнение теплопроводности — дифференциальное уравнение в частных производных второго порядка, которое описывает распределение температуры в заданной области пространства и ее изменение во времени.

Ход работы

Задание.

Рассмотрите задачу Коши для однородного уравнения теплопроводности.
Рассмотрите задачу Коши для неоднородного уравнения теплопроводности.

Список используемой литературы

1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

Практическое занятие 3

Тема: «Решение задач линейного программирования симплекс– методом. Нахождение начального решения транспортной задачи. Решение транспортной задачи методом потенциалов»

Цель: Закрепить практические навыки по решению задач линейного программирования симплекс-методом. Закрепить практические навыки по решению транспортной задачи методом потенциалов

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Задача линейного программирования — это задача поиска неотрицательных значений параметров, на которых заданная линейная функция достигает своего максимума или минимума при заданных линейных ограничениях.

Симплекс-метод — алгоритм решения оптимизационной задачи линейного программирования путём перебора вершин выпуклого многогранника в многомерном пространстве. Алгоритм является универсальным методом, которым можно решить любую задачу линейного программирования.

Если вам тоже ничего не понятно из этого определения, то вы на верном пути. Чаще всего статьи про симплекс-метод очень сильно углубляются в дебри теории задачи линейного программирования, из-за чего очень легко потерять суть и так ничего и не понять. Мы постараемся описать алгоритм симплекс-метода так, чтобы показать, что в нём нет ничего страшного и на самом деле он весьма простой. Но сначала нам всё-таки потребуется ввести несколько определений.

Целевая функция — функция, максимум (или минимум) которой нужно найти. Представляет собой сумму произведений коэффициентов на значения переменных: $F = c_1 \cdot x_1 + c_2 \cdot x_2 + ... + c_n \cdot x_n$

Ограничение — условие вида $a_1 \cdot x_1 + a_2 \cdot x_2 + ... + a_n \cdot x_n$ v b, где вместо v ставится один из знаков: $\leq$, = или $\geq$

План — произвольный набор значений переменных $x_1 ... x_n$.

Одна из самых распространенных и востребованных оптимизационных задач в логистике – транспортная задача. В классическом виде она предполагает нахождение оптимального (т.е. сопряженного с минимальными затратами) плана грузоперевозок. Например, у нас есть сеть розничных магазинов, которым требуется определенное количество товаров. Также имеется ряд складов поставщиков, где требуемые товары хранятся. При этом на каждом складе различный объем запасов этих товаров. Кроме этого нам известны тарифы – затраты на перевозку 1 товара от каждого склада к каждому магазину. Возникает необходимость разработать такой план перевозок, чтобы магазины получили требуемое количество товаров с наименьшими затратами на транспортировку. Вот именно в таких случаях (и во множестве других) приходится решать транспортную задачу.

Ход работы

Задание 1.

Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется a1 кг первого сорта, a2 кг второго сорта и a3 кг третьего сорта. На изготовление продукции вида В расходуется b1 кг первого сорта, b2 кг второго сорта, b3 кг третьего

сорта. На складе фабрики имеется всего материала первого сорта с1 кг, второго сорта с2 кг, третьего сорта с3 кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида α руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида β руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

a1= 19, a2= 16, a3= 19, b1= 31, b2= 9, b3= 1, c1= 1121, c2= 706, c3= 1066,

α=16, β=19.

Задание 2.

ЗАДАНИЕ. Из трех холодильников A_i, $i=\overline{1,3}$, вмещающих мороженную рыбу в количествах a_i т, необходимо последнюю доставить в пять магазинов B_j, $j=\overline{1,5}$ в количествах b_j т. Стоимости перевозки 1т рыбы из холодильника A_i в магазин B_j заданы в виде матрицы $C=((c_{ij}))$, 3x5.

Написать математическую модель задачи и спланировать перевозки так, чтобы их общая стоимость была минимальной.

a_1=320, b_2=140,
a_2=280, b_3=110,
a_3=250, b_4=230,
b_1=150, b_5=220

$$C=\begin{pmatrix} 20 & 23 & 20 & 15 & 24 \\ 29 & 15 & 16 & 19 & 29 \\ 6 & 11 & 10 & 9 & 8 \end{pmatrix}$$

Список используемой литературы

1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

Практическое занятие 4

Тема: «Применение метода стрельбы для решения линейной краевой задачи.»

Цель: закрепить практические навыки по решению линейной краевой задачи мотодом стрельбы

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

«Метод стрельбы сводит решение краевой задачи для ОДУ к решению итерационной последовательности задач Коши.

Ход работы

Задание. Планируется деятельность четырех промышленных предприятий (системы) на очередной год. Начальные средства: усл. ед. Размеры вложения в каждое предприятие кратны усл. ед. Средства x,выделенные k-мупредприятию (k=1, 2, 3, 4), приносят в конце года прибыль . Функции заданы таблично (табл. 4.1). Принято считать, что:

- прибыль не зависит от вложения средств в другие предприятия;
- прибыль от каждого предприятия выражается в одних условных единицах;
- суммарная прибыль равна сумме прибылей, полученных от каждого предприятия.

Определить, какое количество средств нужно выделить каждому предприятию, чтобы суммарная прибыль была наибольшей.

Таблица 4.1 x

Список используемой литературы

1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

Практическое занятие 5

Тема: «Задача о распределении средств между предприятиями»

Цель: закрепить практические навыки по решению линейной краевой задачи о распределении средств между предприятиями

Оснащение: ПК, учебная и справочная литература.

«Метод стрельбы сводит решение краевой задачи для ОДУ к решению итерационной последовательности задач Коши.

Задание. Планируется деятельность четырех промышленных предприятий (системы) на очередной год. Начальные средства: $s_0 = 5$ усл. ед. Размеры вложения в каждое предприятие кратны усл. ед. Средства x,выделенные k-мупредприятию (k=1, 2, 3, 4), приносят в конце года прибыль $f_k(x)$. Функции $f_k(x)$ заданы таблично (табл. 4.1). Принято считать, что:

- прибыль $f_k(x)$ не зависит от вложения средств в другие предприятия;
- прибыль от каждого предприятия выражается в одних условных единицах;
- суммарная прибыль равна сумме прибылей, полученных от каждого предприятия.

Определить, какое количество средств нужно выделить каждому предприятию, чтобы суммарная прибыль была наибольшей.

Таблица 4.1 x

$f_1(x)$ $f_2(x)$ $f_3(x)$ $f_4(x)$

Список используемой литературы

1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

Практическое занятие 6

Тема: «Задача о замене оборудования**»**

Цель: закрепить практические навыки по решению линейной краевой задачи мотодом стрельбы

Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

Замена оборудования – важная экономическая проблема. Задача состоит в определении оптимальных сроков замены старого оборудования (станков, производственных зданий и т.п.). Старение оборудования включает его физический и моральный износ, в результате чего растут производственные затраты, затраты на ремонт и обслуживание, снижаются производительность труда, ликвидная стоимость. Критерием оптимальности являются, как правило, либо прибыль от эксплуатации оборудования (задача максимизации), либо суммарные затраты на эксплуатацию в течение планируемого периода (задача минимизации).

При построении модели задачи принято считать, что решение о замене выносится в начале каждого промежутка эксплуатации (например, в начале года) и что в принципе оборудование можно использовать неограниченно долго.

Основная характеристика оборудования – параметр состояния – его возраст *t.*

При составлении динамической модели замены процесс замены рассматривают как "- шаговый, разбивая весь период эксплуатации на *n* шагов. Возможное управление на каждом шаге характеризуется качественными признаками, например *Х*е (сохранить оборудование), *X'* (заменить) и Хр (сделать ремонт).

Ход работы

Задание. Оборудование эксплуатируется в течение 5 лет, после этого продается. В начале каждого года можно принять решение – сохранить оборудование или заменить его новым. Стоимость нового оборудования $p0 = 4000$ руб[1]. После t лет эксплуатации ($1 < t < 5$) оборудование можно продать за $g(t) = p0\ T'$ руб. (ликвидная стоимость). Затраты на содержание в течение года зависят от возраста t оборудования и равны r(i) = 600(i + l). Определить оптимальную стратегию эксплуатации оборудования, чтобы суммарные затраты с учетом начальной покупки и заключительной продажи были минимальны.

Список используемой литературы
1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

Практическое занятие 7

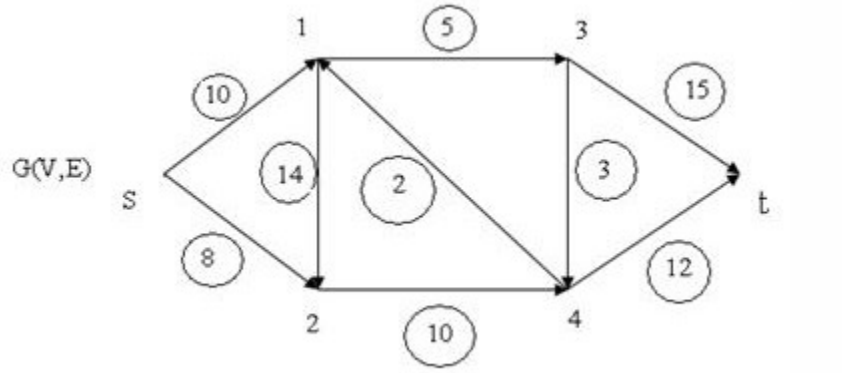
Тема: «Нахождение кратчайших путей в графе. Решение задачи о максимальном потоке**»**
Цель: закрепить практические навыки по решению задачи о максимальном потоке
Оснащение: ПК, учебная и справочная литература.

Теоретические сведения

В теории оптимизации и теории графов, задача о максимальном потоке заключается в нахождении такого потока по транспортной сети, что сумма потоков из истока, или, что то же самое, сумма потоков в сток максимальна.

Ход работы

Задание. Дана сеть G(V,E) (рис.1) с источником s и стоком t. Пропускные способности дуг указаны. Найти максимальный поток из s в t.



Список используемой литературы

1. Зализняк В.Е.Введение в математическое моделирование. Учебное пособие для СПО М: изд-во Юрайт 2020 г.

МДК.02.04 Веб- программирование. Основы интеллектуального труда

Практическое занятие № 1
"Разработка логической и физической структуры сайта"

Цель работы:

1. Изучение особенностей процесса разработки логической и физической структуры сайта

2. Получение первоначальных практических навыков при работе над логической и физической структурой сайта.

Оборудование и программное обеспечение:

1. ПК, локальная сеть.
2. ОС MS Windows 7, офисный пакет MS Office, браузер MS Internet Explorer.

План занятия:

1. Освоение теоретической части занятия в соответствии с разделами 1 - 4.
2. Выполнение практической части занятия согласно п.п.3.1., 4.1.
3. Повторение и закрепление основных изученных понятий и терминов.
4. Ответы по контрольным вопросам.

1. Логическая и физическая структура сайта. Каждый ресурс Интернета, от любительской домашней странички до большого информационного портала, содержит несколько тематических рубрик, соединенных между собой гиперсвязями. Как правило, ссылки на все разделы сайта с краткими анонсами их содержимого приводится на первой, так называемой стартовой странице, которой присваивается имя index.htm (.html). Если тематические рубрики содержат собственные подразделы, каждая из них также имеет свою стартовую страницу, называющуюся index.html.

ПРИМЕЧАНИЕ: *Такое имя файла рекомендуется присваивать всем стартовым документам сайта, поскольку в противном случае при обращении к какому-либо разделу посредством сокращенного URL без указания названия стартовой страницы (например, http://www.mysite.ru/photos/ вместо http://www.mysite.ru/photos/startpage.html) браузер отобразит не саму web-страницу, а перечень хранящихся в данной папке файлов. Подобный набор тематических рубрик с распределенными по соответствующим разделам документами и заранее спроектированными гиперсвязями между всеми страницами ресурса и называется логической структурой сайта. Физическая структура, напротив, подразумевает алгоритм размещения физических файлов по поддиректориям папки, в которой опубликован ваш сайт.*

2. Особенности. Очевидно, что логическая и физическая структуры могут не совпадать, поскольку в общем случае физическая структура ресурса разрабатывается, исходя из удобства размещения файлов. Однако более или менее точное сохранение порядка следования логических разделов в физической структуре сайта позволит вам избежать путаницы при последующем дополнении и обновлении материалов.

СОВЕТ: *Рекомендуется размещать все графические изображения, являющиеся элементами проекта, в отдельной папке с названием "Images", расположенной в корневой директории сайта. Такой подход позволит обновлять хранящиеся в других тематических разделах документы HTML без переноса графики, использовать одни и те же графические файлы во всех разделах сайта и при необходимости удалять целые директории.*

Для того чтобы все гиперссылки на вашей домашней страничке или web-сайте работали корректно, все документы открывались правильно и браузер не выдавал ошибок при обращении к каким-либо разделам ресурса, при создании его физической структуры следует соблюдать несколько **простых правил**:

СОВЕТ: *Назначайте имена директорий, имена и расширения документов HTML и графических файлов с использованием символов только латинского алфавита и только в строчном регистре. Старайтесь, чтобы имена созданных вами файлов и директорий не превышали по длине восьми символов.*

СОВЕТ: *При присвоении имен файлов документам HTML старайтесь следить за тем, чтобы эти имена были ;смысловыми: впоследствии вы легко можете забыть содержимое и назначение какой-либо web-страницы, если имена файлов будут выглядеть, например, как 1.htm, 2.htm, 3.htm и т. д.*

3. Оформление и документирование страниц. Для того чтобы облегчить процесс обновления web-страниц, дополнения разделов или создания новых рубрик, заведите средство документирования проекта — любую электронную таблицу, созданную, например, в Microsoft Excel, или просто разграфленную тетрадку, в которую записывайте соответствие элементов физической структуры вашего проекта его логической структуре. До тех пор пока количество составляющих ваш сайт файлов относительно мало, это может показаться излишним, когда же оно перевалит за первые два десятка, в обилии html-документов и графических элементов будет легко запутаться, особенно если вы создаете несколько проектов одновременно.

3.1. Практическая часть:

В соответствии с приведенным примером выполните таблицу документирования страниц, приведя собственные исходные данные.

Пример оформления средства документирования показан в таблице:

Имя файла	*Директория*	*Описание*
index.html	**/mysite**	Стартовая страница сайта http://www.mysite.ru/
index.html	**/mysite/family**	Стартовая страница раздела "моя семья"
мама.htm	**/mysite/family**	Рассказ о моей маме
brother.htm	**/mysite/family**	Рассказ о моем брате
Almaty-jpg	**/mysite/photos**	Моя фотография в Алматы

Из всего сказанного становится очевидным, что физическая структура сайта скрыта от посетителей вашего ресурса: они могут наблюдать только логическую структуру, причем именно так, как она представлена при помощи элементов навигации. Отсюда следует вполне логический вывод: строение системы навигации должно если не полностью повторять, то хотя бы максимально соответствовать разработанной вами логической структуре сайта/

4. Заглавная страница.

Один из критериев, руководствуясь которым можно разделить различные web-сайты на две основные категории, — это наличие заглавной страницы (splash) или отсутствие таковой.

Заглавная страница представляет собой html-документ, который не включает в себя какую- либо содержательную информацию и элементы навигации. Файлу заглавной страницы присваивается имя index.html, при этом стартовая страница называется иначе и вызывается посредством организации гиперссылки .с за- главной страницы, загружающейся при обращении к сайту первой. Заглавная страница содержит, как правило, логотип компании-владельца данного ресурса, счетчик посещений и предложение выбора кодировки кириллицы, либо вы- бора между английской и русской версиями сайта.

При обращении к сайтам, не оснащенным заглавной страницей, первой отображается стартовый документ, включающий какое-либо информационное наполнение, элементы навигации и иногда анонсы составляющих данный ресурс тематических рубрик.

Использовать или не использовать заглавную страницу при создании собственного проекта в сети Интернет — дело вкуса каждого web-мастера. Дать какие-либо исчерпывающие рекомендации на этот счет трудно, поскольку окончательное решение зависит прежде всего от ваших художественных предпочтений и иногда — от желания заказчика.

4.1. Практическая часть:

Приведите примеры заглавной страницы, определяя в таблице, выполненной согласно практической части 3.1.

Контрольные вопросы к практическому занятию № 1:

1. Что означает имя - index.htm (.html)?
2. Что означает понятие URL?
3. Исходя из каких соображений разрабатывается физическая структура ресурса?
4. Что представляет собой заглавная страница?
5. Для чего документы одного типа размещаются в определенных каталогах (папках)?

Практическая работа № 2
Создание HTML документа с графикой и картами. Вставка видео/аудио в документ HTML

Задание 1. Размещение графики на Web-странице

Информация.
Тэг <IMG …> позволяет вставить изображение в документ. Изображение появится в том месте документа, в котором записан этот тэг. Команда записывается с одиночным тэгом, т.е. закрывающий тэг не применяется.

Графика в Web, как правило, распространяется в трех форматах: GIF, JPG, PNG. Для выполнения упражнения считаем, что графический файл Wagon.gif хранится в рабочем каталоге HTML, где находится и наша Web-страница.

1. Внесите изменения в файл RASP.HTM:

```
<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#330066">
<P ALIGN=CENTER>
<FONT COLOR="#008080" SIZE="7"><B> Расписание </B></FONT><BR>
<FONT SIZE="6"> <I>занятий на вторник</I></FONT><BR><BR>
<IMG SRC="Wagon.gif">
</P>
</BODY>
</HTML>
```

Тэг <IMG …> **имеет немало атрибутов (см. таблицу 2), которые можно задавать дополнительно. Они могут располагаться где угодно в тэге после кода** IMG.

Атрибуты изображения

Таблица 2

Атрибут	Формат	Описание
ALT	<IMG SRC="Wagon.gif" ALT="картина">	Если броузер не воспринимает изображение, вместо него появляется заменяющий текст.
BORDER	<IMG SRC="Wagon.gif" BORDER="3">	Задает толщину рамки вокруг изображения. Измеряется в пикселах.
ALIGN	<IMG SRC="Wagon.gif" ALIGN=TOP>	Выравнивает изображение относительно текста: по верхней части изображения – TOP, по нижней – BOTTOM, по средней – MIDDLE.
HEIGHT	<IMG SRC="Wagon.gif" HEIGHT=111>	Задает вертикальный размер изображения внутри окна броузера.
WIDTH	<IMG SRC="Wagon.gif" WIDTH=220 >	Задает горизонтальный размер изображения внутри окна броузера.
VSPACE	<IMG SRC="Wagon.gif" VSPACE="8">	Добавляет верхнее и нижнее пустые поля.
HSPACE	<IMG SRC="Wagon.gif" HSPACE="8">	Добавляет левое и правое пустые поля.

Задание 2. Атрибуты изображения.

1. Самостоятельно внесите изменения в файл RASP.HTM, опробовав использование таких атрибутов графики как ALT, BORDER, HEIGHT, WIDTH.

Примечание
Всегда обращайте внимание на размеры (объем в байтах) своего графического файла, т.к. это влияет на время загрузки Web-страницы.

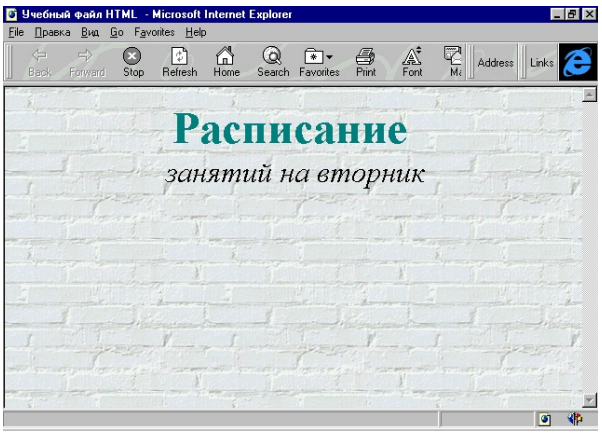
Задание 3. Фоновое изображение графики на Web-странице

Информация.
Фоновое изображение – это графический файл с изображением небольшой прямоугольной плашки. При просмотре в броузере эта плашка многократно повторяется, заполняя все окно, независимо от его размеров.
Графика, используемая в качестве фоновой, задается в тэге <BODY> в начале файла HTML.

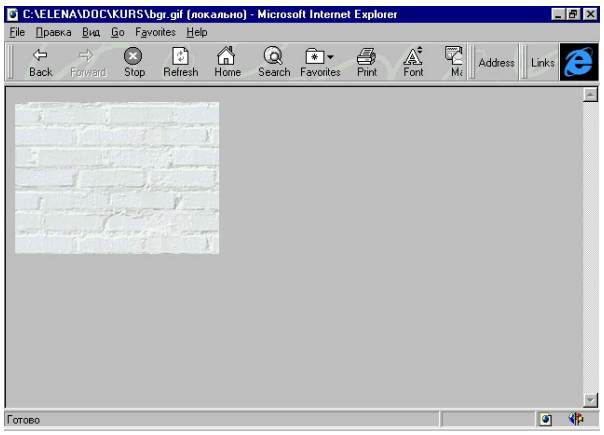
1. Внесите изменения в файл RASP.HTM:

```
<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY BACKGROUND="BGR.GIF" TEXT="#330066">
<P ALIGN=CENTER>
<FONT COLOR="#008080" SIZE="7"><B> Расписание </B></FONT><BR>
<FONT SIZE="6"> <I>занятий на вторник</I></FONT><BR><BR>
</P>
</BODY>
</HTML>
```

На экране вы увидите:



На самом деле графический файл BGR.GIF выглядит так:



Задание 4. Таблицы.

Информация.
Таблицы представляют собой особую часть HTML-документа. Данные в ней организованы в виде прямоугольной сетки, состоящей из вертикальных столбцов и горизонтальных рядов. Каждая клетка таблицы является ячейкой.
Ячейки могут содержать в себе текст, графику или другую таблицу.
Таблица состоит из трех основных частей:

- **название таблицы,**
- **заголовки столбцов,**
- **ячейки.**

Таблица заполняется горизонтальными рядами ячейка за ячейкой слева направо. Заполнение начинается с левого верхнего угла и заканчивается правым нижним. Каждая ячейка должна быть заполнена. Для создания пустых ячеек используются пробелы.

Теги оформления таблиц

Тег	Форма записи	Примечание
TABLE	<TABLE>текст</TABLE>	Объявление таблиц.
TR	<TR>текст</TR>	Тег строки.
TD	<TD>текст</TD>	Тег данных.

Атрибуты тега <TABLE>

Атрибут	Форма записи	Примечание
BORDER	<TABLE BORDER=X>	Задает рамку вокруг таблицы.
WIDTH	<TABLE WIDTH=XX%>	Задает ширину таблицы как XX % от ширины страницы или как XX пикселов.
BGCOLOR	<TABLE BGCOLOR="#RRGGBB">	Задает цвет фона таблицы.

Атрибуты тегов <TD> и <TR>

Атрибут	Форма записи	Примечание
ALIGN	<TD ALIGN=X>	Устанавливает выравнивание по горизонтали (Right, Left, Center)
VALIGN	<TD VALIGN=X>	Устанавливает выравнивание по вертикали (Top, Middle, Bottom, Baseline)
BGCOLOR	<TD BGCOLOR="#RRGGBB">	Задает цвет фона ячейки.

1. Запустите стандартную программу Блокнот (Notepad).

2. Наберите в окне редактора:

```
<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<TITLE> Расписание занятий 5 классов </TITLE>
</HEAD>
<BODY BGCOLOR="FFFFFF">
<P ALIGN=CENTER>
<FONT COLOR="RED" SIZE="6" FACE="ARIAL"><B> 5 класс </B></FONT><BR>
</P>
<FONT COLOR="BLUE" SIZE="4" FACE="COURIER"><B> Понедельник </B></FONT><BR>
<TABLE BORDER="1" WIDTH=100% BGCOLOR="99CCCC">
<TR BGCOLOR="CCCCFF" ALIGN=CENTER>
<TD>Урок</TD> <TD>5 А</TD> <TD>5 Б</TD> <TD>5 В</TD>
</TR>
<TR>
<TD>1</TD> <TD>Русский язык</TD> <TD>Литература</TD> <TD>История</TD>
</TR>
<TR>
<TD>2</TD> <TD>Алгебра</TD> <TD>Информатика</TD> <TD>Англ.язык</TD>
</TR>
```

```
<TR>
<TD>3</TD> <TD>История</TD> <TD>Информатика</TD> <TD> Алгебра </TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

3. Сохраните файл под именем 5.HTM.
4. Для просмотра созданной Web-страницы загрузите броузер Microsoft Internet Explorer.

Задание 5.

1. Дополните полученную Web-страницу по аналогии расписанием на последующие дни: ВТОРНИК, СРЕДУ, ЧЕТВЕРГ, ПЯТНИЦУ, СУББОТУ.

Практическая работа № 3
Создание изображения и использование его на Web-странице.
Работа с таблицами

Информация

Построение гипертекстовых связей

Важнейшим свойством языка HTML является возможность включения в документы ссылок на другие документы. Возможны ссылки:

- **на удаленный HTML файл,**
- **на некоторую точку в текущем HTML-документе,**
- **на любой файл, не являющийся HTML-документом.**

В качестве ссылки можно использовать текст или графику.

Ссылки в пределах одного документа

Такие ссылки требуют наличие двух частей: *метки* и самой *ссылки*. Метка определяет точку, к которой происходит переход по ссылке. Ссылка использует имя метки. Ссылки выделяются цветом или подчеркиванием, в зависимости от того, как настроен броузер. Для изменения цвета ссылки используется атрибуты LINK= **и** VLINK= **тэга <BODY …>**

Ссылка:

```
<A HREF="#ПН">Понедельник</A>
```

Перед именем метки (ПН**), указывающей куда производится ссылка, ставится символ #. Между символами > и < располагается текст (**Понедельник**), на котором производится щелчок для перехода по ссылке.**

Метка:

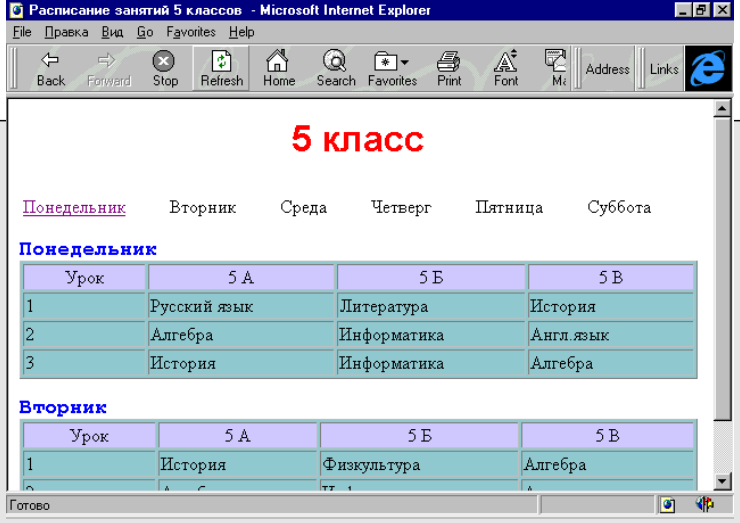
```
<A NAME="ПН">Понедельник</A>
```

Задание 1. Ссылки в пределах одного документа.

1. Дополните файл 5.HTM таблицей, содержащей название дней недели, поместив ее в начало Web-страницы:

```
…
<TABLE WIDTH=100%>
<TR >
<TD>Понедельник</TD>
<TD>Вторник</TD>
<TD>Среда</TD>
<TD>Четверг</TD>
<TD>Пятница</TD>
<TD>Суббота</TD>
</TR>
</TABLE>
<BR>
…
```

2. Вставьте в файл 5.HTM метку, указывающую ПОНЕДЕЛЬНИК:




```
…
<FONT COLOR="BLUE" SIZE="4" FACE="COURIER"><B>
<A NAME="ПН">Понедельник </A></B></FONT><BR>
…
```

3. Вставьте ссылку для выбранной метки:

```
…
<TABLE WIDTH=100%>
<TR >
<TD> <A HREF="#ПН">Понедельник</A> </TD>
<TD>Вторник</TD>
<TD>Среда</TD>
…
```

4. Сохраните файл.

5. Просмотрите полученную Web-страницу.

На экране вы должны увидеть:

Задание 2. Ссылки на другой HTML-документ.

Информация

Ссылки на другой HTML-документ

Ссылки позволяют щелчком по выделенному слову или фразе перейти к другому файлу.

Ссылка:

<A HREF="5.HTM">5 классы</A>

После имени файла 5.HTM**, указывается между символами > и < текст (**5 класс**), на котором производится щелчок для перехода на этот файл.**

1. Загрузите в броузер файл RASP.HTM.
2. Внесите изменения в файл:

```
<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#330066">
<P ALIGN=CENTER>
<FONT COLOR="#008080" SIZE="7"><B> Расписание </B></FONT><BR>
<FONT SIZE="6"> <I>занятий на вторник</I></FONT><BR><BR>
<IMG SRC="Wagon.gif">
```

```
</P>
<CENTER>
<TABLE WIDTH=60%>
<TR ><TD> <A HREF="5.HTM">5 класс</A> </TD><TD>6 класс</TD> </TR>
<TR ><TD>7 класс</TD><TD>8 класс</TD> </TR>
<TR ><TD>9 класс</TD><TD>10 класс</TD> </TR>
<TR><TD>11 класс</TD><TD> </TD></TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

2. Сохраните файл.

3. Просмотрите полученную Web-страницу.

Подведите курсор к ссылке "5 класс" и по щелчку мыши вы перейдете на другую Web-страницу (файл 5.HTM).

Задание 3. Графическая ссылка на другой HTML-документ.

1. Внесите изменения в файл 5.HTM так, чтобы в конце страницы была ссылка на головную страницу "Расписание занятий" (файл RASP.HTM). В качестве ссылки используется графический файл:

```
…
</TR>
</TABLE><BR>
<CENTER>
<A HREF="RASP.HTM"><IMG SRC="HOME.GIF" BORDER="0" ></A>
</CENTER>
</BODY>
</HTML>
```

В качестве ссылки выступает рисунок ("Стрелка Вверх"), находящийся в файле HOME.GIF.

Итоговое задание

1. Разработайте Web-страницы, рассказывающие о вашем классе.

На головной странице поместите рассказ о классе, классном руководителе. Рассказ об учениках разместите на отдельных Web-страницах. Укажите ссылки на страницы учеников с головной страница. Не забудьте установить ссылки возврата с Web-страниц учеников на головную страницу.

Как подготовить хорошую Web-страницу – несколько советов

1. **Следует обратить внимание на простоту и логику расположения информации на ваших страницах.**
2. **Один из способов сделать информацию более легкой для восприятия – это оставить на странице достаточно свободного места, не содержащего ни текста, ни рисунков. Страница, содержащая много информации, только отпугнет посетителя. Попытайтесь представить информацию в виде списков или таблиц, так чтобы достаточно легко можно было найти наиболее важные сведения.**
3. **Не размещайте одно изображение сразу за другим. Попробуйте распределить их по документу, оставив достаточно свободного пространства.**
4. **Информация должна размещаться частями, легкими для восприятия. Обратите внимание на длину абзацев. Если абзац слишком длинный, разбейте его на несколько.**
5. **Если Web-страница имеет большой объем, то возможно, вам следует вставить ссылки, позволяющие пользователю быстро перемещаться между частями одного документа. Иногда имеет смысл вместо одного документа длиной в пять страниц подготовить одну страницу, содержащую перечень тем, каждая из которых будет раскрыта в отдельных Web-страницах, и установить ссылки на соответствующие страницы.**
6. **Использование графики может дополнительно привлечь пользователей. Но необходимо помнить о времени загрузки вашей страницы, которое определяется количеством и объемом графической информации. Красивая картинка не произведет никакого впечатления, если для того, чтобы ее увидеть, приходится ждать пять минут, пока они загружаются.**

Практическая работа № 4
«Создание гиперссылок»

Цель работы: Научиться создавать гиперссылки на выбранные документы.

Задачи работы:

1. Овладеть методикой работы по созданию гиперссылок.
2. Получить навыки работы с полосой навигации;

Обеспечивающие средства: Сборник описаний практических работ; операционная система Windows XP, программа Internet Explorer; программа «Блокнот»; персональный компьютер.

Требования к отчету: Итоги практической работы представить в виде файла lab3.html на диске.

Технология работы:

1. Создать документ, в котором, в заголовке окна браузера должна быть надпись «Практическая 3»
2. Задать цвет для непосещаемой, посещаемой и активной гиперссылок.
3. Задать фон и соответствующий шрифт документу.
4. Использовать шрифты Arial и Courier New
5. Практическая №1- гиперссылка на соответствующий файл –должна быть справа на экране;
6. Практическая №2- гиперссылка на соответствующий файл –должна быть справа на экране
7. Заголовок (по центру и соответствующим шрифтом);
8. « Предметы» оформить виде вложенных списков по образцу предложенному ниже;
9. «Специальности» -заголовок (по центру экрана и соответствующим шрифтом) написать сокращенное и расшифрованное название специальностей с использованием списков определений.
10. Используя вложенные списки создать документ согласно образцу:

Изучаемые предметы

- Предметы преподаваемые на первом курсе:

- Математика
- История
- Программирование
  - Теория
  - Практика
- Химия
- Физика
- География

❖ Предметы преподаваемые на втором курсе
❖ Предметы преподаваемые на третьем курсе:

Специальности :

- ***ФИН*** *– Финансы* ***0603***
- ***БХ*** *- Экономика и бухгалтерский учет* ***0601***
- ***АС*** *– Автоматизированные системы обработки информации и управления* ***2202***
- ***ПВ –*** Правоведение ***0201***

11. Сохранить файл как lab3.txt в блокноте и как lab3.html для просмотра в браузере.

Практическая работа № 5
«Создание Web – страницы»

Цель работы: Научиться создавать Web-страницы.

Задачи работы:

1. Познакомиться с языком HTML.
2. Овладеть техникой создания Web-страниц.

Обеспечивающие средства: Сборник описаний практических работ; операционная система Windows XP, программа Internet Explorer; программа «Блокнот», персональный компьютер.

Требования к отчету: Итоги практической работы представить в виде файла lab1.html на диске.

Технология работы:

1.Создайте папку «Сайт» в папке «Мои документы». Откройте программу «Блокнот» в качестве простого инструмента для создания веб-страниц. Блокнот — это несложный текстовый редактор, используемый для создания простых документов. Наиболее часто программа «Блокнот» используется для просмотра и редактирования текстовых (TXT) файлов,

для создания файлов веб-страниц (HTML). Программа «Блокнот» поддерживает только основное форматирование, поэтому случайное сохранение специального форматирования в документах, в которых должен остаться чистый текст, исключено. Это особенно полезно при создании HTML-документов для веб-страниц, так как особые знаки или другое форматирование могут не отображаться на опубликованных веб-страницах. Все документы HTML имеют одну и ту же структуру, определяемую фиксированным набором тегов структуры.

Документ HTML всегда начинается с тега <HTML> и заканчивается закрывающим тегом </HTML>.

Внутри документа выделяются два основных раздела: раздел заголовков и тело документа, идущие друг за другом.

Основное содержание размещается в теле документа, которое ограничивается парным тегом <BODY>. Простейший правильный документ HTML, содержащий все теги, определяющие структуру, имеет вид:

<HTML>

<HEAD> <TITLE> Заголовок документа </TITLE ></HEAD>

<BODY>

Текст документа

</BODY>

</HTML>

Задание:

2.Создайте Web-страницу, в которой должны присутствовать цветной текст, список, рисунок, таблица, используя стандартную программу Блокнот, используя примеры приведенной ниже таблицы.

Запустите Internet Explorer.

Откройте созданный файл.

Теги и примеры их оформления	Пояснения
<HTML> <HEAD><TITLE>Курсовая работа</TITLE></HEAD> <BODY>	Раздел заголовков содержит информацию, описывающую документ в целом, и ограничивается тегами <HEAD> </HEAD>. Раздел заголовков должен содержать общий заголовок документа, ограниченный парным тегом <TITLE>
<P><FONT SIZE=”14”FACE=”ARIAL”COLOR=”RED”>	Обычный абзац начинается с тега <P> Тег <FONT> управляет параметрами

Содержание </FONT>	шрифта, содержит атрибуты COLOR= цвет текста (например, “GREEN”, “RED”, и т.д.), FACE= гарнитура шрифта или имя шрифта (например, ARIAL и т.д.), SIZE= размер шрифта.
<H1> Введение </H1> <H1> 1. Использование информационных технологий при решении экономических задач </H1> 1.1 Классификация ИТ <H2>	Язык HTML поддерживает 6 уровней заголовков от <H1> до <H6>
Текст до ссылки. <A HREF=”first.htm”> Ссылка. </A>	Гипертекстовая ссылка определяется парным тегом <A>. Обязательным является HREF= (знак равенства показывает, что необходимо задать значение этого атрибута, т.е. адрес документа, на который указывается ссылка)
<IMG SRC= “picture.gif” ALINE=”MIDDLE”>.	Вставка графического элемента (используется только два формата –GIF, JPEG). Для подготовки изображения можно использовать PAINT. Для вставки рисунка используется текстовый элемент, задаваемый непарным тегом <IMG> с обязательным атрибутом SRC=, задающим адрес файла с изображением. Атрибут ALINE= режим взаимодействия изображения с текстом (BOT-TOM – рисунок выше текста, MIDDLE – рисунок в середине текста,LEFT – левее, а RIGHT – правее текста).
</BODY> </HTML>	
<HTML> <HEAD> <TITLE> Заголовок документа </TITLE></HEAD> <BODY> <P> Содержание <UL>	Списки Упорядоченные (нумерованные) списки создаются при помощи парных тегов <OL>, маркированные списки при помощи <UL>. Эти списки могут содержать только элементы списка, определяемые парным тегом <LI>. Закрывающий тег </LI>

<table>
<tr>
<td><LI> <A HREF= “vvedenie.htm”> Введение

</A>

<OL>

<LI>Использование информационных технологий при решении экономических задач

<LI>Решение конкретной экономической задачи с использованием Excel

</OL></td>
<td>можно опускать. Например, Содержание документа можно оформлять списком и использовать гиперссылки.</td>
</tr>
<tr>
<td><LI><A HREF=”conclusion.htm”> Заключение</A>

</UL>

<BODY>

</HTML></td>
<td>Содержание

<u>Введение</u>

1. Использование информационных технологий при решении экономических задач

2. Решение конкретной экономической задачи с использованием Excel

<u>Заключение</u></td>
</tr>
<tr>
<td><HTML>

<HEAD><TITLE> Заголовок документа

</TITLE></HEAD>

<BODY>

<TABLE>

<CAPTION>Таблица №1 </CAPTION>

<TH> Ф.И.О.<TH> Адрес

<TR><TD> Панова И.И.

<TD>Мира 6-21

<TR><TD> Мишина В.П.</td>
<td>Таблица в языке HTML задается парным тегом <TABLE>. Заголовок таблицы определяется парным тегом <CAPTION>, строки таблицы задаются при помощи тегов <TR>. Ячейки в заголовках столбцов -парным тегом <TH>. Обычные ячейки - <TD>. Закрывающиеся теги можно опускать.

Таблица №1

<table>
<tr><td>Ф.И.О.</td><td>Адрес</td></tr>
<tr><td>Панова И.И.</td><td>Мира 6,21</td></tr>
<tr><td>Мишина В.П.</td><td>Победы 47,154</td></tr>
<tr><td>Новикова Е.Н.</td><td>Московская 23-4</td></tr>
</table>
</td>
</tr>
</table>

<TD>Победы 47-154 <TR><TD> Новикова Е.Н. <TD>Московская 23-4 </TABLE> </BODY> </HTML>	

3.Сохранить файл как lab1.txt в блокноте и как lab1.html для просмотра в браузере.

Практическая работа № 6

«Табличная верстка макета сайта»

Цель работы: 1. Изучение особенностей табличной верстки макета и получение практических навыков в создании сайтов. 2. Закрепление теоретического материала.

Оборудование и ПО: ПК, операционная система Windows7, офисный пакет Microsoft.

Порядок выполнения работы:

1. Ознакомление с методическими рекомендациями по табличной верстке макета сайта. *Просмотр обучающих роликов к практическому занятию № 10.*
2. Получение задания для практического выполнения.
3. Выполнение практической части задания в соответствии с рекомендациями.

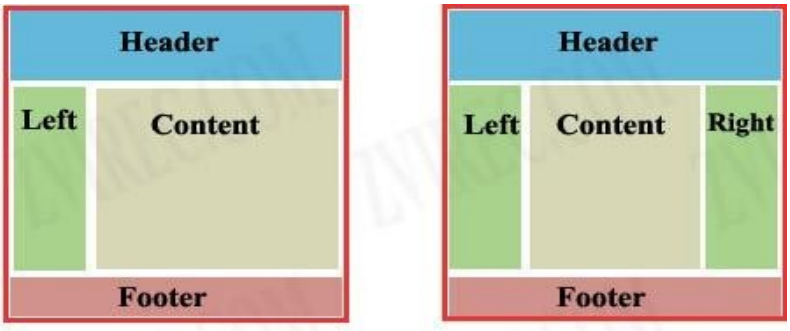
1. Особенности табличной верстки (универсальный табличный макет)

Web-страница должна иметь внутреннюю структуру, так называемый базовый каркас. Часто этот каркас представляет собой таблицу – набор аккуратно выстроенных прямоугольников, в которых размещаются компоненты web-страницы. Границы между ячейками такой таблицы можно сделать видимыми или, ради большей элегантности, невидимыми. В этом случае говорят о *табличной верстке* сайтов.

В последнее время очень популярной стала *блочная верстка* сайтов или, как ее еще называют, *div-верстка*. Суть ее заключается в том, что базовый каркас (шаблон) страницы формируется из блоков, в качестве которых выступает html-тег **DIV**. С помощью стилей эти блоки позиционируются на странице определенным образом, формируя каркас, который уже затем наполняют содержимым.

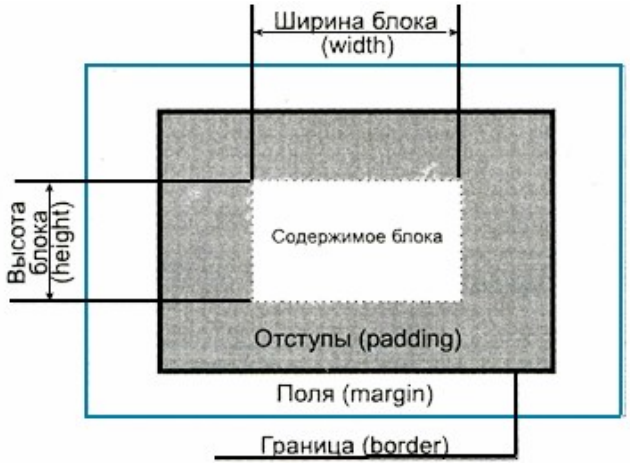
Главный плюс табличной верстки - простота реализации и кроссбраузерность (одинаково выглядят во всех браузерах).

Рассмотрим, по какому принципу строятся табличные макеты страницы и рассмотрим построение различных вариаций 2-х наиболее популярных шаблонов, приведенных на рисунке ниже.



Таким образом, изучив данный материал, Вы научитесь создавать самые популярные табличные макеты страницы, и, причем, по всем правилам. А макет – это уже практически готовый сайт. Вам останется лишь наполнить соответствующие блоки информацией.

2. Верстаем двухколонный макет фиксированной ширины (меню слева)



#container { width:600px;
height:100%; border:1px
solid gray; margin:0 auto;
}
/*Делаем нужное выравнивание в ячейках таблицы*/
#container td{**MARGIN (Поля) –** это расстояние от границы (рамки) блока, до ближайших элементов, или, если их нет, до краев документа.

PADDING (Отступы) – как бы внутреннее расстояние, между границей (рамкой) и содержимым блока.

i. Файл style.css

/*Делаем чтобы макет прижимался вплотную, т.е. убираем любые возможные отступы*/
body,html{ margin:
0px; padding:0px;
}
/*Стиль общей таблицы. Если нужен резиновый макет, нужно задать ширину 100% и убрать рамку, т.к. с ней будет некрасиво*/

```
vertical-align:top
        }
/*Стиль шапки сайта*/ #header {
background-color:#999999;  border-
bottom:1px          solid          black;
height:80px;
        }
/*  Стили для внутренней таблицы */ #maket {
height:100%; width:100%;
        }
/*Стили  левой  колонки  (там,  где  обычно  меню)*/
#left_column{
width:180px;
background-color:#0099CC;
        }
/*Стили  колонки с основным содержанием*/ #main_column {
padding:10px;
        }
/*  Конец стилей для внутренней таблицы              */
/*стиль подвала сайта*/ #footer{
background-color:#999999;
border-top:1px     solid     black;
height:15px;
        }
```

ii. Файл index.html

```
<!DOCTYPE        HTML        PUBLIC        "-//W3C//DTD        HTML        4.01        Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Универсальный макет сайта</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<!--Вставляем главную таблицу с тремя строками - шапка, средняя часть и подвал-->
<table  cellpadding="0" cellspacing="0" id="container">
         <tr>
           <td id="header" >Шапка сайта</td>
         </tr>
          <tr>
           <td >
<!--Вставляем  табличку 100% ширины, с двумя колонками, одна фиксированной ширины,
        другая - без указания ширины (в таком случае она автоматически растягивается на всю оставшуюся ширину)-->
           <table cellpadding="0" cellspacing="0" id="maket">
             <tr>
               <td id="left_column">Меню</td>
     <td  id="main_column"><p>Содержание  сайта  любой  текст  здесь  Содержание  сайта  любой  текст  здесь
Содержание сайта любой текст здесь </p>
       <p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь </p>
       <p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь </p>
       <p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь </p></p></td>
           </tr>
         </table>
```

```
<!--Конец внутренней таблицы-->
          </td>
         </tr>
          <tr>
           <td id ="footer">Подвал</td>
         </tr>
         </table>
<!--Конец главной таблицы-->
</body>
</html>
```

3. Верстаем трехколонный макет фиксированной ширины Файл style.css

```
/*Делаем чтобы макет прижимался вплотную, т.е. убираем любые возможные отступы*/
body,html{
margin:0px; padding:0px;
        }
/*Стиль общей таблицы. Если нужен резиновый макет, нужно задать ширину 100% и убрать рамку, т.к. с ней будет некрасиво*/
#container  {  width:900px;
height:100%;      border:1px
solid gray; margin:0 auto;
        }
/*Делаем нужное выравнивание в ячейках таблицы*/
#container td{
vertical-align:top
        }
/*Стиль шапки сайта*/ #header {
background-color:#cc66ff;  border-
bottom:1px        solid        black;
height:80px;
        }
#header h1 {
          margin: 0; /* Обнуляем отступы для заголовка первого уровня, находящегося в шапке. Это нужно обязательно делать, при использовании заголовков. Если используются параграфы, то тоже нужно обнулить отступы для них. */
                    padding: 10px 0; /* Задаем поля */
        }
/*  Стили для внутренней таблицы */ #maket {
height:100%; width:100%;
        }
/*Стили  левой  колонки*/
#left_column{ width:150px;
background-color:#0099CC;
        }
/*Стили  колонки с основным содержанием*/ #main_column {
padding:10px;
        }
# main_column h1 {
margin:0px; /* Обнуляем отсупы для заголовка первого уровня, находящегося в блоке контента.*/
        }
# main_column p {
margin:0px; /* Обнуляем отсупы для параграфов я, находящегося в блоке контента.*/
padding:5px; /*задаем поля, т.е. чтобы был промежуток между строками, чтобы было понятно, что это параграф :)*/
        }
/*Стили правой колонки*/
```

```
#right_column{ width:150px
;
background-color:#339966;
        }
/*  Конец стилей для внутренней таблицы              */
/*стиль подвала сайта*/ #footer{
background-color:#33ff99;
border-top:1px     solid     black;
height:15px;
        }
#footer p{
margin:0px;   /*  обнуляем  отступы*/  padding:
10px 0; /*  задаем поля */
        }
```

iii. Файл index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Универсальный макет сайта</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<!--Вставляем главную таблицу с тремя строками - шапка, средняя часть и подвал-->
<table  cellpadding="0" cellspacing="0" id="container">
          <tr>
            <td id="header" ><H1>Шапка сайта</H1></td>
          </tr>
            <tr>
            <td >
<!--Вставляем  табличку 100% ширины, с двумя колонками, одна фиксированной ширины,
        другая - без указания ширины (в таком случае она автоматически растягивается на всю оставшуюся ширину)-->
            <table cellpadding="0" cellspacing="0" id="maket">
              <tr>
                <td id="left_column">Меню</td>
                <td id="main_column"><H1>Главная колонка </H1>
<p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь
</p>
        <p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь </p>
        <p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь </p>
        <p>Содержание сайта любой текст здесь Содержание сайта любой текст здесь Содержание сайта любой текст
здесь Содержание сайта любой текст здесь Содержание сайта любой текст здесь </p></td>
              <td id="right_column">Новости</td>
            </tr>
            </table>
<!--Конец внутренней таблицы-->
            </td>
          </tr>
           <tr>
            <td id ="footer">Подвал
            <p>&copy; Все права защищены </p></td>
          </tr>
          </table>
<!--Конец главной  таблицы-->
</body>
```

</html>

4. Выполнение практической части

Практическая часть задания выполняется по одному из 10-ти вариантов. Варианты имеют разные исходные данные и одинаковую задачу – построение html-документа по одному из приведенных выше примеров **в разделах 2**(*двухколонный макет*) и **3**(*трехколонный макет*) в описательной части методического пособия. Обратите внимание на необходимость создания файла style.css

Вариант 1	*Вариант 2*	*Вариант 3*
Исходные данные: 1. трехколонный макет; 2. заполнить header; 3. Разместить изображение в контенте.	**Исходные данные:** 1. двухколонный макет; 2. Заполнить header; 3. Разместить произвольный текст и изображение в контенте; 4. Указать разделы в оглавлении слева.	**Исходные данные:** 1. трехколонный макет; 2. Заполнить footer; 3. Разместить произвольный текст в контенте.
Вариант 4	*Вариант 5*	*Вариант 6*
Исходные данные: 1. двухколонный макет; 2. Заполнить footer; 3. Разместить простой список в контенте; 4. Указать разделы в оглавлении слева.	**Исходные данные:** 1. трехколонный макет; 2. Заполнить header; 3. Разместить произвольный текст в контенте; 4. Указать разделы в оглавлении слева.	**Исходные данные:** 1. двухколонный макет; 2. Заполнить header; 3. Разместить произвольный текст в контенте; 4. Указать разделы в оглавлении слева.
Вариант 7	*Вариант 8*	*Вариант 9*
Исходные данные: 1. трехколонный макет; 2. Заполнить header и footer; 3. Разместить изображение в контенте; 4. Добавить ссылки в правом разделе	**Исходные данные:** 1. двухколонный макет; 2. Заполнить footer; 3. Разместить изображение в контенте; 4. Указать разделы в оглавлении слева.	**Исходные данные:** 1. трехколонный макет; 2. Заполнить header; 3. Разместить произвольный текст и изображение в контенте; 4. Указать разделы в оглавлении слева
Вариант 10		
Исходные данные: 1. двухколонный макет; 2. Заполнить footer; 3. Разместить изображение и простой список в контенте; 4. Указать разделы в оглавлении слева.		

Контрольные вопросы:

1. В чем заключаются особенности табличной верстки сайта в связи с понятием «каркас» и «компоненты»?
2. Перечислите и охарактеризуйте типы макетов сайта.
3. Для чего создается файл style.css?

Лабораторная работа №7.

«Подключение стилей CSS к сайту»

Цель работы: Изучить способы подключения стилей CSS к сайту.

В данной лабораторной работе рассматриваются способы подключения стилей CSS3 к сайту. Существует три способа подключения стилей к сайту.

Первый способ используется больше для обучения или для минимального изменения свойств отдельного тега, он увеличивает вес страницы и тем самым увеличивает время загрузки страницы, помимо этого увеличивает количество кода, что существенно усложняет работу с ним. Этот способ предполагает написание стилей в атрибуте тега *style* (пример <p style=”width: 10px;”> </p>).

Второй способ используется чаще чем первый, он подходит если используется небольшое количество стилей, либо когда возникает необходимость на отдельной странице сайта подключить стили не используемые больше нигде на сайте. Стили во втором способе подключаются к странице в части <head></head> с использованием тега <style><style>.

Третий самый распространенный способ подключения стилей это подключение внешнего файла со стилями, который имеет расширение .css и содержит внутри себя правила и свойства CSS. Данный способ имеет множество достоинств перед предыдущими и существенно облегчает работу с оформлением сайта.

Ход работы

Способ 1. Подключение стилей с использованием атрибута тега *style*.

Как известно из уроков по HTML мы уже знаем что каждый тег имеет определенный набор атрибутов. Некоторые атрибуты уникальные и применяются лишь к определенным тегам, другие же универсальные и могут быть применены к большинству тегов. Одним из таких уникальных атрибутов является атрибут *style*, который позволяет добавить стили CSS к данному тегу.

Синтаксис

```
<h1 style="color: red; font-size: 15pt;">Свойства с помощью атрибута</h1>
```

Все стили записываются через точку с запятой. Значение атрибута помещается в кавычки. Данный способ не рекомендуется использовать везде и постоянно, так как он убивает напрочь суть универсальности стилей. Помимо этого данный способ написания стилей увеличивает вес страницы, что влияет на скорость загрузки сайта. Данный стиль будет применяться в приоритете другим стилям так как обладает большей значимостью.

Способ 2. Подключение стилей с использованием тега *<style></style>* в части *<head></head>*.

Второй способ подключения стилей является более распространенным, он позволяет создать стили для каждой отдельной страницы используя парный тег *<style></style>*.

Синтаксис

```
<head>
<style> div {


}
#menu {
```

color: red; font-size:
18pt;

Background-color: black;
Color: white;}

</style>
</head>

Данный способ добавляет некоторую универсальность стилям, т.е. стили написанные таким способом будут применяться к соответствующим тегам на всей странице. Данный способ немного увеличивает весь страницы и соответственно увеличивает время загрузки. Данный способ подходит в том случае, если вы хотите оформить какую либо страницу вашего сайта определенными стилями, которые не будут повторяться на других веб страницах сайта. Основное что следует запомнить это то что данные стили будут применяться только на той странице в коде которой в части HEAD он будет написан.

Способ 3. Подключение внешнего файла со стилями CSS.

Данный способ является универсальным и самым распространенным. Он позволяет создать стили CSS в отдельном текстовом документе, который имеет расширение .css (например: style.css). Плюсы данного способа в том что все стиле для всего сайта записываются в одном документе, что существенно облегчает дальнейшую работу с внешним видом сайта. На сайте всегда присутствуют элементы которые на всех страницах сайта отображаются одинаково (например это может быть меню, шапка, подвал, какие либо кнопки и прочие элементы).

Подключение внешнего файла CSS

<head>

<link rel="stylesheet" href="https://web-legko.ru/mysite.css"> **(Подключение файла стилей с стороннего сайта)**

<link rel="stylesheet" href="css/style.css"> **(Подключение файла стилей из директории сайта)**

</head>

Значение атрибута тега <link> — rel остаётся неизменным независимо от кода, как приведено в данном примере.

Значение атрибута href задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Таким способом есть возможность подключать стили с другого сайта.

Практическое задание

1. Создайте страницу HTML, назовите ее index.html.
2. В данном документе напишите «скелет» странцы HTML

```
<!DOCTYPE html>
<html>
<head>
        <title>Практическая работа «Подключение CSS к сайту»</title>
</head>
<body>

</body>
</html>
```

3. Внутри тега BODY поместите тег DIV
4. К данному тегу с помощью атрибута *style* примените стиль CSS который будет изменять цвет текста на красный

*** Практическая работа CSS ***

5. Сохраните и просмотрите результат
6. Далее в части HEAD поместите тег <*style*> и запишите в него следующий код

```
<style>
div {


}

color: green; width: 200px;
height: 200px;
backgroung-color: #000006;
</style>
```

7. Сохраните и просмотрите результат в браузере
8. Заметьте, что цвет текста в данном случае для одного и того же элемента задается двумя способами сразу.
9. Далее в папке где мы сохранили файл index.html, создаем файл *style.css.* и записываем в него следующий код

body {

div {
font-size: 20px;
color: yellow;
}

color: #887700;
float: right;
text-align: center;
}

10. Сохраните файл и возвращаемся к редактированию страницы HTML
11. Теперь необходимо подключить файл со стилями к странице. С помощью кода

<link rel="stylesheet" href="style.css">

(так как файл со стилями у нас находится в той же папке что и файл index.html, то мы прописываем относительный путь для файла стилей)
Данный код помещаем в часть HEAD.

Итог:

По итогу выполнения данной работы мы рассмотрели способы подключения стилей CSS к сайту HTML, рассмотрели достоинства и недостатки каждого из способов. По окончании работы ответьте на контрольные вопросы для закрепления знаний.

Ответить на контрольные вопросы

1. Если к одному тегу написаны стили всеми тремя способами, то в каком порядке и с каким приоритетом они будут применяться?
2. Какой из способов является наиболее оптимальным для изменения конкретного элемента на одной конкретной странице HTML?
3. Какой из способов следует использовать при разработке и создании большого проекта?
4. В чем недостатки каждого из способов подключения стилей?

Практическая работа №8
Тема: «Оформление прямоугольных блоков средствами CSS. Отступы между блоков. Рамки»

Цель: Ознакомиться с блочной разметкой страницы. Научиться управлять размещением блочных элементов на странице.

Теоретическая часть

Блочная модель

Для браузера каждый тег – это контейнер, у которого есть содержимое, внутренний отступ, внешние поля, а также рамка. Блок занимает пространство на странице равное сумме ширины содержимого, отступа, поля и рамки.

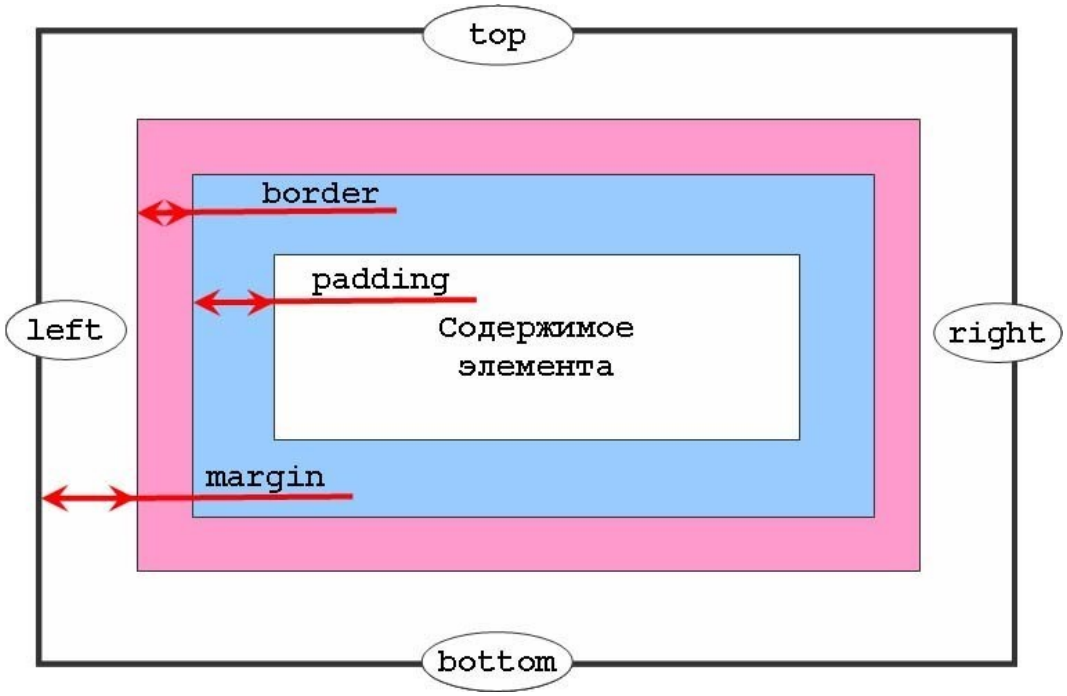


Рис. 1. Блочная модель элемента

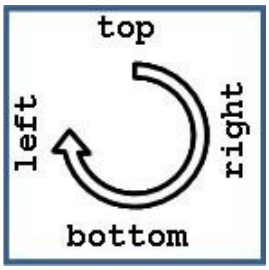


Рис. 2. Порядок установки параметров

Для всех приведенных ниже атрибутов разрешается использовать 1, 2, 3 или 4 значения, перечисляя их через пробел. В качестве значений указываются числа в любом допустимом для CSS формате. В случае применения процентов, отсчет ведется относительно ширины блока.

Атрибуты блочных элементов

Margin

Возможные атрибуты:

- margin-top - ширина верхнего поля.
- margin-right - ширина правого поля.
- margin-bottom - ширина нижнего поля.
- margin-left - ширина левого поля.
- inherit - применяется значение родительского элемента.

```
p {
margin: 20px 30px 5px 0;
}
```

f. padding

Возможные атрибуты:

- padding-top - ширина верхнего промежутка.
- padding-right - ширина правого промежутка.
- padding-bottom - ширина нижнего промежутка.
- padding-left - ширина левого промежутка.
- inherit - применяется значение родительского элемента.

```
h1{
padding: 10px 100px;
}
```

g. border

Возможные атрибуты:

- border-width - толщина границ.
- border-style - стиль границ.
- border-color - цвет границ.
- inherit - применяется значение родительского элемента.

```
p{
border:5px groove lightgrey;
}
```

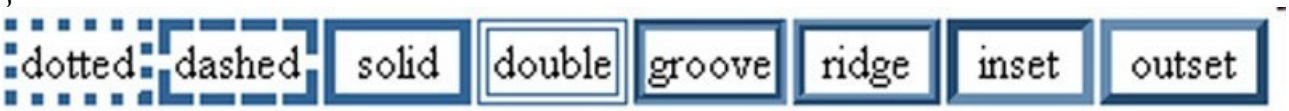


Рис. 3. Тип границ элемента

Float (обтекание блока, плавающий блок)

Возможные атрибуты:

- right - выравнивает блок по правому краю, а остальные элементы будут обтекать его с левой стороны, начиная с самого верха;
- left - выравнивает блок по левому краю, а остальные элементы будут обтекать его с правой стороны, начиная с самого верха;
- none - это значение стоит по умолчанию, блок остаётся на месте;
- inherit - наследует значение у родительского блока.

Clear (запрет на обтекание)

Возможные атрибуты:

- right - отменяет обтекание структурного блока с правой стороны;
- left - вотменяет обтекание структурного блока с левой стороны;
- both - отменяет обтекание блока с двух сторон;
- none - отменяет действие данного свойства;
- inherit - наследует значение у родителя.

i. Ширина блочных элементов

- Если ширина не указана явно тогда она получает значение auto (width: auto) и элемент занимает всю доступную ширину контентной области родителя. В данной ситуации width родителя = реальной ширине дочернего блока.
- Изменение ширины контента родителя влечет за собой изменение ширины дочерних элементов.
- Ширина элемента не может стать меньше указанного min-width (если конечно оно задано) даже если ширина родителя окажется меньше.

- Ширина элемента не может стать больше указанного max-width даже если размеры родителя ему это позволяют.
- Внешние отступы (margin-left, margin-right) влияют на реальную ширину блока.
- Если ширина явно указана в абсолютных единицах, тогда элемент будет иметь заданную ширину независимо от ширины родителя. Внешние отступы в таком случае тоже уже не влияют на размер.
- Если ширина явно указана в относительных единицах, то она будет рассчитываться относительно ширины контентной области родителя. Будет каждый раз пересчитана при изменениях width родителя. Внешние отступы на размер элемента влияния оказывать не будут.

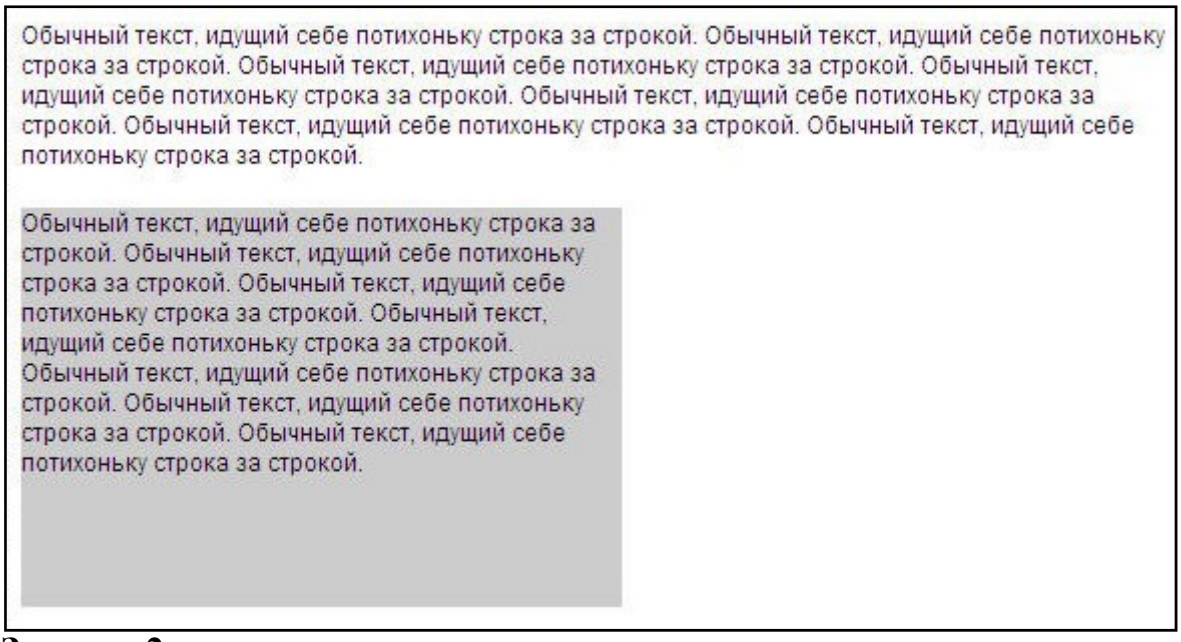
ii. Высота блочного элемента

- По умолчанию высота принимает значение auto (height: auto) и зависит от реальной высоты содержимого. Соответственно, если меняется высота содержимого, меняется высота и блока.
- Если блоку заданы минимальная или максимальная высота (min-height, max-height), или высота (height) явно указана — поведение блочного элемента аналогично поведению с min-width/max-width/width.

Практическая часть

Задание 1

Поместить текст в блок. Группа Блок позволяет заключить текст в блок. Вверху обычный текст, а внизу текст после применения стиля Блок.

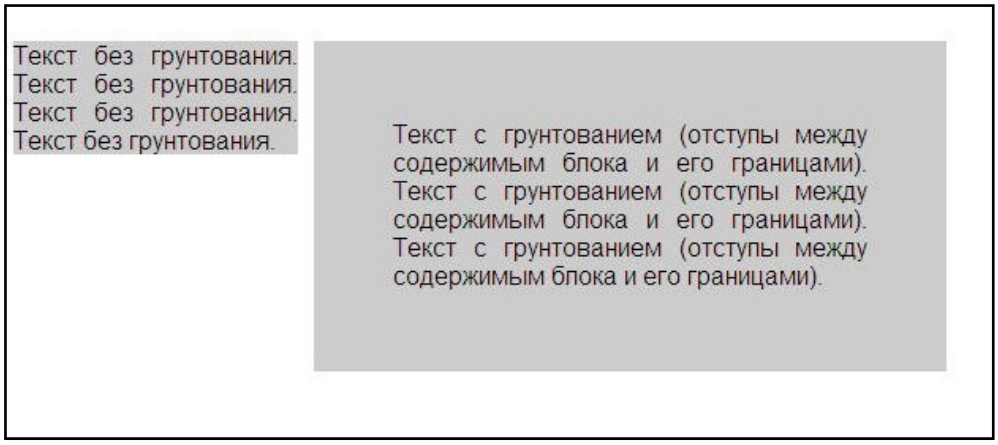


Задание 2.

Создать плавающие элементы по образцу. "Плавающие" и не "плавающие" объекты. Задайте фоны и границы.

Задание 3.

Задание внутренних отступов в блоке. Параметры отступа - грунтования (Padding). Слева обычный блок, справа - со значениями Padding (грунтование 50 пикселей со всех сторон). Задайте фон.



Задание 4.

Работа с внешними отступами (margin). Параметры **Отступ** (Margin) определяют отступы между внешними границами блока и его содержимым. Вверху - без отступа, внизу со значениями отступа 10 пикселей справа. Добавьте границы в каждом блоке и измените фоны.

Задание 5

Создайте блочный каркас сайта (каждый элемент – отдельный блок div)

Меню сайта	Основной контент сайта
Футер сайта	

Практическая работа №9

CSS Селекторы. CSS Включения.

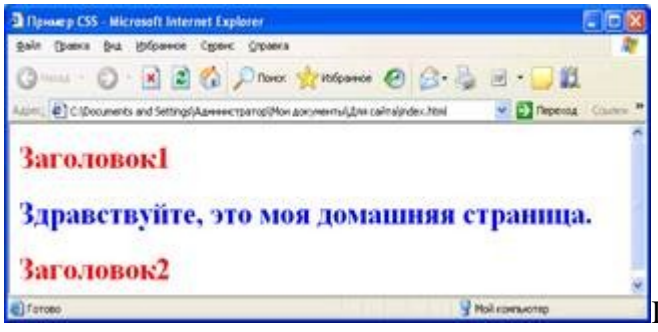
Классовые селекторы (Class Selectors):

Синтаксис: **селектор.класс {свойства}**

А что делать, если нужно некоторые заголовки отобразить по-другому? CSS реализует возможность присваивать стили не всем одинаковым элементам страницы, а избирательно – для этого используется параметр CLASS = "имя класса" или идентификатор ID=«имя элемента», присваивающиеся любому элементу страницы. Рассмотрим эти возможности подробнее.

Класс позволяет задать разные правила форматирования для одного элемента определённого типа или всех элементов документа. Имя класса указывается в селекторе правила после имени тега и отделяется от него точкой. Можно определить несколько правил форматирования для одного элемента и с помощью параметра CLASS соответствующего тега применять разные правила форматирования.

CLASS - атрибут элемента в HTML, определяющий его класс. В CSS можно описать собственные стили для различных классов одних и тех же элементов.



ПРИМЕР: *Содержимое таблицы стилей:*

```
H1.blue {color:blue; size:20pt;}
```

Все элементы H1 с атрибутом CLASS="blue" станут синими.

<body>

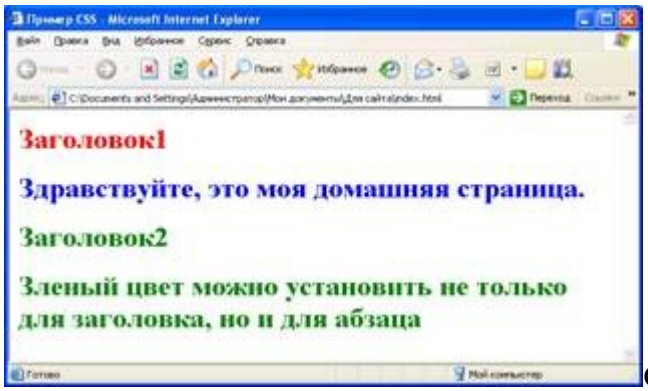
<h1>Заголовок1</h1>

**<h1 class="blue"> Здравствуйте, это моя домашняя страница. **

<h1>Заголовок2</h1>

</body>

Классы могут так же быть описаны без явного привязывания их к определенным элементам.



Синтаксис: **.класс {свойства}**

ПРИМЕР: *Содержимое таблицы стилей:*

```
.green {color:green;}
```

В данном случае все элементы с атрибутом CLASS="green" станут зелеными.

<body>

<h1>Заголовок1</h1>

**<h1 class="blue"> Здравствуйте, это моя домашняя страница. **

<h1 class="green">Заголовок2</h1>

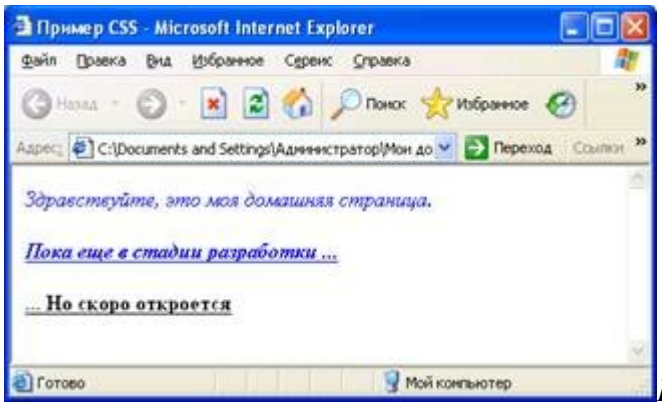
** Зленый цвет можно установить не только для заголовка,**

**но и для абзаца **

</body>

i. Идентификаторы

Присвоение стилей с помощью идентификаторов применяется в случае, если данному идентификатору соответствует только один элемент на странице. Если элементов, которым необходимо присвоить такой стиль, несколько – это уже класс.



ID селекторы (ID Selectors):

Синтаксис: **#id {свойства}**

Правила таблиц стилей регламентируют использование уникального идентификационного имени элемента в качестве селектора, предваряя его символом #:

Идентификаторы используются в основном для придания одному или нескольким элементам одного класса индивидуальных свойств. Скажем, Вы создали класс blue - синий курсив. Но Вам понадобился жирный подчеркнутый текст синим курсивом. Конечно, можно создать новый класс, но зачем? Проще описать ID. Например "boldunderline". И все элементы класса blue с значением ID "boldunderline" станут жирным подчеркнутым синим курсивом. Произойдет как бы синтез свойств класса blue и идентификатора boldunderline.

ПРИМЕР: *Содержимое таблицы стилей:*

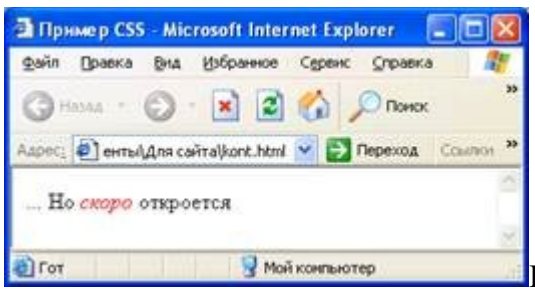
```
.blue {color:blue; font-style:italic}
#boldunderline {text-decoration:underline; font-weight:bold}
```

HTML-код основного документа:

```
<body>
<p class="blue"> Здравствуйте, это моя домашняя страница. </p>
<p class="blue" id="boldunderline"> Пока еще в стадии разработки ... </p>
<p id="boldunderline">... Но скоро откроется </p>
</body>
```

Как видно из примера, атрибут ID может быть использован без указания класса (последний параграф примера. Тогда параграф будет обладать только свойствами ID "boldunderline" (в примере - жирный, подчеркнутый текст).

Контекстные селекторы (Contextual Selectors):



Контекстные селекторы - это сочетания нескольких обыкновенных селекторов. Стиль задается только элементами в заданной последовательности в зависимости от каскадного порядка.

ПРИМЕР: *Содержимое таблицы стилей:*

```
P I {color:red;}
```

В данном примере все элементы I внутри элементов P будут иметь заданный стиль.

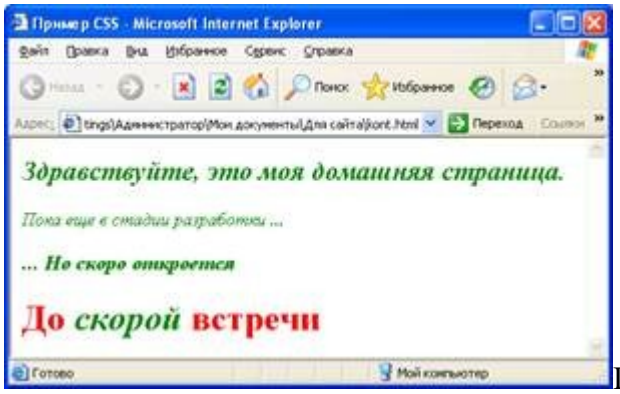
<body>

** ... Но <I>скоро</I> откроется **

</body>

Придание нескольким элементам одинаковых свойств:

Скажем Вам нужно придать нескольким элементам Вашей Web - страницы одинаковых свойств. В этом случае при определении селекторы перечисляются через запятую перед блоком свойств.



ПРИМЕР: *Содержимое файла* ***styles.css****:*

```
h2,h3,h4,p,b {color:green; font-style:italic;}
```

Все элементы h2, h3, h4, p и b будут зелеными с курсивом.

<body>

<h2> Здравствуйте, это моя домашняя страница. </h2>

** Пока еще в стадии разработки ... **

<h3>... Но скоро откроется </h3>

<h1>До скорой встречи</h1>

</body>

Псевдоклассы и псевдоэлементы. Псевдоклассы и псевдоэлементы - это особые классы и элементы, присущие CSS и автоматически определяемые поддерживающими CSS браузерами.

Синтаксис: **селектор:псевдокласс { свойства }**
селектор.класс:псевдокласс { свойства }
селектор:псевдоэлемент { свойства }
селектор.класс:псевдоэлемент { свойства }

Псевдоклассы различают разные типы одного элемента, создавая при определении собственные стили для каждого из них. Например,

```
a:link,a:visited {color:blue}
```

```
a:active {color:red}
a:hover {text-decoration:none}
```

В данном примере все элементы ссылки будут синими. При нажатии (в активном состоянии) поменяют цвет на красный. И при подведении курсора мышки исчезнет подчеркивание.

Псевдоэлементы позволяют объединять несколько стилей для какого-либо объекта. Например, вы можете задать свойства для первой буквы параграфа. Для этого вы назначаете для тега **P** псевдоэлемент **first-letter**, в котором устанавливаете различные стили:

```
p:first-letter {float:right;font-size:2em;color:red;}
```

Список псевдоклассов и псевдоэлементов :

Anchor Pseudo Classes - эти псевдоклассы элемента <a rel="nofollow ugc" target="_blank" href=" ">, обозначающего ссылку. Псевдоклассы этого элемента: (ссылка), active (активная ссылка), visited (посещенный ранее URL), hover (псевдокласс, возникающий при поднесении курсора к ссылке, не работает в Netscape).

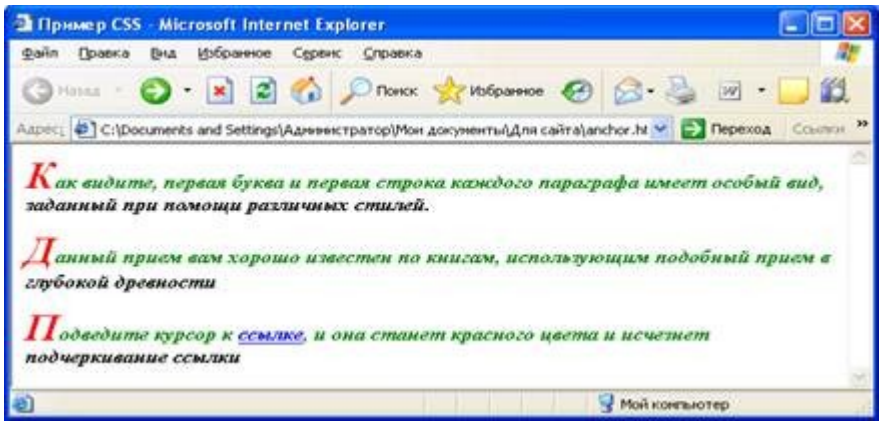
First Line Pseudo-element - first-line. Этот псевдоэлемент может быть использован с block-level элементами (p, h1 и т.д.). Он изменяет стиль первой строки этих элементов.

First Letter Pseudo-element - first-letter. Похож на first-line, но влияет не на всю строку, а только на первый символ.

:first-letter	Устанавливает стили для первой буквы объекта
:first-line	Устанавливает стили для первой строки объекта
:hover	Устанавливает стили для элемента, когда пользователь подводит курсор мыши к ссылке. Данный псевдокласс часто используется вместе с псевдоклассами :active, :link и :visited
:active	Устанавливает стили для элемента, когда ссылка становится активной, но переход по ссылке еще не совершен. Данный псевдокласс часто используется вместе с псевдоклассами :hover, :link и :visited
:link	Устанавливает стили для элемента, когда ссылка не является часто посещаемой. Данный псевдокласс часто используется вместе с псевдоклассами :hover, :active и :visited
:visited	Устанавливает стили для элемента, когда ссылка недавно посещалась. Данный псевдокласс часто используется вместе с псевдоклассами :hover, :active и :link. (Используйте данный псевдокласс перед ними, чтобы не подавлять их поведение)

ПРИМЕР: *Содержимое таблицы стилей*:

```
A.pr:visited { color:blue }
A.pr:active { font-weight:bold; color:brown }
A.pr:link { color:#00FF00 }
A.pr:hover { color:red; text-decoration: none}
P{color: bleak;font-style: italic;font-weight: bolder}
p:first-letter { float:right;font-size:2em;color:red;}
p:first-line {color:green;}
```



HTML-код основного документа:

<body>

<P>Как видите, первая буква и первая строка каждого параграфа имеет особый вид,

**
заданный при помощи различных стилей. **

** Данный прием вам хорошо известен по книгам, использующим подобный прием**

**в глубокой древности **

** Подведите курсор к ссылке,**

и она станет красного цвета и исчезнет подчеркивание ссылки

</body>

Примечание: описания нескольких свойств для одного селектора, контекстуального селектора, класса, индивидуально именованного стиля или группы объединенных селекторов отделяются друг от друга точкой с запятой ";".

Практическое занятие №10

Применение каскадных таблиц CSS для оформления текста в HTML-документе

Цель.

Теоретические сведения

Семейство шрифтов

Семейство шрифтов текста задается с помощью свойства **font-family**.

Свойство **font-family** должно содержать несколько имен шрифтов, как «резервные» системы. Если браузер не поддерживает первый шрифт, он пытается использовать следующий шрифт и т.д.

Начните со шрифта, который вы хотите установить, и закончите с общей семьей шрифтов, чтобы браузеры могли выбрать аналогичный шрифт в общей семье, если нет других доступных шрифтов.

Примечание: Если имя шрифта содержит более чем одно слово, оно должно быть в кавычках, как **font-family: 'Times New Roman'**. Более одного семейства шрифтов указываются в списке разделенном запятыми:

```
p {
  font-family: Georgia, 'Times New Roman', Times, serif;
}
```

Стиль шрифта

Свойство **font-style** в основном используется, например, для указания текста курсивом, имеет три значения:

```
font-style: normal|italic|obligue
```

– normal - текст отображается нормально
– italic - текст отображается курсивом
– oblique - текст «наклонный» (oblique очень похож на курсив, но менее поддерживается)

Размер шрифта

Свойство **font-size** устанавливает размер текста в формате:

```
font-size: абсолютный размер | относительный размер | значение | % | inherit
```

Возможность управлять размером текста играет важную роль в веб-дизайне. Однако, вы не должны использовать корректировку размера шрифта, чтобы параграфы выглядели как заголовки или заголовки выглядели как параграфы.

Важно!

Как правило, по умолчанию для большинства браузеров

1em = 12pt = 16px = 100% = medium

Единицы «px» и «pt», не лучшим образом подходят для веб-документов, т.к не масштабируются, что заставляет нас использовать «em» и "%", которые масштабируется, а значит лучше подходят для мобильных приложений (подробнее см. в лекциях).

Выравнивание текста

Свойсто **text-align** отвечает за выравнивание текста по горизонтали. Текст можно выравнивать по центру (center), по правому/левому (right/left) краю или по ширине. Когда текст выровнен по ширине (text-align:justify), каждая строка имеет одинаковую длину (как в журналах и газетах).

Различные маркеры списка элементов

Тип маркера пункта списка указывается со свойством **list-style-type**

Значения для неупорядоченных списков

Значение	Описание
none	Без маркера
disc	По умолчанию. Маркер в виде заполненного круга
circle	Маркер круг
square	Маркер квадрат

Значения для упорядоченных списков

Значение	Описание
armenian	Маркер - традиционная армянская нумерация
decimal	Маркер - число
decimal-leading-zero	Маркер число с нулями в начале (01, 02, 03, и так далее)
georgian	Маркер - традиционная грузинская нумерация (an, ban, gan, и так далее)
lower-alpha	Маркер - нижняя-альфа (a, b, c, d, e, и так далее)
lower-greek	Маркер - нижний греческий (alpha, beta, gamma, и так далее)
lower-latin	Маркер - нижний латинский (a, b, c, d, e, и так далее)
lower-roman	Маркер - нижний-римский (i, ii, iii, iv, v, и так далее)
upper-alpha	Маркер - верхняя-альфа (A, B, C, D, E, и так далее)
upper-latin	Маркер - верхний латинский (A, B, C, D, E, и так далее)
upper-roman	Маркер - верхней-римский (I, II, III, IV, V, и так далее)

Кроме того, можно определить свой собственный маркер списка, сделать это можно так:

```
ul { list-style-image: url(images/book.gif); }
```

Свойства цвета и фона

Древние реликты (HTML)

Для начала взглянем на фон с точки зрения HTML. Точка зрения, надо сказать, не самая лучшая, но будем соблюдать хронологию. Итак, для работы с фоном у нас имеется два атрибута:

1. **bgcolor** - задает фоновый цвет элемента. Присутствует у элементов BODY, TABLE, TR, TH и TD. В спецификации HTML 4.01 помечен как нежелательный для использования;
2. **background** - задает фоновое изображение для элемента. Согласно спецификации HTML 4.01 присутствует только у элемента BODY и помечен как нежелательный для использования, однако браузеры поддерживают данный атрибут для элементов TABLE и TD.

Естественно, набор крайне ограничен и морально устарел.

Альтернатива новой эры - CSS

Присвоение элементам цветовых и фоновых значений помогают создать зрелищную WEB-страницу. В CSS можно использовать множество свойств, которые придадут вашей странице требуемую привлекательность.

color - Определяет цвет текста

Значение:

- любое соответствующее стандарту значение цвета.
- **Inherit** - применяется значение родительского элемента.

```
p{color:red}
p{color:rgb(255,0,0)}
p{color:#ff0000}
```

background-color- Определяет цвет фона.

Значение:

- любое соответствующее стандарту значение цвета.
- **Transparent** - фон элемента делается прозрачным (по умолчанию).
- **Inherit** - применяется значение родительского элемента.

```
body{
   background-color: black;
  }
```

background-image - Определяет фоновое изображение элемента.

Значение:

- **none** - фоновое изображение не устанавливается.
- любое соответствующее стандарту значение URL фонового изображение.
- **Inherit** - применяется значение родительского элемента.

```
h1{
  background-image: url(pictures.gif);
 }
```

background-repeat - Определяет направление, по которому экран будет заполняться копиями фонового изображения.

Значение:

- **repeat** - фоновое изображение повторяется по горизонтали и по вертикали (по умолчанию).
- **repeat-x** - фоновое изображение повторяется только по горизонтали.
- **repeat-y** - фоновое изображение повторяется только по вертикали.
- **no-repeat** - фоновое изображение не повторяется.
- **inherit** - применяется значение родительского элемента.

```
body {
    background-image : url(fon.png) ;
     background-repeat: no-repeat;
     background-attachment: scroll ;
   }
```

background-attachment: - Устанавливает прокрутку

При наличии фонового рисунка, это свойство устанавливает, будет ли фоновое изображение прокручиваться с содержимым страницы, или будет заблокировано.

Значение:

- **scroll** - фон прокручивается вместе с содержимым;
- **fixed** - фон строго зафиксирован.

```
div{
  background-image: url(pictures.gif);
   background-attachment: fixed ;
 }
```

opasity: - Установка полупрозрачного фона

Свойство CSS 3 opacity задает значение прозрачности и варьируется от 0 до 1, где ноль это полная прозрачность элемента, а единица, наоборот, непрозрачность. У свойства opacity есть

особенность — прозрачность распространяется на все дочерние элементы, и они не могут превысить значение прозрачности своего родителя. Получается, что непрозрачный текст на полупрозрачном фоне быть не может.

Результат применения данного свойства Вы можете наблюдать на этой странице.

```
div{
   border-radius:20px;      /* Закругленные уголки */
   background-color:white;  /* Цвет фона */
    opacity: 0.9;           /* Полупрозрачный фон */
  }
```

background-position: - задает местоположение фонового изображения.

В качестве первого значения данного свойства должна задаваться величина смещения изображения по горизонтали, а в качестве второго величина смещения по вертикали. Величина смещения может быть указана как с помощью пикселей (px), процентов (%) и сантиметров (cm) (background-position:50px 30px;), так и с помощью предопределенных ключевых слов (background-position:right top; background-position:center center;).

```
body
{
background-image:url('spider2.gif');
background-repeat:no-repeat;
background-position:40px 60px;
}
```

Свойства фона могут быть заданы одной строкой с помощью свойства **background**, которое объединяет в себе все вышеперечисленные свойства:

```
background: transparent | background-color | background-image |
background-repeat | background-attachment | background-position;
Например:
background:#ffffff no-repeat url(fon.gif);
```

Тень текста с помощью CSS

Свойство «TEXT-SHADOW»

Чтобы добавить тень для текста, существует правило «text-shadow».

Синтаксис:

```
text-shadow: «X» «Y» «Амплитуда» «Цвет»;
```

Значение:

1. **X** – это горизонтальное смещение тени (положительное значение – смещение тени вправо, отрицательное – смещение тени влево).
2. **Y** – вертикальное смещение тени (положительное значение – смещение тени вниз, отрицательно значение – смещение тени вверх);
3. **Амплитуда** – чем выше ее значение, тем больше степень размытия;
4. **Цвет** – цвет тени

Пример:

```
h1
{
text-shadow: 4px 1px 3px #999; /*
тень заголовка*/
}
```

Результат:

Примеры использования:

Только тень

Tahoma с тенью

Tahoma с контуром

Tahoma вдавленный

Tahoma объёмный

Tahoma неоновый

Tahoma неоновый

Tahoma неоновый

Tahoma, много оттенков

Задание к практическому занятию

Задание 1. Контурный текст

Контурный текст характерен тем, что каждая буква обводится линией, цвет которой отличается от цвета текста (рис. 1). Лучше всего этот эффект смотрится с рубленым шрифтом большого размера, например, заголовков. Для основного текста применение контура лишь ухудшает читабельность.

Контурный текст

Рисунок 1 - Контурный текст

Контур можно создать двумя методами. В первом методе устанавливаем нулевое смещение тени и небольшой радиус её размытия, буквально, 1-2 пиксела (листинг 1). Повышение значения размытия превращает контур в свечение вокруг текста, а это уже другой эффект.

Листинг 1 - Контурный текст

«practiceWEB_10_01.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   .stroke {
    font: 2em Arial, sans-serif;
    text-shadow: red 0 0 2px;
   }
  </style>
 </head>
 <body>
   <p class="stroke">Контурный текст</p>
 </body>
</html>
```

Контур, сделанный этим методом, продемонстрирован на рис. 1. Контур получается слегка размытым, поэтому для тех, кто хочет получить чёткую линию, предназначен второй метод. Он заключается в использовании четырёх резких теней одного цвета, они смещаются в разные углы на один пиксел (листинг 2).

Листинг 2 - Четыре тени для контура

«practiceWEB_10_01_1.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   .stroke {
    font: 2em Arial, sans-serif;
    text-shadow: red 1px 1px 0, red -1px -1px 0,
                 red -1px 1px 0, red 1px -1px 0;
   }
  </style>
 </head>
 <body>
  <p class="stroke">Контурный текст</p>
 </body>
</html>
```

Вид такого контура показан на рис. 2. Заметно, что контур получается более выразительным.

Контурный текст

Рисунок 2 - Контур с помощью четырёх теней

Задание 2. Трёхмерный текст

Для добавления эффекта трёхмерного текста, показанного на рис. 3 применяется одновременно несколько теней которые смещаются относительно друг друга на один пиксел по горизонтали и вертикали.

Десятикамерный холодильник

Рисунок 3 - Трёхмерный текст

Лично мне подобный текст напоминает надписи в стиле ретро и опять же лучше всего он подходит для заголовков, а не для основного текста веб-страницы.

Число теней зависит от того, насколько вы хотите «выдвинуть» текст вперёд. Большее количество повышает «глубину» букв, меньшее, наоборот, понижает трёхмерность. В листинге 3 используется пять теней одного цвета.

Листинг 3 - Тень для добавления трёхмерности

«practiceWEB_10_02_.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   .stroke {
    font: bold 3em Arial, sans-serif;
    color: #0d3967;
    text-shadow: #cad5e2 1px 1px 0, #cad5e2 2px 2px 0,
                 #cad5e2 3px 3px 0, #cad5e2 4px 4px 0,
                 #cad5e2 5px 5px 0;
   }
  </style>
 </head>
 <body>
  <h1 class="stroke">Десятикамерный холодильник</h1>
 </body>
</html>
```

Для всех теней ставим нулевой радиус размытия и одинаковый цвет. Различаются тени только значениями смещения.

Задание 3. Тиснение текста

Для создания эффекта тиснения текста или, по-другому, рельефа, цвет текста должен совпадать с цветом фона. Одна часть «выступающих» над поверхностью букв словно освещена, другая же часть находится в тени (рис. 4).

Рисунок 4 - Рельефный текст

Для добавления подобного эффекта нам понадобится две тени — белую тень мы смещаем влево вверх на один пиксел, а тёмно-серую вправо вниз (листинг 4).

Листинг 4 - Рельефный текст

«practiceWEB_10_03_.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   body {
    background: #f0f0f0; /* Цвет фона веб-страницы */
   }
   .stroke {
    font: bold 3em Arial, sans-serif;
    color: #f0f0f0; /* Цвет текста, совпадает с цветом фона
    */
    text-shadow: #fff -1px -1px 0,
                 #333 1px 1px 0;
   }
  </style>
 </head>
 <body>
  <h1 class="stroke">Рельефный текст</h1>
 </body>
</html>
```

Рельеф выигрышнее всего смотрится именно на сером фоне, поэтому эффект подойдёт не для каждой цветовой схемы сайта. Кстати, легко получить вдавленный, а не выдавленный текст, достаточно поменять местами цвета тени.

```
text-shadow: #333 -1px -1px 0,
             #fff 1px 1px 0;
```

Задание 4. Свечение

Свечение вокруг текста — это та же самая тень, только она яркая и контрастная. Таким образом, для создания эффекта свечения достаточно изменить цвет тени и поставить желаемый радиус размытия. Поскольку свечение вокруг текста должно быть равномерным, то смещение тени надо задать нулевым. На рис. 5 показан пример свечения разных цветов.

Рисунок 5 - Свечение текста

Листинг 5 - Свечение

«practiceWEB_10_04_.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   .light {
    text-shadow: #5dc8e5 0 0 10px; /* Свечение голубого
    цвета */
    color: #0083bd;
   }
   .dark {
    text-shadow: red 0 0 10px; /* Свечение красного цвета */
   }
  </style>
 </head>
 <body>
  <h1 class="light">Светлая сторона</h1>
  <h1 class="dark">Тёмная сторона</h1>
 </body>
</html>
```

Задание 5. Размытие

Тень сама по себе размывается, так что если оставить только тень, а сам текст скрыть, то мы получим размытые буквы (рис. 6), причём степень размытия легко регулировать через параметр ***text-shadow***.

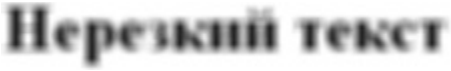


Рисунок 6 - Текст с размытием

Для сокрытия оригинального текста достаточно задать цвет как ***transparent*** (листинг 6). Цвет тени после этого выступает цветом текста, а радиус размытия устанавливает степень нерезкости букв.

Листинг 6 - Размытие текста

«practiceWEB_10_05_.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   .blur {
    text-shadow: #000 0 0 5px;
    color: transparent; /* Прозрачный цвет текста */
   }
  </style>
 </head>
 <body>
  <h1 class="blur">Нерезкий текст</h1>
 </body>
</html>
```

Задание 6. Тень и псевдоклассы

Тень не обязательно добавлять непосредственно к тексту, свойство ***text-shadow*** прекрасно сочетается с псевдоклассами ***:hover*** и ***:first-letter***. За счёт этого получаются интересные эффекты с текстом вроде контурной первой буквы абзаца или свечения ссылки при наведении на неё курсора мыши. В листинге 7 показаны такие приёмы.

Листинг 7 - Использование псевдоклассов

«practiceWEB_10_06_.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текст</title>
  <style>
   a:hover { /* Вид ссылки при наведении на неё курсора */
    text-shadow: #5dc8e5 0 0 5px;
    color: #000;
   }
   p:first-letter { /* Первая буква абзаца */
    font-size: 2em;
    text-shadow: red 1px 1px 0, red -1px -1px 0,
                 red -1px 1px 0, red 1px -1px 0;
   }
  </style>
 </head>
  <body>
   <p>Нишевый проект тормозит <a href="1.html">традиционный канал</a>, не
   считаясь с
      затратами. Структура рынка, отбрасывая подробности, стабилизирует
      департамент маркетинга и продаж, используя опыт предыдущих кампаний.
      Построение бренда, безусловно, спонтанно отталкивает конвергентный
      PR, отвоевывая рыночный сегмент. Инвестиция синхронизирует ролевой
      социальный статус, повышая конкуренцию. Торговая марка естественно
      обуславливает план размещения, используя опыт предыдущих кампаний.</p>
 </body>
</html>
```

Задание 7. Создайте тени текста при наведение курсора

Текст до наведения курсора

Текст после наведения курсора

Задание 8. Светящийся текст

Для этого эффекта необходимо использовать стекированные ***text-shadow***, чтобы создать эффект свечения. Можно использовать несколько значений для ***text-shadow***, чтобы накладывать их друг на друга и создавать другие потрясающие эффекты. В задании увеличивается радиус размытия тени и придали ей ярко-синий цвет. Что объясняет неоново-синее свечение.

«practiceWEB_10_08.html

```html
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
    <style>
   body{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    min-height: 100vh;
    scroll-behavior: smooth;
}
section {
    height: 100vh;
    font-size: 80px;
    font-weight: bold;
    transition: all 200ms;
    background-repeat: no-repeat;
    background-size: cover;
    background-attachment: fixed;
    background-position: 50% 50%;
    display: grid;
    place-items: center;
    z-index: 1;
    cursor: pointer;
    position: relative;
    padding: 20px;
}
.glow-demo {
    background: #111;
}
/*Waves*/
.glow span {
    color: #fff;
    transition: all 300ms;
}
.glow span:hover {
    text-shadow: 0 0 10px #0698a5,
                 0 0 30px #0698a5,
                 0 0 80px #0698a5,
                 0 0 120px #0698a5,
                 0 0 200px #0698a5;
}
  </style>
 </head>
 <body>
 <section class="glow-demo">
        <div class="glow">
            <span>G</span><span>L</span><span>O</span><span>W</span>
        </div>
 </section>
 </body>
</html>
```

Задание 9. Создайте эффект отображения текста

Текст до наведения курсора

Текст после наведения курсора

Контрольные вопросы

1. Дайте определение понятию стиль.
2. Укажите свойства шрифта, изображения, документа, поддающиеся изменению с помощью стилевых спецификаций.
3. Назовите основные статические фильтры.
4. Опишите пример использования статического фильтра к изображению.
5. Охарактеризуйте способы включения и использования стилей в документе.
6. Приведите пример позиционирования текста на странице с помощью стилей. Чем такой способ отличается от использования таблиц и фреймов?
7. Для чего используется z-index?

Практическая работа №11

CSS Ссылки. CSS Списки

Методические разработки практических работ по информатике по темам "Язык разметки HTML" и "Современные web-технологии" (для учащихся средних классов).

При проведении занятий с использованием данных методических разработок для создания html- и css-файлов рекомендуется использовать обычный текстовый редактор, поддерживающий подстветку синтаксиса языков HTML и CSS, а также желательно включающий возможность работы с несколькими файлами в многостраничном режиме.

Например, для Windows подойдет Notepad++ (notepad-plus-plus.org). В текстовом редакторе Notepad++ для того, чтобы выполнялся перенос длинных строк, надо в меню выбрать команду `Вид -> Перенос строк`. Также очень хорошим текстовым редактором является Notepad2 (flos-freeware.ch/notepad2.html). Однако он не поддерживает многостраничный режим работы. Установка переноса строк в нем выполняется с помощью команды `View -> Word Wrap`.

1. HTML–документ. Абзацы, разрывы строк, выравнивание

iii. 1 Структура HTML-документа

Исходный код HTML-документа состоит из тегов и содержания.

Содержание предназначено для отображения в окне браузера. Теги определяют его структуру (разметку): какие части являются заголовками, какие абзацами, а какие иными элементами.

У тегов могут быть различные атрибуты с заданными свойствами. В старых версиях языка HTML атрибуты использовались для оформления содержания: установки размера и цвета шрифта, выравнивания текста, установки отступов и другого. Сейчас содержимое HTML-документа принято оформлять с помощью языка CSS.

Любой HTML документ всегда включает контейнеры `html, head` и `body`, которые вложены друг в друга следующим образом:

```
<html>
<head>
…
</head>
<body>
…
</body>
</html>
```

Задание 1. Создайте файл и задайте ему структуру, которая приведена выше. Сохраните его.

В контейнере `head` обычно присутствует контейнер `title`, содержимое которого отображается в заголовке окна документа.

Задание 2. Добавьте в документ контейнер title:

```
<title>ЭВМ – электронно-вычислительная машина</title>
```

Сохраните файл и откройте в браузере. Найдите введенное вами содержание title.

Содержимое body отображается в окне браузера.

С помощью тега-контейнера `p` размечают абзацы. Одиночный тег `br` позволяет перейти на новую строку без создания нового абзаца, т.е. создает разрыв строки.

Задание 3. Добавьте в контейнер `body` следующее содержимое:

```
<p>Появление персональных компьютеров в начале семидесятых годов (параллельно с постепенной эволюцией крупных ЭВМ) сейчас расценивают как революционный переворот. Масштабы его влияния на человеческое общество сравнивают с последствиями от изобретения книгопечатания.</p>
<p>В мире уже сейчас имеются миллионы и миллиарды ЭВМ. <br/>Их число продолжает неуклонно расти!</p>
```

Сохраните. Обновите документ в браузере. Отметьте, сколько абзацев вы видите, где находится разрыв строки.

iv. 2 Выравнивание абзацев. Старый стиль

Выравнивание абзацев определяется значениями `left` (по левому краю), `right` (по правому), `center` (по центру) и `justify` (по ширине). Эти значения могут быть присвоены свойству `align` (выравнивание), которое допустимо для многих тегов.

Так, например, выравнивание абзаца по центру можно задать так:

```
<p align="center"> …
```

Задание 4. Для созданных ранее абзацев задайте выравнивание по ширине (для первого абзаца) и по правому краю (для второго).

v. 3 Использование языка CSS

Абзацы правильнее выравнивать с помощью CSS (языка описания внешнего вида HTML-документа). Для этого в отдельном файле или контейнере `head` определяют стили оформления для различных элементов. Например, раздел `head` может содержать такой контейнер `style`:

```
<style type="text/css">
        .rt {text-align:right;}
        .jtf {text-align:justify;}
</style>
```

Далее следует указать желаемые стили для абзацев. Точка говорит о том, что мы имеем дело с классом. Следовательно, в теге p должно быть записано, например, так:
`<p class="rt">…`

Задание 5. Удалите ранее добавленный атрибут align у абзацев. Выровняйте абзацы, используя язык CSS.

vi. Результат практической работы

```
<html>

<head>

<title>ЭВМ – электронно-вычислительная машина</title>

<style type="text/css">
        .rt {text-align:right;}
        .jtf {text-align:justify;}
</style>

</head>

<body>
<p class="jtf">Появление персональных компьютеров в начале семидесятых годов (параллельно с постепенной эволюцией крупных ЭВМ) сейчас расценивают как революционный переворот. Масштабы его влияния на человеческое общество сравнивают с последствиями от изобретения книгопечатания.</p>
<p class="rt">В мире уже сейчас имеются миллионы и миллиарды ЭВМ. <br/>Их число продолжает неуклонно расти!</p>
</body>

<html>
```

2. Заголовки. Теги strong, em, span

Задание 1.

Создайте html-страницу с текстом представленным ниже. При этом для создания структуры документа используйте следующие теги разметки:
`p` – абзац;
`h1` – заголовок 1-го уровня;
`h2` – заголовок 2-го уровня.

Системы счисления

Системы счисления — это способ записи чисел. Обычно, числа записываются с помощью специальных знаков (цифр), но бывают исключения. Например, в арабской системе счисления используются цифры (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), а в римской — некоторые латинские прописные буквы (I, V, X, L, C, D, M). Арабская и римская системы счисления имеют еще одно существенное отличие. Арабская система счисления является позиционной, а римская — непозиционной.

Сравнение позиционной и непозиционной систем счисления

В позиционных системах счисления количество, обозначаемое цифрой в числе, зависит от ее позиции, а в непозиционных системах счисления "вес" цифры не зависит от ее позиции. Например:

11 – здесь первая единица обозначает десять, а вторая – 1.
II – здесь обе единицы обозначают единицу.
345, 259, 521 – здесь цифра 5 в первом случае обозначает 5, во втором – 50, а в третьем – 500.
XXV, XVI, VII – здесь, где бы ни стояла цифра V, она везде обозначает пять единиц. Другими словами, величина, обозначаемая знаком V, не зависит от его позиции.

Задание 2.

Обрамите текст, выделенный жирным шрифтом на образце выше, контейнером `strong`, а выделенный курсивом — `em`. Посмотрите результат в браузере.

Задание 3.

Создайте в контейнере `head` следующую таблицу стилей:

```
<style type="text/css">
        span {font-family: Arial;}
        span.blue {color: #00008b;}
        span.red {color: #b22222;}
</style>
```

Заключите в соответствующие контейнеры `span` текст, выделенный красным и синим цветом на образце (при этом не забудьте указывать классы — см. предыдущее занятие). Посмотрите результат в браузере.

Задание 4.

Добавьте в код стиль для заголовков:

```
h1, h2 {
        background-color: #003399;
        color: white;
}
```

Посмотрите результат.

vii. Результат практической работы

```
<html>
<head>
<title>Системы счисления</title>
<style type="text/css">
        span {font-family: Arial;}
        span.blue {color: #00008b;}
        span.red {color: #b22222;}
        h1, h2 {
                background-color: #003399;
                color: white;
        }
</style>
</head>
```

```
<body>
<h1>Системы счисления</h1>
<p><em>Системы счисления</em> — это способ записи чисел. Обычно, числа записываются с помощью специальных знаков (цифр), но бывают исключения. Например, в арабской системе счисления используются цифры (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), а в римской — некоторые латинские прописные буквы (I, V, X, L, C, D, M). Арабская и римская системы счисления имеют еще одно существенное отличие. Арабская система счисления является позиционной, а римская — непозиционной.</p>
<h2>Сравнение позиционной и непозиционной систем счисления</h2>
<p>В <em>позиционных системах счисления</em> количество, обозначаемое цифрой в числе, зависит от ее позиции, а в <em>непозиционных системах счисления</em> "вес" цифры не зависит от ее позиции. Например:</p>
<p><span class="red">11</span> - здесь первая единица обозначает десять, а вторая - 1.</p>
<p><span class="blue">II</span> - здесь обе единицы обозначают единицу.</p>
<p><span class="red">345, 259, 521</span> - здесь цифра 5 в первом случае обозначает 5, во втором - 50, а в третьем - 500.</p>
<p><span class="blue">XXV, XVI, VII</span> - здесь, где бы ни стояла цифра V, она везде обозначает пять единиц. Другими словами, величина, обозначаемая знаком V, не зависит от его позиции.</p>
</body>
<html>
```

3. Подключение внешней таблицы стилей. Отступы и обтекание

После каждого задания следует смотреть результат в браузере и объяснять причину изменений.

1. Создайте html-файл с приведенным ниже текстом, разметьте в нем заголовок первого уровня и абзацы.

Операционные системы

Операционная система — это особенная программа, которая позволяет компьютеру работать так, как мы это представляем. Без нее для обычного пользователя компьютер представлял бы всего лишь груду железа. Все другие программы, которые мы обычно используем в повседневной жизни (текстовые и графические редакторы, игры, интернет-браузеры и др.), могут работать, только если на компьютере установлена операционная система.

Первые операционные системы появились давно и не были похожи на современные. Они представляли собой наборы небольших программ для программистов. Позже количество функций, возлагаемых на операционную систему, увеличивалось, и на данный момент они представляют собой достаточно сложные программные комплексы.

В мире существует огромное количество операционных систем, но лишь некоторые из них нашли широкое распространение.

На сегодняшний день наиболее популярными являются операционные системы семейства Windows, которые являются проприетарным (коммерческим) продуктом корпорации Microsoft. Преимуществом Windows считается дружественный для пользователя интерфейс. Из недостатков отмечают ненадежность системы.

Linux представляет собой множество Unix-подобных операционных систем (дистрибутивов), которые чаще всего являются свободно распространяемыми. Одной из уникальных особенностей систем GNU/Linux является отсутствие единого географического центра разработки. Linux и программы для нее пишутся миллионами программистов, рассредоточенных по всему миру.

2. Создайте файл prac3.css, подключите его к html-документу. Для этого в контейнере head пропишите строку:

```
<link rel="stylesheet" href="prac3.css" />
```

3. Задайте для абзацев и заголовка отступы справа и слева по 15% от размера рабочей области окна браузера:

```
p, h1 {
        margin-left: 15%;
        margin-right: 15%;
}
```

4. В файле html для предпоследнего абзаца пропишите его принадлежность к классу left (<p class="left">), а для последнего к классу right. В таблице стилей для обоих классов задайте рамки вокруг абзацев и установите ширину для них не более 400 пикселей. Например:

```
p.left, p.right {
        border: 6px;
        border-style: groove;
        border-color: silver;
        padding: 15px;
        font-size: 0.9em;
        width:350px;
}
```

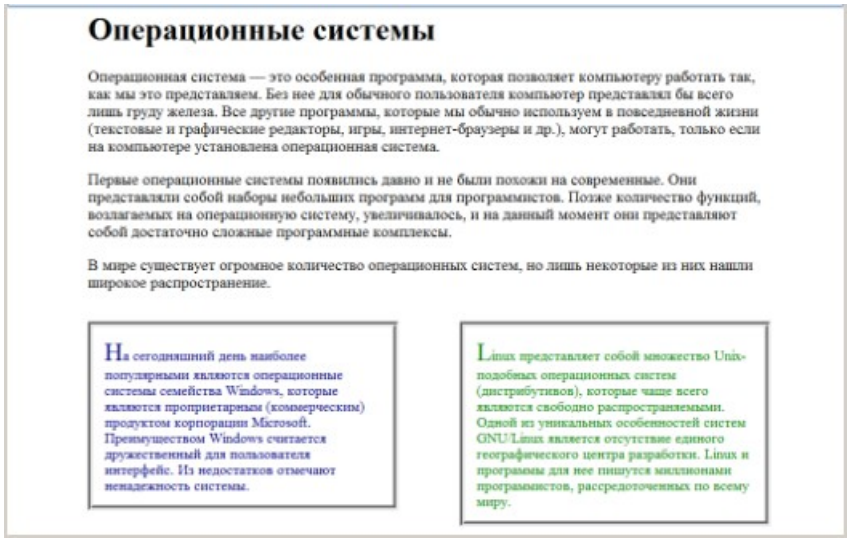
5. Отдельно для каждого класса укажите отличающие их свойства:

```
p.left {
        float:left;
        margin-right: 0%;
        color: navy;
}
p.right {
        float:right;
        margin-left: 0%;
        color: green;
}
```

6. Отдельно оформите первую букву обоих абзацев (буквицу):

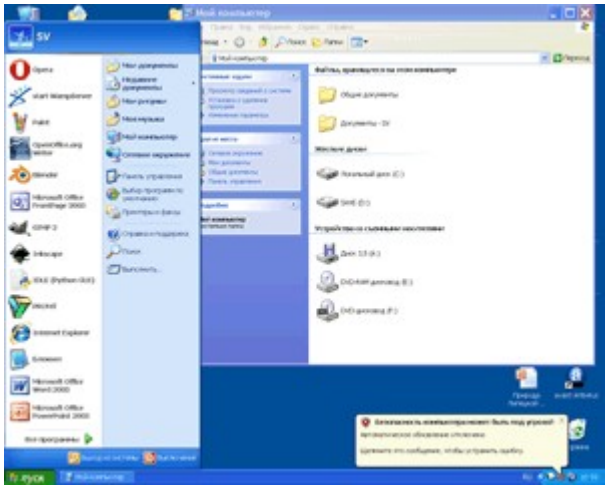
```
p.left:first-letter, p.right:first-letter{
        font-size: 1.8em;
}
```

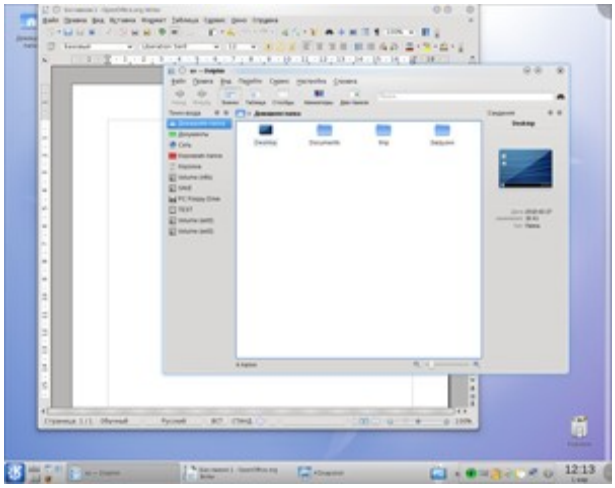
Результат практической работы должен выглядеть примерно так:



4. Тег Div. Вставка в html-документ изображений, их свойства и оформление

После каждого задания следует смотреть результат в браузере и объяснять причину изменений.
Изображения для практической работы:





1. Откройте на редактирование html-файл, созданный на прошлом занятии. Добавьте в него теги <img>. Первое изображение вставьте перед первым абзацем (вне абзаца), второе и третье изображения в конце соответственно предпоследнего и последнего абзацев (перед закрывающим тегом </p>).

```
...
<img src="structure_os.png" width=400 height=299 alt="Операционная система в структуре ЭВМ">
...
<img src="win_s.gif" title="Операционная система Windows XP">
```

```
...
<img src="linux_s.png" title="Операционная система AltLinux">
...
```

2. Опишите общие стилевые свойства тега img в файле *.css (который был создан на прошлом уроке).

```
img {
        display: block;
        margin: auto;
        padding-top: 10px;
}
```

3. Создайте класс image:

```
.image {
        float: right;
        margin: 10 0 10 10px; /* top right bottom left*/
        padding: 10 10 10 10px;
        border: 1px dotted gray;
        text-align: center;
        width: 420px;
        font-size: smaller;
        font-family: "Courier";
}
```

В html-документе обрамите в контейнер `<div class="image">...</div>` первое изображение и абзац, который является подписью к этом изображению.

```
<div class="image">
<img src="structure_os.png" width=400 height=299 alt="Операционная система в структуре
ЭВМ">
<p>Схема, иллюстрирующая место операционной системы в многоуровневой структуре компьюте-
ра</p>
</div>
```

4. Определите для тела документа серый цвет фона.

```
body {
        background-color: silver;
}
```

5. Удалите описание стиля для обычного абзаца и заголовка. Вместо него вставьте описание для идентификатора all.

```
#all {
        width: 900px;
        margin: auto;
        padding: 0 30 0 30px; /* top right bottom left*/
        border-left: 4px groove white;
        border-right: 4px groove white;
        font-size: 1.2em;
        background-color: white;
}
```

6. Обрамите в контейнер `<div id="all">...</div>` все содержимое тела html-документа.

5. Создание и оформление гипертекстовых ссылок

1. Создайте шесть файлов HTML (structure.html, tehnologies.html, history.html, reference.html, hyperlink.html, philosophers.html), поместите их вместе в один каталог. Содержимое файлов и его оформление смотрите в образце.

2. В том же каталоге создайте файл index.html, со списком ссылок на ранее созданные файлы. Озаглавьте список заголовком третьего уровня. В браузере проверьте работоспособность ссылок.
3. Подключите к документу index.html файл style.css, описав в нем стили для списка (присвойте списку определенных класс) и ссылок, таким образом, чтобы меню (группа ссылок) выглядело как ряд вертикальных кнопок.

viii. Примерный результат практической работы

Файл index.html:

```
<html>
<head><title>Web и гиперссылки</title>
<link rel="stylesheet" href="style.css" />
</head>
<body>
<h3>Web и гиперссылки</h3>
<ul class="menu">
<li><a href="structure.html">Структура Web</a></li>
<li><a href="tehnologies.html">Технологии Web</a></li>
<li><a href="history.html">История Web</a></li>
<li><a href="reference.html">Что такое ссылка</a></li>
<li><a href="hyperlink.html">Гиперссылки</a></li>
<li><a href="philosophers.html">Философы</a></li>
</ul>
</body>
</html>
```

Файл style.css:

```
ul.menu{
  list-style:none;
  width: 180px;
}
ul.menu li {
  padding:10px;
  border:3px outset #E6E6FA;
  margin-bottom: 5px;
}
a {
  color: #23594C;
  text-decoration: none;
}
ul.menu li:hover {
  background-color:#bfbfbf;
}
```

6. Создание шаблона сайта

1. Откройте файл index.html, созданный на прошлом занятии. В теле документа создайте контейнеры div в указанной ниже последовательности. Список ссылок поместите в `<div id="left"> … </div>`, а заголовок третьего уровня исправьте на заголовок первого уровня и поместите в контейнер `<div id="header"> … </div>`.

```
<div id="all">
  <div id="header">

  </div>
  <div id="left">

  </div>
  <div id="content">
```

```
  </div>
</div>
```

2. Скопируйте содержимое тела файла structure.html в контейнер `<div id="content"> … </div>`. После этого удалите файл structure.html и измените адрес в ссылке, которая указывала на данную страницу:
`<a href="index.html">Структура Web</a>`

3. Посмотрите результат, открыв в браузере файл index.html.

4. Откройте style.css на редактирование и пропишите в нем следующие таблицы стилей для тегов div. Посмотрите на результат в браузере. Объясните произошедшие изменения.

```
#all{
        width: 1000px;
        margin: auto;
        background-color: #fff;
}
#header{
        height: 100px;
        border: #E6E6FA dashed 1px;
}
#left{
        width: 250px;
        float: left;
        margin-top: 20px;
}
#content {
        width: 708px;
        border: #E6E6FA dashed 1px;
        margin-top: 20px;
        padding: 0 20 0 20px;
}
```

5. В качестве фона тела документа установите изображение:

```
body {
        background-image: url(<a href="http://www.png" title="www.png">www.png</a>);
}
```

6. Нарисуйте изображение на тему «Всемирная паутина» высотой не более 80 пикселей, а шириной не более 200 пикселей; сохраните его в формате PNG. Пропишите тег img с адресом этого изображения перед заголовком в контейнере `<div id="header"> … </div>`.
`<img id="logo" src="logo.png">`

7. Опишите специальный стиль для логотипа:

```
img#logo {
        float: left;
        margin: 10px;
}
```

8. Вставьте в другие html-страницы теги div и ссылку на таблицу стилей аналогично содержимому файла index.html.

7. Оформление таблиц с помощью CSS

Задание 1. Создайте файлы table1.html и связанный с ним style1.css. В первом на языке HTML опишите таблицу представленную ниже, а во втором - ее внешний вид.

HDD	WD Caviar 3.1 Gb	50$
	Quantum FB ST 6.4Gb	90$
Video	Matrox G400	115$
	Voodoo III	120.50$

Задание 2. Создайте файлы table2.html и связанный с ним style2.css. В первом на языке HTML опишите таблицу представленную ниже, а во втором - ее внешний вид.

С каждым днем в Интернете появляется все больше бяк и бук. Это особенные существа, роль которых в развитии современного общества не понятна, но тем не менее само их присутствие заметно. Буки и бяки требуют особого обращения к себе, если обращаться к ним как к нормальным человеческим особям, то вы поняты не будете.	Вот таблица, которая показывает сколько бук, бяк и других обитает в Интернете: буки 65% населения бяки 20% населения другие 15% населения Данные статистического штаба

ix. Ответ к заданиям

Задание 1.

Файл table1.html

```
<html>
<head>
<title>Таблицы</title>
<link rel="stylesheet" href="style1.css" />
</head>
<body>
<table>
<tr>
<th rowspan=2>HDD</th>
<td>WD Caviar 3.1 Gb</td>
<td> 50$</td>
</tr>
<tr>
<td>Quantum FB ST 6.4Gb</td>
<td> 90$</td>
</tr>
<tr>
<th rowspan=2>Video</th>
```

```
<td>Matrox G400</td>
<td>115$</td>
</tr>
<tr>
<td>Voodoo III</td>
<td> 120.50$</td>
</tr>
</table>

</body>
</html>
```

Файл style1.css

```
table {
        margin: auto;
        width: 400px;
        border: #2f4f4f inset 5px;
        padding: 5px;
}
td,th {
        border: groove 1px;
        padding: 5 10 5 10px;
}
```

Задание 2.

Файл table2.html

```
<html>
<head>
<title>Буки и бяки</title>
<link rel="stylesheet" href="style2.css" />
</head>
<body>
<table cellpadding="10">
<tr>
<td class="w200">
С каждым днем в Интернете появляется все больше бяк и бук. Это особенные существа, роль
которых в развитии современного общества не понятна, но тем не менее само их присутствие
заметно.  Буки и бяки требуют особого обращения к себе, если обращаться к ним как к
нормальным человеческим особям, то вы поняты не будете.</td>
<td class="w5"></td>
<td class="w200">Вот таблица, которая показывает сколько бук, бяк и других обитает в Ин-
тернете:<br><br>
        <table class="white" cellspacing="3">
        <tr>
        <td  class="w50">буки</td>
        <td>65% населения</td>
        </tr>
        <tr>
        <td>бяки</td>
        <td>20% населения</td>
        </tr>
        <tr>
        <td>другие</td>
        <td>15% населения</td>
        </tr>
        </table>
<br><br>
Данные статистического штаба
</td>
</tr>
</table>
```

```
</body>
</html>
```

Файл style2.css

```
body {
        background-color: #000066;
}
table {
        margin: auto;
        border: ridge 5px white;
}
td.w200 {
        width: 200px;
        vertical-align: top;
        background-color: #99ccff;
}
td.w50 {
        width: 50px;
}
td.w5 {
        width: 5px;
        background-color: #ffffff;
}
table.white {
        border-width: 1px;
}
table.white tr {
        background-color: #ffffff;
}
```

8. Упражнение на использование верхнего и нижнего регистров

Создайте html-файл со следующим содержимым. Запомните теги, отвечающие за перевод символов в верхний и нижний регистры.

```
<html>
<head>
<title>Верхний и нижний индекс</title>
</head>
<body>
<p>Тег  <strong>sup</strong> отображает текст со сдвигом вверх (верхний индекс) и умень-
шанием размера текущего шрифта на единицу. Например:</p>
<ul>
<li>Microsoft <sup>TM</sup></li>
<li>x<sup>4</sup> = x<sup>2</sup> * x<sup>2</sup></li>
</ul>

<p>Тег  <strong>sub</strong> отображает текст со сдвигом вниз (нижний индекс) и уменьша-
нием размера текущего шрифта на единицу. Например:</p>
<ul>
<li>C<sub>i</sub> = A<sub>i</sub> + B<sub>i</sub></li>
<li>2H<sub>2</sub> + O<sub>2</sub> = 2H<sub>2</sub>O</li>
</ul>
</body>
</html>
```

9. Списки

Создайте html-страницу, содержащую представленные ниже списки. Тип маркеров для классов задайте в связанном с ней файле lists.css.

Неупорядоченные списки

- Газета
- Журнал
- Книга
- Комикс

- Комедия
- Трагедия
- Драма

- Водород
- Кислород
- Углерод
- Азот

Многоуровневый список

- Глава 1
  - Пункт 1.1
  - Пункт 1.2
- Глава 2
  - Пункт 2.1
  - Пункт 2.2
- Глава 3
  - Пункт 3.1
  - Пункт 3.2

Упорядоченные списки

1. Утро
2. День
3. Вечер
4. Ночь

a. Нарисовать
b. Сгруппировать
c. Задать действие

A. Взять чашку
B. Взять ложку
C. Положить кофе
D. Положить сахар
E. Залить кипятком
F. Размешать

i. Младшие классы (1-4)
ii. Средние классы (1-8)
iii. Старшие классы (9-11)

I. Включить
II. Пропылесосить
III. Выключить

Свойство, определяющее тип маркера, - `list-style-type`. Используемые в примере значения:

- circle — пустая окружность;
- square — квадрат;
- lower-latin — строчные латинские (английские) буквы;
- upper-latin — прописные латинские (английские) буквы;
- lower-roman — римские цифры из строчных букв;
- upper-roman — обычные римские цифры.

Заполненная окружность и арабские цифры используются для маркированных и нумерованных списков по умолчанию.

При создании многоуровневых списков вложенный список помещается внутрь соответствующего тега `li` внешнего списка.

Пример результата практической работы

Файл lists.html

```
<html>
<head>
<title>Списки</title>
<link rel="stylesheet" href="lists.css" />
</head>
<body>
<h2>Списки</h2>
<h3>Неупорядоченные списки</h3>
```

```
<ul>
<li>Газета</li>
<li>Журнал</li>
<li>Книга</li>
<li>Комикс</li>
</ul>

<ul class="circle">
<li>Комедия</li>
<li>Трагедия</li>
<li>Драма</li>
</ul>

<ul class="square">
<li>Водород</li>
<li>Кислород</li>
<li>Углерод</li>
<li>Азот</li>
</ul>

<h3>Упорядоченные списки</h3>
<ol>
<li>Утро</li>
<li>День</li>
<li>Вечер</li>
<li>Ночь</li>
</ol>

<ol class="l_l">
<li>Нарисовать</li>
<li>Сгруппировать</li>
<li>Задать действие</li>
</ol>

<ol class="u_l">
<li>Взять чашку</li>
<li>Взять ложку</li>
<li>Положить кофе</li>
<li>Положить сахар</li>
<li>Залить кипятком</li>
<li>Размешать</li>
</ol>

<ol class="l_r">
<li>Младшие классы (1-4)</li>
<li>Средние классы (1-8)</li>
<li>Старшие классы (9-11)</li>
</ol>

<ol class="u_r">
<li>Включить</li>
<li>Пропылесосить</li>
<li>Выключить</li>
</ol>

<h3>Многоуровневый список</h3>
<ul class="circle">
<li> Глава 1
        <ul>
        <li>Пункт 1.1</li>
        <li>Пункт 1.2</li>
        </ul>
</li>
<li> Глава 2
        <ul>
        <li>Пункт 2.1</li>
```

```
        <li>Пункт 2.2</li>
        </ul>
</li>
<li> Глава 3
        <ul>
        <li>Пункт 3.1</li>
        <li>Пункт 3.2</li>
        </ul>
</li>
</ul>
</body>
</html>
```

Файл lists.css

```
ul.circle {
        list-style-type: circle;
}

ul.square {
        list-style-type: square;
}

ol.l_l {
        list-style-type: lower-latin;
}

ol.u_l {
        list-style-type: upper-latin;
}

ol.l_r {
        list-style-type: lower-roman;
}

ol.u_r {
        list-style-type: upper-roman;
}

li ul {
        list-style-type: square;
}
```

Практическая работа №12
" Web- страница с горизонтальными и вертикальными ориентированными блокоми навигации. Интерактивное горизонтальное меню навигации средствами CSS."

CSS меню

- Вертикальное меню
- Горизонтальное меню
- Выпадающее меню

Меню - раздел веб-сайта, предназначенный помогать посетителю перемещаться по сайту. Любое меню представляет собой список ссылок, ведущих на внутренние страницы сайта. Самым простым способом добавить панель навигации на сайт является создание меню с помощью CSS и HTML.

Вертикальное меню

Первым шагом создания вертикального меню будет создание маркированного списка. Также нам нужно будет иметь возможность идентифицировать список, поэтому мы добавим к нему атрибут id с идентификатором "navbar". Каждый элемент <li> нашего списка будет содержать по одной ссылке:

```
<ul id="navbar">

  <li><a rel="nofollow ugc" target="_blank" href="#">Главная</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">Новости</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">Контакты</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">О нас</a></li>

</ul>
```

Наша следующая задача заключается в сбросе стилей списка, установленных по умолчанию. Нам нужно убрать внешние и внутренние отступы у самого списка и маркеры у пунктов списка. Затем зададим нужную ширину:

```
#navbar {

  margin: 0;

  padding: 0;

  list-style-type: none;

  width: 100px;

}
```

Теперь пришло время стилизовать сами ссылки. Мы добавим к ним фоновый цвет, изменим параметры текста: цвет, размер и насыщенность шрифта, уберем подчеркивание, добавим небольшие отступы и переопределим отображение элемента <a> со строчного на блочный. Дополнительно были добавлены левая и нижняя рамки к пунктам списка.

Самой важной частью наших изменений является переопределение строчных элементов на блочные. Теперь наши ссылки занимают все доступное пространство пунктов списка, то есть для перехода по ссылке нам больше не нужно наводить курсор точно на текст.

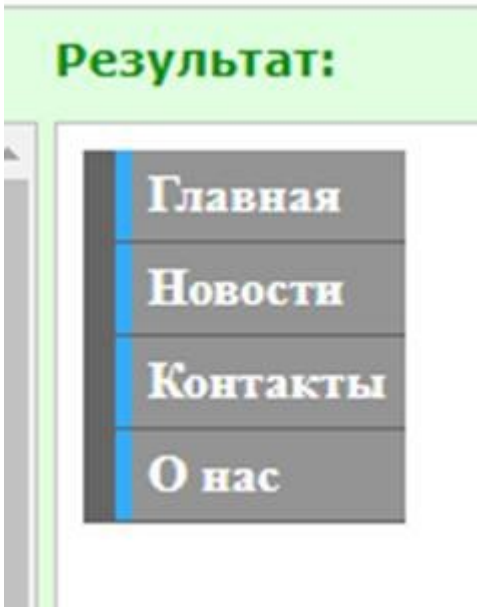
```
#navbar a {

  background-color: #949494;

  color: #fff;

  padding: 5px;

  text-decoration: none;

  font-weight: bold;
```

```
  border-left: 5px solid #33ADFF;
  display: block;
}
#navbar li {
  border-left: 10px solid #666;
  border-bottom: 1px solid #666;
```

Итоговый код: (Стилевой файл прописать отдельно !!)

```
!DOCTYPE html>
<html>
  <head>
    <title>Название документа</title>
    <style>
      #navbar {
        margin: 0;
        padding: 0;
        list-style-type: none;
        width: 100px;
      }
      #navbar li {
        border-left: 10px solid #666;
        border-bottom: 1px solid #666;
      }
      #navbar a {
        background-color: #949494;
        color: #fff;
        padding: 5px;
        text-decoration: none;
```

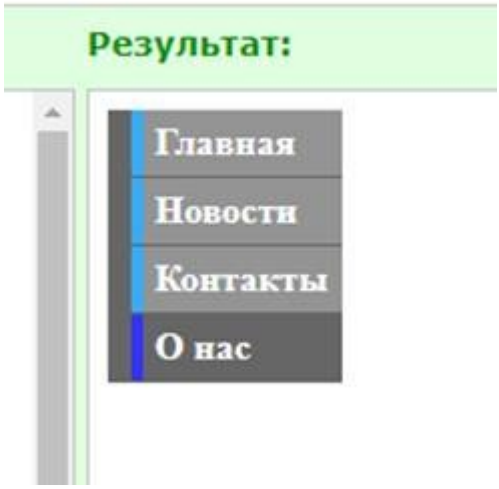
```
      font-weight: bold;
      border-left: 5px solid #33ADFF;
      display: block;
    }
  </style>
 </head>
 <body>
  <ul id="navbar">
    <li><a rel="nofollow ugc" target="_blank" href="#">Главная</a></li>
    <li><a rel="nofollow ugc" target="_blank" href="#">Новости</a></li>
    <li><a rel="nofollow ugc" target="_blank" href="#">Контакты</a></li>
    <li><a rel="nofollow ugc" target="_blank" href="#">О нас</a></li>
  </ul>
 </body>
</html>
```



При наведении курсора мыши на пункт меню его внешний вид может изменяться, привлекая к себе внимание пользователя. Создать такой эффект можно с помощью псевдо-класса :hover.

Вернемся к рассмотренному ранее примеру вертикального меню и добавим в таблицу стилей следующее правило:

```
#navbar a:hover {
  background-color: #666;
  border-left: 5px solid #3333FF;
```



h. Горизонтальное меню

В предыдущем примере мы рассмотрели вертикальную панель навигации, которую чаще всего можно встретить на сайтах слева или справа от области с основным контентом. Однако меню с навигационными ссылками также часто располагается и по горизонтали в верхней части веб-страницы.

Горизонтальное меню можно создать путем стилизации обычного списка. Свойству display для элементов <li> нужно присвоить значение inline, чтобы пункты списка располагались друг за другом.

Для размещения пунктов меню по горизонтали, сначала создадим маркированный список с ссылками:

```
<ul id="navbar">
  <li><a rel="nofollow ugc" target="_blank" href="#">Главная</a></li>
  <li><a rel="nofollow ugc" target="_blank" href="#">Новости</a></li>
  <li><a rel="nofollow ugc" target="_blank" href="#">Контакты</a></li>
  <li><a rel="nofollow ugc" target="_blank" href="#">О нас</a></li>
</ul>
```

Напишем для нашего списка пару правил, сбрасывающих стиль используемый для списков по умолчанию, и переопределим пункты списка с блочных на строчные:

```
#navbar {
  margin: 0;
  padding: 0;
  list-style-type: none;
```

```
}
#navbar li { display: inline; }
```

Теперь нам осталось лишь определить стилевое оформление для нашего горизонтального меню:

```
#navbar {
  margin: 0;
  padding: 0;
  list-style-type: none;
  border: 2px solid #0066FF;
  border-radius: 20px 5px;
  width: 550px;
  text-align: center;
  background-color: #33ADFF;
}
#navbar a {
  color: #fff;
  padding: 5px 10px;
  text-decoration: none;
  font-weight: bold;
  display: inline-block;
  width: 100px;
}
#navbar a:hover {
  border-radius: 20px 5px;
  background-color: #0066FF;
}
```

i. Выпадающее меню

Меню, которое мы будем создавать, будет иметь основные навигационные ссылки, расположенные в горизонтальной панели навигации, и подпункты, которые будут отображаться только после наведения курсора мыши на тот пункт меню, к которому эти подпункты относятся.

Сначала нам нужно создать HTML-структуру нашего меню. Основные навигационные ссылки мы поместим в маркированный список:

```
<ul id="navbar">

  <li><a rel="nofollow ugc" target="_blank" href="#">Главная</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">Новости</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">Контакты</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">О нас</a></li>

</ul>
```

```
<ul id="navbar">

  <li><a rel="nofollow ugc" target="_blank" href="#">Главная</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">Новости</a></li>

  <li><a rel="nofollow ugc" target="_blank" href="#">Контакты</a>

    <ul>

      <li><a rel="nofollow ugc" target="_blank" href="#">Адрес</a></li>

      <li><a rel="nofollow ugc" target="_blank" href="#">Телефон</a></li>

      <li><a rel="nofollow ugc" target="_blank" href="#">Email</a></li>

    </ul>

  </li>

  <li><a rel="nofollow ugc" target="_blank" href="#">О нас</a></li>

</ul>
```

```
#navbar ul { display: none; }

#navbar li:hover ul { display: block; }
```

```
#navbar, #navbar ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
#navbar li { float: left; }
#navbar ul li { float: none; }
```

Затем нам нужно сделать так, чтобы наше выпадающее подменю не смещало контент, расположенный под панелью навигации, вниз. Для этого мы зададим пунктам списка position: relative;, а списку, содержащему подпункты position: absolute; и добавим свойство top со значением 100%, чтобы абсолютно позиционированное подменю отображалось точно под ссылкой.

```
#navbar ul {
  display: none;
  position: absolute;
  top: 100%;
}
#navbar li {
  float: left;
  position: relative;
}
#navbar { height: 30px; }
```

Высота для родительского списка была добавлена специально, так как браузеры не учитывают в качестве содержимого элемента плавающий контент, то без добавления высоты наш список будет проигнорирован браузером и контент, следующий за списком, будет обтекать наше меню.

Теперь нам осталось стилизовать оба наших списка и выпадающее меню будет готово:

```
#navbar ul {
  display: none;
  background-color: #f90;
  position: absolute;
```

```
  top: 100%;
}
#navbar li:hover ul { display: block; }
#navbar, #navbar ul {
  margin: 0;
  padding: 0;
  list-style-type: none;
}
#navbar {
  height: 30px;
  background-color: #666;
  padding-left: 25px;
  min-width: 470px;
}
#navbar li {
  float: left;
  position: relative;
  height: 100%;
}
#navbar li a {
  display: block;
  padding: 6px;
  width: 100px;
  color: #fff;
  text-decoration: none;
  text-align: center;
```

```
}

#navbar ul li { float: none; }

#navbar li:hover { background-color: #f90; }

#navbar ul li:hover { background-color: #666; }
```

Результат:

Создайте самостоятельно:

7. Повторите страницу по данному по образцу:

Вы можете открыть этот пример в отдельной вкладке браузера.

8. Повторите страницу по данному по образцу:

Вы можете открыть этот пример в отдельной вкладке браузера.