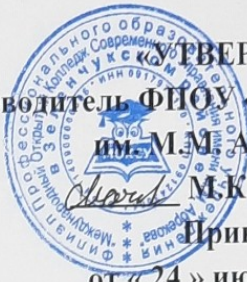


Филиал профессионального образовательного учреждения
«Международный Открытый Колледж Современного Управления
имени М.М. Абрекова» в Зеленчукском районе

ОДОБРЕНО
на заседании
Педагогического
совета ПОУ «МОКСУ
им. М.М. Абрекова»
Протокол № 1
от «24» июня 2023г.

Руководитель ФПОУ «МОКСУ
им. М.М. Абрекова»
М.К. Хубиева
Приказ № 245
от «24» июня 2023г.



Рабочая программа профессионального модуля

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем»

(МДК 01.01 Разработка программных модулей;
МДК 01.02. Поддержка и тестирование программных модулей;
МДК 01.03. Разработка мобильных приложений;
МДК 01.04. Системное программирование)

Специальность: *09.02.07. Информационные системы и программирование*
(базовая подготовка)

Квалификация выпускника: *Программист*

Форма обучения: *очная*

Обсуждено
на заседании цикловой комиссии
«22» июня 2023 г.
Протокол № 1

Составитель программы:
Блимготова М.Т.
Гатаева З.А.

ст. Зеленчукская, 2023г.

Согласовано:

Специальная астрофизическая обсерватория Российской академии наук.

Директор: Валявин Геннадий Геннадьевич

Договор № 23/1605 от 16.05.2023 г.

М.П.

Рецензент:

Байрамкулова Радмила Азреталиевна

Содержание

1. Аннотация.....	4
2. Требования к результатам освоения содержания дисциплины.....	5
3. Структура и содержание учебной дисциплины.....	7
3.1. Структура дисциплины.....	7
3.2. Содержание учебной дисциплины.....	8
4. Методические указания для обучающихся по освоению дисциплины.....	21
5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине.....	27
6. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, в том числе комплект лицензионного программного обеспечения, электронно- библиотечные системы, современные профессиональные базы данных и информационные справочные системы.....	28
7. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины.....	29
8. Перечень ресурсов информационно-коммуникационной сети «Интернет», необходимых для освоения дисциплин.....	30
9. Оценочные материалы для проведения промежуточной аттестации обучающихся по дисциплине.....	31
9.1. Описание показателей и критериев оценивания, описание шкал оценивания.....	31
9.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и(или) опыта деятельности, характеризующих формирование компетенций в процессе освоения образовательной программы.....	33
9.3. Типовые контрольные задания и иные материалы, необходимые для процедуры оценивания знаний, умений, навыков и(или) опыта деятельности, характеризующих формирование компетенций в процессе освоения образовательной программы.....	37
10. Лист регистрации изменений, дополнений.....	177

Аннотация

Рабочая программа ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», изучаемая в рамках программы подготовки специалистов среднего звена (ППССЗ), составлена в соответствии с требованиями ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование», утверждённого приказом Министерства образования и науки Российской Федерации от 09 декабря 2016 г. № 1547 (ред. от 01.09.2022 г.).

Рабочая программа предназначена для преподавания дисциплины студентам очной формы обучения по специальности 09.02.07. «Информационные системы и программирование».

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения профессионального модуля должен:

Иметь практический опыт:

- в разработке кода программного продукта на основе готовой спецификации на уровне модуля;
- использовании инструментальных средств на этапе отладки программного продукта;
- проведении тестирования программного модуля по определенному сценарию;
- использовании инструментальных средств на этапе отладки программного продукта;
- разработке мобильных приложений.

Цели:

- осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;
- осуществлять разработку кода программного модуля на современных языках программирования;
- уметь выполнять оптимизацию и рефакторинг программного кода; - оформлять документацию на программные средства.

Задачи:

- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования;
- способы оптимизации и приемы рефакторинга;
- основные принципы отладки и тестирования программных продуктов.

Место дисциплины в структуре ППССЗ

Профессиональный модуль 01 «Разработка модулей программного обеспечения для компьютерных систем» относится к дисциплинам профессионального учебного цикла.

Настоящая дисциплина включена к общепрофессиональным дисциплинам учебного плана по подготовке специалистов по специальности 09.02.07. «Информационные системы и программирование».

Дисциплины профессионального модуля изучаются на базе основного общего образования на 2 курсе в 4 семестрах, на базе среднего общего образования на 1 курсе в 2 семестрах, на базе основного общего образования на 3 курсе в 5-6 семестрах, на базе среднего общего образования на 2 курсе в 3-4 семестрах, на базе основного общего образования на 4 курсе в 7-8 семестрах, на базе среднего общего образования на 3 курсе в 5-6 семестрах.

2. Требования к результатам освоения содержания дисциплины

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций в соответствии с ФГОС СПО и ППССЗ по специальности 09.02.07. «Информационные системы и программирование».

Результатом освоения профессионального модуля является овладение обучающимися видом профессиональной деятельности Разработка модулей программного обеспечения для компьютерных систем, в том числе профессиональными (ПК) и общими (ОК) компетенциями:

	Наименование общих компетенций
ПК 1.1	Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием
ПК 1.2	Разрабатывать программные модули в соответствии с техническим заданием
ПК 1.3	Выполнять отладку программных модулей с использованием специализированных программных средств
ПК 1.4	Выполнять тестирование программных модулей
ПК 1.5	Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6	Разрабатывать модули программного обеспечения для мобильных платформ
ОК 1	Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;
ОК 2.	Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности;
ОК 3	Планировать и реализовывать собственное профессиональное и личностное развитие. Планировать и реализовывать собственное профессиональное или личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях;
ОК 4	Эффективно взаимодействовать и работать в коллективе и команде;
ОК 5	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;
ОК 6	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения;
ОК 7	Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях;
ОК 8	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности;
ОК 9	Пользоваться профессиональной документацией на государственном и иностранном языках.

3. Структура и содержание учебной дисциплины

3.1. Структура дисциплины

Общая трудоемкость дисциплины и виды учебной работы

Виды учебной работы	Трудоемкость часов
Максимальная учебная нагрузка (всего)	618
ПМ.01 Разработка модулей программного обеспечения для компьютерных систем	
МДК. 01.01. Разработка программных модулей	174
МДК. 01.02. Поддержка и тестирование программных модулей	92
МДК. 01.03. Разработка мобильных приложений	256
МДК. 01.04. Системное программирование	96
МДК. 01.01. Разработка программных модулей	174
Аудиторная работа:	156
Лекции (Л)	80
Семинарские занятия	-
Практическое занятие (ПЗ)	76
Лабораторные работы (ЛР)	-
Курсовое проектирование, курсовые работы	-
Самостоятельная работа	4
ПАТТ	14
Промежуточная аттестация	Диф. зачет/ экзамен
МДК. 01.02. Поддержка и тестирование программных модулей	92
Аудиторная работа:	90
Лекции (Л)	44
Семинарские занятия	-
Практическое занятие (ПЗ)	46
Лабораторные работы (ЛР)	-
Курсовое проектирование, курсовые работы	-
Самостоятельная работа	2
Промежуточная аттестация	Диф. зачет
МДК. 01.03 Разработка мобильных приложений	256
Аудиторная работа:	228
Лекции (Л)	114
Семинарские занятия	-
Практическое занятие (ПЗ)	114
Лабораторные работы (ЛР)	-

Курсовое проектирование, курсовые работы	-
Самостоятельная работа	14
ПАТТ	14
Промежуточная аттестация	зачет/зачет/экзамен
МДК. 01.04 Системное программирование	96
Аудиторная работа:	88
Лекции (Л)	44
Семинарские занятия	44
Практическое занятие (ПЗ)	-
Лабораторные работы (ЛР)	40
Курсовое проектирование, курсовые работы	-
Самостоятельная работа	8
Промежуточная аттестация	Диф. зачет

3.2. Содержание учебной дисциплины

МДК. 01.01. Разработка программных модулей			
Наименование разделов и тем	Содержание учебного материала, практические занятия, самостоятельная работа обучающихся	Объём часов	Форма ТКУ/баллы
Тема 1.1.1 Жизненный цикл ПО	Понятие ЖЦ ПО. Этапы ЖЦ ПО.	2	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие	-	
	Практическое занятие	2	
	Лабораторное занятие	-	
	Самостоятельная работа:	-	
Тема 1.1.2 Структурное программирование	Технология структурного программирования. Инструментальные средства оформления и документирования алгоритмов программ Оценка сложности алгоритма: классификация, классы алгоритмов, неразрешимые задачи.	10	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Условные конструкции. Оценка сложности алгоритмов сортировки. Оценка сложности алгоритмов поиска. Оценка сложности рекурсивных алгоритмов. Оценка сложности эвристических алгоритмов	10	
	Лабораторное занятие		

	Самостоятельная работа:		
Тема 1.1.3 Объектно-ориентированное программирование	Основные принципы объектно-ориентированного программирования. Классы: основные понятия. Перегрузка методов. Операции класса. Иерархия классов. Синтаксис интерфейсов. Интерфейсы и наследование. Структуры. Делегаты. Регулярные выражения. Коллекции. Параметризованные классы. Указатели. Операции со списками.	14	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Работа с классами. Абстрактные классы. Статистические классы Перегрузка методов. Определение операций в классе. Создание наследованных классов Работа с объектами через интерфейсы. Использование стандартных интерфейсов. Работа с типом данных структура. События Коллекции. Параметризованные классы. Использование регулярных выражений Операции со списками.	14	
	Лабораторное занятие		
	Самостоятельная работа:	2	
Тема 1.1.4 Паттерны проектирования	Назначение и виды паттернов. Основные шаблоны. Порождающие шаблоны. Структурные шаблоны. Поведенческие шаблоны.	10	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Использование основных шаблонов. Использование порождающих шаблонов. Использование структурных шаблонов. Использование поведенческих шаблонов	10	
	Лабораторное занятие		
	Самостоятельная работа:		
Тема 1.1.5. Событийно-управляемое программирование	Событийно-управляемое программирование. Элементы управления. Диалоговые окна. Обработчики событий. Введение в графику	14	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Форма и ее свойства. Диалоговые окна и сообщения Однострочные текстовые редакторы. Элементы переключателей и выбора. Обработка данных в табличном виде. Разработка приложений Браузера. Сохранение/чтение настроек приложения, используя Ini файлы. Сохранение/чтение настроек приложения, используя реестр Windows. Обработка	16	

	строковой информации. Обработка графической информации. Drag-And-Drop при разработки программных продуктов. Разработка приложения с использованием текстовых компонентов. Разработка приложения с несколькими формами. Разработка приложения с не визуальными компонентами. Разработка игрового приложения. Разработка приложения с анимацией.		
	Лабораторное занятие		
	Самостоятельная работа:		
Тема 1.1.6 Оптимизация и рефакторинг кода	Методы оптимизации программного кода. Цели рефакторинга. Методы рефакторинга	10	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Оптимизация и рефакторинг кода	10	
	Лабораторное занятие		
	Самостоятельная работа:		
Тема 1.1.7 Разработка пользовательского интерфейса	Правила разработки интерфейсов пользователя	10	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Разработка интерфейса пользователя.	10	
	Лабораторное занятие		
	Самостоятельная работа:		
Тема 1.1.8 Основы ADO.Net	Работа с базами данных Доступ к данным. Создание таблицы, работа с записями. Способы создания команд	10	ПК 1.1, ПК 1.2 ОК 1 – ОК 9
	Семинарское занятие		
	Практическое занятие Создание приложения с БД. Создание запросов к БД . Создание хранимых процедур	10	
	Лабораторное занятие		
	Самостоятельная работа:		

МДК. 01.02. Поддержка и тестирование программных модулей			
Наименование разделов и тем	Содержание учебного материала, практические занятия, самостоятельная работа обучающихся	Объем	Форма

	2. Составление плана тестирования учебной программы. 3. Тестирование «белым ящиком» 4. Тестирование «черным ящиком» 5. Модульное тестирование 6. Интеграционное тестирование		
	Лабораторное занятие	-	
	Самостоятельная работа:	-	
Тема 1.2 Документир ование	1. Процессы тестирования. Место тестирования в процессе разработки ПО. Тестирование и качество. Оценка качества продукта. Метрики Базовые принципы тестирования. Планирование работ по тестированию. Виды деятельности, осуществляемые при составлении плана тестирования. Артефакты тестирования. Модели зрелости тестирования ПО (TMMi). Инструментарий тест менеджера.	2 2	П К 1 . 1 , П К
	2. Риски тестирования. Создание условий работы в команде. Стили управления. Выстраивание отношений. Коммуникации. Постановка задач. SMART. Хронофаги – поглотители времени. 3. Команда тестирования. Оперативное планирование. Приоритет задач. Матрица Эйзенхауэра. Работа с почтой. 4. Оценка трудозатрат на тестирование. Определение задач, которые должны быть выполнены. Оценка трудоемкости задач. Эмпирическое правило Брукса. Практические соображения. Метод анализа видов ошибок и их влияния (FMEA). Упрощенный вариант. 5. Unit тестирование. Понятие Unit Test. Типы тестирования. Интеграционное тестирование. Верификация и валидация. Свойства хорошего Unit теста. Расположение Unit тестов. Именованые проектов. Именованые методов и классов. Какой код тестировать. Unit Test Frameworks. Подход AAA. Пример создания простого Unit теста. Более сложный пример со значением, определяющим сложность пароля. Работа атрибутов TestInitialize и ClassInitialize. Класс Assert. 6. Использование MSTest. Свойство TextContext. Data-driven test. Ordered и Gineric		1 . 2 О К 1 – О К 9

	<p>тесты. Атрибут DeploymentItem. Как заставить себя писать Unit тесты.</p> <p>Польза Unit тестирования. Test Driven Development.</p> <p>7. Виды Unit тестов. Виды тестовых объектов. Inversion of Control (IOC). Шаблон Service Locator.</p> <p>8. Шаблон Dependency Injection. Constructor Injection. Property Injection. Interface. Injection. Factory Method. Extract and Override Factory Injection.</p> <p>9. Проблемы инкапсуляции.</p> <p>10. Mock-объекты. Разница между Mock и Stub 2.</p> <p>11. Библиотека moq. Пример с библиотекой Logging Library.</p> <p>12. Настройка Mock-объектов, когда мы используем Moq framework out и ref параметры.</p> <p>13. Как проверять аргументы, передаваемые в mock-объект</p> <p>14. Как можно контролировать выполнение приложения. Исключения.</p> <p>15. Средства разработки технической документации.</p> <p>16. Технологии разработки документов.</p> <p>17. Документирование программного обеспечения в соответствии с Единой системой программной документации.</p> <p>18. Автоматизация разработки технической документации</p> <p>Автоматизированные средства оформления документации.</p>		
	Семинарское занятие		
	<p>Практическое занятие</p> <p>1. Составление списка задач по тестированию.</p> <p>2. Создание оперативного плана.</p> <p>3. Unit тестирование</p> <p>4. Создание сценариев тестов</p> <p>5. Описание результатов тестирования</p> <p>6. Оформление документации на программные средства с использованием инструментальных средств.</p>	2 2	
	Лабораторное занятие		
	<p>Самостоятельная работа:</p> <p>Подготовка докладов и рефератов, создание компьютерных презентаций.</p>	2	

МДК. 01.03. Разработка мобильных приложений

Наименование разделов и тем	Содержание учебного материала, практические занятия, самостоятельная работа обучающихся	Объем часов	Форма ТКУ / баллы
Тема 1.3.1 Основные платформы и языки разработки мобильных приложений	1. Введение в Java технологии. Особенности языка программирования Java. Описание Java технологий. Использование интегрированной среды разработки. 2. Введение в язык программирования Java. Языковые лексемы Java. Введение в систему типов языка Java. Работа с примитивными типами и константами. Операции языка Java. Преобразование простых типов. 3. Методы и операторы Java. Создание и вызов методов. Перегрузка и методы с переменным числом аргументов. Операторы ветвления. Операторы циклов. 4. Создание и использование массивов Java. Одномерные массивы. Многомерные массивы. Работа с массивами и класс Arrays. 5. Работа со строками в Java. Основные методы класса String. Сложение и преобразование строк. Классы	6 2	ПК 1 . 2 , ПК 1 . 6 ОК 1 — ОК 1 1

	<p>динамических строк. Ввод данных с клавиатуры и класс Scanner. Регулярные выражения.</p> <p>6. Техники тестирования. Индекс массива. Использование одномерных массивов. Создание одномерных массивов. Двумерные массивы. Использование двумерных массивов. Массивы, состоящие из одного элемента. Трехмерные массивы. Использование трехмерных массивов. Четырехмерные массивы. Зубчатые массивы. Использование зубчатых массивов. Ключевое слово <code>params</code>.</p> <p>7. Разработка классов на Java. Обзор основных концепций ООП. Объявление класса. Члены класса и создание объектов класса. Модификаторы доступа. Модификаторы <code>final</code> & <code>static</code>. Использование пакетов, директив импорта и переменной среды <code>CLASSPATH</code>.</p> <p>8. Наследование и полиморфизм. Наследование как механизм повторного использования кода. Конструктор при наследовании Преобразование типов и операция <code>instanceof</code>. Виртуальные методы и позднее связывание. Абстрактные классы и методы.</p> <p>9. Пакет <code>java.lang</code>. Класс <code>Object</code> и переопределение его методов. Метаданные и рефлексия. Классы <code>System</code> и <code>Math</code>.</p> <p>10. Обработка ошибок в Java. Концепция исключений в</p>		
--	---	--	--

	<p>Java. Использование операторов try, catch и finally. Проверяемые и непроверяемые исключения. Создание своих классов исключений. Оператор try для освобождения ресурсов.</p> <p>12. Потоки данных в Java. Обзор классов потоков. Работа с байтовыми потоками. Работа с потоками символов. Использование класса java.io.File. Сжатие файлов. Сериализация объектов в Java. Работа с файловой системой в NIO Использование интерфейса Path. Работа с атрибутами файлов. Основные возможности класса Files. Использование класса Files для обхода дерева каталогов. Мониторинг изменений в файловой системе.</p> <p>13. Пакет java.util. Форматирование данных. Работа с датой и временем. Класс Locale и глобализация кода. Локализация и класс ResourceBundle. Генерация псевдослучайных чисел.</p> <p>14. Коллекции в Java. Иерархия классов коллекций. Концепция параметризованных типов данных. Работа с параметризованным методов и интерфейсом. Обзор возможностей списков, множеств и словарей в Java. Вложенные классы в Java. Внутренние классы. Вложенные классы. Анонимные классы. Перечисления в Java.</p> <p>15. Лямбда-выражения. Синтаксис лямбда-выражений. Ссылки на</p>		
--	---	--	--

<p>методы. Функциональные интерфейсы. Паттерны проектирования. Обзор паттернов. Паттерн одиночка. Паттерн композиция. Паттерн наблюдатель.</p>		
Семинарское занятие	-	
<p>Практическое занятие 1. Установка JDK/IDE, настройка параметров среды. 2. Activity, создание графического приложения 3. Разработка интерфейса мобильного приложения программно в коде Java 4. Разработка интерфейса мобильного приложения в XML 5. Контейнеры и позиционирование элементов в мобильных приложениях 6. Ресурсы 7. Работа с изображениями 8. Адаптеры 9. Установка инструментария и настройка среды для разработки мобильных приложений 10. Установка среды разработки мобильных приложений с применением виртуальной машины</p>	4 0	
Лабораторное занятие	-	
<p>Самостоятельная работа: 1. Работа с конспектами лекций, учебной и специальной литературой. 2. Подготовка к практическим занятиям, оформление результатов практических занятий, отчётов и подготовка к их защите. 3. Подготовка докладов и рефератов, создание компьютерных презентаций. 4. Выполнение индивидуальных заданий.</p>	6	

Тема 1.3.2 Создание и тестировани е модулей для мобильных приложений	1 .	<p>Обзор платформы Android. История возникновения Android. Преимущества Android. Архитектура Android. Особенности платформы Android. Основные компоненты Android. Безопасность и полномочия (Permissions). Установка и настройка компонентов среды разработки.</p> <p>Активности и ресурсы. Активности (Activity) в Android. Создание Активности. Жизненный цикл Активности. Стеки Активностей. Состояния Активностей. Отслеживание изменений состояния Активности. Ресурсы. Отделение ресурсов от кода программы. Создание ресурсов. Простые значения. Визуальные стили и темы. Изображения. Разметка. Анимация. Меню</p> <p>Приложения и пользовательский интерфейс. Использование внешних ресурсов в коде приложения. Использование ресурсов внутри ресурсо. Локализация приложения с помощью внешних ресурсов. Класс Application. Обработка событий жизненного цикла приложения. Понятие контекста. Пользовательский интерфейс. Представления (View). Разметка (Layout)</p>	4	<p>П К 1 . 2 , П К 1 . 6 О К 1 – О К 1 1</p>
	2 .	<p>Намерения, меню и работа с данными. Адаптеры в Android. Использование Адаптеров для привязки данных. Намерения в Android. Использование Намерений для запуска Активностей. Неявные намерения. Сохранение</p>	4	

		<p>состояния и настроек приложения.</p> <p>Общие Настройки (Shared Preferences). Работа с файлами. Использование статических файлов как ресурсов. Меню в Android. Дочерние иконтекстные меню. Описание меню с помощью XML.</p> <p>СУБД, контент-провайдеры и использование сетевых сервисов. Диалоги в Android. Работа с базами данных в Android. Курсоры (Cursor) и ContentValues. Работа с СУБД SQLite. Работа с СУБД без адаптера. Особенности работы с БД в Android. Выполнение запросов для доступа к данным. Изменение данных в БД. Использование SimpleCursorAdapter.</p> <p>Контент-провайдеры. Использование контент-провайдеров.</p> <p>Создание контент-провайдеров. Использование интернет-сервисов</p>		
3	.	<p>Широковещательные приемники (Broadcast Receivers).</p> <p>Применение широковещательных приемников.</p> <p>Жизненный цикл Приемника.</p> <p>Регистрация Приемника.</p> <p>Использование Ordered Broadcast. Использование PendingIntent. Извещения (Notifications) в Android.</p> <p>Взаимодействие с Извещениями. Управление Извещениями. Создание Извещений. Обновление Извещений. Фрагменты (Fragments). Создание Фрагментов. Добавление пользовательского интерфейса. Добавление фрагментов к Активностям. Управление Фрагментами. Транзакции с Фрагментами.</p> <p>Взаимодействие Фрагментов и Активностей. Жизненный цикл Фрагментов.</p>	4	

4	<p>Процессы и потоки (Threads). Жизненный цикл процесса. Потоки. Фоновые потоки.</p> <p>Использование AsyncTask.</p> <p>Сервисы (Services).</p> <p>Описание Сервисов в Манифесте приложения.</p> <p>Запуск Сервисов. Остановка Сервисов. Связанные Сервисы. Сервисы и Извещения. Сервисы переднего плана (Foreground Services). Жизненный цикл Сервисов</p>	4	
5	<p>Виджеты (Widgets).</p> <p>Описание Виджетов в Манифесте приложения.</p> <p>Создание разметки Виджета.</p> <p>Класс AppWidgetProvider.</p> <p>Создание Виджета.</p> <p>Использование Конфигурационной Активности.</p> <p>Использование Preview Image.</p> <p>Обновление Виджетов</p>	4	
6	<p>Работа с картами SD и внутренним хранилищем устройства. Проверка доступности носителя.</p> <p>Доступ к файлам.</p> <p>Совместно используемые файлы и стандартные каталоги.</p> <p>Файлы кэша приложений</p>	4	
7	<p>Загрузчики (Loaders). Обзор API Загрузчиков.</p> <p>Применение Загрузчиков.</p> <p>Запуск и перезапуск Загрузчиков. Использование LoaderManager.</p> <p>Использование LoaderCursor.</p>	4	
8	<p>Беспроводные соединения.</p> <p>Проверка сетевых соединений.</p> <p>Отслеживание состояния соединений.</p> <p>ConnectivityManager и NetworkInfo.</p> <p>Эффективное использование сетевых соединений.</p>	4	
9	<p>Использование AlarmManager и AlarmClock.</p> <p>Типы будильников в Android. Однократные и повторяющиеся события.</p> <p>Области применения AlarmManager</p>	4	

		альтернативы (Timer и Handler). Использование AlarmClock		
	1 0 .	Сенсоры в Android. Обзор сенсоров. Типы сенсоров и получение информации об их доступности. Sensor Framework. Мониторинг состояния сенсоров. Лучшие практики при работе с сенсорами.	4	
	1 1	Телефония и SMS. Совершение звонков из приложения. Определение состояния и параметров телефона. Мониторинг состояния телефонного модуля. Использование SMS. Отправка SMS. Получение SMS.	4	
	1 2	Собственные View. Использование Canvas. Использование SurfaceView. Использование Drawable. Shape Drawable и 2D графика. Модификация существующих View. Создание собственных View	4	
	1 3	Звук и камера в Android. Запись и воспроизведение звука. Основы работы с камерой в Android. Использование имеющихся приложений работы с камерой. Прямое управление камерой. Съемка и сохранение фото и видео.	4	
	Практические занятия:			
	1	Создание эмуляторов и подключение устройств	4	
	2	Настройка режима терминала	4	
	3	Создание нового проекта	4	
	4	Изучение и комментирование кода	4	
	5	Изменение элементов дизайна	4	
	6	Обработка событий: цветовая индикация	4	
	7	Обработка событий: подсказки	4	
	8	Подготовка стандартных модулей	4	

	9	Обработка событий: переключение между экранами	4	
	1 0	Передача данных между модулями	4	
	1 1	Тестирование и оптимизация мобильного приложения	4	
	1 2	Обработка событий: подсказки	4	
	1 3	Обработка событий: цветовая индикация	4	
	1 4	Подготовка стандартных модулей	6	
	1 5	Обработка событий: переключение между экранами	4	
	1 6	Передача данных между модулями	6	
	1 7	Тестирование и оптимизация мобильного приложения	6	
Лабораторное занятие			-	
Самостоятельная работа: 1. Работа с конспектами лекций, учебной и специальной литературой. 2. Подготовка к практическим занятиям, оформление результатов практических занятий, отчётов и подготовка к их защите. 3. Подготовка докладов и рефератов, создание компьютерных презентаций. 4. Выполнение индивидуальных заданий.			8	

МДК. 01.04 Системное программирование

Наименование разделов и тем	Содержание учебного материала, практические занятия, самостоятельная работа обучающихся	О б ъ е м ч а с о в	Ф о р м а Т К У / б а л
--------------------------------	--	--	--

			Л ы
Тема 1.4.1 Программирование на языке низкого уровня	1	Ознакомление со спецификой сборки ПО в GNU/Linux (Ubuntu). Ознакомление с консольными текстовыми редакторами (vi,nano,mcedit). Ручная и автоматическая сборка ПО (Makefile). Модель Клиент-Интерфейс-Сервер (КИС). Статическая сборка библиотек. Совместно используемые библиотеки. Работа с переменными окружения.	2 П К 1 . 2 , П К 1 . 3 , О К 1 — О К 1 0 .
	2	Низкоуровневый ввод-вывод и файловые операции. Обзор механизмов ввода/вывода в Linux (Ubuntu). Файловые дескрипторы. Системные вызовы: open, close, write, read и lseek. Типы файлов. Индексные дескрипторы и жесткие ссылки. Права доступа к файлу.	2
	3	Файловая система proc. Два способа прочесть содержимое директории. Разреженные файлы и специфика их применения. Блокировка областей файла.	2
	4	Межпроцессное взаимодействие. Механизмы межпроцессного взаимодействия. Linux (Ubuntu) Неименованные каналы (pipes) Именованные каналы (named pipes) Сообщения (message queue).	2

		Разделяемая память (shared memory). Семафоры(semaphores).		
	5	Сокеты. Сокеты в файловом пространстве имен (UNIX-сокеты). Парные сокеты(pair sockets). Сетевые сокеты (sockets)	2	
	6	Сигналы. Знакомство с сигналами (signals). Отличие сигналов от других механизмов межпроцессного взаимодействия. Специфика обработки сигналов(signal handling)	2	
	7	Процессы. лонирование процессов — (fork) Замена исполняемого процесса —(exec) Зомби (zombies) — причины возникновения и способы их устранения.	2	
	8	Потоки. Потоки и процессы. Специфика построения многопоточных приложений(multithreading). Досрочное завершение потока. Создание обработчика завершения потока. Средства синхронизации потоков(synchronize primitives). Атрибуты потоков	2	
	9	Демоны (службы). Отличие демона от консольной утилиты. Специфика разработки демонов (daemons). Создание демона, использующего сетевые сокеты.	2	
	10	Консольный ввод-вывод. Специфика разработки консольных приложений. Предотвращение перенаправления вывода. Управление терминалом. Скрытие пароля пользователя при аутентификации	2	
	11	Отображаемая память. Отображение обычного файла. Совместный	2	

		доступ к файлу. Частные отображения. Другие применения mmap. Отображение обычного файла. Совместный доступ к файлу. Частные отображения. Другие применения mmap.		
1 2		Программирование графического интерфейса с помощью GTK+. Специфика разработки 64-битных приложений. Использование библиотеки ncurses. Программирование графического интерфейса с помощью GTK+	2	
Практические занятия				П К 1 . 2 , П К 1 . 3 , О К 1 – О К 1 0 .
1		Переменные окружений.	2	
2		Низкоуровневый ввод-вывод и файловые операции	2	
3		Межпроцессное взаимодействие	2	
4		Сокеты. Сигналы. Процессы. Сетевое программирование сокетов.	4	

	5	Потоки. Использование потоков.	4	
	6	Демоны (службы).	2	
	7	Обмен данными. Работы с буфером экрана.	4	
	Лабораторное занятие		-	
	Самостоятельная работа: 1. Работа с конспектами лекций, учебной и специальной литературой. 2. Подготовка к практическим занятиям, оформление результатов практических занятий, отчётов и подготовка к их защите. 3. Подготовка докладов и рефератов, создание компьютерных презентаций. 4. Выполнение индивидуальных заданий.		4	
Тема 1.4.2. Программирование в Windows	1	Начало работы. Интерфейс.	2	П К 1 . 2 , П К 1 . 3 , О К 1 — О К 1 0 .
	2	Типы данных Windows	2	
	3	Механизм сообщений. Классификация сообщений по функциональным признакам	2	
	4	Простейшая Windows программа	2	
	5	Работа с окнами.	2	

		Функции для работы с окнами. Основы вывода		
	6	Контекст устройства. Режим отображения	2	
	7	Масштаб осей физических координат	2	
	8	Работа с общим контекстом отображения	2	
	9	Работ с общим контекстом отображения	2	
	10	Органы управленияРабота с мышью. Таймер.	2	
	Практические занятия			
	1	Управление каталогами. Управление системным реестром	2	
	2	Изучение исключений и обработчиков событий	2	
	3	Изучение стандартных устройств и консольного вывода	2	
	4	Проверка оборудования	2	
	5	Управление клавиатурой. Управление таймером	2	
	6	Управление видеоадаптером	2	
	7	Изучение работы главной загрузочной панели	4	
	8	Изучение дисковых структур данных	4	
	9	Управление программами	4	
	Лабораторное занятие		-	
	Самостоятельная работа: 1.Работа с конспектами лекций, учебной и специальной литературой. 2. Подготовка к практическим занятиям, оформление результатов практических занятий, отчётов и подготовка к их защите. 3. Подготовка докладов и рефератов, создание компьютерных презентаций. 4. Выполнение индивидуальных заданий.		4	

Учебная практика ПМ 01 Виды работ по разделу 1: 1 Участие в разработке алгоритма решения поставленной задачи и реализация его средствами автоматизированного проектирования 2 Участие в разработке кода программного продукта на основе готовой спецификации на уровне модуля Виды работ по разделу 2: 1 Участие в использовании инструментальных средств на этапе отладки программного продукта; 2 Участие в проведении тестирования программного модуля по определенному сценарию 3 Участие в проведении тестирования программного модуля по определенному сценарию; 4 Участие в использовании инструментальных средств на этапе тестирования программного продукта 5 Участие в анализе алгоритмов в том числе с применением инструментальных средств 6 Участие в осуществлении рефакторинга и оптимизации программного кода Виды работ по разделу 3: 1 Участие в разработке мобильных приложений	180	ПК1.1 – ПК1.6 ОК.01 – ОК.11
Производственная практика ПМ 01 Виды работ по разделу 1: 1 Самостоятельная разработка алгоритма решения поставленной задачи и реализация его средствами автоматизированного проектирования 2 Самостоятельная разработка кода программного продукта на основе готовой спецификации на уровне модуля Виды работ по разделу 2: 7 Использование инструментальных средств на этапе отладки программного продукта; 8 Самостоятельное проведение тестирования программного модуля по определенному сценарию 9 Самостоятельное проведение тестирования программного модуля по определенному сценарию; 10 Использование инструментальных средств на этапе тестирования программного продукта 11 Самостоятельный анализ алгоритмов в том числе с применением инструментальных средств 12 Самостоятельное осуществление рефакторинга и оптимизации программного кода. Обзор и выбор языков программирования и инструментальных средств создания мобильных приложений	216	ПК1.2 – ПК1.6
ПАТТ	1 4	

В результате освоения дисциплины обучающийся должен:

Уметь

- осуществлять разработку кода программного модуля на языках низкого и высокого уровней;

- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;
- осуществлять разработку кода программного модуля на современных языках программирования;
- уметь выполнять оптимизацию и рефакторинг программного кода; - оформлять документацию на программные средства.

Знать

- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования;
- способы оптимизации и приемы рефакторинга;
- основные принципы отладки и тестирования программных продуктов.

4.Методические указания для обучающихся по освоению дисциплины

В процессе преподавания ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем» используются такие виды учебной работы, как лекции, семинары, а также различные виды самостоятельной работы обучающихся по заданию преподавателя.

Методические указания для обучающихся при работе над конспектом лекций во время проведения лекции

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект, что позволит впоследствии вспомнить изученный учебный материал, дополнить содержание при самостоятельной работе с литературой, подготовиться к экзамену.

Следует также обращать внимание на категории, формулировки, раскрывающие содержание тех или иных явлений и процессов, научные выводы и практические рекомендации, положительный опыт в ораторском искусстве. Желательно оставить в рабочих конспектах поля, на которых делать пометки из рекомендованной литературы, дополняющие материал прослушанной лекции, а также подчеркивающие особую важность тех или иных теоретических положений.

Любая лекция должна иметь логическое завершение, роль которого выполняет заключение. Выводы по лекции подытоживают размышления преподавателя по учебным вопросам. Формулируются они кратко и лаконично, их целесообразно записывать. В конце лекции обучающиеся имеют возможность задать вопросы преподавателю по теме лекции.

Методические указания для обучающихся при работе на семинаре

Семинары реализуются в соответствии с рабочим учебным планом при последовательном изучении тем дисциплины.

В ходе подготовки к семинарам обучающемуся рекомендуется изучить основную литературу, ознакомиться с дополнительной литературой, новыми публикациями в периодических изданиях: журналах, газетах и т.д. При этом следует учесть рекомендации преподавателя и требования учебной программы. Рекомендуется также дорабатывать свой конспект лекции, делая в нем соответствующие записи из литературы, рекомендованной преподавателем и предусмотренной учебной программой. Следует подготовить тезисы для выступлений по всем учебным вопросам, выносимым на семинар. Готовясь к докладу или реферативному сообщению, обучающийся может обращаться за методической помощью к преподавателю. Заканчивать подготовку следует составлением плана (конспекта) по изучаемому материалу (вопросу). Это позволяет составить концентрированное, сжатое представление по изучаемым вопросам.

В ходе семинара обучающийся может выступать с заранее подготовленным докладом. Также он должен проявлять активность при обсуждении выступлений и докладов одногруппников.

Поскольку активность обучающегося на семинарских занятиях является предметом контроля его продвижения в освоении курса, подготовка к семинарским занятиям требует ответственного отношения. Не допускается выступление по первоисточнику – необходимо иметь подготовленный письменный доклад, оцениваемый преподавателем наряду с устным выступлением. Не допускается также и распределение вопросов к семинару среди обучающихся группы, в результате которого отдельный обучающийся является не готовым к конструктивному обсуждению «не своего» вопроса. Все вопросы к семинару должны быть письменно проработаны каждым обучающимся.

Методические рекомендации по организации самостоятельной работы

Самостоятельная работа обучающихся направлена на самостоятельное изучение отдельных тем/вопросов учебной дисциплины.

Самостоятельная работа является обязательной для каждого обучающегося, ее объем по дисциплине определяется учебным планом.

При самостоятельной работе обучающиеся взаимодействуют с рекомендованными материалами при минимальном участии преподавателя.

Работа с литературой (конспектирование) Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у обучающихся свое отношение к конкретной проблеме.

Изучая материал по учебной книге (учебнику, учебному пособию, монографии, и др.), следует переходить к следующему вопросу только после полного уяснения предыдущего, фиксируя выводы и вычисления (конспектируя), в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода.

Особое внимание обучающийся должен обратить на определение основных понятий курса. Надо подробно разбирать примеры, которые поясняют определения. Полезно составлять опорные конспекты.

Выводы, полученные в результате изучения учебной литературы, рекомендуется в конспекте выделять, чтобы при перечитывании материала они лучше запоминались.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса.

Вопросы, которые вызывают у обучающегося затруднение при подготовке, должны быть заранее сформулированы и озвучены во время занятий в аудитории для дополнительного разъяснения преподавателем.

Реферат

Подготовка рефератов направлена на развитие и закрепление у обучающихся навыков самостоятельного глубокого, творческого и всестороннего анализа научной, методической и другой литературы по актуальным проблемам дисциплины; на выработку навыков и умений грамотно и убедительно излагать материал, четко формулировать теоретические обобщения, выводы и практические рекомендации.

Рефераты должны отвечать высоким квалификационным требованиям в отношении научности содержания и оформления.

Темы рефератов, как правило, посвящены рассмотрению одной проблемы. Объем реферата может быть от 10 до 15 страниц текста, отпечатанного через 1,5 интервала на компьютере, (список литературы и приложения в объем не входят).

Текстовая часть работы состоит из введения, основной части и заключения.

Во введении обучающийся кратко обосновывает актуальность избранной темы реферата, раскрывает конкретные цели и задачи, которые он собирается решить в ходе своего небольшого исследования.

В основной части подробно раскрывается содержание вопроса (вопросов) темы.

В заключении кратко должны быть сформулированы полученные результаты исследования и даны выводы. Кроме того, заключение может включать предложения автора, в том числе и по дальнейшему изучению заинтересовавшей его проблемы.

В список литературы (источников и литературы) обучающийся включает только те документы, которые он использовал при написании реферата.

В приложении (приложения) к реферату могут выноситься таблицы, графики, схемы и другие вспомогательные материалы, на которые имеются ссылки в тексте реферата.

Методические указания для обучающихся по подготовке доклада

Доклад - сообщение по заданной теме, с целью внести знания из дополнительной литературы, систематизировать материал, проиллюстрировать примерами, развивать навыки самостоятельной работы с научной литературой, познавательный интерес к научному познанию.

Тема доклада должна быть согласованна с преподавателем и соответствовать теме учебного занятия. Материалы при его подготовке, должны соответствовать научно-методическим требованиям и быть указаны в докладе. Необходимо соблюдать регламент, оговоренный при получении задания. Иллюстрации должны быть достаточными, но не чрезмерными. Работа студента над докладом включает отработку умения самостоятельно обобщать материал и делать выводы в заключении, умения ориентироваться в материале и отвечать на дополнительные вопросы слушателей, отработку навыков ораторства, умения проводить диспут.

Докладчик должен знать и уметь:

- сообщать новую информацию;
- использовать технические средства; хорошо ориентироваться в теме всего семинарского занятия;
- дискутировать и быстро отвечать на заданные вопросы; четко выполнять установленный регламент (не более 10 минут);
- иметь представление о композиционной структуре доклада и др.

Структура выступления.

Вступление должно содержать:

- название, сообщение основной идеи;
- современную оценку предмета изложения, краткое перечисление рассматриваемых вопросов;
- живую интересную форму изложения;
- акцентирование внимания на важных моментах, оригинальность подхода.

Основная часть, в которой выступающий должен глубоко раскрыть суть затронутой темы, обычно строится по принципу отчета. Задача основной части – представить достаточно данных для того, чтобы слушатели заинтересовались темой и захотели ознакомиться с материалами. При этом логическая структура теоретического блока не должны даваться без наглядных пособий, аудиовизуальных и визуальных материалов. Заключение – ясное, четкое обобщение и краткие выводы, которых всегда ждут слушатели.

Методические указания для обучающихся по подготовке доклада-презентации

Для подготовки презентации необходимо собрать и обработать начальную информацию.

Последовательность подготовки презентации в программе Microsoft PowerPoint:

1. Четко сформулировать цель презентации: вы хотите свою аудиторию мотивировать, убедить, заразить какой-то идеей или просто формально отчитаться.

2. Определить каков будет формат презентации: живое выступление (тогда, сколько будет его продолжительность) или электронная рассылка (каков будет контекст презентации).

3. Отобрать всю содержательную часть для презентации и выстроить логическую цепочку представления.

4. Определить ключевые моменты в содержании текста и выделить их.

5. Определить виды визуализации (картинки) для отображения их на слайдах в соответствии с логикой, целью и спецификой материала.

6. Подобрать дизайн и форматировать слайды (количество картинок и текста, их расположение, цвет и размер).

7. Проверить визуальное восприятие презентации.

К видам визуализации относятся иллюстрации, образы, диаграммы, таблицы.

Иллюстрация – представление реально существующего зрительного ряда.

Практические советы по подготовке презентации:

- готовьте отдельно: печатный текст, слайды (10-15), раздаточный материал;
- обязательная информация для презентации: тема, фамилия и инициалы выступающего;
- план сообщения;
- краткие выводы из всего сказанного;
- список использованных источников.

Доклад - сообщение по заданной теме, с целью внести знания из дополнительной литературы, систематизировать материал, проиллюстрировать примерами, развивать навыки самостоятельной работы с научной литературой, познавательный интерес к научному познанию.

Тема доклада должна быть согласованна с преподавателем и соответствовать теме учебного занятия. Материалы при его подготовке, должны соответствовать научно-методическим требованиям и быть указаны в докладе. Необходимо соблюдать регламент, оговоренный при получении задания. Иллюстрации должны быть достаточными, но не чрезмерными. Работа студента над докладом включает отработку умения самостоятельно обобщать материал и делать выводы в заключении, умения ориентироваться в материале и отвечать на дополнительные вопросы слушателей, отработку навыков ораторства, умения проводить диспут.

Докладчик должен знать и уметь:

- сообщать новую информацию;
- использовать технические средства; хорошо ориентироваться в теме всего семинарского занятия;
- дискутировать и быстро отвечать на заданные вопросы; четко выполнять установленный регламент (не более 10 минут);
- иметь представление о композиционной структуре доклада и др.

Структура выступления.

Вступление должно содержать:

- название, сообщение основной идеи;

- современную оценку предмета изложения, краткое перечисление рассматриваемых вопросов;
- живую интересную форму изложения;
- акцентирование внимания на важных моментах, оригинальность подхода.

Основная часть, в которой выступающий должен глубоко раскрыть суть затронутой темы, обычно строится по принципу отчета. Задача основной части – представить достаточно данных для того, чтобы слушатели заинтересовались темой и захотели ознакомиться с материалами. При этом логическая структура теоретического блока не должны даваться без наглядных пособий, аудио-визуальных и визуальных материалов. Заключение – ясное, четкое обобщение и краткие выводы, которых всегда ждут слушатели.

Методические указания для обучающихся по подготовке к тесту

Тест – особая форма проверки знаний. Проводится после освоения одной или нескольких тем и свидетельствует о качестве понимания основных понятий изучаемого материала. Тестовые задания составлены к ключевым понятиям, основным разделам, важным терминологическим категориям изучаемой дисциплины.

Для подготовки к тесту необходимо знать терминологический аппарат дисциплины, понимать смысл научных категорий и уметь их использовать в профессиональной лексике.

Владение понятийным аппаратом, включённым в тестовые задания, позволяет преподавателю быстро проверить уровень понимания студентами важных методологических категорий.

Методические указания для обучающихся по выполнению практикумов по решению задач

Практикум по решению задач – выполнение обучающимися набора практических задач предметной области с целью выработки навыков их решения.

Практикумы по решению задач выполняются в соответствии с рабочим учебным планом при последовательном изучении тем дисциплины.

Прежде чем приступать к решению задач, обучающемуся необходимо:

- ознакомиться с соответствующими разделами программы дисциплины по учебной литературе, рекомендованной программой курса;
- получить от преподавателя информацию о порядке проведения занятия, критериях оценки результатов работы;
- получить от преподавателя конкретное задание и информацию о сроках выполнения, о требованиях к оформлению и форме представления результатов.

При выполнении задания необходимо привести развёрнутые пояснения хода решения и проанализировать полученные результаты.

При необходимости обучающиеся имеют возможность задать вопросы преподавателю по трудностям, возникшим при решении задач.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Материально-техническое обеспечение дисциплины включает в себя:

- учебные аудитории для проведения занятий лекционного типа, занятий семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации: мебель аудиторная (столы, стулья, доска аудиторная), стол преподавателя, стул преподавателя, персональный компьютер, мультимедийное оборудование (проектор, экран), наглядные пособия;
- помещения для самостоятельной работы обучающихся, оснащенные компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду организации.

6. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, в том числе комплект лицензионного программного обеспечения, электронно-библиотечные системы, современные профессиональные базы данных и информационные справочные системы

Обучающиеся обеспечены доступом к электронной информационно-образовательной среде Колледжа из любой точки, в которой имеется доступ к сети «Интернет», как на территории организации, так и вне ее.

При проведении образовательного процесса по дисциплине необходимо наличие:

лицензионное программное обеспечение:

- Операционная система Microsoft Windows 10 pro - АКТ №180001220VR от 05 октября 2018 в соответствии с условиями договора №61020926 от 26 сентября 2016 г.;
- Программное обеспечение Microsoft Office Professional 16 - АКТ №180001220VR от 05 октября 2018 в соответствии с условиями договора №61020926 от 26 сентября 2016 г.;
- Программное обеспечение Microsoft Office Professional Plus Education All Languages License Software Assurance Pack Open Value Level E 1 Year Academic Enterprise - АКТ №180001220VR от 05 октября 2018 в соответствии с условиями договора №61020926 от 26 сентября 2016 г.;
- Комплексная Система Антивирусной Защиты Kaspersky Anti-Virus Russian Edition. 2 Desktop 1 year Base Box – Договор поставки № 64 от 08.02.2018;

электронно-библиотечная система:

- Электронная библиотечная система (ЭБС) «Университетская библиотека ONLINE» <http://biblioclub.ru> – Дополнительное соглашение № 1 от 24.08.2018г. к договору №131-05/17 от 15.05.2017г.

современные профессиональные базы данных:

- Официальный интернет-портал базы данных правовой информации <http://pravo.gov.ru>.
- Портал «Информационно-коммуникационные технологии в образовании» <http://www.ict.edu.ru>.

информационные справочные системы:

- Портал Федеральных государственных образовательных стандартов высшего образования <http://fgosvo.ru>.
- Компьютерная справочная правовая система «КонсультантПлюс» – договор об информационной поддержке от 01.01.2015.

Оснащенные базы практики:

Реализация образовательной программы предполагает обязательную учебную и производственную практику.

Учебная и производственная практики проводятся при освоении обучающимися профессиональных компетенций в рамках профессиональных модулей и реализовываются как в несколько периодов, так и рассредоточено, чередуясь с теоретическими занятиями в рамках профессиональных модулей.

Учебная практика реализуется в лабораториях колледжа в соответствии имеющимся оборудованием, инструментами, расходными материалами, которые обеспечивают выполнение всех видов работ, определенных содержанием ФГОС СПО.

Производственная практика проводится на предприятиях (в организациях) города. Оборудование предприятий (организаций) и технологическое оснащение рабочих мест производственной практики соответствует содержанию деятельности и дает возможность обучающемуся овладеть профессиональными компетенциями по всем осваиваемым видам деятельности, предусмотренным программой с использованием современных технологий, материалов и оборудования.

7. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины

7.1 Основная литература

1. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. - Москва: Издательство Юрайт, 2021. - 235 с. - (Профессиональное образование). - ISBN 978-5-534-05047-9. - URL: <https://urait.ru/bcode/472502>

7.2 Дополнительная литература

1. Огнева, М. В. Программирование на языке C++: практический курс: учебное пособие для вузов / М. В. Огнева, Е. В. Кудрина. - Москва: Издательство Юрайт, 2021. - 335 с. - (Высшее образование). - ISBN 978-5-534-05123-0. - URL: <https://urait.ru/bcode/473054>
2. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для среднего профессионального образования / Д. Ю. Федоров. - 3-е изд., перераб. и доп. - Москва: Издательство Юрайт, 2021. - 210 с. - (Профессиональное образование). - ISBN 978-5-534-12829-1. - URL: <https://urait.ru/bcode/487079>
3. Маркин, А. В. Программирование на SQL в 2 ч. Часть 1: учебник и практикум для вузов / А. В. Маркин. - 2-е изд., перераб. и доп. - Москва: Издательство Юрайт, 2021. - 403 с. - (Высшее образование). - ISBN 978-5-534-12256-5. - URL: <https://urait.ru/bcode/471148>
4. Огнева, М. В. Программирование на языке C++: практический курс : учебное пособие для среднего профессионального образования / М. В. Огнева, Е. В. Кудрина. - Москва: Издательство Юрайт, 2021. - 335 с. - (Профессиональное образование). - ISBN 978-5-534-05780-5. - URL: <https://urait.ru/bcode/473118>
5. Подбельский, В. В. Программирование. Базовый курс C#: учебник для вузов / В. В. Подбельский. - Москва: Издательство Юрайт, 2020. - 369 с. - (Высшее образование). - ISBN 978-5-534-10616-9. - URL: <https://urait.ru/bcode/450868>
6. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для вузов / Д. Ю. Федоров. - 3-е изд., перераб. и доп. - Москва: Издательство Юрайт, 2021. - 210 с. - (Высшее образование). - ISBN 978-5-534-14638-7. - URL: <https://urait.ru/bcode/478098>
7. Подбельский, В. В. Программирование. Базовый курс C#: учебник для среднего профессионального образования / В. В. Подбельский. - Москва: Издательство Юрайт, 2020. - 369 с. - (Профессиональное образование). - ISBN 978-5-534-11467-6. - URL: <https://urait.ru/bcode/456697>

8. Перечень ресурсов информационно-коммуникационной сети «Интернет», необходимых для освоения дисциплины

9.3. Типовые контрольные задания и иные материалы, необходимые для процедуры оценивания знаний, умений, навыков и(или) опыта деятельности, характеризующих формирование компетенций в процессе освоения образовательной программы

Раздел 1. МДК. 01.01. Разработка программных модулей
Тема 1.1.1 Жизненный цикл ПО

Вопрос 1.

Жизненный цикл программы — это

Варианты ответов:

- это поиск способов повышения эффективности работы программы (быстрая работа, уменьшение затрат ресурсов ПК);
- это период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации
- это экспериментальный анализ сложности алгоритма или экспериментальное сравнение нескольких алгоритмов, решающих одну и ту же задачу.

Вопрос 2

Программа – это

Варианты ответов:

- это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи за конечное время.
- это комплекс взаимосвязанных программ, предназначенный для поставки, передачи, продажи пользователю.
- это последовательность инструкций, предназначенная для исполнения вычислительной машиной
-

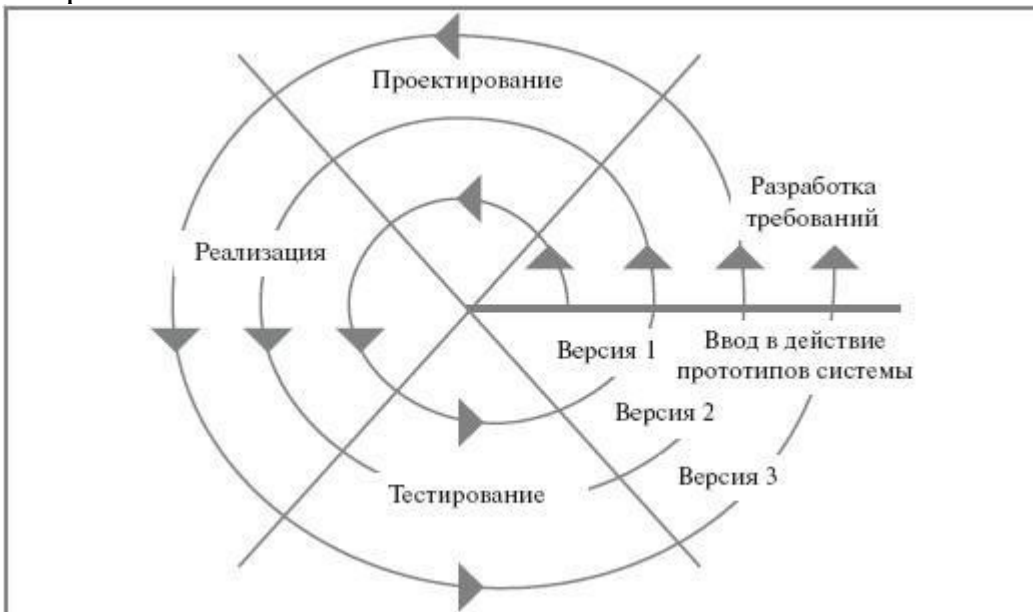
Вопрос 3



Варианты ответов:

- это модель основной характеристикой которой является возможность перехода с одного этапа на следующий только после полного завершения работы на текущем этапе.
- это поэтапная модель с промежуточным контролем.
- это модель в которой на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка.

Вопрос 4



Варианты ответов:

- это поэтапная модель с промежуточным контролем.
- это модель в которой на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка
- это модель основной характеристикой которой является возможность перехода с одного этапа на следующий только после полного завершения работы на текущем этапе.

Вопрос 5

... - это поэтапная модель с промежуточным контролем.

Варианты ответов:

- Каскадная модель
- Инкрементная модель
- Спиральная модель

Вопрос 6

Формы записи алгоритма

Варианты ответов:

- Алгоритмический

- Графический
- Программный
- Словесный
- Циклический

Вопрос 7

Расставьте по порядку этапы жизненного цикла программы

Варианты ответов:

- анализ требований;
- определение спецификаций;
- проектирование;
- кодирование (программирование);
- тестирование и отладка;
- эксплуатация и сопровождение.

Вопрос 8

Ответьте на вопрос

1. Анализ
2. Спецификация
3. Проектирование
4. Кодирование

Варианты ответов

- Заключается в переводе на язык программирования конструкций, записанных на языке проектирования.
- На этом этапе создается общая структура программы, которая должна удовлетворять спецификациям
- На этом этапе необходимо ответить на вопрос: "Что должна делать программа?"
- На этом этапе создается общая структура программы, которая должна удовлетворять спецификациям

Вопрос 9

Тестирование программ

Варианты ответов:

- этап разработки компьютерной программы, в процессе которого проверяется работоспособность программы, не содержащей явных ошибок
- этап разработки компьютерной программы, в процессе которого происходят обнаружение, локализация и устранение явных ошибок в программе
- проверка соответствия функциональности ПО решаемым задачам

Вопрос 10

Отладка программы

Варианты ответов:

- этап разработки компьютерной программы, в процессе которого проверяется работоспособность программы, не содержащей явных ошибок
- этап разработки компьютерной программы, в процессе которого происходят обнаружение, локализация и устранение явных ошибок в программе
- это экспериментальный анализ сложности алгоритма или экспериментальное сравнение нескольких алгоритмов, решающих одну и ту же задачу.

Вопрос 11

Ответьте на вопросы

1. Проектирование
2. Тестирование
3. Отладка

Варианты ответов

- На этом этапе создается общая структура программы, которая должна удовлетворять спецификациям
- этап разработки компьютерной программы, в процессе которого проверяется работоспособность программы, не содержащей явных ошибок
- этап разработки компьютерной программы, в процессе которого происходят обнаружение, локализация и устранение явных ошибок в программе

Тема 1.1.2 Структурное программирование

Работа с простыми типами данных

Задачи этой темы позволяют рассмотреть способы обработки простых типов данных: целых или вещественных чисел. Число при решении предлагаемых задач в этом разделе -- это целое или вещественное со знаком.

1.1.1. Дано натуральное число n . Удалить из записи числа все нули и единицы. Подсчитать количество оставшихся цифр в изображении числа.

1.1.2. Дано произвольное вещественное число. Определить порядок числа и количество цифр в изображении числа.

1.1.3. Дано целое число с количеством цифр в его записи < 10 . Записать число в обратном порядке.

1.1.4. Дано целое число N . Разложить его на простые множители. Вывести на экран исходное число и полученные множители.

1.1.5. Часовая стрелка образует угол j с лучом, проходящим через центр и через точку, соответствующую 12 часам на циферблате: $0 < j < 2\pi$. Определить

значение угла для минутной стрелки для заданного величины j , а также количество полных часов и полных минут.

1.1.6. Дано 10- разрядное число, допустимой для записи целых чисел величины. Определить, является ли введенное число палиндромом. (Число, значение которого при чтении справа налево и слева направо одинаково, называется палиндромом).

1.1.7. Дано некоторое натуральное число. Определить сколько и каких цифр встречается в его записи. Результат обработки: <цифра> - <количество>.

1.1.8. Дано целое число. Найти произведение всех его цифр.

1.1.9. Дано целое натуральное число. Каждую цифру числа заменить на дополнение ее до 10. Вывести на экран исходное число и преобразованное.

1.1.10. Даны два целых числа M и N . Найти все одинаковые цифры в каждом из чисел.

1.1.11. Даны N наборов чисел a, b, c . Определить возможность построения треугольника для каждого из наборов чисел. Результат вывести в форме I а b с результат (можно/ нельзя)

1.1.12. Даны целые положительные числа K и N . Найти $P=(K)N!$. Определить диапазон допустимых N . Учитывать при решении задачи допустимую величину целого числа.

1.1.13. Дано целое число N . Определить число цифр в записи числа и найти сумму цифр числа.

1.1.14. Дано целое число N , определяющее возраст человека. Для этого числа определить наименование: год, года, лет. Например, 31 год, 32 года, 25 лет.

1.1.15. Даны целые числа m, N . Определить сумму любых m цифр числа N .

1.1.16. Дано целое число K , имеющее значения в диапазоне 1..365.

Определить номер дня недели (1..7), на который приходится введенное число для невисокосного года (високосным считается год, который делится на 4, но не 100 и т.п.). Считаем, что первый день года, 1 января, это понедельник.

1.1.17. Определить значение угла в градусах между положением часовой стрелки в начале суток и ее положением в H часов, M минут и S секунд ($0 \leq H \leq 11, 0 \leq M \leq 59, 0 \leq S \leq 59$).

1.1.18. Дано вещественное число. Определить первую цифру дробной части числа.

1.1.19. Дано целое число N . Выделить из числа все четные цифры.

1.1.20. Найти наибольший общий делитель двух натуральных чисел.

ПРИМЕР 1.1.1

Идет N -ая секунда суток. Определить полное количество часов и минут в заданный момент времени, считая от начала суток.

```
program Number_1_1;
```

```
var hour,minute,n:longint;
```

```
{Количество секунд в полных сутках равно 86 400, что требует  
рассмотрения решения задачи только с длинными целыми}
```

```
ch:char;
```

```
Begin
```

```

writeln(' Работа первой программы');
Repeat
writeln('Ведите значение секунд');
readln(n);
hour:=n div 3600;
minute:= n div 60 -hour*60;
writeln('Получено часов= ',hour,' минут= ',minute);
writeln('Еще хотите проверить время?- у or n');
readln(ch);
until ch='n';
End.

```

Вычисление рядов.

В этом разделе задачника необходимо использовать простейшие циклические алгоритмы для вычисления сумм и произведений. Для решения предлагаемых задач надо разработать алгоритмы, обеспечивающие накопление сумм, вычисление факториала натуральных чисел, накопление степеней чисел и вычисление знакопеременных членов рядов ε .

$$1.1.31. y = \sum_{i=1}^n (-1)^{i-1} \cdot x^i / i, \text{ где } |x| < 1$$

$$1.1.32. y = \sum_{k=0}^n (-1)^k / k! \cdot (x/2)^{(n-k)}, \text{ где } |x| < 1$$

$$1.1.33. y = \sum_{i=1}^n (-x)^{2i} / 2i!$$

$$1.1.34. y = \sum_{i=1}^n (-1) \cdot x^{2i} / i!, \text{ где } x > 1$$

$$1.1.35. y = \sum_{i=1}^n (-1)^i \cdot x^i / (i+1)$$

$$1.1.36. y = \sum_{k=1}^n x^{2k} / (2^k \cdot k!)$$

$$1.1.37. y = \sum_{i=1}^n 2x^i / i, \text{ где } |x| < 1$$

$$1.1.38. y = \sum_{k=1}^n (-1)^k \cdot x^k / (k \cdot 2^k)$$

$$1.1.39. y = 2 \sum_{i=1}^n x^{2i+1} / (2i+1)$$

$$1.1.40. y = \sum_{k=1}^n (-1)^k \cdot (k+1) / (2k+1)!$$

$$1.1.41. y = \sum_{i=1}^n (-1)^i / (2i+1) \cdot i, \text{ где } |x| < 1$$

$$1.1.42. y = \sum_{k=1}^n k! / (1/2 + 1/3 + \dots + 1/(k+1))$$

$$1.1.43. y = \sum_{i=1}^n (-1)^{i+1} / \ln^i(x+i) \cdot i!$$

$$1.1.44. y = \sum_{i=1}^n (-1)^i \cdot 2^i / k!$$

$$1.1.45. y = \sum_{k=1}^n k! / \sum_{j=1}^k (x^j / (j+1))$$

$$1.1.46. y = \sum_{k=1}^n (-1)^k \cdot (k+1)! / x^k$$

$$1.1.47. y = \sum_{k=1}^n (-1)^k \cdot (k+1) / k!$$

$$1.1.48. y = \sum_{k=1}^{\infty} (-1)^{k+1} \cdot 2^k / (k \cdot (k+1))$$

$$1.1.49. y = \sum_{k=1}^{\infty} (-1)^k \cdot (k+1) / (2k+1)!!$$

$$1.1.50. y = \sum_{j=1}^{\infty} (x^j + \prod_{j=1}^j ((-1)^j + j))$$

ПРИМЕР 1.1.2.

Дан ряд $s = \sum_{i=1}^{\infty} (-1)^i X^i / i!$. Найти сумму ряда для любого заданного количества членов ряда n и значения X.

```
program Number_1_2;
{Этот пример показывает простейшую алгоритмизацию}
Begin
writeln('Работа второй программы');
Repeat
writeln('Ведите количество членов ряда');
readln(n);
writeln('Введите значение x');
readln(x);
s:=0;
p:=1;
for i:=1 to n do
Begin
p:=-p*x/i;
s:=s+p;
end;
writeln('Результат вычисления ряда S=',s:10:5);
writeln('Будете еще вычислять ряд?');
readln(ch);
until ch='n';
End.
```

ПРИМЕР 1.2.1

```
program Number_2;
{Дана строка, в которой слова}
Begin
writeln('Работает программа- пример на использование строк');
Repeat
writeln('Введите исходную строку');
readln(s);
writeln('Исходная строка = ',s);
k:=length(s);{Определяем длину введенной строки}
i:=1;
while (i<=k-1) do
Begin
if s[i]=' ' then
Begin
Repeat
if s[i+1]=' ' then
Begin
delete(s,i,1);
k:=k-1;
end;
end;
end;
end;
```

```

b:=false;
End
else b:=true;
until b;
end;
i:=i+1;
end;
writeln('Результат преобразования = ',s);
writeln('Будете еще преобразовывать строки? ');
readln(ch);
until ch='n';
End.

```

Задачи для освоения правил работы с массивами данных типа последовательностей, матриц, таблиц. Работа с подобными типами данных зачастую требует выполнения одинаковых или подобных операций, для описания которых целесообразно использовать процедуры и функции.

- 1.3.1. Дана последовательность из букв латинского алфавита размерности N. Упорядочить последовательность по алфавиту с помощью процедуры. Вывести исходную и упорядоченную последовательности букв.
- 1.3.2. Ввести произвольную последовательность чисел. Упорядочить с помощью процедуры последовательность в порядке возрастания. Вывести результат упорядочивания и номера чисел в исходной последовательности.
- 1.3.3. Ввести последовательность слов, в каждом из которых от 2 до 6 латинских букв. Слова разделены запятыми. Последнее слово заканчивается точкой. Написать процедуру, определяющую одинаковых “соседей” (слова справа и слева — совпадают). Напечатать всех “соседей” и исходную последовательность слов.
- 1.3.4. Ввести последовательность из N элементов. Подсчитать количество элементов, равных 0. С помощью процедуры обеспечить удаление всех нулевых элементов последовательности и сжатие последовательности. Вывести исходную последовательность, количество нулей, результат сжатия последовательности и количество элементов в преобразованной последовательности.
- 1.3.5. Ввести неупорядоченную последовательность из 20 элементов. Написать процедуры упорядочивания последовательности и вставки K новых элементов на правильное место в упорядоченную последовательность. Вывести результаты всех обработок последовательности.
- 1.3.6. Ввести последовательность из N целых чисел. Упорядочить по возрастанию четные (по значению, а не по номеру) элементы последовательности. Нечетные элементы должны остаться на своих местах. Упорядочивание описывается процедурой.
- 1.3.7. Ввести последовательность из N целых чисел. Выбрать из последовательности нулевые значения и составить последовательность из их

номеров. Выбор элементов последовательности описывается процедурой. Вывести исходную последовательность и результат обработки последовательности.

1.3.8. Вводится последовательность из чисел, определяющих координаты вершин треугольников. Выбрать из этих значений тот треугольник, который будет иметь наибольшую площадь. Вычисление площади выполнить функцией. Вывести исходную последовательность и результат поиска.

1.3.9. Вводится произвольный текст, содержащий не более 50 слов. С помощью функции выбрать из текста те слова, в которых первая буква слова встречается еще хотя бы один раз. Вывести исходный текст и результат его обработки.

1.3.10. Вводится последовательность из N элементов. С помощью функции подсчитать количества положительных, отрицательных и нулевых элементов последовательности. Вывести исходную последовательность и результаты ее обработки.

1.3.11. Дана матрица действительных чисел размерности $M \times N$. С помощью процедуры обеспечить получение сумм элементов матрицы по столбцам, строкам и диагоналям, параллельным главной диагонали. Вывести исходную матрицу и результаты вычислений в соответствующих строках и столбцах.

1.3.12. Задана последовательность из слов, в каждом из которых не более 7 букв. С помощью процедуры обеспечить перестановку слов в обратном порядке. Вывести исходную последовательность и результат преобразования.

1.3.13. Задана последовательность из N слов. С помощью функции обеспечить выбор из последовательности всех одинаковых слов. Оставить в исходной последовательности одинаковые слова по одному разу. Вывести исходную последовательность, последовательность одинаковых слов и результат преобразования последовательности.

Задача 2.1. Расчет по формуле

Написать программу, которая переводит температуру в градусах по Фаренгейту в градусы Цельсия по формуле $C = 5/9 (F - 32)$, где C — температура по Цельсию, F — температура по Фаренгейту

Листинг 2.1. Вычисление по формуле — классы ввода-вывода

```
#include // 1 using namespace std; // 2
int main() { // 3
double fahr, cels; // 4
cout << endl << "Введите температуру по Фаренгейту" << endl; // 5
cin >> fahr; // 6
cels = 5 / 9 * ( fahr - 32 ); // 7
cout << "По Фаренгейту: " << fahr << ", по Цельсию: " << cels << endl; // 8
return 0; // 9
```

Тема 1.1.3 Объектно-ориентированное программирование

1. Создать класс для выполнения арифметических действий с дробями. Используйте целые переменные для представления закрытых данных класса — числителя и знаменателя. Для

задания начальных значений параметров использовать конструктор. Создать методы:- сложения чисел;- вычитания чисел;- умножения чисел;- деления чисел;- вывод на экран результатов операций в виде a/b , где a - числитель, b - знаменатель. Исходные данные и результаты расчетов хранить:- в текстовых файлах;- в бинарных файлах.

2. Создать класс для обработки комплексных чисел, полями которого будет являться действительные и мнимые части числа, а методами – сложение, вычитание, умножение, деление двух комплексных чисел, а также нахождение модуля комплексного числа и вывода результатов на экран. Исходные данные и результаты расчетов хранить:- в текстовых файлах;- в бинарных файлах.

3. Создать класс для обработки квадратных матриц. Методами этого класса должны быть:- сложение матриц;- умножение матриц;- нахождение нормы матрицы ($\sqrt{\sum_{i,j} a_{ij}^2}$);- вывод результатов расчетов на экран. Исходные данные и результаты расчетов хранить:- в текстовых файлах;- в бинарных файлах. Дополнительные массивы не использовать. Все операции проводить в файле.

4. Создать класс для обработки полиномов. Степенной полином представить односвязным списком ненулевых коэффициентов. Элемент списка должен содержать показатель степени и само значение коэффициента. Методами этого класса должны быть:- вычисление значения полинома при заданном x ;- вычисление первой производной полинома при заданном значении x ;- вывод результатов расчета на экран. Исходные данные и результаты расчетов хранить:- в текстовых файлах;- в бинарных файлах.

5. Создать класс для обработки одномерных массивов. Методами этого класса должны быть:- добавление элемента в конец массива;- добавление элемента vi - тую позицию;- удаление элемента из i - той позиции;- вывод результата на экран;- сложение двух массивов;- умножение двух массивов. Исходные данные и результаты расчетов хранить:- в текстовых файлах;- в бинарных файлах. Дополнительные массивы не использовать. Все операции проводить в файле.

6. Создать класс для обработки текста. Методами этого класса должны быть: - изменение регистра; - подсчет количества слов в тексте; - подсчет количества символов в каждом слове. Исходные данные и результаты расчетов хранить: - в текстовых файлах; - в бинарных файлах. Дополнительные массивы не использовать. Все операции проводить в файле.

7. Создать класс для обработки структур вида: фамилия, должность, оклад. Данные хранить и обрабатывать: - в текстовом файле; - в бинарном файле. Методами данного класса должны быть: - удаление элемента структуры из файла; - перестановка заданных элементов структуры в файле.

8. Создать класс для обработки структур вида: номер, название прибора, стоимость. Данные хранить и обрабатывать:- в текстовом файле;- в бинарном файле. Методами данного класса должны быть: - вставка элемента структуры в нужную позицию файла; - сортировка элементов структуры по заданному полю.

9. Разработать программу на основе классов для работы с цепными списками строк (строки произвольной длины) операциями включения в список, вывод строки произвольной длины на экран, удаления из списка элемента с заданным значением данного, удаления всего списка или конца списка, начиная с заданного элемента.

10. Разработать программу на основе классов для определения одномерных массивов строк фиксированной длины. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы индексов, выполнение операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, вывод на экран элементов массива.

11. Разработать программу на основе классов для работы с одномерными массивами целых чисел (векторов). Предусмотреть возможность обращения к отдельному элементу массива с контролем выхода за пределы индексов, возможность задания произвольных границ индексов при создании объекта и выполнения операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов, умножения и деления всех элементов массива на скаляр, вывод элементов массива на экран.

12. Разработать программу с использованием классов для обработки одномерных массивов строк. Каждая строка задается длиной и указателем на выделенную для нее область памяти. Предусмотреть возможность обращения к отдельным элементам массива по индексам, выполнение операций поэлементного сцепления двух массивов, слияния двух массивов с исключением повторяющихся элементов, вывод элементов массива на экран.

13. Разработать программу на основе классов для определения одномерных массивов строк фиксированной длины. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы индексов, выполнение операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, вывод на экран элементов массива.

14. Разработать программу с использованием классов для обработки одномерных массивов строк. Каждая строка задается длиной и указателем на выделенную для нее область памяти. Предусмотреть возможность обращения к отдельным элементам массива по индексам, выполнение операций поэлементного сцепления двух массивов, слияния двух массивов с исключением повторяющихся элементов, вывод элементов массива на экран.

15. Разработать программу с использованием классов для описания многочленов, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения и вычитания многочленов, вывод на экран описания многочлена.

16. Разработать объект-вектор. Даны две последовательности A и B. Последовательности хранятся в файлах. Количество элементов в последовательностях неизвестно. Для обработки последовательностей описать методы: шейкер-сортировки, создания последовательности из несовпадающих элементов каждой из последовательностей, ввода и вывода всех последовательностей. Вывод результирующей последовательности сопровождать выводом номеров элементов в исходных последовательностях. Сортировку сравнить по скорости с методом простых перестановок.

17. Разработать объект-вектор. Дана последовательность A. Последовательность хранится в файле. Количество элементов последовательности неизвестно. Для обработки последовательности описать методы: сортировки простым вставками, определения сумм отрицательных и положительных элементов, суммы всех элементов, а также индексов

только положительных и только отрицательных элементов. Сортировку сравнить по скорости с методом простых перестановок.

18. Разработать объект-вектор. Даны последовательности A и B . Последовательности хранятся в файлах. Количество элементов в последовательностях неизвестно. Для обработки последовательностей описать методы: сортировки простым включением, создания последовательности из элементов, не входящих в одну из последовательностей, но входящих в другую, ввода и вывода всех последовательностей. Сортировку сравнить по скорости с методом пузырька.

19. Разработать объект-вектор. Даны последовательности A и B . Последовательности хранятся в файлах. Количество элементов в последовательностях неизвестно. Для обработки последовательностей описать методы: быстрой сортировки, создания последовательности из элементов, входящих в каждую из последовательностей только по одному разу, ввода и вывода всех последовательностей. Сортировку сравнить по скорости с методом простых перестановок.

20. Разработать объект-вектор. Даны последовательности A и B . Последовательности хранятся в файлах. Количество элементов в последовательностях неизвестно. Для обработки последовательностей описать методы: сортировки простым выбором, создания последовательности из 2-х исходных последовательностей, ввода и вывода всех последовательностей. Вывод результирующей последовательности сопровождать выводом номеров элементов в исходных последовательностях. Сортировку сравнить по скорости с методом простых перестановок.

21. Разработать объект-вектор. Дана последовательность A . Последовательность хранится в файле. Количество элементов в последовательности неизвестно. Для обработки последовательностей описать методы: сортировки простым включением, создания последовательности, в которой сначала стоят все положительные элементы, а затем все отрицательные, ввода и вывода всех последовательностей. Вывод результирующей последовательности сопровождать выводом номеров элементов в исходной последовательностях. Сортировку сравнить по скорости с методом простых перестановок.

22. Разработать объект-вектор. Дана последовательность A . Последовательность хранится в файле. Количество элементов в последовательности неизвестно. Для обработки последовательностей описать методы: сортировки простым выбором, обработки последовательности, в форме --,...--, ввода и вывода всех последовательностей. Сортировку сравнить по скорости с методом простых перестановок.

23. Разработать объект-вектор. Даны последовательности A . Последовательность хранится в файле. Количество элементов в последовательности неизвестно. Для обработки последовательностей описать методы: сортировки простым включением, создания последовательности, в которую входят только те элементы последовательности, которые стоят между максимальным и минимальным элементами неупорядоченной последовательности, ввода и вывода всех последовательностей. Вывод результирующей последовательности сопровождать выводом номеров элементов в исходной последовательностях. Сортировку сравнить по скорости с методом пузырька.

24. Разработать объект-вектор. В файле хранятся сведения о жителях некоторой улицы: фамилия, адрес, год рождения. Все сведения записываются в памяти в виде последовательности. Вводится некоторая текущая дата. В последовательности найти всех жителей, имеющих право участвовать в выборах (возраст ≥ 18). Упорядочить последовательность методом простого выбора по фамилиям жителей. Полученные последовательности вывести на экран.

25. Разработать объект-вектор. В файле хранятся сведения об абонентах телефонной сети: фамилия, адрес, номер телефона. Записать все сведения в памяти в виде последовательности. Все записи в файле не упорядочены. Упорядочить сведения по фамилиям абонентов методом быстрой сортировки. Удалить из сведений ушедшего абонента, внести в последовательность данные о новом абоненте, изменить данные об абоненте. Вывести все сведения на экран.

26. Разработать объект-вектор. Данные о фондах библиотеки хранятся в файле: автор книги, название книги, шифр издания (тоже строка), год издания, количество книг данного названия. Из всех сведений создать последовательность из книг одного автора. Упорядочить последовательность по фамилиям авторов методом простого включения. Обеспечить внесение в полученную последовательность новых сведений. Вывод на экран всех результатов обработки обязателен.

27. Вводится произвольная последовательность целых чисел и записывается в памяти в виде динамической последовательности. Разработать объект-вектор, обеспечивающий формирование последовательности, удаление из последовательности первого отрицательного числа, всех отрицательных чисел, любого заданного по номеру числа, определения длины результирующей последовательности, сохранение последовательности в файле, вывод последовательности и всех ее модификаций на экран.

28. Имеется N населенных пунктов. Все пункты пронумерованы и имеют свои наименования. Некоторые из пунктов соединены попарно дорогами. Определить можно ли попасть по этим дорогам из любого K -ого пункта в пункт M . Информация о дорогах задается в виде последовательности пар чисел i, j , указывающих номера соединенных дорог парами пунктов i, j . Если пункты не соединены, тогда в паре i, j номера равны 0. Признак конца последовательности, - это пара нулей в разделе признаков соединения. Разработать объект, обеспечивающий формирование исходной последовательности, сохранение всех сведений в файле, обработка сведений о возможных траекториях движения, вывод последовательностей полученных траекторий движения по возможным дорогам. Обработка последовательности по всем заданиям обеспечивается через текстовое меню.

29. Дана действительная матрица размерности $M \times N$. С помощью алгоритма сортировки выбором обеспечить выполнение методов обработки строк матрицы: упорядочить строки матрицы по убыванию значений первых элементов каждой строки, по возрастанию сумм элементов строк, по возрастанию значений наименьших элементов строк, по убыванию значений наибольших элементов строк, создание и сохранение исходной матрицы в файле, вывод всех вариантов преобразования. Объект, содержащий все названные методы обработки последовательностей строк, должен работать под управлением текстового меню.

30. В файлах создаются две последовательности A и B . Последовательности произвольные из действительных чисел. Выполнить сортировки последовательностей алгоритмом деления пополам: взять сначала 1 и $\lfloor 1/2 \rfloor$ в качестве границ места очередного элемента X последовательности, Далее сдвигать эти границы следующим образом: 1 и K -ый

номера выбираются, где K равно целой части суммы индексов границ последовательности, если $A_k < X$. В качестве нижней границы берется K , а верхняя граница не меняется. При противоположном соотношении сортируемых чисел без изменения остается нижняя граница. Когда границы совпадут, тогда сортировка завершается. Получить последовательность натуральных чисел $K, K+1, \dots, K+m$, где K_i - это номер места элемента последовательности. В прислании этой последовательности с упорядоченной последовательностью A без нарушения признака упорядоченности. Методы объекта должны также обеспечить вывод всех вариантов последовательностей и выполнение всех пунктов задания через текстовое меню.

31. Разработать объект для обработки результатов некоторой лотереи. В качестве первого из результатов розыгрыша лотереи имеется последовательность выигравших шестизначных номеров билетов. Для каждого из выигравших билетов есть последовательность сумм выигрышей. Количество выигрышей M . Выигрыши - это последовательность $P_1, P_2, \dots, P_M$, где P_i - это выигрыш i -го билета. Определить суммарный выигрыш, выпавший на любые билеты из массива выигравших билетов. Обеспечить сохранение данных розыгрышей в файлах и вывод сведений на экран. Обработка всех сведений по розыгрышу выполняется через меню.

32. В файле задать произвольную последовательность целых чисел. Создать упорядоченную последовательность из неповторяющихся чисел исходной последовательности. Элементы, которые уже попали в результирующую последовательность не надо пытаться вставить еще раз в последовательность. Определить количество сравнений, необходимых для решения этой задачи алгоритмом быстрой сортировки. Обеспечить в методах объекта вывод исходной последовательности, преобразованной последовательности и тех элементов, которые были исключены из последовательности в результате преобразования. Все преобразования выполнять через текстовое меню.

33. Создать файл из слов различной длины. Считаем, что слова не имеют более 8 букв в длину. Между словами может быть произвольное число пробелов. Разработать объект, в котором последовательность слов упорядочивается следующим образом: сначала по алфавиту однобуквенные слова, затем двухбуквенные и т.д. В каждой последовательности одинаковые слова записывать по одному разу. Обеспечить вывод всех видов последовательностей, а также тех слов, которые не вошли в результирующую последовательность. Обработку последовательностей выполнять через меню.

34. Дана последовательность несовпадающих между собой натуральных чисел. Длина последовательности не более 100. Элементы X_i и X_j последовательности считаются взаимосвязанными, если значение элемента $X_j = X_i$. Последовательность связанных элементов образует цепь. Описать методы объекта для определения цепей в последовательности элементов, создание последовательностей типа цепи, определение суммы элементов в каждой из цепей, минимальную и максимальные длины цепей. Все исходные данные должны быть записаны в файл. Обработка последовательности выполняется через меню.

35. Создать данные для обработки сведений о жителях некоторой улицы. Определить список лиц, которые могут голосовать (≥ 18). Разработать объект-список с операциями по обработке списков жителей. Перечень операций: удалить из списка, вставить в список нового жителя, напечатать по запросу полный список жителей, сохранить список в файле, прочитать список из файла, создать список избирателей, сохранить в файле список избирателей,

редактировать список дополнением сведений, исправлением их или удалением устаревших сведений. Выполнение всех видов работы обеспечить через текстовое меню.

36. Разработать объект список с методами обработки сведений о жителях некоторого района. В списке указано: фамилия, имя, отчество, пол, дата рождения, адрес. Выделить в новый список адреса, по которым живут люди пенсионного возраста с учетом различий в определении пенсионного возраста для мужчин и женщин, по введенной дате опроса. Обеспечить обработку данных по созданным спискам: Формирование общего списка жителей. 11 Создание подмножества списка пенсионеров. Создание полного списка жителей и сохранение его в файле. Вывод всех модификаций списков. Выполнение всех видов работ обеспечить через текстовое меню.

37. Разработать объект список с методами обработки сведений о жителях некоторого района. В списке указано: фамилия, имя, отчество, пол, дата рождения, адрес, место работы. Выделить в новый список адреса, по которым живут люди старше 20 лет, по введенной дате опроса. Обеспечить обработку данных по спискам: Формирование общего списка жителей и сохранение этого списка в файле. Формирование списка работающих. Удаление из списка жителя. Запись в список нового жителя. Сортировка списка по адресам. Сохранение сформированного списка в файле. Вывод всех типов списков на экран по соответствующему запросу пользователя. Выполнение всех видов работы обеспечить через текстовое меню.

38. Разработать объект список для обработки сведений об абонентах телефонной сети. Обеспечить следующие виды работ со списком: Формирование списка абонентов в формате: фамилия, адрес, номер телефона. Сохранить созданный список в файле. Вывод всех типов запрашиваемых списков на экран. Например, вывести абонентов на букву "Я" или "А" и т.д. Упорядочить список по фамилиям абонентов. Ввести в список нового абонента. Изменить данные об абоненте. Удалить абонента из списка. Все изменения данных должны сохраняться в файле. Выполнение всех видов работы обеспечить через текстовое меню.

39. Имеется упорядоченный список рабочих одного цеха. Список этот введен и сохранен в некотором файле. Известно, что каждый из рабочих должен отработать 42 часа в неделю. Каждый день рабочему разрешается работать не более 10 часов. Обязательные часы присутствия на работе с 10 до 16 часов. Работа выполняется только в одну смену (с 8 утра до 8 вечера). Разработать объект список, обеспечивающий учет отработанного времени для каждого рабочего. Данные о каждом рабочем вводятся в следующей форме: фамилия И.О., день недели, время прихода, время ухода. Обеспечить обработку списка по следующим сведениям: Ввод сведений о каждом рабочем. Сортировка сведений по алфавиту. Запись списка рабочих в файл. Вывод результатов обработки для каждого рабочего и для всего списка. Исправление данных по каждому из рабочих. Выполнение всех видов работы обеспечить через текстовое меню.

40. Создать объект список для описания структуры данных о библиотеке и методов обработки этих данных. В сведениях указываются: номер раздела, количество записей в каждом разделе (если количество записей равно нулю, то раздел пуст). 13 № Количество записей. Название раздела N КШифр. В каждом разделе под шифром этого раздела содержатся сведения обо всех книгах раздела по формату: шифр, автор (фамилия, имя, отчество), наименование книги, год издания, место издания. Это также список. Обеспечить следующие методы по обработке данных о библиотеке: 1. Ввод данных о библиотеке. 2. Сохранение данных в файле. 3. Поиск автора по вариантам запросов: а) по шифру; б) по фамилии автора. 4. Вывод авторов каждого из разделов. 5. Ввод нового автора в свой раздел. 6. Удаление утерянной книги. 7. Сохранение всех изменений в файле. Выполнение всех видов работы обеспечить через текстовое меню.

41. Задан некоторый произвольный текст. Создать объект-список для обработки слов, входящих в текст. Формат представления результатов обработки: слово, количество повторений слова в тексте, номер строки текста, в которой слово находится в тексте. Описать следующие методы обработки текста: 1. Ввод текста и сохранение его в файл. 2. Выбор слов из текста, создание списка отдельных слов текста. 3. Упорядочивание списка слов по частоте появления в тексте. 4. Внесение исправлений в текст и сохранение исправленного текста в файл. 145. Запись словаря из найденных слов в файл (каждое из найденных слов в словаре встречается только один раз). 6. Вывод списка слов и текста на экран. Выполнение всех видов работы обеспечить через текстовое меню.

42. По каждой учебной группе составляется ведомость успеваемости из показателей: 2 - успевающие по предмету в соответствии с графиком занятий; 1 - имеющие текущие задолженности по предмету; 0 - неуспевающие по предмету. Полный формат сведений может быть таким: Фамилия студента, группа, пять предметов с показателями успеваемости данного студента по этим предметам. Разработать объект-список из сведений об учебной группе, обеспечивающий методы обработки ведомости по показателям: список успевающих на 2; список успевающих на 1, 2; список успевающих на 0, 1, 2; список успевающих на 0. Обеспечить следующие виды работ со списком: Ввод списка учащихся. Исправление сведений. Сортировка списка по показателям успеваемости. Запись списка в файл. Вывод результатов обработки ведомости успеваемости. Выполнение всех видов работы обеспечить через текстовое меню.

43. Некоторая бригада из рабочих работает по методу бригадного подряда. Известно: список всех работ, объем каждого вида работ, расценки за единицу объема работ. Создать объект-список для обработки сведений о членах бригады с указанием объема выполненных работ. Определить коэффициент трудового участия каждого рабочего в бригаде и начисленную ему зарплату. Обеспечить следующие виды работ со списком: Ввод списка бригады с показателями участия в работе. Сохранение списка в файл. Вывод по фамильному списку с указанием начисленной зарплаты. Сортировка списка по коэффициенту участия в работе. Запись отсортированного списка в файл. Внесение исправлений в данные о работе каждого члена бригады. Вывод по фамильному списку и списка с выполненными работами на экран. Выполнение всех видов работы обеспечить через текстовое меню.

44. В библиотечном коллекторе имеется N наименований книг. Для каждой книги указаны цена одного экземпляра и количество экземпляров каждого наименования книги. Для библиотеки отпущено M рублей для покупки книг. Требуется купить K наименований книг таким образом, чтобы заявка была удовлетворена по количеству приобретаемых разных книг и наиболее полно была использована выделенная сумма. Разработать объект-список для учета всех типов книг -- тех, которые находятся в коллекторе и тех, которые может купить библиотека. Для выполнения задачи обеспечить: Составление каталога книг коллектора. Выбор книг на заданную сумму покупки. Вывод списка выбранных книг. Внесение исправлений в список купленных книг. Запись в файлы списков книг коллектора и списка выбранных книг. Выполнение всех видов работы обеспечить через текстовое меню.

45. Разработать объект-список для учета учащихся по группам. Все сведения об учащихся представлены в следующем формате: 1-ая строка индекс группы начальный номер в списке; 2-ая строка фамилия имя отчество следующий номер n - ая строка фамилия имя отчество следующий номер. Если вместо номера указаны 00, то этот элемент

списка должен быть последним в списке. Обеспечить обработку списка указанного формата: Напечатать список в исходном виде. Напечатать список в правильной последовательности. Упорядочить список по алфавиту (с сохранением исходных номеров элементов списка). Сохранить список в файле. Выполнение всех видов работы обеспечить через текстовое меню.

46. В репертуаре театра спектаклей. В каждом из спектаклей занято К актеров. Вводится список актеров по каждому спектаклю. Разработать объект список для обработки сведений по занятости актеров в спектаклях за каждый календарный месяц. Обеспечить создание исходного списка актеров театра, создание списка занятых актеров в любом количестве названных спектаклей, редактирование компонент списка, вывод сведений по занятости: По спектаклям (фамилия актера, количество) По фамилиям (спектакль, количество). Выполнение всех видов работы обеспечить через текстовое меню. 47. При проведении конференции регистрируют всех участников. Составляется список по сведениям об участниках: фамилия, город, организация. Разработать объект список для обработки сведений об участниках конференции и обеспечить вывод сведений обо всех участниках, об участниках из одного города, об участниках из одной организации. Списки должны быть упорядочены по алфавиту. Выполнение всех видов работы обеспечить через текстовое меню.

48. Составить список абонентов телефонной сети: фамилия, имя, отчество, адрес, номер телефона. Разработать объект список, обеспечивающий поиск абонента: по фамилии (учесть обработку однофамильцев), по номеру телефона. Методы объекта должны обеспечить следующие виды работ: Ввод сведений в файл. Создание списка всех абонентов. Создание списков абонентов по заданной категории номеров телефонов (Диапазон номеров: 100- 199, 200- 299 и т.д.) Обеспечить сортировку любого типа списка абонентов по алфавиту. Выполнение всех видов работы обеспечить через текстовое меню. 49. Сведения об участниках соревнований по художественной гимнастике занести в некоторый файл. Формат сведений: фамилия, страна, баллы по 4 видам соревнований (из 10 баллов). Баллы формируются случайным образом. Файл сведений содержит не известное заранее количество компонент. Разработать объект список. Выполнить группировку участников по следующим сведениям: а) По странам; б) по фамилиям; в) выбрать 3-х лучших участников. Сортировки по всем признакам обеспечить на основании представления списка в виде стека. На вершине стека всегда участник с фамилией или из страны с начальной буквой алфавита, или с наилучшим результатом.

50. Разработать объект список для обработки сведений о жителях города. Для каждого города в файл заносятся сведения о количестве жителей, из них численность трудоспособного населения, число пенсионеров, число безработных. Создать список из сведений в виде очереди и обеспечить выдачу сведений по городам по следующим показателям: по числу жителей, по числу безработных, по числу пенсионеров. Обеспечить исправление любых сведений: удалить жителя любого города, дополнить сведения по любому городу, исправить сведения по любому жителю по его фамилии и адресу.

Тема 1.1.4 Паттерны проектирования

Задача 1 «Быстрый доступ к размеру изображения»

Реализовать класс, позволяющий получать информацию о размере изображения в BMP-файле со следующим предположением. Если имя файла заканчивается на _WWWxNNN, где WWW и NNN – десятичные числа (необязательно состоящие из трёх цифр), то WWW –

ширина, а ННН – высота изображения. В этом случае для получения размера необязательно открывать файл.

vasya_1024x768.bmp – размер 1024x768.

kolya_300x30 – размер 30x30.

petya_200xmax.bmp – неверный формат.

В последнем случае, поскольку формат не соответствует, модуль должен все же открыть ВМР-файл и считать размер явным образом.

Работу с самим ВМР-файлом вынести в отдельный класс (можно воспользоваться классом Bitmap). Создаваемый класс должен инкапсулировать его (отношение композиции).

Задача 2. «Векторный графический редактор» Справка:

Реализовать векторный графический редактор, работающий в пакетном режиме.

На вход редактору поступает два файла. Один файл содержит в себе 2D-сцену в приблизительно следующем формате:

```
rect[1] 10 10 20 20
```

```
rect[2] 50 50 70 70
```

```
line[O_o] 1 1 90 7
```

```
ellipse[cool] 5 5 90 10
```

Первый элемент каждой строки содержит тип объекта (прямоугольник, линия, эллипс) и его идентификатор в квадратных скобках. Идентификатор представляет из себя строку (в которой не содержится символа `]`) и уникален в рамках одного типа объектов. Далее идут координаты `x1 y1 x2 y2` для прямоугольника и линии, `x y w h` для эллипса.

Второй файл содержит команды для редактора:

```
translate rect[1] 50 70
```

```
scale line[O_o] 1 1.5
```

```
rotate 90
```

```
undo
```

Команда `translate` смещает указанный объект (задаются `dx` и `dy`). `scale` изменяет масштаб (задаются коэффициенты масштаба `sx` и `sy`). `rotate` поворачивает объект вокруг его центра на заданное количество градусов (также можно реализовать более сложную версию с поворотом вокруг заданной точки). После имени команды идёт идентификатор объекта. Если он не указан, то операция применяется ко всем объектам.

Команда `undo` отменяет последнюю операцию (у неё нет аргументов).

Векторный редактор должен прочитать файл с исходной сценой (с обработкой ошибок), файл команд (с обработкой ошибок), применить команды к сцене и записать сцену в новый файл.

Задача 3 «Данные о системе обучения»

Реализовать систему хранения информации о системе обучения. Информация опирается на следующие основные классы: Student, Teacher, Course. Эти классы связаны друг с другом посредством агрегации, ассоциации и проч.

Атрибуты классов

student

id — уникальный численный идентификатор.

name — имя.

courses - курсы, которые посещает студент

teacher

id — уникальный численный идентификатор.

experience - стаж в годах

name — имя

courses - курсы, которые ведет преподаватель

course

id — уникальный численный идентификатор.

name - название курса

teacher - id учителя, который ведет курс

students - id студентов, посещающих курс

Обеспечить загрузку и сохранение этой базы данных в текстовый файл формата TSV (Tab-Separated Values). Каждая строка в этом файле должна начинаться со слов student, teacher, course. Далее будут идти необходимые атрибуты.

Для сохранения и восстановления связей между различными объектами использовать поля id.

Например, в строке с записью о курсе можно сохранить teacher_id и students_id, а после загрузки всего файла создать связь с реальным объектом.

Задача 4 «Данные о футбольном чемпионате»

Реализовать систему хранения информации о футбольном чемпионате. Информация опирается на следующие основные классы: Team (команда), Player (игрок), Match (матч). Эти классы связаны друг с другом посредством агрегации, ассоциации и проч.

Атрибуты классов

Team

id — уникальный численный идентификатор.

name — имя.

players — игроки, играющие за данную команду в рамках чемпионата.

Player

id — уникальный численный идентификатор.

name — имя

team — команда.

Match

id — уникальный численный идентификатор.

date — дата.

location — место.

result — счёт.

team1 — первая команда.

team2 — вторая команда.

players — игроки, участвовавшие в матче.

Обеспечить загрузку и сохранение этой базы данных в текстовый файл формата TSV (Tab-Separated Values). Каждая строка в этом файле должна начинаться со слов team, player, match. Далее будут идти необходимые атрибуты.

Для сохранения и восстановления связей между различными объектами использовать поля id.

Например, в строке с записью о матче можно сохранить team1_id и team2_id, а после загрузки всего файла создать связь с реальным объектом.

Задача 5 «Логгер»

Задание: Реализовать логгер.

Описание:

Примерный интерфейс логгера.

```
class logstream {  
    //реализация ...  
public:  
    static void set_log_file (const std :: string & name );  
  
    logstream (void );  
    ~logstream (void );  
    template < typename T >  
    inline logstream & operator << ( const T & rhs ) {
```

```
// ...
return* this;
}
// ...
};
```

set_log_file – статический метод класса, позволяющий назначить файл журнала. Если в момент вызова журнал уже был открыт, то старый файл сохраняется и закрывается, отрывается новый файл, и в дальнейшем, журнал ведётся уже в него.

конструктор – конструируется "легковесный" объект, предоставляющий доступ к одному единственному файловому потоку, ведущему запись в журнал.

деструктор – "легковесный" объект уничтожается, при этом файловый поток, ведущий запись в журнал сбрасывает все буферизованные данные в файл, но сам файл не закрывает и продолжает вести запись.

operator << – перегруженный оператор вывода в поток. При передаче в поток манипулятора endl, происходит принудительный сброс буферизованных данных в файл журнала.

Допускается реализация, в которой файловый поток закрывается в моменты когда нет ни одного "легковесного" logstream. В этом случае, при создании хотя бы одного logstream, файловый поток открывается снова в режиме дозаписи файла.

Пример использования

```
int f(int a) {
    logstream log;
    log << "f(0x" << std::hex << a << ")" << std::endl;
    int r = a * a;
    log << "f() -- return " << r << std::endl;
    return r;
}

void main(void) {
    logstream::set_log_file("test.log");
    logstream log;
    log << "main()" << std::endl;
    f(15);
    log << "main() -- return" << std::endl;
}
```

Задача 6 «Подсветка синтаксиса C++ в HTML»Задание

Используя паттерн декоратор, реализовать набор классов, позволяющий разнообразным образом переводить C++ код в HTML страницы: с подсветкой КЛЮЧЕВЫХ СЛОВ или ТОКЕНОВ, расстановкой номеров строк и пр. Описание:

1. Реализовать класс, который принимает ссылку на текст в конструкторе, и имеет функцию write(ostream & out), которая записывает этот текст в выходной поток.

2. Реализовать несколько классов-декораторов над вышеописанным классом HTMLDecorator – простое преобразование исходного кода в HTML файл. Для этого достаточно добавить открывающие и закрывающие тэги в начало и конец файла, а так же заменить спецсимволы: одинарные и двойные кавычки, знаки больше меньше и знак амперсанда.

Задача 7 «Решение численных уравнений»Задача

Используя паттерн шаблонный метод написать систему классов, которая позволит применять различные сходящиеся итерационные алгоритмы, которые все имеют схожий алгоритм:

Проверить некоторое условие с некоторой точностью. Если верно - вернуть текущий результат X_k

Если нет - вычислить следующий член последовательности предполагаемых результатов $X_{k+1} = g(X_k)$ и т.д.

В этой задаче необходимо численно решить одномерное уравнение $f(x)=0$ на интервале $[a, b]$ (в предположении, что на заданном интервале есть только один корень). Приблизительно так. Хочется уметь это делать разными методами:

Метод хорд.

Метод Ньютона.

Необходимо реализовать эти два алгоритма, используя паттерн проектирования «Шаблонный метод».

Обратите внимание, что метод Ньютона кроме основной функции f требует задания её производной f' . Учтите этот факт при разработке интерфейса классов.

Задача 8 «Численное решение уравнения $f(x)=0$ »

Хорошо известны методы численного решения одномерного уравнения $f(x)=0$ на интервале $[a, b]$ (в предположении, что на заданном интервале есть только один корень). Это:

Метод бисекции (деление отрезка пополам).

Метод хорд.

Метод Ньютона.

Необходимо реализовать эти три алгоритма, используя паттерн проектирования «Стратегия», чтобы они были взаимозаменяемы.

Обратите внимание, что метод Ньютона кроме основной функции f требует задания её производной f' . Учтите этот факт при разработке интерфейса классов.

Задача 9 «Режим шифрования»Задание

Реализовать несколько режимов шифрования для произвольного блочного шифра (например простого XOR-шифра), используя паттерн стратегия.

Описание

Under construction

Задача 10 «Обнаружение утечек памяти»Задание

Реализовать обнаружение утечек памяти с C++ программах.

Описание

Простейший способ обнаружения утечки памяти с помощью функции `_DumpMemoryLeaks` в Visual Studio.

```
// main.cpp
void main(void) {
// код
_CrtDumpMemoryLeaks(); // здесь вся выделенная память
должна быть освобождена
}
```

Объект класса `MemoryDump` должен быть создан в самом начале `cpp` файла, так чтобы он, среди других глобальных и статических объектов всегда создавался первым. Тогда при деинициализации гарантируется обратный порядок вызова деструкторов, и поэтому `_CrtDumpMemoryLeaks()` в деструкторе будет вызвана уже после вызова деструкторов у всех остальных глобальных объектов.

Второе приближение

Если в проекте присутствует несколько `cpp` файлов, то потребуется создать несколько таких глобальных объектов (т.к. порядок инициализации / деинициализации глобальных объектов в разных модулях не определён и может быть любым). Но при этом требуется чтобы `_CrtDumpMemoryLeaks()` вызывалась только в последнем вызове деструктора.

Для этого потребуется преобразовать наш класс в соответствии с паттерном одиночка: создание и уничтожение объекта происходит множество раз, но реально существует всегда только один экземпляр объекта, который один раз создаётся (первый вызов конструктора) и только один раз уничтожается (последний вызов деструктора).

Результат

Необходимо оформить класс и всю необходимую логику в виде отдельного модуля, а также сформулировать дополнительные правила использования этого модуля (если требуется).

Приведённый выше код можно использовать в качестве тестовых примеров для проверки работы модуля.

Тема 1.1.5.Событийно-управляемое программирование

1. Разработка приложения с использованием текстовых компонентов

Цель работы: изучить правила работы текстовыми компонентами

Теоретический материал

В языках C и C++ файл рассматривается как поток (stream), представляющий собой последовательность считываемых или записываемых байтов. При этом последовательность записи определяется самой программой.

C++ Builder позволяет работать с файлами тремя различными способами:

Работа с текстовыми компонентами

Каждый файл в программе на С++ должен быть связан с некоторым указателем. Этот указатель имеет тип FILE (определен в stdio.h) и используется во всех операциях с файлами. Синтаксис операции следующий:

```
# include < stdio.h >
```

```
FILE *fin, *fout;
```

Для работы с файлом его необходимо открыть функцией **fopen**, первый параметр которой содержит имя файла и путь к нему, а второй - режим открытия файла. Функция возвращает указатель на файл. Часто используемые режимы открытия файла:

г	Открывает файл для чтения

```
fout=fopen("a:\\out.txt","w");
```

При записи обмен происходит не непосредственно с файлом, а с некоторым буфером. Информация из буфера переписывается в файл только при его переполнении или при закрытии файла.

После окончания работы с файлом он обязательно должен быть закрыт функцией **fclose(FILE *)**. Если файл не удалось открыть, то возвращается нулевой указатель (NULL). Например: # include < stdio.h >

```
FILE *lw;
```

```
if (lw=fopen("a.text","r")) == NULL){
```

```
Memo1->Lines->Add("Файл не открыт");
```

```
return; } fclose(lw);
```

Работа с текстовыми файлами

Для записи в текстовый файл наиболее часто используется функция **fprintf**:
int *fprintf(FILE *stream, const char *format[]); где параметр format определяет строку форматирования аргументов, заданных своими адресами. Обычно эта строка состоит из последовательности символов "%", после которых следует символ типа

```
char *fgets(char *st, int n, FILE *stream);
```

где st - указатель на буфер, в который считывается строка; n - число читаемых символов; stream - указатель на файл. Строка читается до тех пор, пока не будет прочитано n-1 символов, или до конца строки \n. В конце прочитанной строки ставится нулевой символ.

Для проверки достижения конца файла используется функция feof(F).

Чтение форматированных данных можно осуществлять с помощью функции fscanf: int *fscanf(FILE *stream, const char *format[]); Строка форматирования строится аналогично fprintf.

Следует обратить внимание на то, что при чтении данных всегда указываются адреса переменных (&), а не сами переменные.

Пример записи и чтения данных из файла:

```
# include < stdio.h >
Memo1->Clear(); FILE *lw;
// Запись данных в файл
if ((lw=fopen("a.text","wf")) == NULL) {
Memo1->Lines->Add("Файл не удалось
создать"); return; }
int num=10; char st[12] = "ИНФОРМАЦИЯ",sr[30];
fprintf(lw,"%s \n В группе %i человек",st, num);
fclose(lw);
// Чтение данных из файла
if ((lw=fopen("a.text","rt")) == NULL) {
Memo1->Lines->Add("Файл не удалось открыть
"); return; return; }
while (!feof(lw)) { fgets(sr,30,lw);
if (sr[strlen(sr)-1] == '\n') sr[strlen(sr)-1]=0; Memo1-
>Lines->Add(sr); } fclose(lw);
```

В C++ определены три класса файлового ввода/вывода: **ifstream** - входные данные для чтения; **ofstream** - выходные файлы для записи; **fstream** - файлы для чтения и записи.

Очень удобно применять следующие операции: поместить в поток (<<) и извлечь из потока (>>). Пример программы:

```
#include <fstream.h> Memo1-
>Clear();
// Ввод данных в файл
ofstream lw ("a.text");
if (!lw) {Memo1->Lines->Add("Файл не удалось
создать "); return;} int num=10; double k=5.67; char
s[20] = "ИНФОРМАЦИЯ"; lw << num << " <<k << "
<< s << endl; lw.close();
// Чтение данных из файла
ifstream lx ("a.text");
if (!lx){ Memo1->Lines->Add("Файл не удалось открыть ");return;}
lx >> num >> k >> s;
Memo1->Lines->Add(ToIntStr(num));
Memo1->Lines->Add(FloatToStr(k)); Memo1-
>Lines->Add(s); lx.close();
```

Помимо этих операции поместить в поток можно еще с помощью функций put и write. Возможности ввода/вывода можно существенно расширить, используя манипуляторы потока.

Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент работы программы. Поэтому их можно разместить в любом удобном месте формы. Оба рассматриваемых компонента имеют идентичные свойства и различаются только внешним видом. После вызова компонента появляется диалоговое окно, с помощью которого выбирается имя программы и путь к ней. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержатся в свойстве FileName. Для фильтрации файлов, отображаемых в окне просмотра, используется свойство Filter, а для задания расширения файла, в случае, если оно не задано пользователем, - свойство DefaultExt. Если необходимо изменить заголовок диалогового окна, используется свойство Title.

Для установки компонентов TOpenDialog и TSaveDialog на форму необходимо на странице Dialogs меню компонентов щелкнуть мышью по пиктограмме и поставить её в любое свободное место формы.

Установка фильтра производится следующим образом. Выбрав соответствующий компонент, дважды щелкнуть по правой части свойства Filter инспектора объектов. Появится окно Filter Editor, в левой части которого записывается текст, характеризующий соответствующий фильтр, а в правой части - маска. Для OpenDialog1 установим значения маски, как показано на рис. 7.1. Формат *.dat означает, что будут видны все файлы с расширением dat, а формат *.* - что будут видны все файлы (с любым именем и с любым расширением).



Для того чтобы файл автоматически записывался с расширением .dat, в свойстве DefaultExt запишем требуемое расширение - .dat.

Аналогичным образом настроим SaveDialog1 для текстового файла (расширение .txt).

Ход работы

Составить программу, вводящую в файл или читающую из файла ведомость абитуриентов, сдавших вступительные экзамены. Каждая запись должна содержать фамилию, курс, группу и средний балл. Вывести список абитуриентов, записанный в файл и прочитанный из файла. Запись произвести в стиле C++ в текстовый файл.

Unit1.cpp

```
#define n 5 //ограничим
число студентов struct stud { char fam[15]; int
kurs; int grup; float ball;}; stud st[n],sz[n]; int
kol=0;
```

```

//-----
----- void __fastcall
TForm1::Button1Click(TObject *Sender){ if(kol<n){
strcpy(st[kol].fam,Edit1->Text.c_str());
st[kol].kurs=StrToInt(Edit2->Text);
st[kol].grup=StrToInt(Edit2->Text);
st[kol].ball=StrToFloat(Edit4->Text);
Edit1->Text=""; Edit2->Text="";
Edit3->Text=""; Edit4->Text="";      112
Label5->Caption="Число записей
"+IntToStr(kol+1); kol++; }
else Button1->Enabled=false;
}
//-----
----- void __fastcall
TForm1::Button3Click(TObject *Sender)
{int i=0; Memo1->Clear();
ifstream in("z1.txt");
if(!in) {ShowMessage("Не удастся открыть файл."); }
while(!in.eof())
{
in>>sz[i].fam; in>>sz[i].kurs; in>>sz[i].grup;
in>>sz[i].ball; if (in) {
Memo1->Lines->Add(IntToStr(i));
Memo1->Lines->Add(sz[i].fam);
Memo1->Lines->Add(IntToStr(sz[i].kurs));
Memo1->Lines->Add(IntToStr(sz[i].grup));
Memo1->Lines-
>Add(FloatToStrF(sz[i].ball,ffGeneral,5,3)); i+
+; } }
in.close(); }
//-----
----- void __fastcall
TForm1::Button2Click(TObject *Sender)
{int i; AnsiString aaa; Memo2->Clear();
//ofstream out("z1.txt");
ofstream
out("z1.txt",ios::app); for
(i=0; i<kol; i++){
out<<st[i].fam<<" "<<st[i].kurs<<" "<<st[i].grup
<<"
"<<st[i].ball<<endl; } for
(i=0; i<5; i++){

```

```

Memo2->Lines->Add(st[i].fam);
Memo2->Lines->Add(IntToStr(st[i].kurs));
Memo2->Lines->Add(IntToStr(st[i].grup));
Memo2->Lines->Add(FloatToStrF(st[i].ball,ffGeneral,5,3));
}
out.close(); }
//-----

```

2. Разработка приложения с несколькими формами

Цель работы: изучить способ разработки приложений с несколькими формами **Ход работы**

1. Создайте новый проект, выбрав в главном меню пункт **File / New Application**.

2. Измените значение свойства **Name** на **MainForm**, а свойства **Caption** — на **Multiple Forms Test Program**. 3. Сохраните проект. Дайте модулю имя **Main**, а проекту — имя **Multiple**.

4. Теперь поместите в форму кнопку (**Button**). Присвойте свойству **Name** значение **ShowForm2**, а свойству **Caption** — значение **Show Form 2**.

5. Выберите в главном меню пункт **File / New Form** (или щелкните на кнопке **New Form** панели инструментов), чтобы создать новую форму. После создания новая форма будет иметь имя **Form1** и разместится точно поверх главной формы. Надо, чтобы новая форма имела меньший размер и была более-менее отцентрирована относительно главной формы.

6. Измените размер и положение новой формы так, чтобы она была примерно в два раза меньше главной формы и располагалась в ее центре. Для перемещения формы используйте строку заголовка. Размер можно изменить перетаскиванием нижнего правого угла.

7. Измените значение свойства **Name** новой формы на **SecondForm**, а свойства **Caption** — на **A Second Form**.

8. Выберите в главном меню пункт **File / Save** (или щелкните на кнопке **Save File** панели инструментов) и сохраните файл под именем **Second**.

9. Выберите компонент **Label** и поместите его в новую форму. Измените текст свойства **Caption** на **This is the second form**. Измените размер и цвет текста. Отцентрируйте сообщение относительно формы. Теперь ваша форма должна выглядеть примерно как на рис. 3.1.



Рис. 3.1 Вид формы к этому моменту

10. Щелкните на главной форме. Обратите внимание, что вторая форма теперь закрыта главной формой. Дважды щелкните на кнопке **Show Form 2**. На экране появится окно **редактора кода** и курсор будет размещен как раз там, где вам нужно начинать ввод текста. 11. Введите текст, приведенный ниже (вам нужно набрать только одну строку в скобках):

```
//-----
void __fastcall TMainForm::ShowForm2Click(TObject *Sender)
{
    SecondForm->ShowModal();
}
//-----
```

12. Запустите программу

Вы получили сообщение об ошибке **Undefined symbol 'SecondForm'** (Не определено обозначение 'SecondForm').

Странно...

SecondForm должно быть правильным именем, ведь именно это имя мы дали второй форме...

Дайте подумать... Ага! Вспомните, что у нас есть два исходных файла с соответствующими заголовками. Но в модуле **MainForm** нет объявления переменной **SecondForm** (которая является указателем на экземпляр класса **TSecondForm**). Мы должны указать, где расположено объявление класса. Для этого нужно включить заголовочный файл для **SecondForm** в исходный файл **MainForm** с помощью директивы **#include**. Переключитесь в окно редактора кода и щелкните на вкладке **Main.cpp** для отображения модуля главной формы. Перейдите к началу файла. Первые несколько строк должны выглядеть следующим образом:

```
//-----
#include <vcl.h>
#pragma hdrstop

#include "Main.h"
//-----
```

Вы видите, что сюда включен заголовочный файл **Main.h**, но нет файла **Second.h**, потому что мы не давали указания **C++Builder** включить этот файл. Давайте сделаем это сейчас.

1. Выберите в главном меню пункт **File / Include Unit Hdr**. На экране появится диалоговое окно **Include Unit**, показанное на рис. 3.2.



Рис.3.2 Диалоговое окно **Include Unit**

2. Перед вами находится список доступных модулей. В данном случае единственным модулем в списке является **Second**. Щелкните на имени **Second**, а затем на кнопке **OK**, чтобы закрыть диалоговое окно.

Диалоговое окно **Include Unit** отображает только те модули проекта, которые еще не включены в данный модуль. Включенные модули не содержатся в списке.

После щелчка на кнопке **OK** **C++Builder** мгновенно добавит в файл директиву **#include** для **Second.h**:

```
//-----
#include <vcl.h>
#pragma hdrstop

#include "Main.h"
#include "Second.h"
//-----
```

Теперь в модуль **Main** включено объявление класса из модуля **Second**. Щелкните на кнопке **Run**, чтобы запустить программу. На этот раз компиляция пройдет без задержек и программа заработает. Когда вы щелкнете на кнопке **Show Form 2**, на экране появится вторая форма. Как видите, C++Builder помогает вам управлять модулями. Вы должны использовать опцию **Include Unit Hdr**, чтобы обеспечить модулям доступ к объявлениям классов, использующихся в других модулях.

Поместите на форму компоненты **Label1**, **Edit**, **Label2**, **Button1** и **Button2** со страницы **Standard** палитры компонент как показано на рис.4.¹¹⁵

В метке **Label** имеется свойство **Word Wrap** — допустимость переноса слов длинной надписи, превышающей длину компонента, на новую строку. Чтобы такой перенос мог осуществляться, надо установить свойство **WordWrap** в **true**, свойство **AutoSize** в **false** (чтобы размер компонента не определялся размером надписи) и сделать высоту компонента такой, чтобы в нем могло поместиться несколько строк. Если **WordWrap** не установлено в **true** при **AutoSize** равном **false**, то длинный текст, не помещающийся в рамке метки, просто обрезается.

Перейдите в обработчик события **OnClick** сделав двойной щелчок на компоненте **Button1** на форме.

При вводе из окна числовой информации надо использовать функции **StrToInt(s)** - преобразование строки **s** в целое значение и **StrToFloat(s)** — преобразование строки **s** в значение с плавающей запятой (описание смотрите в лекции). После десятичной точки должно быть 4 цифры.

Затем выполните перевод температур по формуле: $C = (F - 32) * 5 / 9$, где **C** - это температура по шкале Цельсия, а **F** -- по шкале Фаренгейта.

Потом вам надо занести в метку смешанную информацию, состоящую из строк символов и чисел. Для этого воспользуйтесь функциями **FloatToStrF(n)** и **IntToStr(k)**, переводящие соответственно число **n** с плавающей запятой и целое число **k** в строки. При вызове функции **FloatToStrF(n,f,k,m)** указывают: **f** — формат; **k** — точность; **m** — количество цифр после десятичной точки. Формат определяет способ изображения числа: **ffGeneral** — универсальный; **ffExponent** — научный; **ffFixed** — с фиксированной точкой; **ffNumber** — с разделителями групп разрядов; **ffCurrency** — финансовый. Точность — нужное общее количество цифр.

Для формирования текста, состоящего из нескольких фрагментов, можно использовать операцию "+", которая для строк означает их склеивание (конкатенацию).

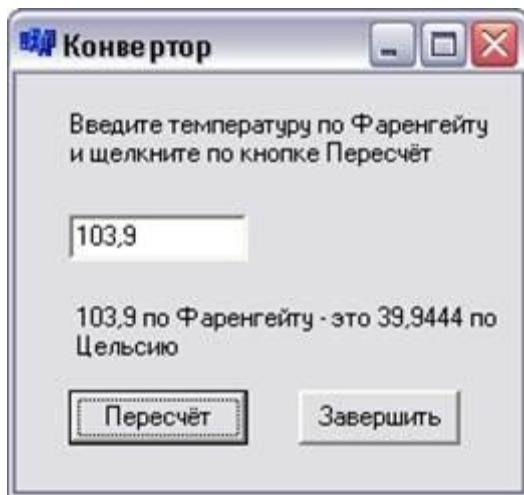


Рис.4 Вид формы конвертора

Сделайте, чтобы в TEdit можно было вводить только числа.

Для этого вам надо поместить следующий код в OnKeyPress любого TEdit'a:

```
// Key – код нажатой клавиши
```

```
// проверим является ли символ
```

```
допустимым if ((Key >= '0') && (Key <=
```

```
'9'))// цифра return;
```

```
else if ((Key == '.') || (Key ==
```

```
',')) {
```

```
// DecimalSeparator - глобальная переменная - разделитель целой и дробной
части, содержит символ, используемый в качестве разделителя при записи
дробных чисел.
```

```
if ((Edit1->Text).Pos(DecimalSeparator) !=
```

```
0) Key = 0; // разделитель уже введен
```

Получить выполненную работу или консультацию специалиста по вашему учебному проекту

```
else// если ещё нет
```

```
Key = DecimalSeparator;
```

```
return;
```

```
}
```

```
if (Key == VK_BACK)// клавиша
```

```
<Backspace> return;
```

```
if (Key == VK_RETURN)// клавиша <Enter>
```

```
{
```

```
Edit1->SetFocus();//делаем компонент активным: в его поле можно набирать
и выводить текст; return; } // остальные клавиши запрещены
```

```
Key = 0;// код запрещенных символов заменим нулем, в результате символ в
поле редактирования не отобразится
```

Для того, чтобы после нажатия на кнопку завершить форма закрылась используйте метод Close(), который закрывает форму.

3. Разработка приложения с не визуальными компонентами

Цель работы: изучить способ разработки приложения с не визуальными компонентами.

Ход работы

Задание 1. Установите на форму компонент PopUpMenu (находится на странице Standart). Присвойте свойству формы PopUpMenu значение PopUpMenu1. Дважды щелкните по компоненту PopUpMenu1 и в открывшемся конструкторе меню создайте следующие пункты (рис.1).

Создайте обработчики для кнопок:

«Свернуть форму»: Form1.WindowState:=wsMinimized;

«Восстановить форму»:

Form1.WindowState:=wsNormal; Для кнопки

«Закрыть» обработчик напишите самостоятельно.

Задание 2. На форме установите Image, OpenPictureDialog.

Создайте меню:

File: Open, (разделитель), Exit, (разделитель) - невидимый, Most Recent-невидимый Options: Center, Stretch, Transparent

About

Var current:string;

procedure TForm1.Open1Click(Sender:

TObject); begin if

OpenPictureDialog1.Execute then begin

if Current<>" then

begin

MostRecent1.Caption:=Current;

N2.Visible:=true;

MostRecent1.Visible:=true;

end;

Current:=OpenPictureDialog1.FileName;

Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName); end;

end;

procedure TForm1.MostRecent1Click(Sender:

TObject); var S : string; begin

S:= MostRecent1.Caption;

Image1.Picture.LoadFromFile(S); 117

MostRecent1.Caption:=Current;

```
Current:=S;  
end;
```

```
procedure TForm1.Center1Click(Sender:  
TObject); begin  
Center1.Checked := not  
Center1.Checked; Image1.Center :=  
Center1.Checked; end;
```

Задание 3. Измените меню File и соответствующие методы для отображения имён и повторного открытия не одной, а трёх последних картинок.

Добавьте в главное меню пункт Lines: Add, (разделитель) – невидимый. Добавьте методы:

```
procedure TForm1.Add1Click(Sender:  
TObject); var I : TMenuItem;  
begin  
I:=TMenuItem.Create(Lines1);  
I.Caption:= Memo1.SelText;  
I.OnClick:=Put;  
Lines1.Add(I);  
N3.Visible:=true;  
end;
```

```
procedure TForm1.Put(Sender:  
TObject); begin  
Memo1.Text:=Memo1.Text + (Sender as  
TMenuItem).Caption; end;
```

Задание 4. Остальные обработчики событий напишите самостоятельно. Добавьте в меню Text пункт "Alignment: Left", циклически изменяющий свойство Memo1.Alignment и своё свойство Caption для выравнивания по левому краю, по центру и по правому краю.

4. Разработка игрового приложения

Цель работы: ознакомиться с особенностями использования игровых ресурсов изучить способы разработки игрового приложения

Теоретический материал

Из чего состоит игра: игровые ресурсы

Любая *игра* состоит из множества частей. Но среди них можно выделить две основных. Первая – это игровые ресурсы, и вторая – это программный код.

Игровые ресурсы – это графические, музыкальные и иные ресурсы, которые используются для оформления игры. Рассмотрим различные форматы, в которых хранятся игровые ресурсы.

Графические файлы

Графические файлы используются для хранения изображений. Как правило, это – графические представления игровых объектов, которые применяются в двумерных или в псевдотрехмерных играх, когда трехмерный вид игровых объектов достигается лишь художественными средствами, без использования технологий трехмерной графики. Так же эти файлы могут быть использованы для создания элементов оформления игры и как текстуры для наложения на трехмерные модели.

JPEG

JPEG – это стандарт хранения файлов изображений, разработанный специально для хранения цифровых фотографий. *JPEG* расшифровывается как *Joint Photographic Experts Group* – именно так называлась *рабочая группа*, которая разработала этот стандарт. Особенностью *JPEG* является возможность сжатия изображения с потерями качества. Как правило, в *JPEG* выделяют 10 уровней сжатия (от 1 до 10), однако эти изображения можно сжимать и с более точным указанием уровня сжатия. Чем сильнее сжатие – тем сильнее потери качества. На невысоких уровнях сжатия *JPEG*-файлы практически не содержат так называемых артефактов сжатия. В случае с *JPEG* это проявляется в заметном искажении изображения, особенно – содержащего четкие границы между цветами, четкие линии. Дело в том, что при сжатии по алгоритму *JPEG* изображение разбивается на фрагменты 8x8 пикселей. После этого цветовая информация о пикселях одного квадрата сжимается, яркостная же информация остаётся в более сохранном виде.

Как правило, в *JPEG* хранят различные двумерные изображения, которые можно использовать для организации элементов интерфейса пользователя, для создания различных экранов игры (экран приветствия, экраны с информацией о набранных очках и так далее). В *JPEG* можно хранить изображения (их называют *спрайтами*), которые используются в двумерных играх в качестве игровых объектов (преимущественно – прямоугольной формы), в качестве фоновых изображений.

Лучше всего использовать *JPEG* для хранения цветных изображений с плавными цветовыми переходами. В частности, это могут быть фотографии, нарисованные игровые объекты и так далее.

JPEG-файлы имеют расширение JPG или *JPEG*.

Для создания *JPEG*-файлов с успехом можно использовать популярные графические редакторы – такие, как Adobe Photoshop различных версий.

PNG

Формат *PNG* (*Portable Network Graphics*) отличается от формата JPG тем, что использует алгоритмы сжатия, сжимающие изображение без потери качества. В результате оказывается, что изображения, имеющие четкие цветовые переходы (такие, как схемы, графики, изображения для простых элементов управления) могут быть довольно сильно сжаты в *PNG*-файлах. Если изображение имеет плавные цветовые переходы – лучшим выбором для его сжатия будет JPG, хотя и *PNG* вполне можно использовать. Как и JPG, данный формат поддерживает полноцветные изображения, однако изображение можно создать с использованием нескольких фиксированных цветов, что идет на пользу размеру графического файла, и, естественно, при правильном подборе цветов, не сказывается на его качестве.

Формат *PNG* был разработан для замены устаревшего, но все еще популярного формата GIF. Основное назначение GIF-файлов – простая интернет-графика.

PNG-файлы имеют расширение *.PNG*. Для их создания можно использовать растровые графические редакторы, такие, как Adobe Photoshop.

Еще одна важная особенность формата *PNG* – поддержка прозрачности. То есть при создании файла можно указать, какие из пикселей изображения считать прозрачными. В

результате *PNG* идеально подходит для хранения графического представления двумерных игровых объектов сложной формы.

TGA

TGA (Truevision Graphic Adapter), *TARGA* - Truevision Advanced Raster Graphics Adapter – это графический формат, который используется преимущественно для хранения игровых *текстур*. Этот формат был специально создан для хранения *текстур*, он используется во многих существующих играх именно для этих целей. Он поддерживает полноцветные изображения (8, 16, 24, 32-битные), поддерживают 8-ми битный *альфа-канал* для указания прозрачных участков изображения.

Текстура – это растровое (точечное) изображение, которое накладывается на поверхность полигона, являющегося частью трехмерной модели. Текстуры позволяют придавать трехмерным моделям цвета, создавать видимость рельефа, мелких деталей модели. При проектировании *3D* модели создание мелких деталей обычно оказывается слишком ресурсоемким, использование же *текстур* позволяет получать привлекательно выглядящие модели, построенные с использованием сравнительно небольшого количества трехмерных элементов. Сегодня трехмерные модели создаются с высоким уровнем детализации – мощности современных компьютеров хватает для реалистичной обработки тысяч моделей. В играх прошлых лет модели обычно выглядят схематично (например, профиль колеса автомобиля в играх, использующих низкополигональные модели, может выглядеть как многоугольник, но не как *окружность*), однако текстурирование позволяет даже таким моделям выглядеть достаточно реалистично.

Текстуры различаются глубиной цвета и разрешением – чем выше и то и другое – тем выше качество изображения. В применении к текстурам существует такое понятие, как *тексель* – число пикселей, приходящееся на минимальную единицу текстуры.

TGA-файлы имеют расширение *.TGA*

Для создания *TGA*-файлов можно применять всё тот же Adobe Photoshop и другие редакторы. Например, бесплатный редактор *Paint.NET* (<http://www.getpaint.net/>)

DDS

DDS (*Direct Draw Surface*) – это графический формат, разработанный специально для использования в *DirectX SDK* ([http://msdn.microsoft.com/en-us/library/bb943990\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb943990(VS.85).aspx)). Он предназначен преимущественно для хранения *текстур*. Благодаря особенностям формата – поддержке хранения сжатых и несжатых *текстур*, аппаратной поддержке, этот формат идеален для хранения игровых *текстур*.

DDS-файлы имеют расширение *DDS*. Для их создания можно использовать *плагины* для обычных графических редакторов, для работы с такими файлами созданы специальные наборы утилит и *плагинов*.

BMP

BMP (*Bitmap*) – это графический формат, который хранит изображение в несжатом виде. Он поддерживает алгоритм сжатия, который, однако, используется достаточно редко. Создавать *BMP*-файлы можно практически во всех растровых редакторах. Особенность *BMP* – широкое распространение при создании графических интерфейсов пользователя в различных Windows-программах. *BMP*-файлы обычно имеют сравнительно большой объём.

Файлы трехмерных моделей

Трехмерные модели используются в трехмерных, объемных играх. Для придания им реалистичного вида, на такие модели обычно накладывают файлы *текстур*.

FBX, X

FBX (Autodesk *FBX*), – это универсальный формат для хранения трехмерных моделей и сопутствующей информации. Он используется во множестве трехмерных приложений, в частности, весьма популярен в компьютерных играх.

Существуют *плагины* для популярных программ создания трехмерной графики, позволяющие конвертировать созданные в них модели в этот формат. Найти ПО для работы с *FBX*-файлами можно на <http://www.autodesk.ru/>.

Формат X – это формат трехмерных файлов, который используется *DirectX*.

Файлы описания шрифтов, SPRITEFONT

SPRITEFONT-файл - это xml-файл с настройками шрифта. Он содержит инструкции, касающиеся визуализации шрифта в игровом окне XNA-проекта. Файлы этого типа имеют расширение .SPRITEFONT.

Звуковые файлы

MP3 – сжатый звуковой файл. Этот формат имеет огромную популярность за счёт возможности сильного сжатия звука с приемлемым качеством. *WAV* – обычно представляет собой несжатый звуковой файл, как правило, может использоваться для воспроизведения коротких звуковых фрагментов (звуковых эффектов) высокого качества.

Все вышеперечисленные форматы файлов – это игровые ресурсы, которые могут быть включены в игровой проект. Учитывая тенденцию к сближению технологий XNA и *Silverlight*, которая наблюдается сегодня, нельзя строго разделить перечисленные игровые ресурсы между двумя технологиями.

Игровая терминология

Игровая терминология складывается под сильным влиянием англоязычных названий. Часто в русском языке довольно сложно найти хорошую аналогию принятым английским терминам, поэтому в сфере игровой терминологии существует двойственность – довольно часто создатели игр пользуются англоязычными терминами в то время, как и русскоязычные названия так же находят применение. Рассмотрим некоторые термины, которые используются в игровой индустрии.

Sprite – этот термин часто заменяют русскоязычным неологизмом *спрайт* – словарь Lingvo 12 определяет понятие "*спрайт*" как "элемент динамического графического отображения". В игровой индустрии синонимами слова *спрайт* являются такие слова, как "изображение", "картинка", иногда пользуются словом "*текстура*", однако обычно это понятие несет несколько иную смысловую нагрузку. Как правило *спрайт* – это двумерное изображение, причем, в узком смысле слова это – лишь изображение, а в широком – это игровой *объект*, который обладает гораздо более широким набором возможностей, нежели обычное изображение. Спрайты имеют прямоугольную форму, однако в компьютерных играх часто встречаются непрямоугольные объекты. Это достигается за счет задавания прозрачных областей при рисовании *спрайта*. Термин "анимированный *спрайт*" относится к *спрайту*, который выводится с использованием анимации, создающей иллюзию движения, перемещения каких-либо частей изображения – движение рук и ног персонажа при перемещении, движение колёс автомобиля, лопастей пропеллеров самолёта и т.д. *Анимация* обычно реализуется поочередной сменой нескольких статичных изображений, специально подготовленных для того, чтобы создать иллюзию движения.

Texture – *текстура*. Обычно *текстурами* называют двумерные изображения, которые "накладывают" на трехмерные модели. В терминологии XNA понятие текстуры и *спрайта* при разговоре о двумерных игровых объектах совпадает.

Background – фон. Так называется изображение, обычно – соответствующее размерами размерам игрового поля, которое является фоном для других изображений. Фон может быть неподвижным и подвижным. Подвижный фон (*scrolling background*) используется в играх, называемых скроллерами (scrollers). Скроллинг – это один из принятых игровых терминов. Он означает прокрутку, перемещение содержимого окна. Скроллинговые игры чрезвычайно распространены среди двумерных игр. Например, использование скроллингового фона позволяет создать иллюзию движения в двумерном гоночном *симуляторе*. Фон в двумерной игре может состоять из нескольких частей, движущихся с различной скоростью – это позволяет создать эффект трехмерности игрового мира.

2D-game – двумерная *игра* – *игра*, в которой использованы двумерные изображения.

3D-game – трёхмерная *игра* – использующая трехмерные модели и трехмерный игровой мир.

Tile – тайл – небольшое изображение, которое используется для конструирования уровней в играх. *Tile* можно перевести как "мозаика" или "черепица".

Polygone (полигон, многоугольник) – пространственный многоугольник, который используется для создания трехмерных объектов. Как правило, в компьютерной графике используются треугольники.

Pixel (пиксель) – наименьший элемент растрового изображения, точка, отображаемая на экране. Обычно в пикселях измеряют разрешение *текстур* (например – 1024x768), экранное разрешение монитора, размеры игровых окон. Слово *Pixel* – это аббревиатура от *Picture's Element*.

Texel (тексель) – точка текстуры в трехмерном пространстве. Слово *Texel* – это сокращение от *Texture Element*.

Voxel (воксель) – точка трехмерного изображения. Это слово – аббревиатура от *Volumetric Pixel* – объемный пиксель.

Texture Filtering (фильтрация *текстур*) – уменьшение искажений при наложении *текстур* на трехмерный объект.

Camera (камера) – так называют точку в игровом пространстве, с которой игрок видит игровой мир. С точки зрения положения камеры игры можно подразделить на игры от первого лица (камера расположена так, что реализует вид как бы "из глаз" персонажа), игры с видом слева-сверху (*стратегии*), игры с видом сзади – камера располагается обычно позади игрока и немного выше его. Существуют и игры, где камера может принимать различные позиции, например, реализующие вид из глаз персонажа и вид сзади

Transparency (прозрачность) – прозрачными могут быть части двумерных изображений – это позволяет создавать изображения сложной формы, которые, фактически, ограничены прямоугольником. Прозрачными бывают и объекты в трехмерном игровом мире – например – это может быть прозрачная вода или стекло, определенным образом преломляющие свет. *Light Model* (модель освещения) – способы моделирования освещения объектов.

Игровая документация

Если вы начинаете разрабатывать серьезный игровой проект – вам понадобится игровая документация. Эта лекция посвящена вопросам разработки такой документации. В частности, мы поговорим о концепт-документе, дизайн-документе и плане разработки игры.

В работе над этим материалом использованы образцы документов, рекомендованных компанией 1С для заполнения желающим сотрудничать с ней разработчикам. Актуальные версии образцов документов можно найти на (http://games.1c.ru/4_files/desdocpack.zip). Автор курса выражает признательность Сергею Герасеву – Менеджеру внешней разработки *игровых программ* 1С (gers@1c.ru) за содействие.

План разработки игры

План разработки игры – это рабочий документ, который направлен на детализацию некоторых положений, имеющих в ранее составленных документах и на планирование производства игры. Он может иметь следующую структуру:

1. Анализ рынка
 - 1.1. Целевая аудитория
 - 1.2. Хиты и сравнение
2. Технический анализ
3. Ресурсы проекта
 - 3.1. Персонал
 - 3.2. Оборудование
 - 3.3. Программное обеспечение
 - 3.4. Финансовые ресурсы

Риски проекта

Календарный план

Документ `gameproposaltemplate.doc` содержит шаблон плана разработки от 1С, документ `gameproposal.doc` – пример плана разработки.

5. Разработка игрового приложения

Цель работы: создание игрового приложения

Ход работы

Создаем файл main.cpp

```
#include <Windows.h>
```

```
int WINAPI WinMain(HINSTANCE,HINSTANCE,LPSTR,int)
{
    return 0;
}
```

Пока что он ничего не делает. Создаем каркас — класс Game Game.h

```
#ifndef _GAME_H_
#define _GAME_H_

class Game
{ private:
    bool run;
public:
    Game();
    int Execute();

    void Exit();
};

#endif
```

Game.cpp

```
#include "Game.h"

Game::Game()
{
    run = true;
}

int Game::Execute()
{
    while(run);
    return 0;
}

void Game::Exit()
{
    run = false;
}
```

Создаем файл Project.h, он нам очень пригодится в будущем

```
#ifndef _PROJECT_H_
#define _PROJECT_H_

#include <Windows.h>
```

```
#include "Game.h"
```

```
#endif
```

Изменяем main.cpp

```
#include "Project.h"
```

```
int WINAPI WinMain(HINSTANCE,HINSTANCE,LPSTR,int)
{
    Game game;
    return game.Execute();
}
```

1.2. Графика

Создаем 2 класса — Graphics для отрисовки графики и Image для отрисовки картинок

Graphics.h

```
#ifndef _GRAPHICS_H_
#define _GRAPHICS_H_
```

```
#include "Project.h"
```

```
#include "Image.h"
class Image;
```

```
class Graphics
```

```
{ private:
```

```
    SDL_Surface* Screen;
```

```
public:
```

```
    Graphics(int width, int height);
```

```
    Image* NewImage(char* file);
```

```
    Image* NewImage(char* file, int r, int g, int b);    bool
```

```
DrawImage(Image* img, int x, int y);
```

```
    bool DrawImage(Image* img, int x, int y, int startX, int startY, int endX, int endY);
```

```
    void Flip();
```

```
};
```

```
#endif
```

Image.h

```
#ifndef _IMAGE_H
#define _IMAGE_H
```

```
#include "Project.h"
```

```
class Image
```

```
{ private:
```

```
    SDL_Surface* surf; public:
```

```
    friend class Graphics;
```

```

        int GetWidth();
        int GetHeight();
    };

#endif

```

Изменяем Project.h

```

#ifndef _PROJECT_H_
#define _PROJECT_H_

#pragma comment(lib,"SDL.lib")

#include <Windows.h>
#include <SDL.h>

#include "Game.h"
#include "Graphics.h"
#include "Image.h"

#endif

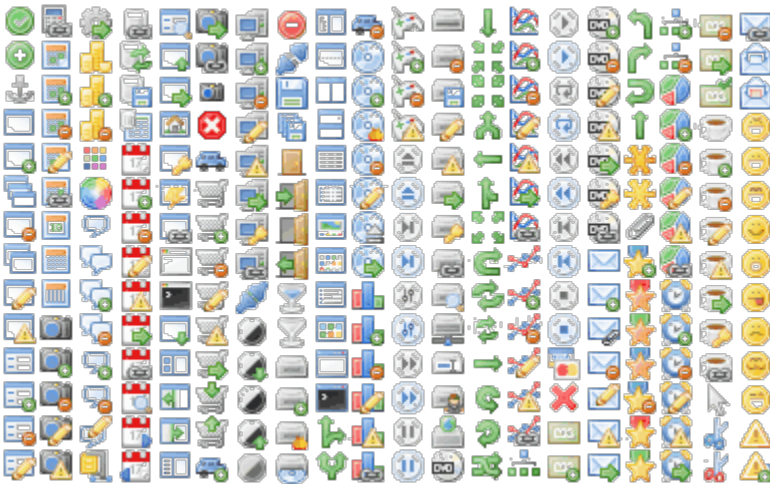
```

SDL_Surface — класс из SDL для хранения информации об картинке

Рассмотрим Graphics

NewImage — есть 2 варианта загрузки картинки. Первый вариант просто грузит картинку, а второй после этого еще и дает прозрачность картинке. Если у нас красный фон в картинке, то вводим r=255,g=0,b=0

DrawImage — тоже 2 варианта отрисовки картинки. Первый рисует всю картинку целиком, второй только часть картинки. startX, startY — координаты начала части картинки. endX, endY — конечные координаты части картинки. Этот метод рисования применяется, если используются атласы картинок. Вот пример атласа:



(изображение взято из веб-ресурса interesnoe.info)

Рассмотрим Image

Он просто держит свой сурфейс и дает право доступа к своим закрытым членам классу Graphics, а он изменяет сурфейс.

По сути, это обертка над SDL_Surface. Также он дает размер картинки

Graphics.cpp

```

#include "Graphics.h"

```

```

Graphics::Graphics(int width, int height) {
    SDL_Init(SDL_INIT_EVERYTHING);
    Screen = SDL_SetVideoMode(width,height,32,SDL_HWSURFACE|SDL_DOUBLEBUF);
}

Image* Graphics::NewImage(char* file)
{
    Image* image = new Image();
    image->surf = SDL_DisplayFormat(SDL_LoadBMP(file));

    return image;
}

Image* Graphics::NewImage(char* file, int r, int g, int b) {
    Image* image = new Image();
    image->surf = SDL_DisplayFormat(SDL_LoadBMP(file));

    SDL_SetColorKey(image->surf, SDL_SRCCOLORKEY | SDL_RLEACCEL,
        SDL_MapRGB(image->surf->format, r, g, b));

    return image;
}

bool Graphics::DrawImage(Image* img, int x, int y)
{
    if(Screen == NULL || img->surf == NULL)    return
false;

    SDL_Rect Area;
    Area.x = x;
    Area.y = y;

    SDL_BlitSurface(img->surf, NULL, Screen, &Area);

    return true;
}

bool Graphics::DrawImage(Image* img, int x, int y, int startX, int startY, int endX, int endY)
{
    if(Screen == NULL || img->surf == NULL)    return
false;

    SDL_Rect Area;
    Area.x = x;
    Area.y = y;

    SDL_Rect SrcArea;
    SrcArea.x = startX;
    SrcArea.y = startY; SrcArea.w = endX;
    SrcArea.h = endY;

    SDL_BlitSurface(img->surf, &SrcArea, Screen, &Area);

    return true;
}

void Graphics::Flip()
{
    SDL_Flip(Screen);
    SDL_FillRect(Screen,NULL, 0x000000);
}

```

В конструкторе инициализируется SDL и создается экран.

Функция Flip должна вызываться каждый раз после отрисовки картинок, она представляет получившееся на экран и чистит экран в черный цвет для дальнейшей отрисовки. Image.cpp

```
#include "Image.h"

int Image::GetWidth()
{
    return surf->w;
}

int Image::GetHeight()
{
    return surf->h;
}
```

Надо изменить Game.h, Game.cpp и main.cpp
Game.h

```
#ifndef _GAME_H_
#define _GAME_H_

#include "Project.h"
class Graphics;

class Game
{ private:
    bool run;

    Graphics* graphics;
public:
    Game();
    int Execute(int width, int height);

    void Exit();
};

#endif
```

Тут мы добавляем указатель на Graphics и в Execute добавляем размер экрана

Game.cpp

```
#include "Game.h"

Game::Game()
{
    run = true;
}

int Game::Execute(int width, int height)
{
    graphics = new Graphics(width,height);

    while(run);

    SDL_Quit();
}
```

```

        return 0;
    }

    void Game::Exit()
    {
        run = false;
    }

```

Ничего особенного, разве что не пропустите функцию `SDL_Quit` для очистки SDL

main.cpp

```

#include "Project.h"

int WINAPI WinMain(HINSTANCE,HINSTANCE,LPSTR,int)
{
    Game game;
    return game.Execute(500,350);
} Создаем экран размером 500 на 350.

```

1.3. Ввод

Создаем Input.h

```

#ifndef _INPUT_H_
#define _INPUT_H_

#include "Project.h"

class Input
{ private:
    SDL_Event evt;

public:
    void Update();

    bool IsMouseButtonDown(byte key);    bool
    IsMouseButtonUp(byte key);
    POINT GetButtonDownCoords();

    bool IsKeyDown(byte key);    bool
    IsKeyUp(byte key);
    byte GetPressedKey();

    bool IsExit();
};

```

#endif

`SDL_Event` — класс какого-нибудь события, его мы держим в `Input`'е для того, чтобы не создавать объект этого класса каждый цикл

Ниже расположены методы, не представляющие особого интереса. Примечание: методы с 128 окончанием `Down` вызываются, когда клавиша была нажата, а с окончанием `Up` — когда опущена.

Input.cpp

```

#include "Input.h"

void Input::Update()
{
    while(SDL_PollEvent(&evt));

```

```

}

bool Input::IsMouseButtonDown(byte key)
{
    if(evt.type == SDL_MOUSEBUTTONDOWN)
        if(evt.button.button == key)
            return true;

    return false;
}

```

```

bool Input::IsMouseButtonUp(byte key)
{
    if(evt.type == SDL_MOUSEBUTTONUP)
        if(evt.button.button == key)
            return true;

    return false;
}

```

```

POINT Input::GetButtonDownCoords()
{
    POINT point;
    point.x = evt.button.x; point.y =
    evt.button.y;

    return point;
}

```

```

bool Input::IsKeyDown(byte key)
{
    return (evt.type == SDL_KEYDOWN && evt.key.keysym.sym == key);
}

```

```

bool Input::IsKeyUp(byte key)
{
    return (evt.type == SDL_KEYUP && evt.key.keysym.sym == key);
}

```

```

byte Input::GetPressedKey()
{
    return evt.key.keysym.sym;
}

```

```

bool Input::IsExit()
{
    return (evt.type == SDL_QUIT);
}

```

Здесь мы обрабатываем наш объект событий в функции Update, а остальные функции просто проверяют тип события и его значения.

Изменяем теперь Game.h и Game.cpp

```

#ifndef _GAME_H_
#define _GAME_H_

#include "Project.h"

#include "Graphics.h" class
Graphics; #include "Input.h"
class Input;

class Game
{ private:
    bool run;

```

```

        Graphics* graphics;
        Input* input;
public:
        Game();
        int Execute(int width, int height);

        Graphics* GetGraphics();
        Input* GetInput();

        void Exit();
};

#endif

```

Как видно, мы добавили указатель на Input и создали методы-возвращатели Graphics и Input

Game.cpp

```

#include "Game.h"

Game::Game()
{
    run = true;
}

int Game::Execute(int width, int height)
{
    graphics = new Graphics(width,height);
    input = new Input();

    while(run)
    {
        input->Update();
    }

    delete graphics;
    delete input;

    SDL_Quit();
    return 0;
}

Graphics* Game::GetGraphics()
{
    return graphics;
}

Input* Game::GetInput()
{
    return input;
}

void Game::Exit()
{
    run = false;
}

```

130

6. Разработка приложения с анимацией

Цель работы: рассмотреть способы разработки приложений с анимацией

Ход работы

Задание: Создать несложную анимацию, по дороге будут ездить на встречу друг другу автомобиль и мотоцикл.

Используемые компоненты:

TImage (Графический холст) находится на вкладке Additional.

TTimer (Таймер) находится на вкладке System.

В папке "Picture" расположены изображения fon.bmp, avto.bmp и moto.bmp. Это наши заготовки, которые мы будем использовать в программе. Первый способ

1. Запускаем Delphi и создаем новое приложение: File->New->VCL Forms Application – Delphi. 2. Помещаем на форму 3 компонента TImage. Один из них загружаем фон, а в два других изображения мотоцикла и автомобиля соответственно и в режиме таймера будем изменять положения компонент TImage с изображением автомобиля и мотоцикла относительно фона.

3. В компонент Image1 в свойство Picture загружаем подготовленный нами файл fon.bmp, свойство Align устанавливаем alClient. А форму растянем до размеров фона.

4. В компонент Image2 в свойство Picture загружаем подготовленный нами файл avto.bmp и перемещаем его на дорогу.

5. В компонент Image3 в свойство Picture загружаем подготовленный нами файл moto.bmp и перемещаем его на дорогу.

6. Сравнить вашу форму с примером:

7. В коде программы перед разделом type добавим раздел const и объявим две константы const
scr_width = 640; // ширина формы scr_height = 480; //
высота формы

В разделе var объявим переменные x,y,x1,y1 типа integer.131 var
Form1: TForm1;
x,y,x1,y1:integer;

8. Создайте метод procedure TForm1.Timer1Timer и запишите его выполнение procedure
TForm1.Timer1Timer(Sender: TObject); begin x:=x+2;//текущая координата + шаг для автомобиля
x1:=x1-2;//текущая координата + шаг для мотоцикла
if x>scr_width+image2.Width then x:=-image2.Width;// ограничение справа if x1<-
image3.Width then x1:=scr_width;// ограничение слева
//рисуем image2.Left:=x;
image3.Left:=x1; end;

9. В свойствах таймера свойство интервал устанавливаем в пределах от 1 до 100, в зависимости от того какую скорость анимации вы хотите получить. Второй способ

Второй способ заключается в том, что мы создаем для нашего случая 4 объекта типа TBitmap, в первый загружаем фон, во второй - автомобиль, в третий - мотоцикл, а четвертый будет буфером обмена в котором мы вначале будем формировать картинку, а потом выводить ее на экран. Это необходимо делать для того чтобы избежать мерцания картинки на экране во время движения в результате ее перерисовки. Если эффект мерцания для вас не существен, то можно выводить изображение сразу на экран.

1. Помещаем на форму компонент TImage и компонент TTimer.

2. Для компонента Image1 свойство Align устанавливаем alClient.
А для формы свойство AutoSize устанавливаем True.

3. Объявите константы
const scr_width = 640; // ширина экрана
scr_height = 480; // высота экрана

4. В описании переменных var объявляем четыре переменных для хранения графических картинок, тип tbitmap

var
fon:tbitmap;//Графический образ Фона avto:tbitmap;//Графический образ
автомобиля moto:tbitmap;//Графический образ мотоцикла
scr_buffer:tbitmap;//Графический образ автомобиля
x,y,x1,y1:integer;//координаты автомобиля и мотоцикла 5. Для того
чтобы вывести на экран изображения нужно :

а) Активировать созданные переменные //создаем
объекты fon:=TBitmap.Create; // фон
moto:=TBitmap.Create; // мотоцикл
avto:=TBitmap.Create; // машина

```

scr_buffer:=TBitmap.Create; // буфер обмена
scr_buffer.Width:=scr_width; // ширина буфера
scr_buffer.Height:=scr_height; // высота буфера
б) Загрузить изображения в эти переменные //
загружаем объекты
moto.LoadFromFile('moto.bmp'); // мотоцикл
avto.LoadFromFile('avto.bmp'); // машина
fon.LoadFromFile('fon.bmp'); // фон x:=0; //начальные
координаты машины y:=430; 132
x1:=500; // начальные координаты мотоцикла
y1:=380;
в) Установить прозрачный фон вокруг машины и мотоцикла
avto.transparent:=true; //задаем прозрачность moto.transparent:=true;
г) Вывод изображения на графический холст Image //Определение ление
координат
x:=x+2; //текущая координата + шаг для автомобиля x1:=x1-2; //текущая координата +
шаг для мотоцикла
if x>scr_width+avto.Width then x:=-avto.Width; // ограничение справа if x1<-
moto.Width then x1:=scr_width; // ограничение слева
//рисуем scr_buffer.Canvas.Draw(0,0,fon); //возобновление фона
scr_buffer.Canvas.Draw(x1,y1,moto); //движение мотоцикла
scr_buffer.Canvas.Draw(x,y,avto); //движение машины
form1.Canvas.Draw(0,0,scr_buffer); //копируем содержимое буфера на экран
При этом первая цифра в скобках, это координата x, а вторая цифра соответственно координата y
д) Для того , чтобы изображения стали двигаться, необходимо динамически менять
координаты изобра-жений на холсте, желательно стирая старое изображение. Эту задачу
выполняет компонент Timer.

```

Тема 1.1.6 Оптимизация и рефакторинг кода

1. Оптимизация кода

Цель работы: изучить технологию оптимизации программного кода **Ход работы**

Задание:

Модифицируйте 2 программы, реализованные на C++:

1 программа

```

#include <iostream> #include <locale.h> using
namespace std;
int main()
{
int number; setlocale(LC_CTYPE,"Russian"); cout << "Введите число: "; cin >> number; cin.ignore();
cout << "Вы ввели: "<< number <<"\n"; cin.get(); }

```

Описание: пользователю предлагается ввести цифру, но если он введет например: b6, то ему выдаст - "Вы ввели: 5 (Только номер). Необходимо добиться, чтобы программа различала отрицательные и положительные значения, цифры и буквы, а также автоматически выдавала значение введенного числа в квадрате.

2 программа

```

#include <stdio.h> /* Стандартный заголовочный файл ввода-вывода */ #include <iostream>
/* Библиотека (стандарт) */
#include <locale.h> /* Русификатор */ #include <windows.h> /* Русификатор */ using namespace std; +
int main(int argc, char* argv[])
{
setlocale(LC_CTYPE,"Russian");
double plus, minus, pow, div; // объявление переменных через запятую double a1; //
отдельное объявление переменной a1 double a2; // отдельное объявление переменной a2 cout << "Введите
первое число: ";133 cin >> a1;

```

```
cout << "Введите второе число: "; cin >> a2; plus = a1 + a2; // операция сложения minus = a1 - a2; // операция
вычитания pow = a1 * a2;
// операция умножения div = a1 / a2; // операция деления cout << a1 << "+" << a2 << "=" << plus << endl; cout << a1
<< "-" << a2 << "=" << minus << endl; cout << a1 << "*" << a2 << "=" << pow << endl; cout << a1 << "/" << a2 << "="
<< div << endl; system("pause");
```

return 0; } Описание: простой калькулятор, который может: добавлять, вычитать, умножать и делить.

Необходимо реализовать функцию возведения числа в любую степень, а также умножение отрицательных чисел.

Задание:

–Решите задачу оптимизации кода (файл с кодом возьмите у преподавателя), используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей.

–Оптимизировать программу (можно использовать процедуры или функции).

–Оптимизированная программа должна содержать проверки всех переменных, которые вводятся с клавиатуры.

–Для созданных программ оценить метрические характеристики по Холстеду;

–Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

Расчет метрики Холстеда:

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Метрика Холстеда позволяет оценить размер (в словах) и объем в битах программы на стадии анализа требований. Используя нормы выработки операторов в день можно оценить время на разработку.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы: n1

— число уникальных операторов

+

Опреаторы	Число операторов	Операнды	Число операндов
::	7	a	2
=	8	b	2
==	1	c	2
.	6	mAparam	5
!=	1	mBparam	5
<	1	mCparam	3
-	3	result.x1	3
/	3	result.x2	3
*	2	result.status	2
- binary	3	det	3
return	1	4	3
{	12	2	2
}	12		
(	13		
)	13		

программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов); n2 — число уникальных операндов программы (словарь операндов); N1 — общее число операторов в программе; N2 — общее число операндов в программе.

+, *, /, - это операторы x, y, z, 999, -25, number1

- это операнды

На основании этих характеристик рассчитываются оценки:

Словарь программы (Halstead Program Vocabulary, HPVoc): $n = n1 + n2$; Длина

программы (Halstead Program Length, HPLen): $N = N1 + N2$;

Объем программы (Halstead Program Volume, HPVol): $V = N \log_2 n$;

Сложность программы (Halstead Difficulty, HDiff): $D = (n1/2) \times (N2 / n2)$;

На основе показателя HDiff предлагается оценивать усилия программиста при разработке при помощи показателя HEff (Halstead Effort): $H = D \times V$.

2. Рефакторинг кода

Цель работы: изучить технологию рефакторинга программного кода

Ход работы

Задание: Задание 1:

В рамках лабораторной работы приведите примеры реализации рефакторинга в следующих системах:
–Visual Studio.

–Visual Assist X. –Refactor.

–JustCode

–ReSharper

–CodeIt

Ход выполнения работы

1. Выполнить анализ программного кода разрабатываемого ПО и модульных тестов с целью выявления плохо организованного кода.
2. Используя шаблоны рефакторинга, выполнить реорганизацию программного кода разрабатываемого ПО.
3. Выполнить описание произведенных операций рефакторинга (было-стало-шаблон рефакторинга).
4. В случае необходимости скорректировать проектную документацию (диаграммы классов, последовательностей).
5. Сделать выводы по результатам выполнения работы. Класс Main **Было:** package game; import game.Characters.*; import game.Characters.Character; import game.Energetics.Energetic; import game.Energetics.Lightning; import game.Levels.Block; import game.Levels.Level; import game.Levels.Level_data; import game.Weapon.Bullet; import game.Weapon.Weapon; import javafx.animation.AnimationTimer; import javafx.application.Application; import javafx.scene.Scene; import javafx.scene.image.Image; import javafx.scene.input.KeyCode; import javafx.scene.layout.Pane; import javafx.stage.Stage; import java.io.DataInputStream; import java.io.FileInputStream; import java.io.IOException; import java.util.ArrayList; import java.util.HashMap; public class Main extends Application { public static ArrayList<Block> blocks = new ArrayList<>(); public static ArrayList<Bullet> bullets = new ArrayList<>(); public static ArrayList<Bullet> enemyBullets = new ArrayList<>(); public static ArrayList<EnemyBase> enemies = new ArrayList<>(); static HashMap<KeyCode, Boolean> keys = new HashMap<>(); public static Stage stage; public static Scene scene; public static Pane gameRoot = new Pane(); public static Pane appRoot = new Pane(); public static Menu menu; public static Character booker; public static HUD hud; public static Weapon weapon; public static Elizabeth elizabeth; static VendingMachine vendingMachine; static Tutorial tutorial; private static CutScenes cutScene; public static Energetic energetic; public static Lightning lightning; public static int levelNumber; static Level level; public static AnimationTimer timer = new AnimationTimer() { @Override public void handle(long now) { update(); } }; private static void update() { for (EnemyBase enemy : enemies) { enemy.update(); if (enemy.getDelete()) { enemies.remove(enemy); break; } } Bullet.update(); Controller.update(); booker.update();

```
if (!energetic.getName().equals(""))  
energetic.update(); if (levelNumber > 0)  
elizabeth.update(); if (lightning != null)  
{ lightning.update(); if  
(lightning.getDelete()) lightning = null;  
}
```

```

menu.update(); hud.update();
weapon.update();
if (booker.getTranslateX() > Level_data.BLOCK_SIZE * 295) cutScene = new
CutScenes(levelNumber);
}
@Override
public void start(Stage primaryStage) throws Exception { stage =
primaryStage; scene = new Scene(appRoot, 1280, 720);
appRoot.getChildren().add(gameRoot);
level = new Level();
try (DataInputStream dataInputStream = new DataInputStream(new
FileInputStream("C:/DeadShock/saves/data.dat"))) { levelNumber =
dataInputStream.readInt(); level.createLevels(levelNumber);
level.changeImageView(levelNumber); vendingMachine =
new VendingMachine(); booker = new Character();
booker.setMoney(dataInputStream.readInt());
booker.setSalt(dataInputStream.readByte());
booker.setCountLives(2); weapon = new Weapon();
weapon.setWeaponClip(dataInputStream.readInt());
weapon.setBullets(dataInputStream.readInt());
hud = new HUD(); elizabeth = new Elizabeth();
energetic = new Energetic(); } catch (IOException
e) { levelNumber = 0;
level.createLevels(levelNumber); vendingMachine
= new VendingMachine();
booker = new Character(); weapon = new
Weapon(); hud = new HUD(); energetic =
new Energetic(); tutorial = new
Tutorial(levelNumber);
} switch (levelNumber)
{ case 0:
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 117, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 127, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 148, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 161, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 171, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 185, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 204, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 215, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 228, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 233, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 243, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 252, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 262, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 277, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 280, 200));137 enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 286, 200));
break; case
1:
enemies.add(new EnemyRedEye(Level_data.BLOCK_SIZE * 57, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 67, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 74, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 87, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 104, 150)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 117, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 133, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 156, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 177, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 193, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 201, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 216, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 224, 200)); enemies.add(new

```

```

EnemyComstock(Level_data.BLOCK_SIZE * 246, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 260, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 277, 100));

```

Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 13));	Block("invisible",	Level_data.BLOCK_SIZE	*	34,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 13));	Block("invisible",	Level_data.BLOCK_SIZE	*	36,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 9));	Block("invisible",	Level_data.BLOCK_SIZE	*	60,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 9));	Block("invisible",	Level_data.BLOCK_SIZE	*	61,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 7));	Block("invisible",	Level_data.BLOCK_SIZE	*	106,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 7));	Block("invisible",	Level_data.BLOCK_SIZE	*	107,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 13));	Block("invisible",	Level_data.BLOCK_SIZE	*	168,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 13));	Block("invisible",	Level_data.BLOCK_SIZE	*	170,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 13));	Block("invisible",	Level_data.BLOCK_SIZE	*	196,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 13));	Block("invisible",	Level_data.BLOCK_SIZE	*	197,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 8));	Block("invisible",	Level_data.BLOCK_SIZE	*	232,
Level_data.enemyBlocks.add(new Level_data.BLOCK_SIZE * 8)); break;	Block("invisible",	Level_data.BLOCK_SIZE	*	233,

```

    }
    menu = new Menu();
    appRoot.getChildren().add(menu.menuBox);
    booker.translateXProperty().addListener( ((observable, oldValue, newValue) -> { int offset =
    newValue.intValue();
    if (offset > 600 && offset < gameRoot.getWidth() - 680) { gameRoot.setLayoutX( - (offset -
    600) );
    level.getBackground().setLayoutX((offset - 600) / 1.5);
    }
    if (offset <= 100)
    level.getBackground().setLayoutX(0);
    }));
    vendingMachine.createButtons();
    stage.getIcons().add(new Image("file:/C:/DeadShock/images/icon.jpg"));
    stage.setTitle("DeadShock"); stage.setResizable(false);
    stage.setWidth(scene.getWidth()); stage.setHeight(scene.getHeight());
    stage.setScene(scene); stage.show(); timer.start(); }
    public static void main(String[] args) { launch(args);
    }
}
} Стало: package game; import
game.Characters.*; import
game.Characters.Character; import
game.Energetics.Energetic; import
game.Energetics.Lightning; import
game.Levels.Block; import game.Levels.Level;
import game.Levels.Level_data; import
game.Weapon.Bullet; import
game.Weapon.Weapon; import
javafx.animation.AnimationTimer; import

```

```

javafx.application.Application; import
javafx.scene.Scene; import
javafx.scene.image.Image; import
javafx.scene.input.KeyCode; import
javafx.scene.layout.Pane; import
javafx.stage.Stage; import
java.io.DataInputStream; import
java.io.FileInputStream; import
java.io.IOException; import java.util.ArrayList;
import java.util.HashMap; public class Main
extends Application {
    public static ArrayList<Block> blocks = new ArrayList<>(); public static
    ArrayList<Bullet> bullets = new ArrayList<>(); public static ArrayList<Bullet>
    enemyBullets = new ArrayList<>(); public static ArrayList<EnemyBase>
    enemies = new ArrayList<>();
    static HashMap<KeyCode, Boolean> keys = new HashMap<>(); public static
    Stage stage; public static Scene scene; public static Pane gameRoot = new
    Pane(); public static Pane appRoot = new Pane(); public static Menu menu;
    public static Character booker; public static HUD hud; public static Weapon
    weapon; public static Elizabeth elizabeth;
    static VendingMachine vendingMachine; static
    Tutorial tutorial; private static CutScenes
    cutScene; public static Energetic energetic;
    public static Lightning lightning;
    public static int levelNumber;
    static Level level;
    public static AnimationTimer timer = new AnimationTimer() {
        @Override
        public void handle(long now) {
            update();
        }
    };
    private void initContent()
    { appRoot.getChildren().add(gameRoot); level = new
    Level();
    try (DataInputStream dataInputStream = new
    FileInputStream("C:/DeadShock/saves/data.dat")) { levelNumber =
    dataInputStream.readInt(); level.createLevels(levelNumber);
    level.changeImageView(levelNumber); vendingMachine =
    new VendingMachine(); booker = new Character();
    booker.setMoney(dataInputStream.readInt());
    booker.setSalt(dataInputStream.readByte());
    booker.setCountLives(2); weapon = new Weapon();
    weapon.setWeaponClip(dataInputStream.readInt());
    weapon.setBullets(dataInputStream.readInt());
    hud = new HUD(); elizabeth =
    new Elizabeth(); energetic = new
    Energetic(); } catch (IOException
    e) { levelNumber = 0;
    level.createLevels(levelNumber); vendingMachine = new
    VendingMachine();
    booker = new Character();
    weapon = new Weapon(); hud =
    new HUD(); energetic = new
    Energetic();
    tutorial = new Tutorial(levelNumber);
    }
    createEnemies(); menu =
    new Menu();
    appRoot.getChildren().add(menu.menuBox);
    booker.translateXProperty().addListener( (observable, oldValue, newValue) -> { int offset =
    newValue.intValue();

```

```

if (offset > 600 && offset < gameRoot.getWidth() - 680) { gameRoot.setLayoutX( - (offset -
600) );
level.getBackground().setLayoutX((offset - 600) / 1.5);
}
if (offset <= 100)
level.getBackground().setLayoutX(0);
});
vendingMachine.createButtons();
}
public static void createEnemies() {
switch (levelNumber) { case 0:
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 117, 200));140      enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 127, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 148, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 161, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 171, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 185, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 204, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 215, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 228, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 233, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 243, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 252, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 262, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 277, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 280, 200));
enemies.add(new EnemyComstock(Level_data.BLOCK_SIZE * 286, 200)); break; case 1:
enemies.add(new EnemyRedEye(Level_data.BLOCK_SIZE * 57, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 67, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 74, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 87, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 104, 150)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 117, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 133, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 156, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 177, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 193, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 201, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 216, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 224, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 246, 200)); enemies.add(new
EnemyRedEye(Level_data.BLOCK_SIZE * 260, 200)); enemies.add(new
EnemyComstock(Level_data.BLOCK_SIZE * 277, 100));

```

Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 13));	Level_data.BLOCK_SIZE	*	34,
Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 13));	Level_data.BLOCK_SIZE	*	36,
Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 9));	Level_data.BLOCK_SIZE	*	60,
Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 9));	Level_data.BLOCK_SIZE	*	61,
Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 7));	Level_data.BLOCK_SIZE	*	106,

```

        Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 7));
        Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 13));
        Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 13));
        Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 13));
        Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 13));
        Level_data.enemyBlocks.add(new Block("invisible", Level_data.BLOCK_SIZE * 233,
        Level_data.BLOCK_SIZE * 8));
        break;
    } }
    private static void update() { for
    (EnemyBase enemy : enemies)
    { enemy.update();

```

Тема 1.1.7 Разработка пользовательского интерфейса

1. Пользовательская и программная модели интерфейса Цель работы:

закрепить теоретические знания по разработке пользовательского интерфейса; получить практические навыки по проектированию пользовательской и программной модели интерфейса.

Теоретический материал

Существуют три совершенно различные модели пользовательского интерфейса: модель программиста,

модель пользователя и программная модель. Программист, разрабатывая пользовательский интерфейс, исходит из того, управление какими операциями ему необходимо реализовать в пользовательском интерфейсе, и как это осуществить, не затрачивая ни существенных ресурсов компьютера, ни своих сил и времени. Его интересуют функциональность, эффективность, технологичность, внутренняя стройность и другие не связанные с удобством пользователя характеристики программного обеспечения. Именно поэтому большинство интерфейсов существующих программ вызывают серьезные нарекания пользователей.

С точки зрения здравого смысла хорошим следует считать интерфейс, при работе с которым пользователь получает именно то, что он ожидал. Представление пользователя о функциях интерфейса можно описать в виде пользовательской модели интерфейса.

Пользовательская модель интерфейса - это совокупность обобщенных представлений конкретного пользователя или некоторой группы пользователей о процессах, происходящих во время работы программы или программной системы. Эта модель базируется на особенностях опыта конкретных пользователей, который характеризуется:

- уровнем подготовки в предметной области разрабатываемого программного обеспечения;
- интуитивными моделями выполнения операций в этой предметной области;
- уровнем подготовки в области владения компьютером; П устоявшимися стереотипами работы с компьютером.

Для построения пользовательской модели необходимо изучить перечисленные выше особенности опыта предполагаемых пользователей программного обеспечения. С этой целью используют опросы, тесты и даже фиксируют последовательность действий, осуществляемых в процессе выполнения некоторых операций, на пленку.

Приведение в соответствие моделей пользователя и программиста, а также построение на их базе программной модели (рис. 1.1) интерфейса задача не тривиальная. Причем, чем сложнее автоматизируемая предметная область, тем сложнее оказывается построить программную модель

интерфейса, учитывающую особенности пользовательской модели и не требующую слишком больших затрат как в процессе разработки, так и во время работы. С этой точки зрения объектные интерфейсы кажутся наиболее перспективными, так как в их основе лежит именно отображение

объектов предметной области, которыми ¹⁴² оперируют пользователи. Хотя на настоящий момент времени их реализация достаточно трудоемка.

При создании программной модели интерфейса также следует иметь в виду, что изменять пользовательскую модель не просто. Повышение профессионального уровня пользователей и их подготовки в области владения компьютером в компетенцию разработчиков программного обеспечения не входит, хотя часто грамотно построенный интерфейс, который адекватно отображает сущность происходящих процессов, способствует росту квалификации пользователей.

Интуитивные модели выполнения операций в предметной области должны стать основой для разработки интерфейса, а потому в большинстве случаев их необходимо не менять, а уточнять и совершенствовать. Именно нежелание или невозможность следования интуитивным моделям выполнения операций приводит к созданию искусственных надуманных интерфейсов, которые негативно воспринимаются пользователями.

Иногда кажется, что единственно доступный для изменения элемент - устоявшийся стереотип работы с компьютером. Однако ломка стереотипов - процедура болезненная. На это стоит решаться, если некоторое революционное изменение значительно расширяет возможности пользователя или облегчает его работу, например, переход к Windows-интерфейсам существенно упростил работу с компьютером огромному числу пользователей-непрофессионалов. Ломая же стереотипы по мелочам или неточно следуя принятой концепции, разработчик рискует оттолкнуть пользователей, которые просто не будут понимать, что происходит. В качестве примера можно вспомнить хотя бы путаницу с вызовом программ двойным щелчком правой клавиши мыши по пиктограмме рабочего стола или одинарным, если пиктограммы вынесены на панель Quick Launch (Быстрый Доступ) Windows.

Критерии оценки интерфейса пользователем. Многочисленные опросы и обследования, проводимые ведущими фирмами по разработке программного обеспечения, показали, что основными критериями оценки интерфейсов пользователем являются:

- простота освоения и запоминания операций системы - конкретно оценивают время освоения и продолжительность сохранения информации в памяти;
- скорость достижения результатов при использовании системы - определяется количеством вводимых или выбираемых мышью команд и настроек;
- субъективная удовлетворенность при эксплуатации системы удобство работы, утомляемость и т. д.).

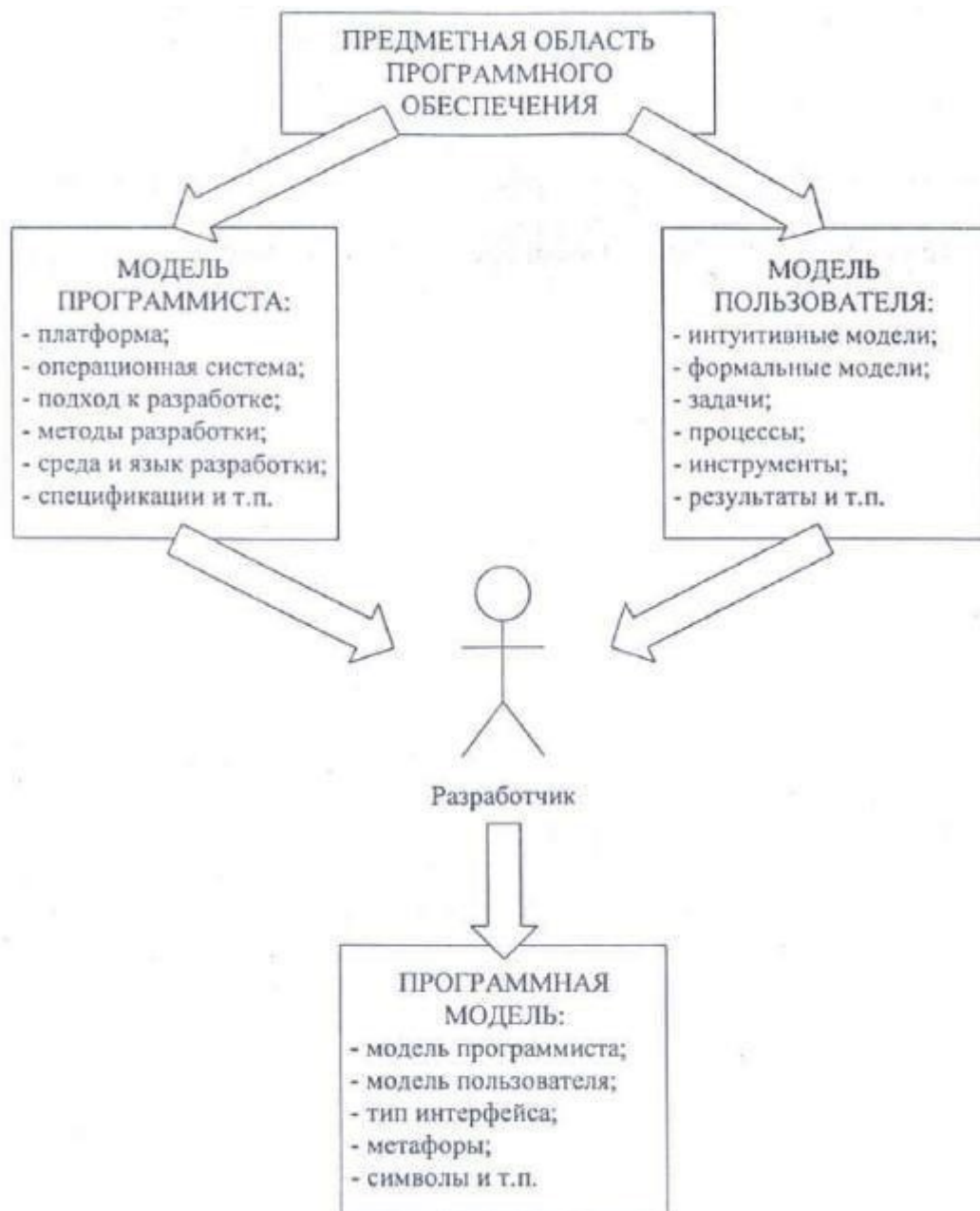


Рис.1.1. Процесс проектирования пользовательского интерфейса

Причем для пользователей профессионалов, постоянно работающих с одним и тем же пакетом, на первое место достаточно быстро выходят второй и третий критерии, а для пользователей непрофессионалов, работающих с программным обеспечением периодически и выполняющих сравнительно несложные задачи - первый и третий.

С этой точки зрения на сегодняшний день наилучшими характеристиками для пользователей профессионалов обладают интерфейсы со свободной навигацией, а для пользователей непрофессионалов - интерфейсы прямого манипулирования. Давно замечено, что при выполнении операции копирования файлов при прочих равных условиях большинство профессионалов используют оболочки типа Far, а непрофессионалы - «перетаскивание объектов» Windows.

Задания для выполнения №1

1. Разработать модель программиста, модель пользователя и программную модель для обучающе-контролирующей программы. Обучающе-контролирующая программа должна быть разработана в одной из сред ООП (Lazarus, Delphi, VB, C++ и т.д.) с использованием всех, ранее изученных классов, компонентов и методов. Также программа должна включать теоретический, практический и контролирующий блоки. Модели представить в виде одного из вариантов: 144 текстовое описание, схема, таблица.

2. Оформить работу в соответствии с требованиями ЕСПД (ГОСТ 19.101-77, ГОСТ 19.102-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.10578, ГОСТ 19.106-78, ГОСТ 19.401-78, ГОСТ 19.604-78). При оформлении использовать MS Office или OpenOffice.org.
3. Сдать и защитить работу.

Варианты заданий

1. Обучающе-контролирующая программа по Lazarus
2. Обучающе-контролирующая программа по Pascal
3. Обучающе-контролирующая программа по Word
4. Обучающе-контролирующая программа по Excel
5. Обучающе-контролирующая программа по Access
6. Обучающе-контролирующая программа по Power Point
7. Обучающе-контролирующая программа по Ramus
8. Обучающе-контролирующая программа по PhotoShop
9. Обучающе-контролирующая программа по теме «Компьютер и его ПО»
10. Обучающе-контролирующая программа по теме «История ВТ»
11. Обучающе-контролирующая программа по HTML

Содержание и оформление отчета по лабораторной работе Отчёт должен содержать:

1. Титульный лист.
2. Аннотацию.
3. Содержание.
4. Основную часть, оформленную в соответствии с требованиями ЕСПД (ГОСТ 19.10177, ГОСТ 19.102-77, ГОСТ 19.103-77, ГОСТ 19.104-78, ГОСТ 19.105-78, ГОСТ 19.106-78, ГОСТ 19.401-78, ГОСТ 19.604-78).
5. Заключение (описание результатов работы).

Защита отчета по лабораторной работе заключается в предъявлении преподавателю полученных результатов (на экране монитора или у доски), демонстрации полученных навыков и ответах на вопросы преподавателя

2. Разработка технического задания

Цель работы: изучение руководящих документов по составлению технического задания. Разработка проекта технического задания на программное обеспечение.

Теоретический материал

Общие сведения

Техническое задание (ТЗ, техзадание) - основной документ, содержащий требования заказчика к системе, в соответствии с которыми осуществляется создание и разработка конечного продукта.

Техническое задание на программу и программное обеспечение разрабатывается в соответствии с требованиями следующих документов:

- ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению;

- ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

Основанием для разработки ТЗ чаще всего является договор. Техническое задание позволяет:

- исполнителю - понять суть задачи, показать заказчику «технический облик» будущего изделия, программного изделия или автоматизированной системы;
- заказчику - осознать, что именно ему нужно;
- обеим сторонам - представить готовый продукт;
- исполнителю - спланировать выполнение проекта и работать по намеченному плану;
- заказчику - требовать от исполнителя соответствия продукта всем условиям,

оговоренным в ТЗ;

- исполнителю - отказаться от выполнения работ, не указанных в ТЗ;

- заказчику и исполнителю - выполнить полную попутную проверку готового продукта;
- избежать ошибок, связанных с изменением требований (на всех стадиях и этапах создания, за исключением испытаний).

Если поставлены сжатые сроки подготовки ТЗ и заказчик не требует оформления документации в соответствии с государственным стандартом, то можно использовать шаблон технического задания по стандарту IEEE Std 830, который предполагает, что детальные требования могут быть обширными и не существует оптимальной структуры для всех систем. По этой причине стандартом рекомендуется обеспечивать такое структурирование детальных требований, которое делает их оптимальными для понимания. Стандартом рекомендуются различные способы структурирования детальных требований для различных классов систем.

Существует и третья альтернатива для выбора шаблона технического задания, когда заказчик предлагает использовать принятый в компании корпоративный шаблон для описания требований к информационным системам.

Для внесения изменений или дополнений в техническое задание на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению

Данный стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

Согласно стандарту техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки; - в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия, допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

В разделе «Введение» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

В разделе «Основания для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка; - организация, утвердившая этот документ, и дата его утверждения; - наименование и (или) условное обозначение темы разработки.

Применительно к специфике учебного процесса основанием может служить задание на курсовое проектирование, приказ по институту от . . за №., договор . . за N., и т.п.

Наименование: «Разработка лабораторной работы № 1».

Шифр: «ЛАБА-001».

В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

Раздел «Требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;146 - требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке; - требования к транспортированию и хранению; - специальные требования.

1. В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т.п.

Например: Программа должна позволять ... вычислять ... строить... создавать ...

Исходные данные: текстовый файл с заданной ...

Входные и выходные данные : графическая и текстовая информация - результаты анализа системы.; текстовые файлы - отчеты о ... диагностика состояния системы и сообщения о всех возникших ошибках.

2. В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т.п.).

Для программного обеспечения надежность зависит не столько от исполнителя, сколько от надежности технических средств и операционной системы, поэтому в разделе обязательно следует написать такую фразу: «Требования к обеспечению надежного (устойчивого) функционирования программы не предъявляются»

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать нескольких минут при условии соблюдения условий эксплуатации технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Перечень аварийных ситуаций также составляет заказчик и согласовывает с исполнителем, т.е. это время на перезагрузку операционной системы, если отказ не фатален и не вызван крахом операционной системы или выходом из строя технических средств.

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных привилегий.

3. В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т.п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

Здесь можно ограничиться следующими фразами: «Условия эксплуатации программы совпадают с условиями эксплуатации ПЭВМ IBM PC и совместимых с ними ПК», «Программа должна быть рассчитана на непрофессионального пользователя» и т.п.

4. В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их основных технических характеристик.

Как правило, тут указывают минимальные требования к технике, на которой будет функционировать разработанное программное обеспечение. Например, IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

- процессор Pentium-1000 с тактовой частотой, ГГц - 10, не менее; -

оперативную память объемом, Гб - 2, не менее; - и так далее.

5. В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования и программным средствам, используемым 147 программой.

При необходимости должна обеспечиваться защита информации и программ.

Например: Программа должна работать автономно под управлением ОС MS Windows версии не ниже 7. Базовый язык программирования — C++.

6. В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

Например, программное изделие должно иметь маркировку с обозначением товарного знака компании-разработчика, типа (наименования), номера версии, порядкового номера, даты изготовления и номера сертификата соответствия Госстандарта России (если таковой имеется). Маркировка должна быть нанесена на программное изделие в виде наклейки, выполненной полиграфическим способом с учетом требований ГОСТ 9181-74.

Упаковка программного изделия должна осуществляться в упаковочную тару предприятия-изготовителя (поставщика).

Упаковка программного изделия должна проводиться в закрытых вентилируемых помещениях при температуре от +15 до +40 °С и относительной влажности не более 80 % при отсутствии агрессивных примесей в окружающей среде.

7. В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

Допускается транспортирование программного изделия в транспортной таре всеми видами транспорта (в том числе в отапливаемых герметизированных отсеках самолетов без ограничения расстояний). При перевозке в железнодорожных вагонах вид отправки - мелкий малотоннажный.

При транспортировании и хранении программного изделия должна быть предусмотрена защита от попадания пыли и атмосферных осадков. Не допускается кантование программного изделия. Климатические условия транспортирования приведены ниже:

- температура окружающего воздуха, °С - от ... до ...;
- атмосферное давление, кПа - ..; - относительная влажность воздуха при 25 °С -

В разделе «Требования к программной документации» должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

В состав программной документации должны входить:

- техническое задание;
- программа и методика испытаний;
- руководство системного программиста;
- руководство оператора; - ведомость эксплуатационных документов.

Программа и методика испытаний потребуется, чтобы показать заказчику, что разработанная исполнителем программа соответствует требованиям согласованного и утвержденного технического задания. После проведения совместных (приемо-сдаточных) испытаний заказчик и исполнитель подпишут акт завершения работы. Таким образом, работа будет закрыта, условия договора выполнены.

В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

В разделе «Стадии и этапы разработки» устанавливают необходимые стадии и сроки разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, определяют исполнителей.

Главное - грамотно определиться со сроками. По возможности, старайтесь равномерно распределить этапы по срокам (и суммам). Помните, что не все проекты доживают до последней стадии. А отчеты должны быть по каждому этапу. Помните также, что больше всего времени займет рабочий проект. Если вы не успеете сделать в срок документацию, то Заказчик имеет полное право вообще не принять работу со всеми вытекающими последствиями.

Основными и неперенными стадиями и этапами являются само техническое задание, эскизный проект, технический и рабочий проекты.

Эскизный проект. На этой стадии детально разрабатываются структуры входных и выходных данных, определяется форма их представления. Разрабатываются общее описание алгоритма, сам алгоритм, структура программы, план мероприятий по разработке и внедрению программы.

Технический проект. Содержит разработанный алгоритм решения задачи, а также методы контроля исходной информации. Здесь же разрабатываются средства обработки ошибок и выдачи диагностических сообщений, определяются формы представления исходных данных и конфигурация технических средств.

Рабочий проект. На этой стадии осуществляются программирование и отладка программы, разработка программных документов, программы и методики испытаний. Подготавливаются контрольно-отладочные примеры. Окончательно оформляются документация и графический материал.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

Например, контроль и приемка разработки осуществляются на основе испытаний контрольно-отладочных примеров. При этом проверяется выполнение всех функций программы. В приложениях к техническому заданию при необходимости приводят:

- перечень научно-исследовательских и других работ, обосновывающих разработку; - схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
- другие источники разработки. **Задание на лабораторную работу**

Разработать проект технического задания на программное обеспечение в соответствии с ГОСТ 19.201-78. Программное обеспечение должно включать в свой состав:

- серверную часть;
- клиента, функционирующего на ПЭВМ;

- мобильного клиента, функционирующего на смартфоне; - веб-клиента, функционирующего в браузере.

По желанию состав программного обеспечения может быть расширен.

Требования к функциональным характеристикам необходимо указать для каждого клиента.

Вопросы для самопроверки 1.

Зачем разрабатывают ТЗ?

2. Какими стандартами регулируется содержимое технического задания?
3. Какие существуют стадии и этапы разработки?
4. Раскройте понятие «время восстановления после отказа».
5. Что должен содержать подраздел «Требования к функциональным характеристикам» раздела «Требования к программе или программному изделию»?
6. Обязательно ли присваивать условное обозначение темы разработки?

3. Разработка интерфейса пользователя. Проектирование пользовательского интерфейса десктопного приложения . Проектирование пользовательского интерфейса десктопного приложения.

Цель работы: изучение принципов проектирования пользовательского интерфейса. Разработка пользовательского интерфейса десктопного приложения.

Теоретический материал

Общие сведения о десктопных приложениях

Десктопные приложения - это программы, требующие наличия оператора (человека, работающего с программой), содержащие в себе всю полную функциональность и способные работать отдельно на любой машине изолированно от других приложений. Microsoft Word, Excel, Блокнот, однопользовательские игры - все это примеры десктопных приложений. Для их работы необходимы лишь достаточные аппаратные ресурсы компьютера, само приложение и набор библиотек, содержащих функции для работы с приложением.

Десктопные приложения могут быть также и многопользовательскими. Например, редактор файлов, который в зависимости от логина и пароля, введенных при запуске, будет давать доступ к различным файлам. И программа, и файлы находятся на одном компьютере, просто производится локальное разграничение доступа для разных пользователей.

Пользовательский интерфейс

Пользовательский интерфейс (UI) - это способ, которым вы выполняете какую-либо задачу с помощью какого-либо продукта, т.е. совершаемые вами действия и то, что вы получаете в ответ.

Программный интерфейс не только решает нашу проблему взаимодействия с приложением, но и делает это взаимодействие максимально комфортным. Нам важно наличие интерфейса, позволяющего при меньшем количестве усилий ознакомиться с возможностями приложения и понять принципы работы в нем.

Чтобы не возникло проблем при использовании какого-либо приложения, можно визуализировать его функциональные возможности в виде понятных элементов, и за этой визуализацией кроется целая кухня UX/UI-дизайна.

Грань между UX (**User Experience**) и UI (**User Interface**) очень тонка, но если разобраться, то становится ясно, что UX помогает понять пользователя. В UX-дизайне больше психологического аспекта, нежели технологического. UX изучает пользователя: как пользователь живет, что он думает, как и что делает, что его окружает. Перед дизайнером ставится задача - помочь обычному человеку легко разобраться с вашим программным продуктом и получить при этом удовлетворение от работы с ним.

А понять пользователя очень важно. Никому не захочется заполнить двадцать полей формы для регистрации на сайте или перешелкать штук пятнадцать вкладок, прежде чем добраться до нужной функции. «Пользователя не следует заставлять взаимодействовать с программой дольше, чем абсолютно необходимо для решения той или иной задачи» (из книги Алана Купера «Психбольница в руках пациентов»). **Этапы разработки пользовательского интерфейса**

Полный цикл разработки интерфейса представлен на рис. 2.1.



Рис. 2.1. Этапы разработки пользовательского интерфейса

Для сокращения общего времени разработки определение стилистики начинается после пользовательских сценариев.

Исследование. На этапе исследования проводится сбор информации о продукте, клиенте, его конкурентах или близких аналогах, сбор статистики использования текущего интерфейса (например, сайта или мобильного приложения), анализ устройств предполагаемой целевой аудитории.

Если уже известно, кто будет воплощать интерфейс в жизнь (разработчики), то знакомимся с ними и выясняем их возможности и ограничения.

Этот этап помогает понять, для кого разрабатывается интерфейс, с какими ограничениями следует его делать (размеры экранов, интерактивность), как не стоит делать (например, быть непохожими на конкурентов).

Пользовательские сценарии. На основе предоставленного описания работы интерфейса создается список задач (пользовательских сценариев), которые может выполнять пользователь в рамках интерфейса. Например, обновить аватарку в профиле.

Все задачи расписываются по шагам, которые необходимо предпринять для решения задачи. Например:

- зайти на сайт;
- авторизоваться;
- перейти в профиль;
- нажать на аватарку;
- выбрать файл;
- подтвердить или изменить кадрирование изображения; - сохранить.

Составленные списки шагов для каждой задачи помогают понять, где путь для решения слишком долг относительно остальных задач. Этап пользовательских сценариев больше всего подходит для сокращения пути решения задач пользователей в рамках интерфейса.

Пример выше можно сократить на несколько шагов. Например, сделать сохранение автоматическим, а обрезание изображения - опциональным.

Структура интерфейса. Полученный список шагов на предыдущем этапе ложится в основу структуры интерфейса. Становится известно количество экранов, их краткое содержание и положение в общей структуре. На данном этапе строится карта экранов (*User Flow Diagram*).

Пример карты экранов приведен на рис. 2.2.

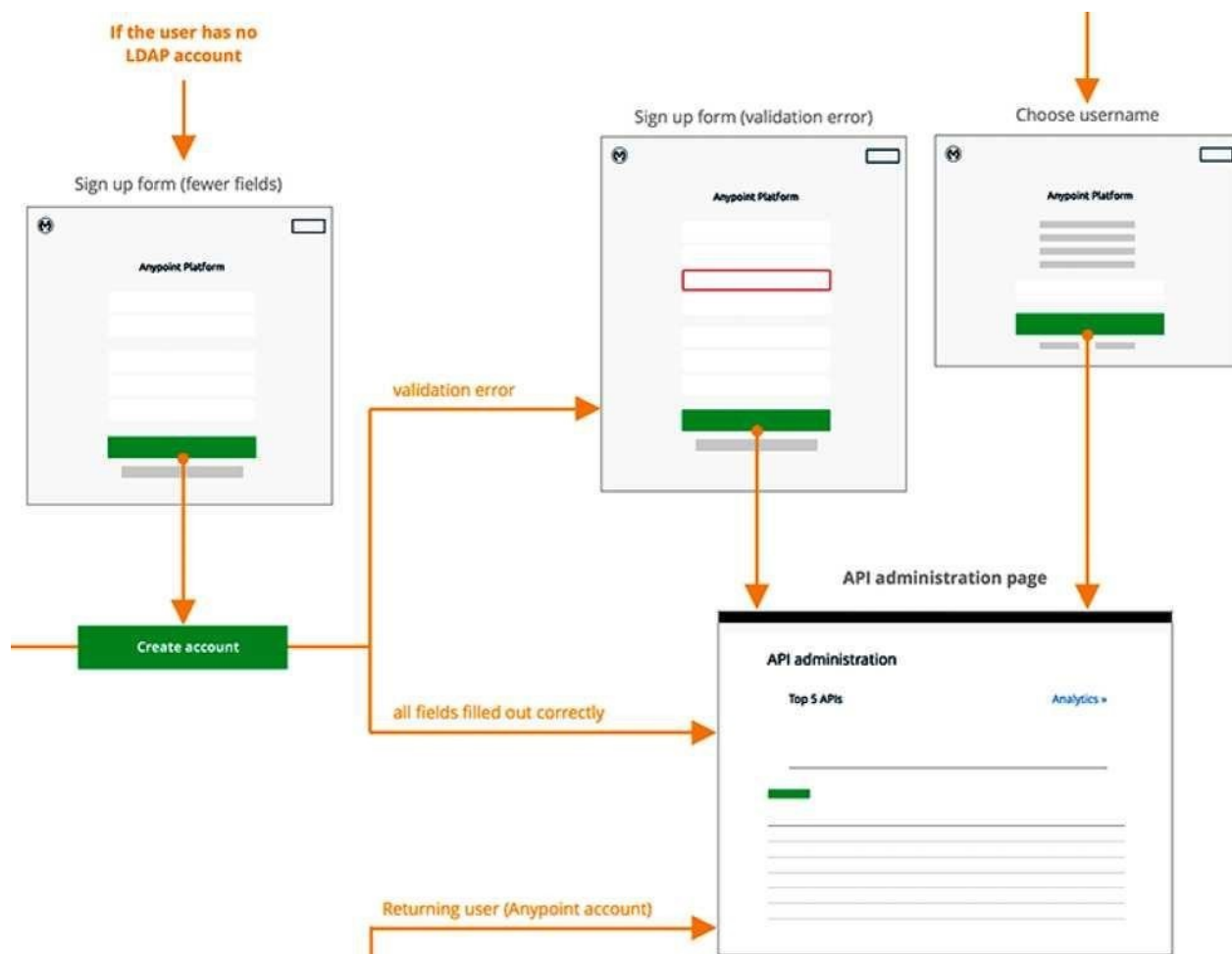


Рис. 2.2. Фрагмент карты экранов (User Flow Diagram)

Прототипирование интерфейса. В большинстве случаев реализуется два схематичных прототипа: черновой и финальный. Исключения составляют небольшие интерфейсы: простенькие мобильные приложения или маленькие сайты.

Черновой прототип представляет собой схематичные изображения экранов, связанные между собой. При черновом варианте на схемах изображены зоны и описания этих зон. Например, список новостей или шапка сайта. Все без деталей.

Черновой прототип помогает более наглядно понять, насколько объемным будет приложение, как много информации будет на каждом экране, как много нужно кликать, чтобы добраться до нужной страницы.

Следующим шагом идет финальный прототип, в котором схемы страниц все еще остаются связанными между собой, но на страницах уже видны все кнопки, тексты, чекбоксы, формы и прочие элементы.

Пример чернового прототипа интернет-магазина приведен на рис. 2.3.

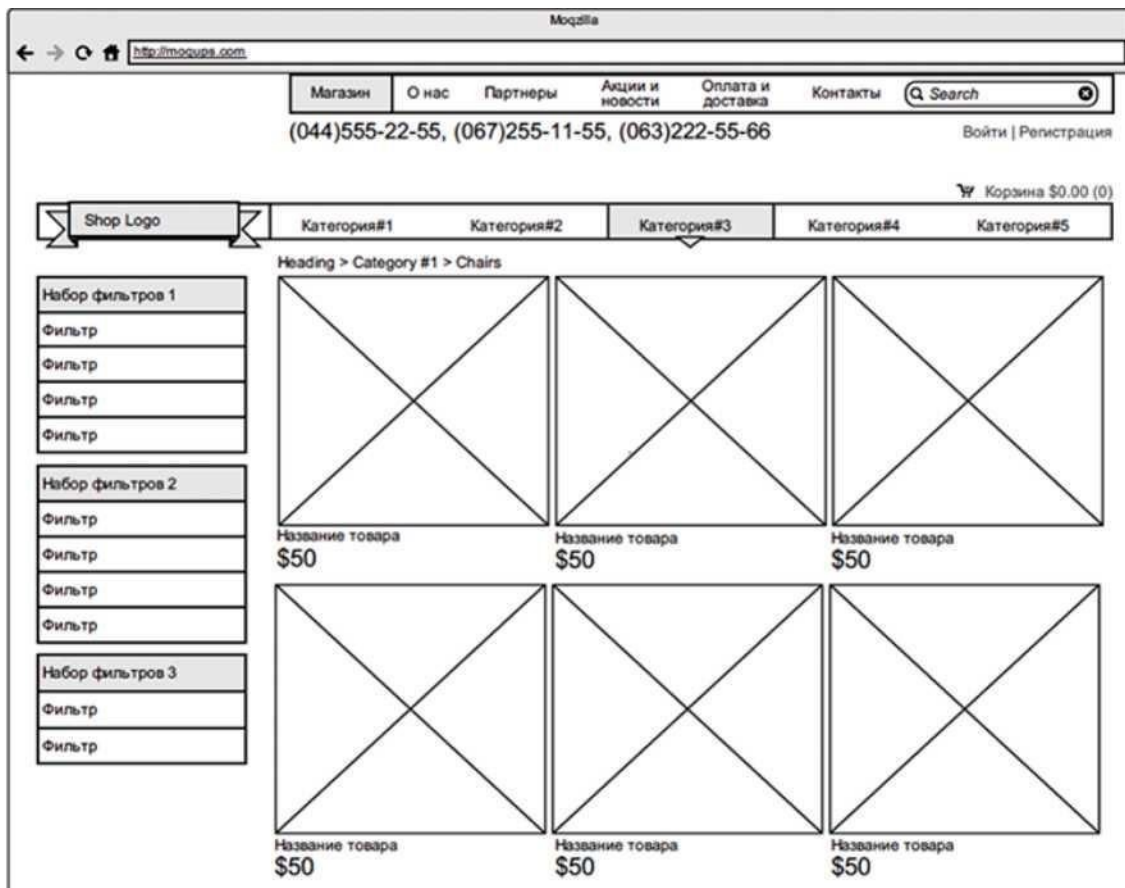


Рис. 2.3. Пример черного прототипа интернет-магазина

В прототипах планируется функционал, расположение элементов страниц относительно друг друга, но никак не оформление. Цвета, изображения, иконки - это все этап оформления. На этапе проектирования невозможно сказать, как они будут взаимодействовать между собой, как будут смотреться вместе, будут ли перекрывать друг друга.

Определение стилистики. После этапа исследования и параллельно с этапами проектирования идет определение будущей стилистики интерфейса.

Для выбора стилистики готовятся несколько наборов изображений (*moodboards*). Эти наборы представлены страничками сайтов, иллюстрациями, кнопками, шрифтовыми композициями, связанными между собой стилистически.

Существует множество различных концепций, например: *material design*, *metro*, *skeuomorphism* и т.д. При выборе стиля интерфейса следует учесть текущие тенденции в дизайне, адаптивность, время на разработку и внедрение дизайна, и много других не менее важных моментов.

UI-kit - набор готовых решений пользовательского интерфейса. Это могут быть кнопки, поля ввода, «хлебные крошки», меню, переключатели, формы - все те элементы, что помогают пользователям взаимодействовать с сайтом или приложением.

Пример набора элементов из *Flat UI-kit* приведен на рис. 2.4.

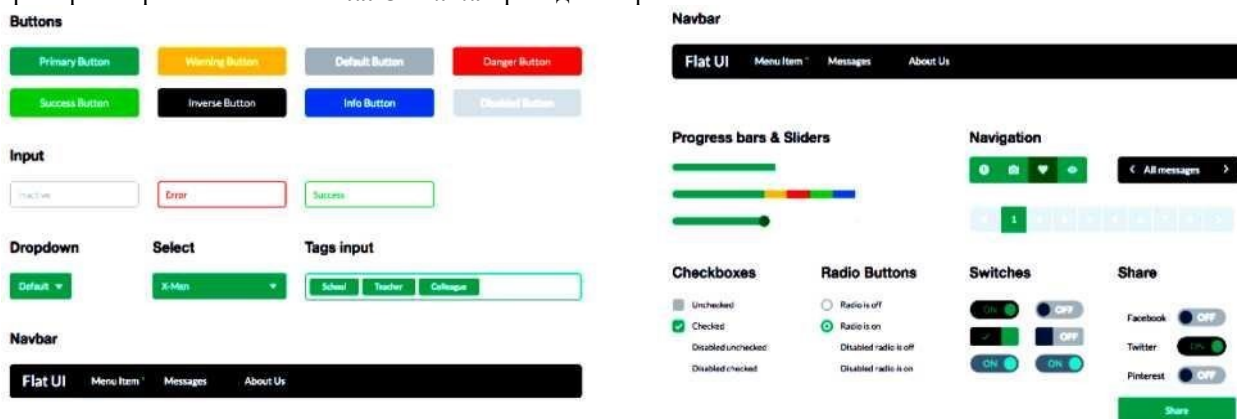


Рис. 2.4. Пример набора элементов из *Flat UI*-кита

Обычно готовый кит представляет собой графику в слоях для работы в **Photoshop** или **Sketch**. В документе хранятся элементы для дизайна интерфейсов, которые можно использовать в неизменном виде или редактировать их и подстраивать под стиль проекта.

Дизайн-концепция. Она призвана показать оформление приложения и будущий вид всего приложения. Если предыдущий этап определения стилистики только дал направление, то дизайн-концепция призвана скрестить выбранное направление с имеющимся содержанием интерфейса.

Дизайн-концепция может быть представлена любым объемом, но стараются его минимизировать для экономии времени. Обычно концепция представлена 1-3 экранами интерфейса. Если речь идет о сайте, то стараются показать вид одной и той же страницы для нескольких устройств.

Для разработки дизайн-концепции используются online-инструменты:

- <https://www.axure.com>
 - <http://mockupbuilder.com> - <https://www.fluidui.com> -
- и т.д.

Оформление всех экранов. После утверждения дизайн концепции настает время оформления всех остальных экранов интерфейса. Дизайн- концепция - это предположение о том, как может выглядеть весь интерфейс. Когда же очередь доходит до оформления всех экранов, тогда и происходит финализация внешнего вида: становится ясно, правильно ли подобраны кегль или интерлиньяж, хорошо ли сочетается толщина линий иконок с текстом, не конфликтует ли оформление форм (кнопок, полей ввода) с другими элементами экрана и многое другое.

Планом для оформления всех экранов являются структура и схематичный прототип интерфейса. Однако случаются отхождения от этого плана. Так при оформлении может выясниться, что всплывающее окно будет намного нагляднее и эффективнее, чем разъезжающийся блок информации посреди экрана.

Все оформленные экраны собираются в интерактивный прототип, который создаст максимально приближенный опыт использования интерфейса без прибегания к услугам разработчиков.

Анимация интерфейса. Часто этот этап начинается еще с момента дизайн-концепции и продолжается на протяжении оформления всех экранов.

Стараются показать только какие-либо нестандартные случаи анимации интерфейса, которые не предусмотрены операционной системой. Например, нету никакой надобности показывать, с какой скоростью будет выезжать следующий экран в интерфейсе приложения. Однако это тоже можно считать анимацией интерфейса.

Для **Material design** есть гайдлайны, которые наглядно объясняют, как надо анимировать и как не надо.

Эти гайдлайны подходят для анимации интерфейсов любой платформы.

Подготовка материалов для разработчиков. На данном этапе уже присутствуют макеты интерфейса во всех состояниях, прототип, связывающий весь интерфейс воедино и видеоролики, показывающие анимацию. Чтобы помочь разработчикам в реализации интерфейса, дизайнеры готовят все необходимые для этого материалы:

- спрайты;
- шрифт со всеми иконками; - **UI-Kit** с повторяющимися элементами интерфейса и их состояниями.

Для иконок и прочей графики из интерфейса, для всех расстояний, отступов, размеров используют специальное программное обеспечение, например, Zeplin, которое самостоятельно готовит иконки и код.

Задание на лабораторную работу

Разработать пользовательский интерфейс для десктопного клиента, описанного в лабораторной работе № 1. Разработку пользовательского интерфейса производить согласно п. 2. настоящего пособия.

1. Произвести анализ аналогов, выявить их достоинства и недостатки, результаты анализа отразить в отчете.
2. Разработать пользовательские сценарии и привести их часть в отчете.
3. Разработать карту экранов в части описанных пользовательских сценариев.
4. Разработать черновой прототип экранов.
5. Подобрать подходящие стилистики для приложения не менее двух.
6. На основании одной из стилистик разработать дизайн концепцию приложения.

Вопросы для самопроверки

1. В чем отличие понятий UI и UX?
2. Какие этапы включает разработка пользовательского интерфейса?
3. Что такое UI-кит?

4. Для чего разрабатывают дизайн-концепцию?
5. В чем отличие чернового прототипа интерфейса от финального?

4. Адаптивный веб-дизайн.

Цель работы: изучить принципы проектирования пользовательских интерфейсов сайта, обеспечивающих его правильное отображение на различных устройствах, подключенных к интернету и динамически подстраивающихся под заданные размеры окна браузера. **Теоретический материал**

Общие сведения

Целью адаптивного веб-дизайна (*responsive web design, RWD*) является универсальность отображения содержимого веб-сайта для устройств, для того чтобы веб-сайт был удобно просматриваемым с устройств различных форматов и с экранами различных разрешений. По технологии адаптивного веб-дизайна не нужно создавать отдельные версии веб-сайта для отдельных видов устройств. Один сайт может работать на смартфоне, планшете, ноутбуке и телевизоре с выходом в Интернет, т.е. на всем спектре устройств. Пример применения адаптивного дизайна приведен на рис. 4.1



Рис. 4.1. Пример адаптивного веб-дизайна

Применение адаптивного веб-дизайна обусловлено следующими аспектами:

большое разнообразие устройств, с которых можно выходить в Интернет. В настоящее время существует множество устройств, которыми люди пользуются, в том числе и для того, чтобы выходить в Интернет. Все они различаются размером экрана, разрешением и, соответственно, тем, как может отображаться на них веб-сайт. Поэтому важно, чтобы ваш сайт хорошо смотрелся и правильно отображался у любого из пользователей, независимо от того, какое устройство он использует; *популярность мобильных устройств с выходом в Интернет и увеличение мобильного интернет-трафика.* С ростом популярности мобильных устройств количества пользователей, которые заходят с них на сайты, заметно увеличилось, поэтому просто игнорировать их уже нельзя - это не один-два человека в полгода, это значительная часть вашей аудитории, и им должно быть удобно пользоваться вашим сайтом (иначе они не будут этого делать); *срочная информация.* Если ваш ресурс содержит новостную/ срочную информацию, высокая вероятность, что пользователю может понадобится прочитать эту информацию именно с телефона (потому что других устройств у него под рукой нет) в данный момент времени, нужно позаботиться о том, чтобы у него была возможность это сделать.

Отличие адаптивного сайта от мобильной версии (приложения) сайта

Мобильные версии сайтов и мобильные приложения, специально разработанные для различных мобильных устройств, также решают проблему с удобством просмотра сайта, но имеют некоторые недостатки: *под каждый тип операционной системы нужно свое приложение/ версия сайта.* Это требует дополнительных ресурсов как временных, так и денежных; *необходимость загрузки приложения.* Для того, чтобы пользоваться вашим приложением, пользователям необходимо его загрузить. Это требует каких-то дополнительных усилий от пользователей, и многие не будут этого делать, если точно не уверены, что приложение им очень нужно, и они планируют регулярно его использовать; *разделение трафика.* С точки зрения продвижения сайта мобильное приложение не удобно тем, что разделяет весь трафик ресурса на трафик сайта и трафик приложения, что будет выглядеть, как меньшая посещаемость сайта; *необходимость интеграции материалов*

сайта. В случае с мобильным приложением необходимо либо синхронизировать сайт с приложением (дополнительные ресурсы), либо делать двойную работу по наполнению сайта и приложения материалами.

В отличие от мобильных приложений, адаптивный дизайн - это один адрес сайта, один дизайн, одна система управления и содержание сайта.

Недостатки адаптивного дизайна

Адаптивный дизайн может дать многочисленные преимущества, но абсурдно предположение, что он подходит для всех веб-проектов. Можно выделить следующие недостатки:

зачастую адаптивный дизайн приводит к неоправданному увеличению времени загрузки сайта на мобильных устройствах. Концепция RWD подразумевает, что пользователи всех платформ получают одинаковый контент, оптимизированный под конкретные разрешения экранов.

В то же время на сайте может быть масса информации, показывать которую мобильным пользователям нет никакой необходимости. Однако она загружается автоматически, как только они заходят на сайт, и зачастую это происходит в местности со слабым сигналом сотовой связи; пользователи не смогут посмотреть полную версию. На большинстве неадаптивных мобильных сайтов внизу страницы присутствует опция «Переключиться на полную версию» или другая аналогичная кнопка. Она позволяет посетителям просматривать сайт для настольных компьютеров, минуя таблицы стилей для мобильной версии. Пример для сайта kinopoisk.ru приведен на рис. 4.2.

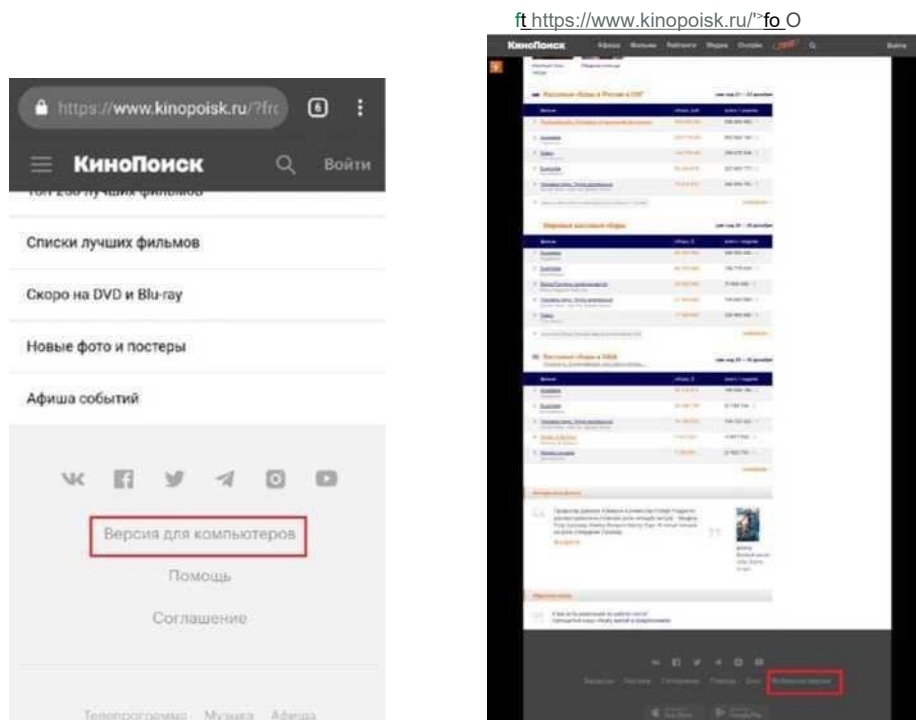


Рис. 4.2. Пример перехода к разным версиям сайта

Принципы адаптивности

Основными принципами при проектировании адаптивного веб-дизайна являются:

1) *проектирование для мобильных устройств с самых ранних этапов ("mobile first")*. Проектирование начинается с адаптивной версии веб-сайта для мобильных устройств. На этом этапе дизайнеры стремятся правильно передать смысл и основные идеи с использованием небольшого экрана и всего одной колонки. Содержимое при необходимости сокращают, удаляя второстепенные информационные блоки и оставляя самое важное;

2) *работа с медиазапросами (media queries)*. Запросы определяют: – тип устройств: проекторы, смартфоны, мониторы, телевизоры и пр.; – условия.

На соответствующий запрос и ответ будут применяться соответствующие устройству параметры отображения из файла стилей css;

3) *применение гибкого макета на основе сетки (flexible, grid-based layout) и использование гибких изображений (flexible images)*.

Рассмотрим основные виды адаптивных макетов, существующие в настоящее время:

а) *резиновый*. Простой в реализации и очевидный для пользователя тип представления сайта. Основные блоки сжимаются до ширины экрана мобильного устройства, где такое невозможно - перестраиваются в одну длинную ленту;

б) *перенос блоков*. Очевидный способ для многоколоночного сайта: при уменьшении ширины экрана дополнительные блоки (сайдбары) переносятся в нижнюю часть макета (рис. 4.3);

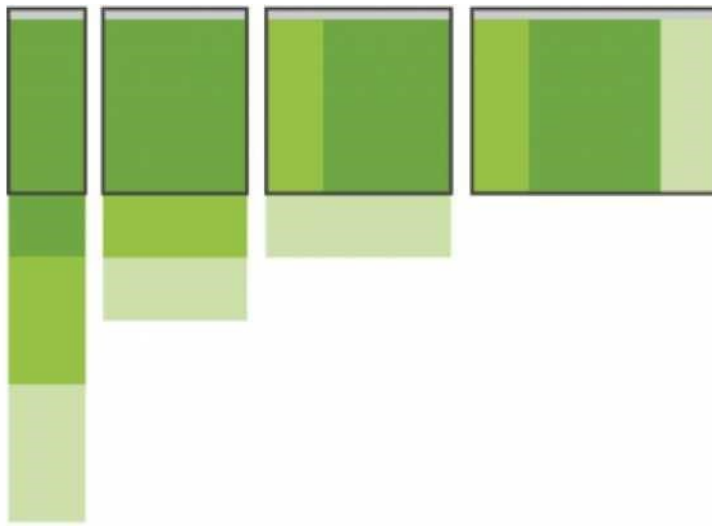


Рис. 4.3. Способ «перенос блоков»

в) *переключение макетов*. Этот способ наиболее удобен при чтении сайта с различных устройств: под каждое разрешение экрана разрабатывается отдельный макет. Способ трудоемок, поэтому менее популярен, чем предыдущие два (рис. 4.4);

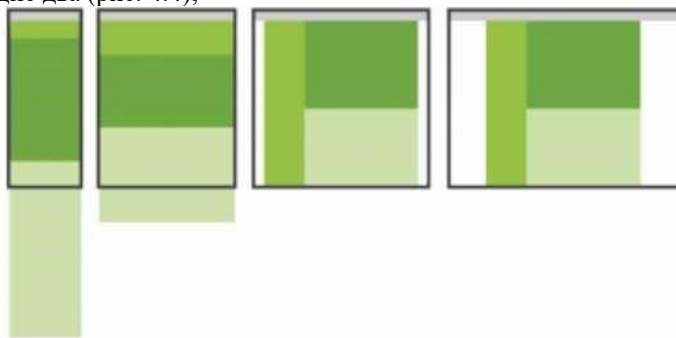


Рис. 4.4. Способ «переключение макетов»

г) *адаптивность «малой кровью»*. Очень простой способ, который подходит для несложных сайтов. Достигается элементарным масштабированием изображений и типографики. Не очень популярен, так как не обладает гибкостью (рис. 4.5);

UULM

Рис. 4.5. Способ «адаптивность «малой кровью»

д) *панели*. Способ, пришедший из мобильных приложений, где дополнительное меню может появляться при горизонтальном или вертикальном этапе. Главный недостаток - неочевидность действий для пользователя: очень непривычно видеть мобильную навигацию на веб-сайте (рис. 4.6).

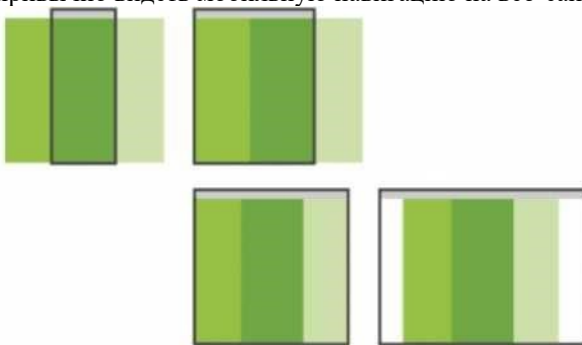


Рис. 4.6. Способ «панели»

Нужно помнить, что представленные выше макеты не являются универсальными решениями - для каждого проекта необходимо выбирать наиболее подходящий под нужды и возможности способ.

Задание на лабораторную работу

Разработать пользовательский интерфейс для веб-клиента, описанного в лабораторной работе № 1. Подготовить макеты для отображения не менее трех страниц для ширины экрана следующего разрешения:

- для смартфонов 320 px, 480 px;
- планшетов 768 px;
- нетбуков и некоторых планшетов 1024 px; - персональных компьютеров 1280 px и более.

Вопросы для самопроверки

1. Раскройте понятие «адаптивный веб-дизайн».
2. Зачем нужен адаптивный веб-дизайн?
3. В чем отличие адаптивного сайта от мобильной версии (приложения) сайта?
4. Каковы недостатки адаптивного веб-дизайна?
5. Перечислите основные виды адаптивных макетов.

5. Разработка протокола взаимодействия веб-сервисов

Цель работы: изучить принципы проектирования протокола взаимодействия вебсервисов с использованием протокола SOAP.

Теоретический материал

Веб-сервисы

Веб-служба, веб-сервис ([англ. web service](#)) - идентифицируемая уникальным веб-адресом (*URL-адресом*) программная система со стандартизированными интерфейсами, а также *HTML-документ* сайта, отображаемый браузером пользователя.

Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определенных протоколах (*SOAP, XML-RPC* и т.д.) и соглашениях (*REST*). Вебслужба является единицей модульности при использовании сервис-ориентированной архитектуры приложения.

На сегодняшний день наибольшее распространение получили следующие протоколы реализации веб-сервисов:

- *SOAP (Simple Object Access Protocol)* — по сути это тройка стандартов *SOAP/WSDL/UDDI*; - *REST (Representational State Transfer)*; - *XML-RPC (XML Remote Procedure Call)*.

На самом деле *SOAP* произошел от *XML-RPC* и является следующей ступенью его развития. В то время как *REST* - это концепция, в основе которой лежит скорее архитектурный стиль, нежели новая технология, основанный на теории манипуляции объектами *CRUD (Create Read Update Delete)* в контексте концепций *WWW*.

Концепция построения веб-сервисов с использованием SOAP

Веб-сервис идентифицируется строкой *URI*. Веб-сервис имеет программный интерфейс, представленный в машинно-обрабатываемом формате [WSDL](#). Другие системы взаимодействуют с этим веб-сервисом путем обмена сообщениями протокола [SOAP](#). В качестве транспорта для сообщений используется протокол [HTTP](#). Описание веб-сервисов и их *API* могут быть найдены средствами [UDDI](#). Концептуальная схема технологии приведена на рис. 5.1, где:

- *SOAP (Simple Object Access Protocol)* - протокол обмена сообщениями между потребителем и поставщиком веб-сервиса;
- *WSDL (Web Services Description Language)* - язык описания внешних интерфейсов вебслужбы;
- *UDDI (Universal Discovery, Description and Integration)* - универсальный интерфейс распознавания, описания и интеграции, используемый для формирования каталога веб-сервисов и доступа к нему.



Рис. 5.1. Концепция веб-сервиса

Связь между протоколами приведена на рис. 5.2.

UDDI

Публикация и поиск сервисов

WSDL

x Описание интерфейсов SOAP

Обмен сообщениями

HTTP, SMTP, FTP, POP ...

Транспортная инфраструктура

Рис. 5.2. Протоколы веб-сервисов

Все спецификации, используемые в технологии, основаны на XML и, соответственно, наследуют его преимущества (структурированность, гибкость и т.д.) и недостатки (громоздкость, медлительность).

SOAP

SOAP - простой протокол доступа к объектам (компонентам распределенной вычислительной системы), основанный на обмене структурированными сообщениями. Как любой текстовый протокол, SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTPS и др., но чаще всего SOAP используется поверх HTTP.

Все сообщения SOAP оформляются в виде структуры, называемой конвертом (*envelop*), включающей следующие элементы:

- идентификатор сообщения (локальное имя);
- опциональный элемент Header (заголовок);
- ноль или более ссылок на используемые пространства имен;
- ноль или более свойств, доступных в этом пространстве имен; - обязательный элемент Body (тело сообщения);
- ноль или более ссылок на используемые пространства имен; - дочерние элементы тела сообщения.

Пример SOAP-запроса на сервер интернет-магазина:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<getProductDetails xmlns="http://warehouse.example.com/ws">
<productID>12345</productID>
</getProductDetails>
</soap:Body>
</soap:Envelope> Пример
```

ответа:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"163 /"> <soap:Body>
<getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
<getProductDetailsResult>
<productID>12345</productID>
<productName>CTaKaH rpaHeHbiA</productName>
<description>CTaKaH граненый. 250 M.n.</description>
<price>9.95</price>
```

```

<currency>
  <code>840</code>
  <alpha3>USD</alpha3>
  <sign>$</sign>
  <name>US dollar</name>
  <accuracy>2</accuracy>
</currency>
<inStock>true</inStock>
</getProductDetailsResult>
</getProductDetailsResponse>
</soap:Body>
</soap:Envelope>

```

WSDL

Язык описания веб-сервисов (*Web services Description Language*, WSDL) предназначен для унифицированного представления внешних интерфейсов веб-службы. Текущая версия протокола (на момент написания этой лекции) [WSDL 2.0](#) и она имеет некоторые отличия от предыдущих версий приведенные в табл. 5.1 и на рис. 5.3.

Таблица 5.1

Основные элементы протокола WSDL

Элемент WSDL 1.1	Элемент WSDL 2.0	Краткое описание
PortType	Interface	Представляет описание интерфейса веб-сервиса (список операций и их параметров)
Service	Service	Список системных функций
Binding	Binding	Специфицирует интерфейсы и задает параметры связывания с протоколом SOAP: стиль связывания (RPC/Document) и транспорт (SOAP). Эта секция доступна и для каждой из операций
Operation	Operation	Определяет операцию, представляемую веб-сервером. WSDL операция — это аналог традиционным функциям и процедурам
Message	—	Сообщение, связанное с определенной операцией. Содержит информацию, необходимую для выполнения данной операции. Каждое сообщение может состоять из нескольких логических частей, описывающих типы данных и имена атрибутов. В версии 2.0 было исключено, так как была внедрена поддержка XML Schema для всех элементов
Types	Types	Описание данных в соответствии с XML Schema

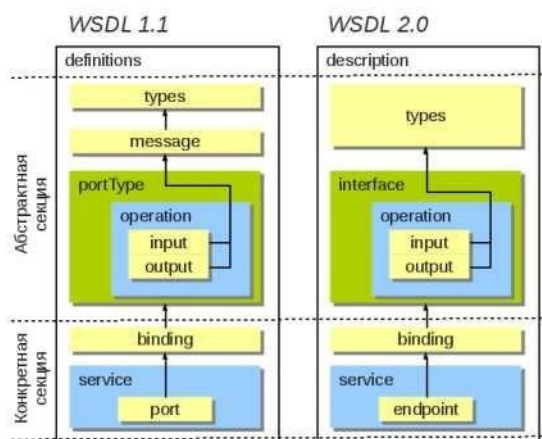


Рис. 5.3. Структура протокола WSDL

В спецификации WSDL 1.1 было определено четыре шаблона обмена сообщениями (типы операций):

- однонаправленные операции (One-way): операция может принимать сообщение, но не будет возвращать ответ;
- запрос-ответ (Request-response): операция может принять запрос и должна вернуть ответ; - вопрос-ответ (Solicit-response): операция может послать запрос и будет ждать ответ на него; - извещение (Notification): операция может послать сообщение, но не будет ожидать ответ.

В версии WSDL 2.0 эти шаблоны изменены и расширены в сторону поддержки сообщений об ошибках (например, шаблон Robust-in-only), но для обратной совместимости поддерживаются типы WSDL 1.1.

Пример описания веб-сервиса на языке WSDL (версия 2.0): <?xml version="1.0"?>

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl" xmlns:soap=
"http://www.w3.org/ns/wsdl/soap" xmlns:hy="http://www.herongyang.com/Service/"
targetNamespace="http://www.herongyang.com/Service/">
```

```
<wsdl:documentation>
```

```
Hello_WSDL_20_SOAP.wsdl
```

```
Copyright (c) 2009 HerongYang.com, All Rights Reserved.
```

```
</wsdl:documentation>
```

```
<wsdl:types>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.herongyang.com/Service/"> <xsd:element
name="Hello" type="xsd:string"/>
```

```
<xsd:element name="HelloResponse" type="xsd:string"/>
```

```
</xsd:schema>
```

```
</wsdl:types>
```

```
<wsdl:interface name="helloInterface" > <wsdl:operation
name="Hello" pattern="http://www.w3.org/ns/wsdl/in-out"
style="http://www.w3.org/ns/wsdl/style/iri">
```

```
<wsdl:input messageLabel="In" element="hy:Hello" />
```

```
<wsdl:output messageLabel="Out" element="hy:HelloResponse" />
```

```
</wsdl:operation>
```

```
</wsdl:interface>
```

```
<wsdl:binding name="helloBinding" interface="hy:helloInterface"
type="http://www.w3.org/ns/wsdl/soap"
```

```
soap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP165" />
```

```
<wsdl:operation ref="hy:Hello"
```

```
soap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/> </wsdl:binding>
```

```
<wsdl:service name="helloService"
```

```
interface="hy:helloInterface"> <wsdl:endpoint
```

```
name="helloEndpoint" binding="hy:helloBinding"
```

address="http://www.herongyang.com/Service/Hello_SOAP_12.php/"> </wsdl:service>
</wsdl:description> В данном примере:

- веб-сервис helloService определен с эндпойнтом helloEndpoint, доступным по адресу http://www.herongyang.com/Service/Hello_SOAP_12.php/;
- эндпойнт helloEndpoint ссылается на связывание helloBinding;
- связывание helloBinding определено с использованием протокола SOAP 1.2 поверх HTTP; - связывание helloBinding ссылается на интерфейс helloInterface;
- интерфейс helloInterface определяет операцию Hello, которая требует элементы входящего и исходящего сообщений;
- каждый элемент Hello/HelloResponse определяет в XML-схеме секции types . Пример описания сложных структур и списков:

```
<types>
<xs:schema xmlns:tns="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xs="http://www.w3.org/2001/XMLSchema" element- FormDefault="qualified"
targetNamespace="http://localhost/">
<complexType name="Message">
<sequence>
<element name="phone" type="string" minOccurs="1" maxOccurs="1"
/>
<element name="text" type="string" minOccurs="1" maxOccurs="1" />
<element name="date" type="dateTime" minOccurs="1" maxOccurs="1"
/>
<element name="type" type="decimal" minOccurs="1" maxOccurs="1"
/>
</sequence>
</complexType>
<complexType name="MessageList">
<sequence>
<element minOccurs="1" maxOccurs="unbounded" name="message"
type="Message"/> </sequence>
</complexType>
<element name="Request">
<element name="messageList" type="MessageList" />
</element>
<element name="Response">
<complexType>
<sequence>
<element name="status" type="boolean" />
</sequence>
</complexType>
</element>
</xs:schema> </types>
```

В данном примере представлено:

- описание структуры Message, состоящей из полей phone, text, date, type;
- описание списка сообщений MessageList, состоящего из неограниченного количества элементов типа Message;
- описание запроса Request со списком сообщений;
- описание ответа Response с булевской переменной, содержащей статус выполнения запроса.

UDDI

Universal Description, Discovery and Integration (UDDI - универсальный интерфейс распознавания, описания и интеграции) - открытый стандарт, утвержденный OASIS, определяющий методы публикации и обнаружения сетевых программных компонентов сервисориентированной архитектуры ([SOA](#)). В практической реализации UDDI представляет собой сетевой реестр ([службу каталогов](#)), представляющий данные и метаданные о веб-сервисах и доступный по адресу: URL: [http:// uddi.xml.org/services](http://uddi.xml.org/services).

Задание на лабораторную работу

Разработать протокол взаимодействия сервера и клиентов, описанных в лабораторной работе № 1:

1. Подготовить WSDL-описание команд сервера.
2. Подготовить примеры SOAP запросов и ответов в соответствии с WSDL-описанием.

Вопросы для самопроверки

1. Что такое веб-сервис?
2. Что такое сервис-ориентированная архитектура?
3. Какие протоколы реализации веб-сервисов получили наибольшее распространение?
4. Опишите структуру SOAP сообщения.
5. В чем отличие спецификаций WSDL 1.1 от WSDL 2.0?

Тема 1.1.8 Основы ADO.Net

1. Теоретические основы Технологии ado.Net

Теоретическая часть

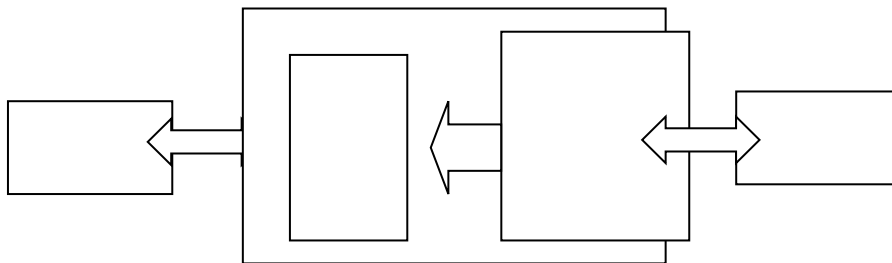
Технология ADO.NET

Microsoft ADO.NET (ActiveX Data Objects) — объектная модель доступа к данным; набор средств, позволяющих приложению управлять и взаимодействовать со своим файловым или серверным хранилищем данных. Библиотеки ADO.NET включают классы, которые служат для подсоединения к источнику данных, выполнения запросов и обработки их результатов.

Объектная модель ADO Net представлена на рисунке 1.

Рисунок 1 – Объектная модель ADO .Net

Подсоединенные объекты используются для управления соединением, транзакциями, для выборки данных и передачи изменений они взаимодействуют непосредственно с БД. Большинство подключенных объектов реализовано в рамках того, что называется поставщиками данных. Поставщики данных Net (NET data provider) - это набор классов, предназначенных для взаимодействия с хранилищем данных определенного типа (рисунок 2).



Каждый поставщик данных .NET реализует одинаковые базовые классы — `ProviderFactory`, `Connection`, `ConnectionStringBuilder`, `Command`, `DataReader`, `Parameter` и `Transaction`, имена которых зависят от поставщика.

Например, у поставщика SQL Client .NET Data Provider есть объект `SqlConnection`, а у поставщика ODBC .NET Data Provider это объект `OdbcConnection`. Но независимо от типа поставщика, объект `Connection` реализует одинаковые базовые функции посредством одних и тех же базовых интерфейсов.

Базовые классы подсоединенных объектов

ProviderFactory — новый класс в [ADO.NET](#) версии 2.0, действующий как фабрика объектов, которая дает возможность создавать образцы других классов для поставщика данных .NET. Каждый класс `ProviderFactory` предлагает метод `Create`, в котором создаются объекты `Connection`, `ConnectionStringBuilder`, `Command`, `Parameter`, `DataAdapter` и `CommandBuilder`.

Класс Connection - применяется для соединения с БД и отсоединения от нее. С помощью свойств этого объекта можно задать тип источника, его расположение прочее. Объект `Connection` выступает в качестве канала, по которому другие классы, например `DataAdapter` и `Command`, взаимодействуют с БД для передачи изменений и выборки их результатов.

ConnectionStringBuilder - новый класс в [ADO.NET](#) версии 2.0. Этот класс делает проще процесс построения строк подключения для поставщика данных .NET. Каждый класс `ConnectionStringBuilder` предоставляет свойства, которые соответствуют опциям, доступным в той самой строке подключения поставщика данных. Создав строку подключения с помощью класса `ConnectionStringBuilder`, можно получить доступ к строке подключения средствами свойства `ConnectionString` класса `ConnectionStringBuilder`.

Transaction используется для выполнения группы команд вместе как неделимую операцию. В классе `Connection` есть метод `BeginTransaction`, позволяющий создавать объекты `Transaction`. С помощью объекта `Transaction` удастся подтвердить или отменить все коррективы, сделанные в ходе транзакции.

Класс Command - может осуществлять запрос к БД, вызов хранимой процедуры или прямой запрос на возврат содержимого конкретной таблицы. Команда может вернуть или нет какой-либо результат (в зависимости от этого выполнение объекта `Command` запускается различными методами).

Parameter - используется для создания параметризованного объекта `Command`. Для использования параметров создаются классы `Parameter`, соответствующие всем параметрам запроса, затем они добавляются в класс `Parameters` объекта `Command`. Класс `Parameter` [ADO.NET](#) предоставляет свойства и методы, позволяющие определить тип данных и значение параметров.

Класс DataReader предназначен для максимально быстрой выборки и просмотра возвращаемых запросом записей. Он позволяет просматривать результаты запроса по одной записи за раз. При переходе к следующей записи содержимое предыдущей отбрасывается. Объект `DataReader` не поддерживает обновление, и возвращаемые им данные доступны только для чтения. Поскольку класс `DataReader` реализует лишь ограниченный набор функций, он очень прост и имеет высокую производительность.

Класс DataAdapter воплощает новую концепцию моделей доступа к данным Microsoft. Это своеобразный шлюз между БД и отсоединенными объектами модели [ADO.NET](#). Он устанавливает подключение или, если подключение уже установлено, содержит достаточно информации, чтобы воспринимать данные автономных объектов и взаимодействовать с базой данных предписанным образом.

Базовые классы автономных объектов

Объекты, составляющие автономную часть модели [ADO.NET](#), не взаимодействуют напрямую с подсоединенными объектами, для этого используется `DataAdapter`. Автономные объекты не должны знать о базе данных, поэтому они могут совместно использоваться различными базами.

При автономной работе с данными живое соединение с БД не понадобится, однако вы не увидите изменений, внесенных другими пользователями после выполнения вами исходного запроса.

При работе в автономном режиме [ADO.NET](#) ведет пул реальных физических подключений для различных запросов, таким образом многократно повышая свою производительность.

Класс DataSet содержит набор данных. Данные в нем отсоединены от БД. Все изменения данных кэшируются в объектах `DataRow`. Кроме того, класс `DataSet` предоставляет функции чтения и записи в файл и область памяти. Можно сохранить только содержимое объекта `DataSet`, только его структуру или и то и другое. [ADO.NET](#) хранит эти данные в виде XML-документа.

Класс Data Table похож на таблицу базы данных. Он состоит из объектов DataColumn, DataRow и различных налагаемых на них ограничений. Он хранит данные в формате строк и столбцов.

Класс DataColumn соответствует столбцу таблицы. В действительности же DataColumn содержит не данные, хранящиеся в объекте DataTable, а информацию о структуре столбца. Такая разновидность информации называется метаданными (metadata).

Класс DataTable содержит свойство — **Constraint** — типа ConstraintsCollection. Оно позволяет создавать объекты ForeignKeyConstraint или UniqueConstraint и ассоциировать различные столбцы с определенными условиями, которым должны соответствовать данные из DataTable. Объект DataTable предоставляет через набор Rows содержимое всех записей данных. Когда вы изменяете содержимое записи, **DataRow** кэширует эти изменения, чтобы позже передать их в БД. Таким образом, при изменении значения поля записи объект DataRow хранит оригинальное и текущее значения поля, что обеспечивает успешное обновление содержимого БД.

DataRelation позволяет задать отношения между различными таблицами, с помощью которых можно проверять соответствие данных из различных таблиц, а также просматривать родительские и дочерние строки из различных объектов DataTable. Кроме того, объекты DataRelation предоставляют свойства, позволяющие обеспечить ссылочную целостность.

Выбрав результаты запроса в объект DataTable, его содержимое можно просматривать разными способами посредством объекта **DataView**. Просматривать содержимое одного объекта DataTable можно одновременно посредством нескольких объектов DataView.

DataSet со строгим контролем типов — это класс, наследованный от класса DataSet и включающий свойства и методы, основанные на указанной вами схеме. Кроме того, этот класс содержит другие классы для объектов DataTable и DataRow, они позволяют создавать более эффективный код доступа к данным

2. Создание базы данных в среде MssqlServerManagement.

Цель работы: ознакомиться с основными конструкциями SQL, технологиями среды MS SQL Server Management, объектами SMO (среды MS Visual Studio) для резервного копирования и восстановления БД.

Ход работы

Задание: Необходимо в среде SQL Server Management Studio:

Создать новую базу данных (далее БД); 2) создать в этой БД таблицы и связи для реализации всех функций предметной области MMM.

Для работы с базами данных MS SQL Server 2005 существует много возможностей - одна из них это инструмент SQL Server Management Studio (далее SSMS).

1. Запустите среду SQL Server Management Studio и подключитесь к локальному серверу MS SQL Server, используя технологию 1.
2. Если соединение прошло успешно, то откроется основное окно среды Management Studio. Рассмотрите подробно все компоненты среды.
3. Среда Management Studio представляет данные в виде окон, выделенных для отдельных типов данных. Сведения о базе данных отображаются в обозревателе объектов и окнах документов. Найдите в открытом окне среды раздел «Обозреватель объектов», вид которого представлен на рисунке 3.

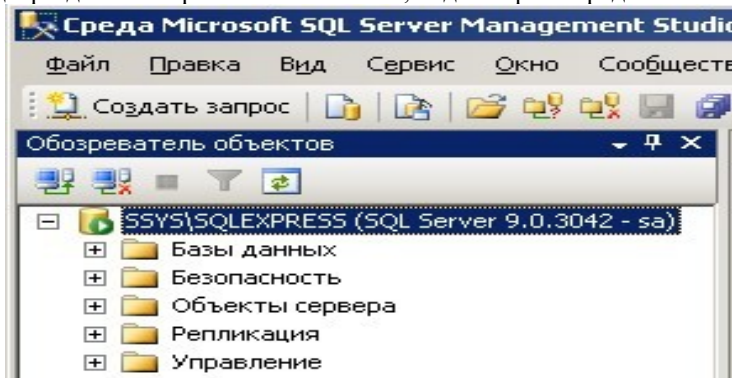


Рисунок 3 – Обозреватель объектов в среде SSMS

Обозреватель объектов является представлением в виде дерева, в котором отображаются все объекты базы данных на сервере. Обозреватель объектов включает сведения по всем серверам, к которым он подключен.

4. Далее найдите в открытом окне среды раздел «Окно документов». Окно документов представляет собой наиболее крупную часть среды Management Studio. В окнах документов могут размещаться редакторы запросов и окна обзора. По умолчанию отображается страница «Подробности обозревателя объектов», подключенная к экземпляру компонента Database Engine на данном компьютере.
5. Найдите на панели инструментов среды кнопку «Создать запрос» и нажмите ее. Откроется окно создания запроса. С помощью данного окна можно создавать и выполнять запросы к БД. Наберите в основной области окна запроса `select * from information_schema.tables` и выполните запрос, нажав на кнопку «Выполнить». В нижней части окна запроса появится таблица с результатами, найдите ее. (Выполненный вами запрос

извлекает информацию о существующих таблицах подключенной сейчас БД – это по умолчанию база данных Master и таблицы в ней системные).

6. Найдите в открытом окне среды раздел «Окно базы данных», вид которого представлен на рисунке 4.

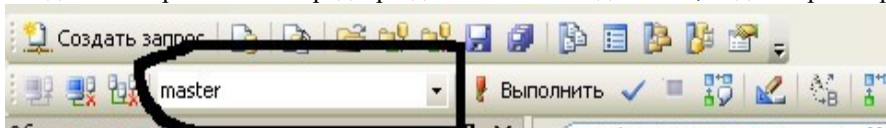


Рисунок 4 – Окно базы данных в среде SSMS

Именно это поле со списком позволяет выбрать применяемую в данный момент базу данных для выполнения запросов. По умолчанию в данном окне выбрана БД для учетной записи в которой зарегистрировался пользователь (для пользователя Sa это по умолчанию БД master) Далее приступим к созданию своей бд, для этого:

7. В своей папке с проектом Лабы_ИСПКЭС создайте новую папку БД_МММ, в которой будет храниться БД.
8. Самый простой способ создать базу данных — воспользоваться графическим интерфейсом SQL Server Management Studio. Создайте новую базу данных на Вашем локальном сервере MS SQL Server, используя технологию 2.
9. Убедитесь в обозревателе объектов в появившейся только что базе данных МММ.
10. Ознакомьтесь с описанием предметной области в приложении 1.
11. Далее приступим к созданию таблиц. Существует несколько методов, рассмотрим один из них. В обозревателе объектов разверните узел БД МММ_ВашеФИО и выберите в списке ветвь Таблицы → вызовите контекстное меню и выберите «Создать».
12. Создадим сначала таблицу Модель. Определим сначала все столбцы и типы данных в окне создания таблицы. Вводите имена столбцов без пробелов, кириллицей, в нижнем регистре. Конструктор таблицы «Модель» изображен на рисунке 5.

Модель *			
Имя столбца	Тип данных	Разрешить значения null	
код_модели	int	☐	
название_модели	varchar(30)	☐	
себестоимость	money	☐	
время_изготовления	smalldatetime	☒	
продажная_цена	money	☒	
описание	nchar(350)	☒	
фото	image	☒	

Рисунок 5 – Структура таблицы «Модель»

13. Сделайте столбец код_модели первичным ключом, используя технологию 3.
14. Закройте окно создания таблицы и сохраните, если еще этого не сделали, таблицу под именем модель.
15. Сделайте столбец код_модели счетчиком, используя технологию 4.
16. Создайте аналогично остальные таблицы для раздела Заказ и Реализация продукции предметной области (таблицы Магазин, Заказ, Состав_заказа, Готовая_продукция). Внимательно определяйте типы данных.
17. Создайте связи между таблицами предметной области, используя технологию 5.
18. Заполните таблицу «Модель» данными, используя информацию из файла каталог (см. Сетевую папку. Местонахождение папки спросить у преподавателя.)

Практическая работа № 1.49. Копирование и восстановление базы данных

Цель работы: ознакомиться с основными конструкциями SQL, технологиями среды MS SQL Server Management, объектами SMO (среды MS Visual Studio) для резервного копирования и восстановления БД.

Задание №1. необходимо создать резервные копии базы данных «МММ» с использованием полного резервного копирования, разностного резервного копирования и резервного копирования журнала транзакций. Ход работы:

1. Запустите SQL Server Management Studio (SSMS), подключитесь к своему экземпляру SQL Server, используя технологию 1.
2. Создайте папку с именем [c:\Student\ВашаПапка\test](#).
3. Откройте окно нового запроса. Измените контекст на базу данных master, используя технологию 6.

Наберите и исполните следующую команду, чтобы создать полную резервную копию базы данных:

BACKUP DATABASE МММ TO DISK = 'C:\.....TEST\AW.BAK'

Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

4. Внесите изменение в таблицу «Модель» базы данных МММ. Добавьте одну запись (придумайте сами)/

5. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать резервную копию журнала транзакций и сохранить только что внесенное изменение:
`BACKUP LOG MMM TO DISK = 'C:\.....TEST\AW1.TRN'`
Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.
6. Внесите еще одно изменение в таблицу «Модель».
7. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать разностную резервную копию базы данных:
`BACKUP DATABASE MMM TO DISK = 'C:\.....\TEST\AWDIFF1.BAK' WITH DIFFERENTIAL`
Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.
8. Внесите еще одно изменение в таблицу «Модель».
9. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать полную резервную копию базы данных в указанном месте на диске:
`BACKUP LOG MMM TO DISK = 'C:\....TEST\AW2.TRN'`
Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

Задание №2. необходимо провести восстановление базы данных «MMM» из сделанных в задании №1 резервных копий. Ход работы:

1. Если необходимо, запустите SSMS, подключитесь к своему экземпляру SQL Server, используя технологию 1.
2. Выполните восстановление БД из первой полной резервной копии (C:\...TEST\AW.BAK) средствами оболочки SSMS. Для этого выполните:
 - В обозревателе объектов вызовите контекстное меню на вашей БД и выберите задачу восстановления базы данных (см. рисунок 6).

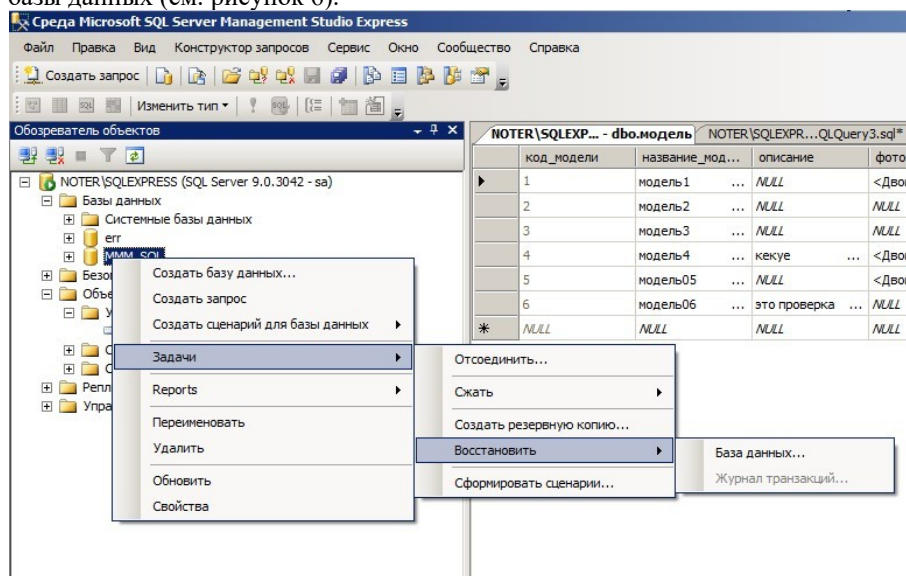


Рисунок 6 – Восстановление БД

- В открывшемся окне необходимо задать следующие параметры восстановления *На закладке «Общие» необходимо выбрать:*
 - a. Базу данных для восстановления (вашу MMM)
 - b. Выбрать источник набора данных для восстановления с устройства → файл C:\...TEST\AW.BAK
 - c. После определения файла-источника данных необходимо флажком выбрать базу данных для восстановления (рисунок 7).



Рисунок 7- Выбор БД для восстановления
 На закладке «Параметры»

- а. необходимо включить опцию «Перезаписать БД» и «оставить БД готовой к использованию», (рисунок 8).

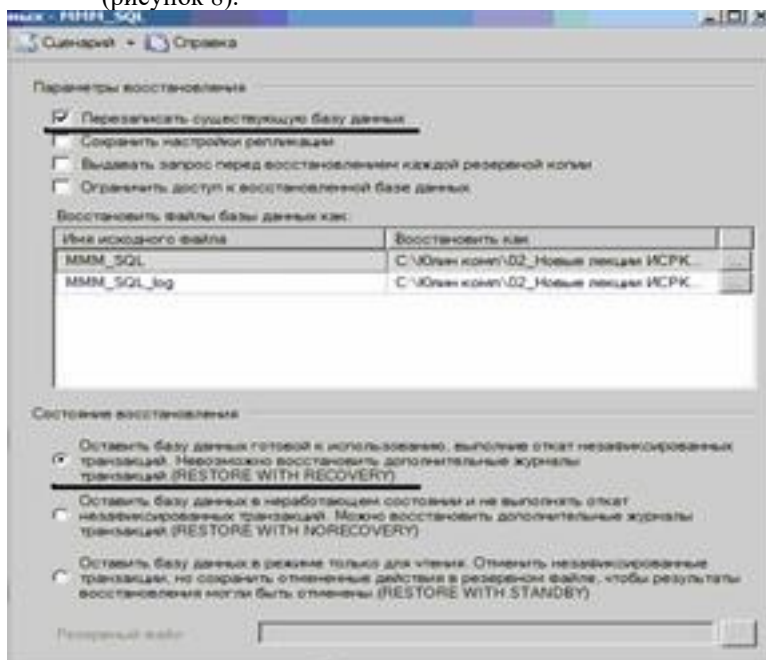


Рисунок 8 – Задание параметров восстановления 3. Нажмите

ОК

- После восстановления БД, откройте таблицу «Модель» и убедитесь, что она не содержит всех добавлений, вносимых вами в процессе выполнения упражнения, так как восстановление происходило из первой резервной копии (без изменений).

Задание №3. необходимо организовывать со стороны клиентского приложения, созданного в Visual Studio удаленное администрирование БД (резервное копирование).

Ход работы:

В Visual Studio

- Создайте новый проект Windows Application и сохраните его в своей папке под именем Лабы_MMM_2 семестр.
- В главную форму добавьте меню, изображенное на рисунке 9:

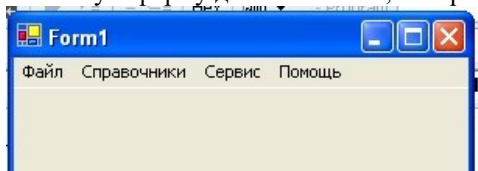


Рисунок 9 – Главное меню проекта *Файл* (*Открыть, Закрыть, Выход*)

Справочники (Модель, Магазин, Дерево моделей)

Заказы (Работа с заказами)

Отчеты (Прайс-лист, Бланк заказов)

Администрирование БД (Резервное копирование, Безопасность) Сервис (Калькулятор)

Помощь (Справка, О программе)

- Добавьте новую форму в проект
- Добавьте на только что созданную форму компоненты в соответствии с рисунком 10.

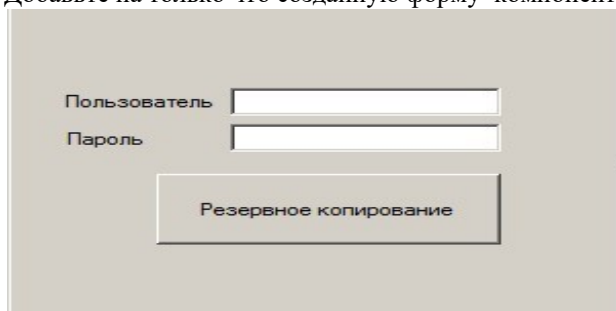


Рисунок 10 – Форма для подключения к серверу

5. Обеспечьте функциональную работу формы (напишите обработчик кнопки «Резервное копирование» с использованием объектов SMO. Описание объектов SMO, их свойств и методов см. в лекционном материале.)
6. Добавьте возможность открытия данной формы при выборе в главной форме пункта меню Администрирование БД → Резервное копирование
7. Запустите проект, проверьте работу формы.
8. Закройте проект
9. Убедитесь в появлении файла резервной копии на диске (файл, который указан в тексте программы).
10. Откройте SSMS. Добавьте в таблицу «Модель» новую строку данных (самостоятельно).
11. Средствами оболочки SSMS, выполните восстановление БД из резервной копии, созданной вашей программой
12. Убедитесь, что после восстановления добавленных строк в таблице «Модель» нет.

Задание №4. Ответьте на вопросы теста и представьте результаты преподавателю.

1. Вы выполняете разностное резервное копирование базы данных AdventureWorks каждые четыре часа, начиная с 04:00. полная резервная копия создается в полночь. Какие данные будут содержаться в разностной резервной копии сделанной в полдень?
 - a. A. Страницы данных, измененные после полуночи.
 - b. B. Экстенды, измененные после полуночи.
 - c. C. Страницы данных, измененные после 08:00
 - d. D. Экстенды, измененные после 08:00.
2. Вы выполняете полное резервное копирование базы данных Adventure Works,, которое завершается в полночь. Разностное резервное копирование выполняется по расписанию каждые четыре часа, начиная с 04:00. Резервное копирование журнала транзакций происходит по расписанию каждые пять минут. Какую информацию будет содержать резервная копия журнала транзакций, созданная в 09:15?
 - a. A. Все транзакции, начатые после 09:10.
 - b. B. Транзакции, завершённые после 09:10.
 - c. C. Страницы, измененные после 09:10.
 - d. D. Экстенды, измененные после 09:10.

7. Разработка формы для работы с данными в среде VisualStudio без кода

Цель работы: ознакомиться с возможностью мастеров среды MS Visual Studio для создания формы для работы с данными.

Задание: Необходимо создать простое (без написания кода) Windows-приложение, которое позволит взаимодействовать с данными в БД – просмотр, удаление, добавление данных в таблицу «Модель». Пример формы изображен на рисунке 11.

Рисунок 11 – Форма для работы с данными из таблицы «Модель»

Ход работы:

1. Добавим новую форму в проект

Откройте в Visual Studio проект, созданный в лабораторной работе № 2. Добавьте в проект новую форму. Добавьте обработчик для открытия только что добавленной формы при выборе в главном меню проекта пункта Справочники→Каталог моделей

2. Добавьте в проект новый «источник данных» с помощью мастера, включающий в себя все столбцы таблицы «Модель», используя технологию 19.
3. Рассмотрим, созданный мастером код. Для этого:
 - В окне Solution Explorer среды Microsoft Visual Studio найдите добавленный элемент – класс DataSet со строгим контролем типов, откройте (Дважды кликнув мышью) и рассмотрите его содержимое.
 - Выберите команду главного меню View→Server Explorer - и на экране появится одноименное окно. В иерархической структуре элементов окна Server Explorer должен появиться элемент, соответствующий тому соединению, которое использовано в мастере Data Source Configuration.
 - В главном меню выберите команду Data→Show Data Sources. Отобразится источник данных, который был создан с помощью мастера. Найдите узлы, соответствующие выбранной таблице «Модель», убедитесь что выделенные столбцы будут расположены в виде вложений под таблицами, в которые они входят. Теперь рассмотрим, как же можно использовать новый источник данных.
4. Добавление элементов в форму с помощью окна Data Source
Найдите в окне Data Source узел для таблицы «Модель» и разверните его (нажав на +). Выберите и перетащите его на форму все узлы-столбцы таблицы кроме столбца с Фото. Обратите внимание, что вместе с добавленными вами элементами на форме появился “Элемент управления навигацией”. Visual Studio добавила навигационный элемент управления и разместила его в верхней части формы.
5. Запустите проект. С помощью элемента BindingNavigator переместитесь к последней модели, после чего измените значение в одном из столбцов — ИмяМодели. Чтобы сохранить изменения, щелкните кнопку, на которой изображена дискета. После щелчка кнопки вы не сможете сразу увидеть на экране результаты завершения процедуры сохранения изменений. Чтобы убедиться, что внесенные изменения успешно сохранены в БД, закройте форму и перезапустите ваш проект. Внесенные изменения должны быть отображены в форме.
6. Изучение кода, сгенерированного в Visual Studio
Отобразите созданную в данной работе форму и откройте ее код события Load.
Из кода, сгенерированных в Visual Studio видно, что событие Load содержит код, который для таблицы моделей вызывает метод Fill в объектах TableAdapter. Метод Fill выполняет запрос, содержащийся в объекте TableAdapter, и сохраняет результаты в классе DataSet. При этом на класс DataSet ссылается таблица Моделей. Благодаря этому коду информация о моделях доступна в классе DataSet со строгим контролем типов в момент появления формы — ее же можно отобразить в элементе управления TextBox.

Событие Click для SaveItem

Код события Click объекта SaveItem элемента управления BindingNavigator, относящегося к информации о модели, состоит из трех строк. В первой строке вызывается **метод Validate** для формы, чтобы элементы управления могли проверить достоверность данных, введенных пользователем. Во второй строке кода вызывается метод EndEdit объекта BindingSource, который относится к информации о моделях. Он передает привязанным и объекту BindingSource элементам управления команду записи ожидающих изменений а источник данных. Без ЭТОЙ строки изменения так и останутся в кэше элементов управления и не будут отправлены в БД посредством следующей строки. В заключительной строке кода вызывается метод Update объекта TableAdapter для получения информации о модели; он передает ожидающие изменения в класс DataSet со строгим контролем типов.

Раздел 2. МДК. 01.02. Поддержка и тестирование программных модулей

Тема 1.1 Отладка и тестирование программного обеспечения

1. Выявление ошибок и причин их появления

Цель работы: изучить проблематику создания сложной программной системы в отношении к разрабатываемой ИС.

Краткие теоретические сведения

Особенности разработки сложных (больших) программных систем. Из года в году увеличиваются разнообразие и сложность систем, получивших в международной научнотехнической практике название систем, интенсивно использующих программное обеспечение – Software Intensive Systems (SIS). В системах такого рода функциональный потенциал определяется программным обеспечением (ПО) или зависит от ПО в существенной мере. В таких системах программные компоненты взаимодействуют друг с другом и компонентами и подсистемами другой природы, датчиками, приборами и людьми, вовлеченными в процесс использования SIS. К числу SIS, например, относятся разнородные автоматизированные системы управления, встроенные бортовые транспортные системы, телекоммуникационные и корпоративные системы, в том числе и на базе web-сервисов. Для разработок SIS типичны крупномасштабные проекты – десятки или сотни разработчиков, месяцы или годы разработки, сотни тысяч или десятки миллионов долларов, комплектование из многочисленных разнородных подсистем, большая часть из которых включает программные системы. Не все программные системы сложны. Существует множество программ, которые задумываются, разрабатываются, сопровождаются и используются одним и тем же человеком. Обычно это начинающий программист или профессионал, работающий изолированно. Нельзя сказать, что все такие системы плохо сделаны или тем более усомниться в квалификации их создателей. Но такие системы, как правило, имеют очень ограниченную область применения и короткое время жизни. Обычно их лучше заменить новыми, чем пытаться повторно использовать, переделывать или расширять. Разработка подобных программ скорее утомительна, чем сложна, так что изучение этого процесса нас не интересует. Какого-либо одного формального признака, отличающего обычную программу от сложной, не существует. В целом сложные программы выгодно отличаются разнообразием предоставляемого сервиса и количеством обрабатываемой информации. Возможно обозначить лишь некоторые качественные характеристики, свойственные сложной программе. Сложная программа характеризуется также более сложным алгоритмом обработки событий. В частности, такая программа предполагает некоторую реакцию на вмешательство пользователя в управляемый процесс или объект. Существенно, что сложные программы предназначены для многократного использования и применения разными пользователями. В связи с этим следует обратить внимание на ряд необходимых свойств программного обеспечения.

Обычно сложная программа обладает следующими свойствами: □

программа решает одну или несколько связанных прикладных задач, зачастую сначала имеющих четкой постановки и настолько важных для каких-либо лиц или организаций, что те приобретают значимые выгоды от ее использования; □ программа не предназначена для решения каких-либо прикладных задач, но от нее зависит эффективное решение этих прикладных задач. Это системные программы, например операционные системы, системы управления базами данных, различные инструментальные системы и т. п.; □

существенно, чтобы программа была удобной в использовании. В частности, она должна включать достаточно полную и понятную для пользователей документацию, возможно, специальную документацию для администраторов, а также набор документов для обучения работе с программой; □

программа должна обладать высокой производительностью, высокой реактивностью или удовлетворять другим требованиям, в противном случае ее использование по назначению (на реальных данных) может привести к значимым для пользователей потерям; □

программа должна обладать высокой надежностью. Неправильная работа программы может нанести ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто; □

для выполнения своих задач программа должна удовлетворять требованиям совместимости, переносимости и интеграции с другими программами и программноаппаратными системами и обеспечивать работу на разных платформах; □

пользователи, работающие с программой, могут приобретать дополнительные выгоды от того, что программа развивается, в нее вносятся новые функции и устраняются ошибки. Поэтому необходимо наличие проектной документации, позволяющей развивать ее, возможно, вовсе не тем разработчикам, которые ее создавали, без больших затрат на обратную разработку (реинжиниринг); □

в разработку программы вовлечено значительное количество людей (десятки и сотни человек). Большую программу практически невозможно написать с первой попытки, с небольшими усилиями и в одиночку; □
большая программа имеет намного большее количество ее возможных пользователей по сравнению с небольшими программами и еще больше тех лиц, деятельность которых будет так или иначе затронута ее работой и результатами. Более подробно теоретические сведения методики изложены в.

Контрольные вопросы

1. Что такое управление процессом разработки?
2. Что такое гибкость программного обеспечения?
3. Как описывается поведение программных систем?
4. Что такое сложность ПО?

2. Тестирование «белым ящиком»

Задание 1. Разработать программу на Python.

Даны длины сторон треугольника, определить вид треугольника и его площадь. Выполнить контроль вводимых чисел.

1. Разносторонний треугольник
2. Равнобедренный треугольник
3. Равносторонний треугольник

Ограничения: - три числа не могут быть определены как стороны треугольника; - если хотя бы одно из них меньше или равно 0; - сумма двух из них меньше третьего.

Задание 2. Подготовить набор тестовых вариантов для обнаружения ошибок в программе. Результат оформить в следующем виде: Таблица 1

А	В	С	Ожидаемый результат	Объект проверки
Значение	Значение	Значение	Что должно получиться	Значения вводимых данных, либо ожидаемый результат
...	...	...	...	...

Задание 3. Разработать программу на Python.

Даны длины сторон треугольника, определить вид треугольника и его площадь. Выполнить контроль вводимых чисел.

1. Остроугольный треугольник
2. Тупоугольный треугольник
3. Прямоугольный треугольник

Ограничения:

- три числа не могут быть определены как стороны треугольника;
- если хотя бы одно из них меньше или равно 0;
- сумма двух из них меньше третьего.

Подготовить набор тестовых вариантов для обнаружения ошибок в программе и оформить результат.

Задание 4. На основании проведенных тестов составьте рекомендации по исправлению ошибок, выявленных в ходе тестирования в виде отчета.

Пример:

1 тест. В ходе проведения первого теста было обнаружено, что при введении некорректных данных площадь все равно высчитывается.

Рекомендуется: в случае, если пользователь введет не корректные данные, следует выводить сообщение с просьбой исправить введенные значения. Добавить в программу проверку введенных значений на соответствие ограничения.

3. Тестирование «черным ящиком»

Задание 1. Написать калькулятор с небольшими багами.

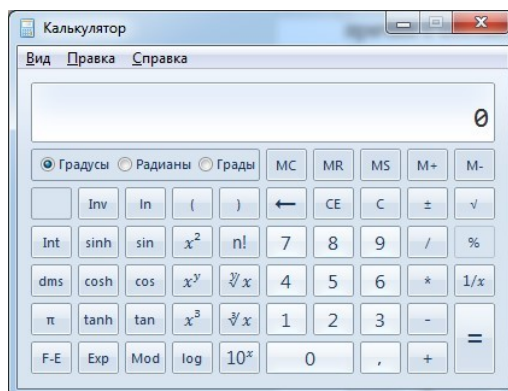


Рисунок 2

Задание 2. Обменяться программой с другими студентами. Провести тестирование и написать отчет в тетради.

Таблица 2

Название теста	Описание сценария	Входные данные	Выходные данные	Удачное/неудачное тестирование	Предложения по исправлению найденных ошибок.	Пожелания пользователей
Функция суммы	Сложение двух положительных чисел; Проверка результата	Первая переменная=3 Вторая переменная=8	Результат= 11	Неудачное	-	Поле для ввода значений и вывода, объединить

4. Модульное тестирование

Задание 1. Создание проекта программы, модули которого будут тестироваться.

Разработаем проект содержащий класс, который вычисляет площадь прямоугольника по длине двух его сторон.

Создадим в Visual Studio новый проект Visual C# -> Библиотека классов. Назовём его MathTaskClassLibrary. Class1 переименуем в Geometry.

В классе реализуем метод, вычисляющий площадь прямоугольника. Для демонстрации остановимся на работе с целыми числами. Код программы приведён ниже.

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace MathTaskClassLibrary
8 {
9     public class Geometry
10     {
11         public int RectangleArea(int a, int b)
12         {
13             return a * b;
14         }
15     }
16 }

```

Рисунок 2

Создание проекта для модульного тестирования в Visual Studio.

Чтобы выполнить unit-тестирование, необходимо в рамках того же самого решения создать ещё один проект соответствующего типа.

Правой кнопкой щёлкните по решению, выберите “Добавить” и затем “Создать проект...”.

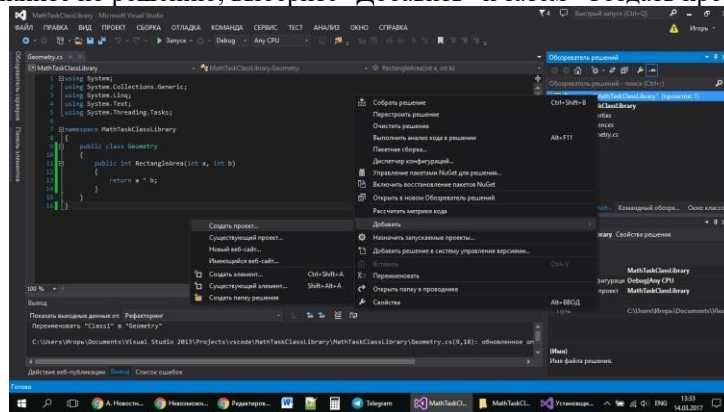


Рисунок 3 В открывшемся окне в группе Visual C# щёлкните “Тест”, а затем выберите “Проект модульного теста”. Введите имя проекта MathTaskClassLibraryTests и нажмите “ОК”. Таким образом проект будет создан.

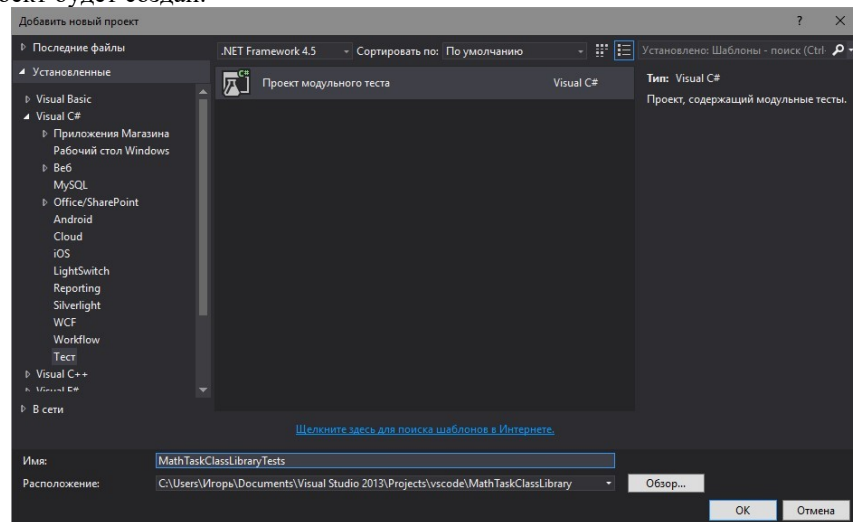


Рисунок 4

Перед Вами появится следующий код:

```
1 using System;
2 using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4 namespace MathTaskClassLibraryTests
5 {
6     [TestClass]
7     public class UnitTest1
8     {
9         [TestMethod]
10        public void TestMethod1()
11        {
12        }
13    }
14 }
```

Рисунок 5

Директива [TestMethod] обозначает, что далее идёт метод, содержащий модульный (unit) тест. А [TestClass] в свою очередь говорит о том, что далее идёт класс, содержащий методы, в которых присутствуют unit-тесты.

В соответствии с принятыми соглашениями переименуем класс UnitTest1 в GeometryTests.

Затем в References проекта необходимо добавить ссылку на проект, код которого будем тестировать.

Правой кнопкой щёлкаем на References, а затем выбираем “Добавить ссылку...”. В появившемся окне раскрываем группу “Решение”, выбираем “Проекты” и ставим галочку напротив проекта MathTaskClassLibrary. Затем жмём “ОК”.

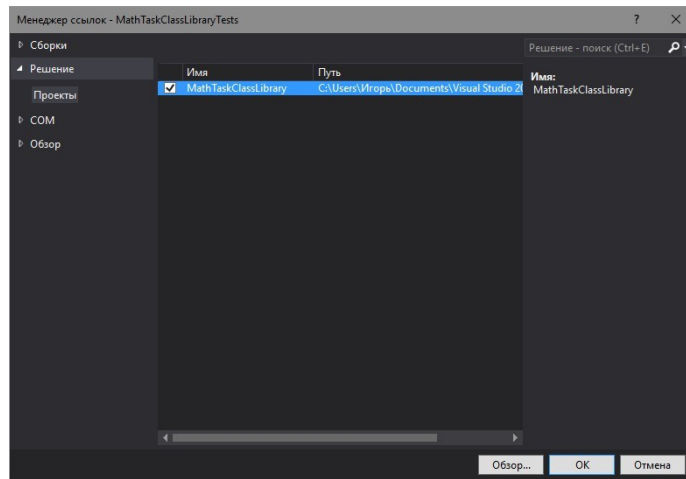


Рисунок 6 Также в коде необходимо подключить с помощью директивы using следующее пространство имён: using MathTaskClassLibrary;

Займёмся написание теста. Проверим правильно ли вычисляет программа площадь прямоугольника со сторонами 3 и 5. Ожидаемый результат (правильное решение) в данном случае это число 15.

Переименуем метод TestMethod1() в RectangleArea_3and5_15returned(). Новое название метода поясняет, что будет проверяться (RectangleArea – площадь прямоугольника) для каких значений (3 и 5) и что ожидается в качестве правильного результата (15 returned). Тестирующий метод обычно содержит три необходимых компонента:

1. исходные данные: входные значения и ожидаемый результат;
2. код, вычисляющий значение с помощью тестируемого метода;
3. код, сравнивающий ожидаемый результат с полученным.

Соответственно тестирующий код будет таким:

```

1 using System;
2 using Microsoft.VisualStudio.TestTools.UnitTesting;
3 using MathTaskClassLibrary;
4
5 namespace MathTaskClassLibraryTests
6 {
7     [TestClass]
8     public class GeometryTests
9     {
10         [TestMethod]
11         public void RectangleArea_3and5_15returned()
12         {
13             // исходные данные
14             int a = 3;
15             int b = 5;
16             int expected = 15;
17
18             // получение значения с помощью тестируемого метода
19             Geometry g = new Geometry();
20             int actual = g.RectangleArea(a, b);
21
22             // сравнение ожидаемого результата с полученным
23             Assert.AreEqual(expected, actual);
24         }
25     }
26 }

```

Рисунок 7

Для сравнения ожидаемого результата с полученным используется метод AreEqual класса Assert. Данный класс всегда используется при написании unit тестов в Visual Studio.

Теперь, чтобы просмотреть все тесты, доступные для выполнения, необходимо открыть окно “Обозреватель тестов”. Для этого в меню Visual Studio щёлкните на кнопку “ТЕСТ”, выберите “Окна”, а затем нажмите на пункт “Обозреватель тестов”.

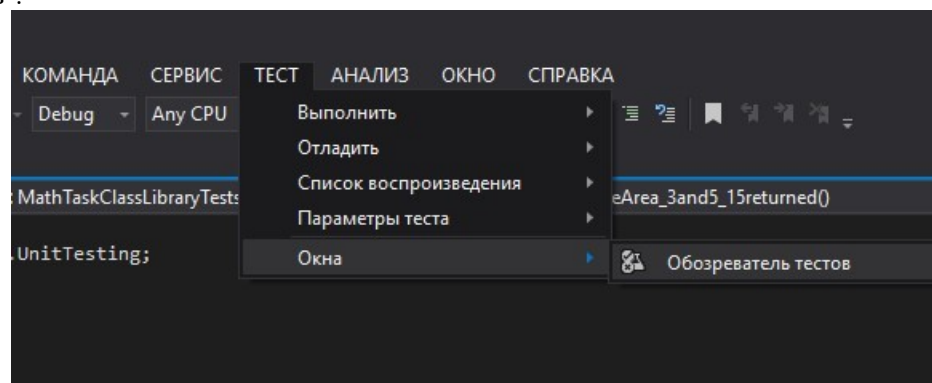


Рисунок 8 В

студии появится следующее окно:

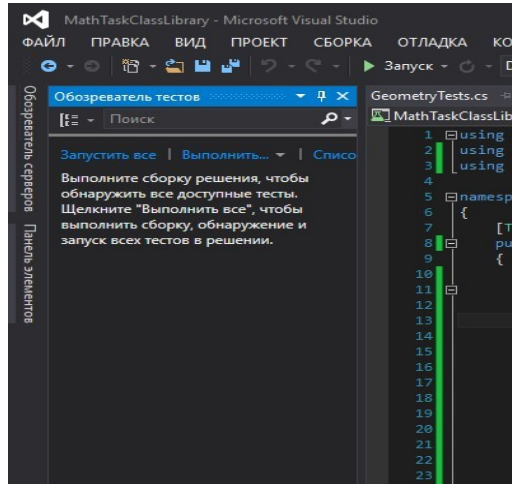


Рисунок 9

В данный момент список тестов пуст, поскольку решение ещё ни разу не было собрано. Выполним сборку нажатием клавиш Ctrl + Shift + B. После её завершения в “Обозревателе тестов” появится наш тест.

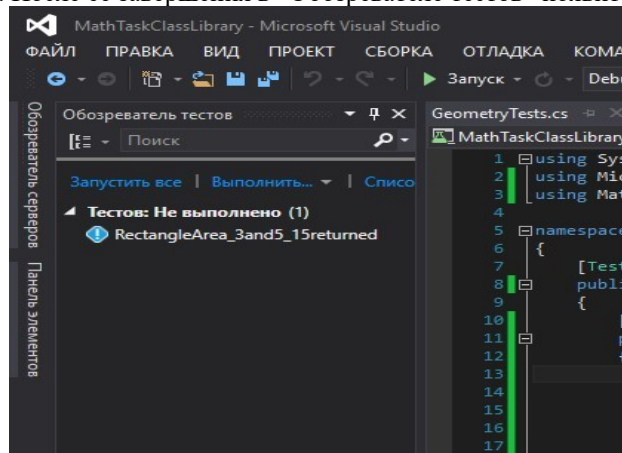


Рисунок 10

Синяя табличка с восклицательным знаком означает, что указанный тест никогда не выполнялся. Выполним его.

Для этого нажмём правой кнопкой мыши на его имени и выберем “Выполнить выбранные тесты”.

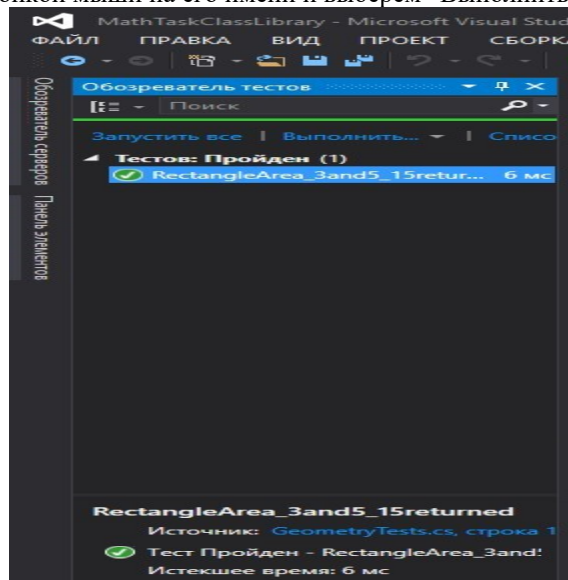


Рисунок 11

Зелёный кружок с галочкой означает, что модульный тест успешно пройден: ожидаемый и полученный результаты равны.

Изменим код метода RectangleArea, вычисляющего площадь прямоугольника, чтобы симитировать провал теста и посмотреть, как поведёт себя Visual Studio. Прибавим к возвращаемому значению 10.

Запустим unit-тест.

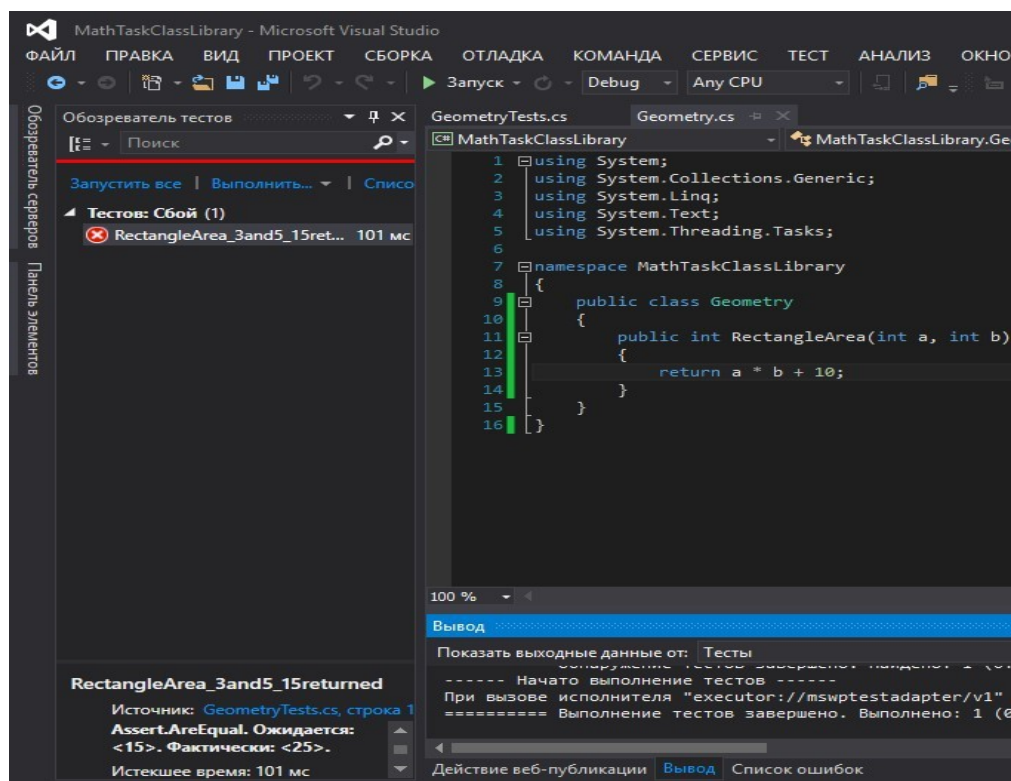


Рисунок 12

Как Вы видите, красный круг с крестиком показывает провал модульного теста, а ниже указано, что при проверке ожидалось значение 15, а по факту оно равно 25.

Задание 2. Разработать программу для подсчета объема цилиндра и создать модульный тест.

5. Интеграционное тестирование

Интеграционное тестирование называют еще тестированием архитектуры системы. С одной стороны, это название обусловлено тем, что интеграционные тесты включают в себя проверки всех возможных видов взаимодействий между программными модулями и элементами, которые определяются в архитектуре системы – таким образом, интеграционные тесты проверяют полноту взаимодействий в тестируемой реализации системы. С другой стороны, результаты выполнения интеграционных тестов – один из основных источников информации для процесса улучшения и уточнения архитектуры системы, межмодульных и межкомпонентных интерфейсов. Т.е., с этой точки зрения, интеграционные тесты проверяют корректность взаимодействия компонент системы. В результате проведения интеграционного тестирования и устранения всех выявленных дефектов получается согласованная и целостная архитектура программной системы, т.е. можно считать, что интеграционное тестирование – это тестирование архитектуры и низкоуровневых функциональных требований. Интеграционное тестирование, как правило, представляет собой итеративный процесс, при котором проверяется совокупность модулей, возрастающая от итерации к итерации. В интеграционном тестировании выделяют три метода выполнения: восходящее тестирование; монолитное тестирование; нисходящее тестирование.

Задание: Согласно варианту провести один из методов интеграционного тестирования.

6. Разработка алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования

Контрольные вопросы

1. Каковы стадии жизненного цикла информационных систем, их основное содержание?
2. Как представляется функциональная модель деятельности в методологии IDEF0?
3. Каковы основные объекты диаграмм функциональной модели по методологии IDEF0?
4. Что обозначают работы в диаграммах функциональной модели, как они отображаются по методологии IDEF0?
5. Для чего предназначены стрелки в диаграммах функциональной модели, каковы их типы и виды?
6. Для чего предназначен словарь стрелок?
7. Каковы типы связей работ по методологии IDEF0?
8. Что такое туннелирование стрелок, для чего оно нужно, каковы виды туннелирования?

7. Использование инструментальных средств на этапе отладки программного модуля

ЗАДАНИЕ

Провести отладку созданной в программы с помощью симулятора-отладчика среды AVR Studio, проделав следующие операции.

1. Выполнить программу в пошаговом режиме, отслеживая изменение содержимого используемых в программе регистров. Обратит внимание на изменение содержимого программного счётчика. Сравнить содержимое программного счётчика при выполнении команд с их адресами в памяти программ, приведёнными в листинге трансляции и окне памяти программ.
2. Выполнить прогон программы. Проверить правильность результата работы программы.
3. Задать точку останова на команде загрузки в РОН числа В. Включить режим отображения сообщений о достижении точки останова. Выполнить прогон программы с контрольными точками. Задать точку останова на команде умножения. Выполнить прогон программы с контрольными точками. Удалить заданные точки останова.
4. Задать точки наблюдения в используемых РОН. Выполнить программу в пошаговом режиме, отслеживая изменение их содержимого.

СОДЕРЖАНИЕ ОТЧЁТА

Отчёт должен содержать: титульный лист с указанием номера и названия лабораторной работы, номера группы и фамилий выполнивших работу; цель работы; краткие теоретические сведения (классификацию средств отладки ПО, перечень основных функций программаторов); список использованных команд отладчика AVR Studio с указанием их назначения.

Контрольные вопросы

1. Классификация средств отладки прикладного ПО встраиваемых МП.
2. Виды и особенности аппаратных средств отладки ПО.
3. Основные функции программных средств отладки ПО.
4. Пошаговое выполнение программы и его возможности.
5. Особенности прогона программы с контрольными точками.
6. Контрольные точки: типы, назначение, использование.

Тема 1.2 Документирование

1. Оформление документации на программные средства с использованием инструментальных средств

Цель работы: Разработать комплект документации на программное средство
Программное документирование – это процесс записи информации, произведенной процессами жизненного цикла.

Процесс содержит набор действий, которые планируют, проектируют, разрабатывают, производят, редактируют, распространяют и сопровождают те документы, в которых нуждаются все заинтересованные лица, такие как менеджеры, инженеры и пользователи программного средства.

Общие требования к составу и содержанию документов, поддерживающих создание программных средств, представлены в ряде стандартов разного ранга. Состав документов широко варьируется в зависимости от класса и характеристик объекта разработки.

Существует несколько стандартов в области обеспечения документирования программных средств.

ISO 12207 - Информационные технологии. Процессы жизненного цикла программного обеспечения.

В этом стандарте документированию посвящен специальный раздел в группе вспомогательных процессов.

ISO 9000- 3 - Общее руководство качеством и стандарты по обеспечению качества.

Управлению качеством документации посвящен специальный раздел 6.2. Эти задачи отражены также в ряде разделов стандарта, непосредственно регламентирующих управление качеством сложных программных средств.

ISO 6592 – Обработка информации. Руководство по разработке документации для вычислительных систем

Главная цель этого стандарта состоит в установлении базисной структуры документации, на основе которой возможно для любого проекта обеспечить эффективное совершенствование и реализацию информационной системы, ПС или БД. В стандарте установлены руководящие принципы создания документов для информационных систем.

ISO 9294 - Информационные технологии. Руководящие положения по управлению документацией на программное обеспечение

Технический отчет этого стандарта представляет руководство по документированию ПС для руководителей, отвечающих за создание программной продукции. Руководство предназначено для помощи в управлении разработкой и эффективном документировании программных проектов.

IEEE 1063-1993 - Пользовательская документация на программное обеспечение.

+В нем представлены наиболее полно общие требования к пользовательской документации на программные средства широкого применения. Стандарт определяет минимальные требования к структуре и содержанию комплекта документов для пользователей программных продуктов.

ГОСТ 34.602—89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы

Настоящий стандарт распространяется на автоматизированные системы для автоматизации различных видов деятельности (управление, проектирование, исследование и т. п.), включая их сочетания, и устанавливает состав, содержание, правила оформления документа “Техническое задание на создание (развитие или модернизацию) системы”.

ГОСТ Р 51904-2002 — Программное обеспечение встроенных систем. Общие требования к разработке и документированию.

Стандарт распространяется на процессы разработки и документирования программного обеспечения встроенных систем реального времени. Стандарт распространяется на все действия, имеющие отношение к разработке программного обеспечения.

ГОСТ 19.101 – Виды программ и программных документов.

Настоящий стандарт устанавливает виды программ и программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения. Виды программных документов и их содержание приведены в таблице 1:

Таблица 19

Вид программного документа	Содержание программного документа
1	2

Спецификация	Состав программы и документации на нее
Ведомость держателей подлинников	Перечень предприятий, на которых хранят подлинники программных документов
Текст программы	Запись программы с необходимыми комментариями
Описание программы	Сведения о логической структуре и функционировании программы
Программа и методика испытаний	Требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля
Техническое задание	Назначение и область применения программы, технические, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний
Пояснительная записка	Схема алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений
Эксплуатационные документы	Сведения для обеспечения функционирования и эксплуатации программы

Порядок выполнения работы:

Оформить пояснительную записку (ПЗ) на программный продукт.

ПЗ на программное средство должна иметь следующую структуру:

- 1) Постановка задачи;
- 2) Входные и выходные данные;
- 3) Среда разработки и обоснование выбора языка программирования;
- 4) Описание алгоритма;
- 5) Описание используемых классов и методов; 6) Заключение.

Приложение А – Техническое задание;

Приложение Б – Технический проект;

Приложение В – Иерархия функциональных диаграмм. Диаграмма сущность-связь.

Диаграмма потоков данных

Приложение Г UML-диаграммы

Приложение Д Листинг программы

Содержание отчета: ПЗ на электронном и бумажном носителе.

2. Тестирование в Microsoft Solutions Framework

Цель работы: провести тестирование в Microsoft Solutions Framework

Тест

В каждом тестовом задании может быть несколько вариантов ответа. После проведения теста студенты могут попробовать обосновать свои неверные ответы.

1. При использовании какого метода интеграционного тестирования сначала все программные модули, входящие в состав системы, тестируются и только затем объединяются для интеграционного тестирования?

1. восходящего
2. монолитного
3. нисходящего
4. с поздней интеграцией
5. с постоянной интеграцией
6. с регулярной интеграцией

Ответ: 1

2. При использовании какого метода интеграционного тестирования подразумевается, что, как только разрабатывается новый модуль системы, он сразу же интегрируется со всей остальной системой?

1. восходящего
2. монолитного
3. нисходящего
4. с поздней интеграцией
5. с постоянной интеграцией
6. с регулярной интеграцией

Ответ: 5

3. Для каких видов интеграционного тестирования нужен драйвер?

1. восходящего
2. монолитного
3. нисходящего
4. с поздней интеграцией
5. с постоянной интеграцией
6. с регулярной интеграцией

Ответ: 1, 5, 6

4. Для каких видов интеграционного тестирования нужны заглушки?

1. восходящего
2. монолитного
3. нисходящего
4. с поздней интеграцией
5. с постоянной интеграцией
6. с регулярной интеграцией

Ответ: 1, 3, 6

5. Для каких видов интеграционного тестирования при разработке часто выполняется интеграция?

1. восходящего
2. монолитного
3. нисходящего
4. с поздней интеграцией
5. с постоянной интеграцией
6. с регулярной интеграцией

Ответ: 3, 5, 6

Роль тестировщика в команде разработчиков ПО

В течение всего курса обсуждались и демонстрировались на примере программной системы "Калькулятор" подходы к тестированию и верификация программного обеспечения.

В конце курса мы поговорим о роли, обязанностях и задачах тестировщика в команде разработчиков программного обеспечения на примере методологии ведения проектов и разработки программного обеспечения Microsoft Solutions Framework (MSF) for Agile Software Development.

Microsoft Solutions Framework

Замечание. Подробнее о подходе MSF можно почитать по адресу <http://www.microsoft.com/rus/msdn/msf/> или по адресу <http://msdn.microsoft.com/vstudio/teamsystem/msf>

Microsoft Solutions Framework (MSF) – хорошо настраиваемый, масштабируемый, полностью интегрируемый набор процессов разработки программного обеспечения, принципов и проверенных практик, предназначенных для того, чтобы предоставить команде разработчиков программного обеспечения именно тот вид управления проектами, который им больше подходит.

MSF — это методология ведения проектов и разработки решений, которая базируется на принципах работы над продуктами как самой фирмы Microsoft, так и других компаний, работающих в области IT-индустрии.

MSF является схемой для принятия решений по планированию и реализации новых технологий в организациях. MSF включает обучение, информацию, рекомендации и инструменты для идентификации и структуризации информационных потоков бизнеспроцессов и всей информационной инфраструктуры новых технологий.

Microsoft Solutions Framework представляет собой хорошо сбалансированный и гибкий набор методик организации процесса разработки, который может быть адаптирован под потребности практически любого коллектива разработчиков и проекта, вне зависимости от его размера и сложности. MSF поддерживает самые различные подходы к организации процесса разработки, что позволяет команде разработчиков выбирать самый подходящий для них путь. Философия MSF утверждает, что не существует единой методологии разработки, которая оптимально будет соответствовать требованиям любых проектов. Но, тем не менее, любому проекту необходимо управление. MSF направлена на помощь в обеспечении этого управления. При этом MSF не налагает предписаний, а позволяет команде разработчиков настраивать предоставленные средства. Средства MSF могут быть применены по отдельности или все вместе. Главное — они позволят добиться успеха для многих типов проектов.

Главными принципами MSF можно назвать производительность, интегрируемость и расширяемость:

Производительность. Один из ключевых принципов MSF направлен на то, чтобы сделать команду разработчиков более производительной.

Производительность в MSF поддерживается хорошо налаженным управлением процесса разработки;

Пинтегрируемость. Решения и управление представлены инструментальными средствами, посредством плавной интеграции любых наборов инструментальных средств, справки и содержания MSF. Все эти элементы легко обновляются через MSDN;

Прасширяемость. Процесс управления и справка полностью настраиваемы в пределах MSF. Разработчики могут выбрать быстрый или более структурированный подход, каждый из которых включает в себя наборы предложенных сценариев, или определить свой собственный подход, используя эти сценарии.

MSF содержит не только рекомендации общего характера, но и предлагает адаптируемую модель коллектива разработчиков, определяющую взаимоотношения внутри коллектива, гибкую модель проектного планирования, основанного на управлении проектными группами, а также набор методик для оценки рисков.

Жизненный цикл процессов в MSF сочетает водопадную и спиральную модели разработки: проект реализуется поэтапно, с наличием соответствующих контрольных точек, а сама последовательность этапов может повторяться по спирали.



Рисунок 49

При таком подходе от водопадной модели берется простота планирования, от классической спиральной – легкость модификаций. Благодаря промежуточным контрольным точкам и обратной спирали верификации облегчается взаимодействие с заказчиком.

При управлении проектом четко ставится цель, которую необходимо достичь в результате, и учитываются ограничения, накладываемые на проект. Все виды ограничений могут быть отнесены к одному из трех видов: ограничения ресурсов, ограничения времени и ограничения возможностей. Эти три вида ограничений и приоритетность задач по их преодолению образуют треугольник приоритетов в MSF.



Рисунок 50

Треугольник приоритетов является основой для матрицы компромиссов – заранее утвержденных представлений о том, какие аспекты процесса разработки будут четко заданы, а какие будут согласовываться или приниматься как есть.

Microsoft выпустила среду разработки, в полной мере поддерживающей основные идеи MSF – Microsoft Visual Studio 2005 Team System. Это первый программный комплекс, представляющий собой не среду разработки для индивидуальных членов коллектива, а комплексное средство поддержки коллективной работы.



Рисунок 51

Замечание. В состав Visual Studio Team Edition входит специальная редакция для тестировщиков – Team Edition for Software Testers, с которой мы и работали на протяжении всего курса.

Также Visual Studio 2005 Team System включает в себя два шаблона методологий MSF, которые можно применять "как есть", настраивать под соответствие своим собственным потребностям или использовать как основу для создания своего собственного подхода для организации и управления процессом разработки программного обеспечения:

- MSF for Agile Software Development
- MSF for CMMI® Process Improvement

Тестировщик в MSF for Agile Software Development
MSF for Agile Software Development

Microsoft Solutions Framework (MSF) for Agile Software Development — управляемый сценариями, основанный на окружении, быстрый процесс разработки программного обеспечения для создания .NET и других объектно-ориентированных приложений. MSF for Agile Software Development включает в себя подходы по управлению требованиями к качеству, такими, как требования к оборудованию или требования к безопасности. На основании окружения MSF помогает решить, как управлять проектом. Этот подход помогает создавать адаптивный процесс, который, преодолевая трудности большинства быстрых процессов разработки программного обеспечения, достигает целей, поставленных при создании проекта перед командой разработчиков.

О ролях

В MSF for Agile Software Development собрана вместе команда равных разработчиков, которая обеспечивает полный набор необходимых составляющих, связанных с созданием, использованием и обслуживанием создаваемого продукта. Каждый член команды, или роль, отвечает за удовлетворения нужд своей клиентуры, причем ни один клиент не является важнее другого. MSF for Agile Software Development содержит все необходимые методики и подходы для уверенности в том, что команда разработчиков принимает правильные решения.



Рисунок 52

Подробную информацию об MSF for Agile Software Development и ролях в этом подходе можно найти на сайте Microsoft в соответствующем разделе.

Наша цель заключается в том, чтобы рассказать о роли тестировщика в команде разработчиков, работающих по этому подходу.

О роли тестировщика

Главная задача тестировщика — обнаружить и сообщить о проблемах программного продукта, которые могут сказаться на качестве. Ключевая задача тестировщика — найти и сообщить о существенных ошибках в продукте, протестировав его. Также в обязанности тестировщика входит точное сообщение о проявлениях ошибки и описание какого-либо обходного пути для уменьшения этих проявлений. Тестировщик описывает ошибки, а также простые в понимании и выполнении шаги для устранения этих ошибок. Он участвует в группе по

разработке стандартов качества продукта. Цель тестирования состоит в том, чтобы доказать, что известные функции работают правильно, и обнаружить новые проблемы продукта.

Далее будет описана последовательность работы тестировщика по подходу MSF for Agile Software Development.

Проверка сценария

Сценарий — тип рабочего элемента, описывающий действия пользователя в системе для достижения им определенной цели. Таким образом, в сценарии записаны шаги, которые необходимо сделать пользователю для достижения определенной цели. При этом не обязательно, чтобы сценарий описывал успешные пути достижения цели — вполне могут быть сценарии, описывающие шаги, которые не приведут к желаемому результату. С другой стороны, достичь одну и ту же цель в системе можно несколькими путями.

Сценарий разделен на задачи тестирования. Эти задачи назначены тестировщикам для разработки и выполнения тестовых ситуаций. Назначение сценария для тестирования означает, что необходимые функциональные возможности были включены в текущую сборку и готовы к тестированию. Проверка того, что сборка содержит функциональность, описанную в сценарии, требует понимания сценария и его граничных условий. Валидационные тесты разрабатываются для того, чтобы покрыть все функциональные возможности и граничные условия сценария. Все ошибки, возникающие в процессе тестирования, сохраняются в виде соответствующего рабочего элемента.

Для проверки сценария, в первую очередь, нужно определить подход к тестированию.

Подход к тестированию — это стратегия, определяющая разработку и выполнение тестов. Он также определяет качественные рекомендации по нагрузке на программный продукт. Подход к тестированию является отправной точкой для тест-плана на ранней стадии проекта, но развивается и изменяется он вместе с проектом. Подход к тестированию — объединение методик, в частности, методик по ручному и автоматизированному тестированию. Перед каждой итерацией документ, описывающий подход к тестированию, должен быть обновлен, чтобы отразить цели тестирования и тестовых данных, которые используются на новой итерации.

Для определения подхода для тестирования необходимо, чтобы сценарий был написан и утвержден.

Если это сделано, то выполняются следующие действия.

1. Определение окружения проекта, т.е. выявление уникальных проектных рисков и пользователей, на которых это может повлиять; выявление специальных ситуаций, которые могут повлиять на тестирование; учет влияния рисков; определение того, что произойдет, если возникнет ошибка.

2. Определение тестовой миссии, т.е. определение проектных целей, которые необходимо удовлетворить в течение тестирования; консультация с проектировщиком и бизнес-аналитиком по техническим неясностям и пользовательским рискам; сотрудничество с бизнес-аналитиком и проектировщиком для создания списка приоритетных технических неясностей и пользовательских рисков.

3. Оценивание возможных технологий, т.е. оценивание доступных инструментальных средств для тестирования; оценивание навыков команды тестировщиков; определение тестовых технологий, возможных и соответствующих проекту, основанных на доступных инструментах и навыках.

4. Определение показателей тестирования, т.е. работа с разработчиками для определения реалистичных порогов показателей покрытия кода для тестируемых модулей; использование тестового окружения, цели тестирования и технологии тестирования для определения тестовых показателей; определение того, что тестовые показатели будут часто включать в себя пороги для различных типов текстов (Web, нагрузочные, стрессовые) или процент автоматизированных тестов; работа с бизнес-аналитиком для определения реалистичных показателей порогов нагрузки продукта; добавление тестовых показателей к соответствующим отчётам и в документ, описывающий подход к тестированию; опубликование подхода к тестированию.

В результате выработки подхода к тестированию показатели тестов определены и будут являться обязательными при модульном тестировании.

Далее необходимо написать валидационные тесты.

Валидационные тесты проверяют, что система выполняет функции, описанные в сценарии. Они разрабатываются как тесты "чёрного ящика" приложения и проверяют области, наиболее важные для конечных пользователей. При разработке этих тестов не учитывается структура исследуемого кода. В качестве тестов используют тестовые ситуации, которые повторяют действия, выполняемые реальными пользователями.

Для создания валидационных тестов необходимо, чтобы сценарий был написан и утвержден, были назначены задачи тестирования сценария и определена область, проверяемая сценарием.

Если это сделано, то выполняются следующие действия.

1. Определение тестируемой области и среды, т.е. изолирование тестируемой области (тесты могут быть запущены как часть итерационных тестов, если они автоматизированы).

2. Определение деталей выполнения тестовых примеров, т.е. определение тестовых данных, необходимых для тестового примера; проверка документа "Подход к тестированию"; добавление тестовых данных к соответствующей итерационной части документа; определение всех ограничений для тестовых примеров, вызываемых в тестовых заданиях; определение всех граничных условий для тестовых примеров, вызываемых в тестовых заданиях; определение всех граничных условий для тестового примера; определение, могут ли тестовые примеры быть автоматическими; определение шагов для выполнения сценария.

3. Реализация тестовых примеров, т.е. создание папки для тестирования сценария, если она ещё не создана; написание документации для ручных тестовых примеров; написание автоматизированных тестовых примеров для итерационных тестов; размещение тестовых примеров в папку тестирования сценария; добавление всех тестов, определенных как итерационные, в папку итерационных тестов; сохранение тестов в версионном контроле и в документе, описывающем подход к тестированию.

В результате написания валидационных тестов все граничные условия и вся функциональность тестового задания будут покрыты.

Далее необходимо произвести выбор и запуск тестового примера.

Выбор и запуск тестового примера. Наиболее важная часть тестирования — выполнение тестов для текущей сборки. Как только тест-кейс выполнен, важно записать результаты по отношению к сценарию или требования к качеству.

Для выбора и запуска тестовой ситуации необходимо, чтобы тестовые примеры для тестового задания были написаны и была доступна сборка с необходимой функциональностью.

Если это сделано, то выполняются следующие действия.

1. Определение проводимых тестов, т.е., определение и уделение первостепенного внимания запускаемым тестам в соответствии с кратким обзором тестового плана в документе подхода к тестированию; проверка заметок сборки, чтобы убедиться, что тестируемая функциональность доступна.
2. Определение тестовых конфигураций, т.е. обнаружение и установление тестовых конфигураций, необходимых при тестировании (включает в себя аппаратные средства, программное обеспечение и тестовые данные).
3. Получение сборки, т.е. извлечение сборки с сервера, обеспечивающего версионный контроль.
4. Запуск тестирования, т.е. выбор тестовой папки для сценария или требования к качеству; выбор тестов для запуска; выполнение каждого шага теста, если это ручной тест; запуск теста, если он автоматизированный; использование тестовых данных из документа о подходе к тестированию текущей итерации.
5. Анализ тестовых результатов, т.е. присоединение сценария или требования к качеству к результатам тестирования; сравнение результатов теста с ожидаемыми результатами; выявление ошибок, если результаты не отвечают ожиданиям: если все тесты сценария или требования к качеству пройдены, закрытие рабочего элемента.
6. Закрытие рабочих элементов, т.е., если все тесты из тестового задания выполнены, закрытие рабочего элемента; уведомление владельца сценария, что нет заблокированных элементов.

В результате выбора и запуска тестовой ситуации тестовое задание будет закрыто, потому что все тесты пройдены, и будут занесены ошибки при непредвиденном поведении системы.

Далее необходимо произвести выявление ошибки.

Выявление ошибки. Всегда нужно проверять, не обнаружена ли уже проблема, перед тем как создавать новую ошибку. Необходимо выполнить шаги для воспроизведения ошибки, таким образом, чтобы она могла быть исследована. В отчет всегда нужно включать как можно больше деталей для помощи команде в определении лучшего способа разрешения ошибки. У каждой выявленной ошибки должен быть ответственный за ее исправление.

Этот этап производится, если тестирование выявило неожиданное поведение системы.

Если это так, то выполняются следующие действия.

1. Определение, не обнаружена ли аналогичная ошибка, т.е. прежде чем создать новую ошибку, проверить БД системы отслеживания ошибок, и убедиться, что проблема еще не идентифицирована; если проблема уже была идентифицирована и ошибка обнаружена, обновить форму отслеживания ошибки соответствующим образом; если проблема уже опубликована и ошибка исправлена, изменить состояние рабочего элемента ошибки с "Исправлено" на "Найдено заново"; добавить новую информацию и документировать новые шаги для воспроизведения ошибки; документировать номер сборки, где было идентифицировано некорректное поведение.

2. Регистрация новой ошибки, т.е., если ошибка не опубликована и может быть воспроизведена, ввести шаги для воспроизведения ошибки и детали результатов выполнения этих шагов в рабочем документе; включить столько деталей, сколько возможно, чтобы помочь команде, ответственной за приоритеты ошибок, понять проблему; там, где это применимо, прикрепить тестовые ситуации, результаты, конфигурации, сценарий или требования к качеству к отчету по ошибке.

3. Назначение ошибки, т.е. определение главной области проблемы; если более чем одна область может быть задействована, то выбрать одну из них; назначение ошибки на владельца затронутой области приложения; сохранение ошибки в базе данных отслеживания ошибок.

В результате этих действий ошибка будет должным образом введена в систему отслеживания ошибок и комментарии будут содержать информацию, необходимую для её исправления, или ошибка будет назначена соответствующим образом..

Далее необходимо произвести исследовательское тестирование.

Исследовательское тестирование – систематический способ проверить продукт без predetermined набора тестов. Существует множество эвристик, которые могут быть применены к проведению исследовательского тестирования. Эти эвристики включают в себя использование ролей, характеристики, переменный анализ, область поиска и тестирование различных состояний. Эвристика, обеспеченная этим руководством, описывает, как продукт протестирован с точки зрения определенной роли с целью создания новых требований. Для того, чтобы выполнить исследовательское тестирование таким образом, необходимо выбрать роль и работать через функциональные возможности системы, пытаясь достичь определённых целей. Если новые цели достигнуты или функциональные возможности не способны удовлетворить потребностей роли, добавить новые или модифицировать существующие сценарии для удовлетворения этих потребностей. Лимит сессий исследовательского тестирования – не более двух часов.

Этот этап производится, если сборка готова к тестированию на новые требования или ошибки и роли опубликованы на портале проекта..

Если это выполнено, то выполняются следующие действия.

1. Установление длины сессии, т.е. установление периода времени для сессии исследовательского тестирования (этот период должен быть не меньше, чем полчаса, но не больше, чем два часа); постановка цели сессии; запуск и журналирование сессии.

2. Проверка ожидания для роли, т.е. выбор роли из списка опубликованных ролей; осуществление прохода по существующим функциональным возможностям, чтобы убедиться, что они соответствуют ожиданиям.

3. Предположение новых целей, т.е. исследование возможности роли; если реализация возможностей сложна для выполнения, определение новой ошибки или добавление нового сценария или требования к качеству в лист сценариев; изменение существующих сценариев или требований к качеству; поиск новых целей или пропущенных функциональных возможностей, необходимых для данной версии продукта; добавление новых сценариев или требований к качеству входимостей в лист сценариев.

В результате этих действий новые сценарии и/или требования будут добавлены для отражения нового понимания системы.

После выполнения всех описанных выше действий закончится проверка сценария, в результате которой можно утверждать, что все валидационные тесты были запущены и все ошибки результатов были опубликованы.

Тестирование требований к качеству

Документы, описывающие требования к качеству системы, характеризуют такие качества системы, как максимальная нагрузка, доступность, надежность и сопровождаемость. Эти требования обычно принимают форму ограничений на то, как система должна работать.

Назначение требований к качеству для тестирования показывает, что сборка готова к тестированию. Во многих случаях сценарии присоединены к требованиям к качеству, чтобы показать области, которые будут проверяться. Тестирование требований к качеству требует, чтобы тесты на устройства, безопасность и нагрузку были завершены и ни один не был заблокирован. По результатам тестов, создаются отчеты для документирования обнаруженных проблем.

Для тестирования требований к качеству, в первую очередь, нужно определить подход к тестированию аналогично тому, как это делалось при тестировании сценариев.

Далее необходимо написать тесты на производительность.

Тесты на производительность измеряют время отклика приложения и гарантируют, что приложение соответствует установленным требованиям к качеству. Для написания тестов на производительность необходимо, чтобы было назначено тестовое задание, показывающее, что необходимо проверить работу требований к качеству.

Если это сделано, то выполняются следующие действия.

1. Понимание цели теста, т.е. цель тестирования должна быть ясно определена (например, не должно быть существенного различия в отклике продукта для пользователей с разной скоростью подключения к Интернету). Выход теста должен быть ясно задокументирован и представлен в виде диапазона значений.

2. Специфицирование тестовых конфигураций, т.е. определение тестовых конфигураций; документирование любых отклонений результатов

теста, которые зависят от тестовых конфигураций; документирование внешних условий, которые могут отразиться на времени тестирования.

3. Планирование тестирования, т.е. планирование тестовых условий, включая предпосылки и установленный сценарий; пересмотр сценария, чтобы определить области, где производительность критична, а где – нет.

4. Документирование тестовых шагов, т.е. перечисление тестовых шагов так, чтобы все тестировщики, которые выполняют рабочие тесты, выполняли каждый шаг одним и тем же образом; отказ перечисления тестовых шагов может отразиться в неправильном измерении работы; использование автоматизации для построения тестовых ситуаций там, где это возможно; добавление тестовых ситуаций в папку для требований к качеству и итерационных тестов; регистрирование тестов и обновление рабочего листа подхода к тестированию с любыми тестовыми данными или другими взглядами на тест.

В результате написания тестов на производительность рабочие тесты будут завершены и зарегистрированы и будет проверено, что требования к качеству, относящиеся к функционированию системы, собраны.

Далее необходимо написать тесты безопасности.

Тесты безопасности, или "тесты проникновения", используют угрозы, найденные в процессе моделирования угроз, чтобы симитировать попытку противника достигнуть определенных целей в программе. Эта форма тестирования может быть разделена на три части: исследование, идентификация недостатка и эксплуатация. Тестирование проникновения может привести к открытию новых уязвимостей, которые становятся требованиями безопасности или ошибками. В результате тестировщики должны знать об угрозах не меньше проектировщиков. Эта форма тестирования требует специальных навыков, таких, как умение думать и действовать как противник.

Для написания тестов безопасности необходимо, чтобы была назначена тестовая задача для требования безопасности, которое выполняется на данной итерации, а модель угрозы была актуальной и опубликованной.

Если это сделано, то выполняются следующие действия.

1. Исследование точки входа, т.е. идентифицирование точки входа системы и функциональности, отвечающей за защиту программы; сбор информации из модели угрозы, для определения ожидаемых направлений нападения; расположение по приоритетам точек входа и задание им уровней доверия.

2. Идентифицирование недостатков, т.е. написание тестовых примеров, использующих прямые и частично случайные тесты, чтобы попытаться обратиться к программе; применение направленных мер, нацеленных на обход определенных мер безопасности; полуслучайные нападения могут манипулировать форматом данных или протокола, чтобы проверить граничные условия или выявить ошибки приложения.

3. Слабости к эксплоитам, т.е. добавление тестовых примеров, эксплуатирующих любые обнаруженные уязвимости, чтобы попытаться обратиться к программе; принятие во внимание количество времени, необходимого для того, чтобы воспользоваться уязвимостью; сохранение этих ручных тестов в соответствующей папке; регистрация их; добавление всех

необходимых тестовых данных в документ, описывающий подход к тестированию.

В результате написания тестов безопасности тестовые примеры будут покрывать все части требований к безопасности и все необходимые тестовые данные будут добавлены в документ, описывающий подход к тестированию.

Далее необходимо выбрать и запустить тестовые примеры аналогично тому, как это делалось при тестировании сценариев.

Далее необходимо провести выявление ошибки аналогично тому, как это делалось при тестировании сценариев.

Далее необходимо произвести исследовательское тестирование аналогично тому, как это делалось при тестировании сценариев.

После выполнения всех описанных выше действий закончится тестирование требований к качеству, в результате которой можно утверждать, что все тесты были запущены и все ошибки результатов были опубликованы.

Стрессовое тестирование

Стрессовое тестирование имеет много общего с тестированием производительности, однако его основная задача – не определить производительность системы, а оценить производительность и устойчивость системы в случае, когда для своей работы она выделяет максимально доступное количество ресурсов либо когда она работает в условиях их критической нехватки. Основная цель стрессового тестирования – вывести систему из строя, определить те условия, при которых она не сможет далее нормально функционировать. Для проведения стрессового тестирования используются те же самые инструменты, что и для тестирования производительности.

Заккрытие ошибок

Ошибка – рабочий элемент, который сообщает о том, что в системе существует или существовала потенциальная проблема. Цель открытия ошибки состоит в том, чтобы точно сообщить об ошибках, причем так, чтобы разработчик, ознакомившийся с ошибкой, смог понять все составляющие проблемы. Описания в сообщении об ошибке должны обеспечивать прослеживание шагов, сделанных при столкновении с ошибкой, таким образом позволяя легко её воспроизводить.

Ошибка может быть закрыта по нескольким причинам: она исправлена, относится к другому релизу, демонстрирует несоответствие, которое не удастся воспроизвести, или дублирует уже открытую ошибку. После того, как она закрыта, никакая работа по ошибке не совершается в течение текущей итерации. Заккрытие ошибок обычно происходит после проверки исправлений.

Для закрытия ошибки необходимо, чтобы она находилась в открытом состоянии и была указана для текущей версии программы

Сначала необходимо провести исправление ошибки.

Исправление ошибки. Проверяя исправления, смотрим на то, что ошибка была корректно исправлена и её исправление не повредило другой функциональности системы. После того, как ошибка исправлена разработчиком, тестировщик должен убедиться, что тестовый пример теперь выполняется. Если это так, то ошибку можно закрывать. Иначе ошибка снова переназначается разработчику.

Для написания исправления ошибки необходимо, чтобы сборка, которая устраняет эту ошибку, и тестовый пример, который осуществляет функциональные возможности, связанные с ошибкой, были найдены.

Если это сделано, то выполняются следующие действия.

1. Попытка воспроизводить ошибку, т.е. следовать шагам выполнения, как первоначально сообщено в описании ошибки, чтобы попытаться воспроизвести ошибку.

2. Поиск неожиданного поведения системы, т.е. проводится изучение смежных функциональных возможностей и попытка найти неожиданное поведение системы, связанное с ошибкой (это особенно важно, если задача разработки не была осуществлена полностью).

3. Получение подробной информации об ошибке, т.е., если описание ошибки непонятное, запрашивать у разработчика дополнительную информацию.

4. Переназначение ошибки, т.е., если тестовый пример не выполняется, то ошибка переназначается назад разработчику.

Далее необходимо провести собственно закрытие ошибки.

Для закрытия ошибки необходимо, чтобы сборка с исправленной ошибкой была обнаружена и тест, который обнаружил ошибку, выполнен; было решено не исправлять ошибку из-за ограничений на времени или ошибка была сдублирована и не может быть воспроизведена или будет задержана к другому релизу.

Если это сделано, то выполняются следующие действия.

1. Обновление дубликата ошибки, т.е., если ошибка является дубликатом существующей, необходимо обновить описание.

2. Новая ошибка не будет исправлена, т.е., если ошибка не должна быть исправлена, обновите описание ошибки, указав причину.

3. Ошибка исправлена, т.е., если, после выполнения шагов по воспроизведению ошибки, она не повторилась, закройте ошибку; укажите в описании ошибки сборку, используемую для проверки исправления, чтобы проконтролировать описание ошибки.

В результате закрытия ошибки, она будет закрыта в системе отслеживания ошибки с описанием поведения после исправления или объяснения того, почему ошибка не была устранена.

В результате закрытия ошибка будет закрыта с соответствующим кодом причины.

Раздел 3. МДК. 01.03. Разработка мобильных приложений

Тема 1.3.1 Основные платформы и языки разработки мобильных приложений

а) Примеры тестов

Раздел 1. Введение в основы разработки мобильных приложений

1. Что означает аббревиатура IDE?

А) Integrated Development Environment;

Б) Integrated Development Environment Application В) обработка информации производится с использованием ЭВМ;

В) Integrated Dev Electronics;

Г) Integrated Driver Electronics. 2. С помощью чего можно запустить приложение Android на телефоне?

А) Через USB кабель;

Б) Через Bluetooth;

В) Через наушники;

Г) Через Wi-Fi;

Д) Через передачу данных.

3. Набор средств программирования, который содержит инструменты, необходимые для создания, компиляции и сборки мобильного приложения называется:

А) Android SDK;

Б) JDK; В)

Android NDK; Г)

плагин ADT.

4. Чем являются Eclipse и IDEA?

А) API (интерфейс прикладного программирования);

Б) SDK (набор средств разработки);

В) IDE (интегрированная среда разработки);

Г) ADT (инструменты разработки под Android).

5. Единица измерения для изменения размеров элементов:

А) px;

Б) sp;

В) dp;

Г) pt.

6. Как называются мобильные приложения, которые разрабатываются исключительно для одной мобильной операционной системы?

А) специальные приложения;

Б) веб-приложения;

В) гибридные мобильные приложения;

Г) нативные мобильные приложения.

7. Расставьте в правильном порядке этапы разработки мобильного приложения

.Подбор контента

.Идея

.Юзабилити-тестирование

.Устранение ошибок

- .Прототип
 - . Дизайн
 - . Разработка
 - . Тестирование
 - . Проектирование интерфейса
 - . Маркетинг
8. Как называются мобильные приложения, которые специально созданы с использованием различных многоплатформенных веб-технологий, таких как JavaScript и HTML5?
 - А) специальные приложения;
 - Б) веб-приложения;
 - В) нативные мобильные приложения;
 - Г) гибридные мобильные приложения.
 9. Развитием и поддержкой ОС Android, главным образом, занимается компания:
 - А) Android;
 - Б) Apple;
 - В) Google;
 - Г) Microsoft.
 10. Ядро какой операционной системы использовалось в качестве базы для ОС Android?
 - А) Linux;
 - Б) Windows;
 - В) Mac OS;
 - Г) OS/2.

**Раздел 2. Языки программирования для разработки мобильных приложений
(Java, Kotlin и др.)**

1. Какой атрибут отвечает за установку внешних отступов?
 - А) gravity;
 - Б) padding; В) margin;
 - Г) bottom.
2. Обязательным атрибутом макета LinearLayout является:
 - А) text;
 - Б) orientation;
 - В) gravity; Г) context.
3. С помощью какого метода происходит взаимодействие пользователя с кнопкой?
 - А) onClick();
 - Б) onCreate();
 - В) onButton();
 - Г) onStart();
4. Что значит атрибут hint в представлении EditText?

- А) Определение текста;
 - Б) Жирный шрифт;
 - В) Подчеркнутый текст;
 - Г) Подсказка для ввода текста.
5. Можно ли выбрать несколько значений в переключателе ToggleButton?
- А) Да, не больше двух;
 - Б) Да, не больше трех;
 - В) Нет, нельзя выбрать несколько значений;
 - Г) Зависит от исходных данных.
6. Что делает строка setContentView() в методе onCreate()?
- А) Определяет разметку и запускает проект;
 - Б) Определяет код;
 - В) Определяет ошибки в проекте;
 - Г) Определяет разметку.
7. Можно ли выбрать несколько значений в переключателе CheckBox?
- А) Да, не больше двух;
 - Б) Да, не больше трех;
 - В) Нет, нельзя выбрать несколько значений;
 - Г) Зависит от исходных данных;
 - Д) Да, в неограниченном количестве.
8. С помощью какого атрибута, можно выделить текст курсивом?
- А) textSize;
 - Б) text;
 - В) textAppearance;
 - Г) textStyle.
9. В какой формат компилируются Java-классы для исполнения на виртуальной машине Dalvik?
- А) .class;
 - Б) .jar;
 - В) .dex;
 - Г) .apk.
10. Как группирует LinearLayout дочерние элементы?
- А) Относительно других элементов;
 - Б) В табличном виде;
 - В) С привязкой друг к другу;
 - Г) По горизонтали, по вертикали.

б) Примеры вопросов устных блиц-опросов

Раздел 1. Введение в основы разработки мобильных приложений

1. Возможности Android Studio.
2. Шаги установки Android Studio.
3. Шаги создания проекта в Android Studio.
4. Структура проекта.

5. Компоненты приложения.
6. Операционная среда Android.
7. Что такое макет приложения?
8. Технологии разработки мобильных приложений для различных платформ.
9. Обзор мобильных платформ, особенности.
10. Техническое задание на разработку мобильного приложения.

Раздел 2. Языки программирования для разработки мобильных приложений

(Java, Kotlin и др.)

1. Особенности языка Java.
2. Особенности языка Kotlin.
3. Способы создания Activity.
4. Способы создания макета приложения. Основные виды Layout.
5. LinearLayout. Основные атрибуты.
6. Жизненный цикл мобильного приложения.
7. TableLayout. Свойства.
8. FrameLayout. Атрибуты.
9. ConstraintLayout. Основные атрибуты. View-элементы. Варианты EditText.
10. Способы обработки нажатия на кнопку.

в) Примеры индивидуальных практических заданий

Раздел 1. Введение в основы разработки мобильных приложений

Задание 1

1. Установить Android Studio на домашний компьютер, если позволяют минимальные требования к установке.
2. Изучить документацию введение по интерфейсу и общему функционалу среды AS <https://developer.android.com/studio/intro>
3. Изучить руководство <https://developer.android.com/guide>

Задание 2. Дана структура проекта в Android Studio (рис. 1). Установить правильное соответствие в таблице. В первой колонке таблицы по порядку следования из структуры проекта, далее по 2-ой и 3-ей колонке правильное соответствие по названию и что внутри.

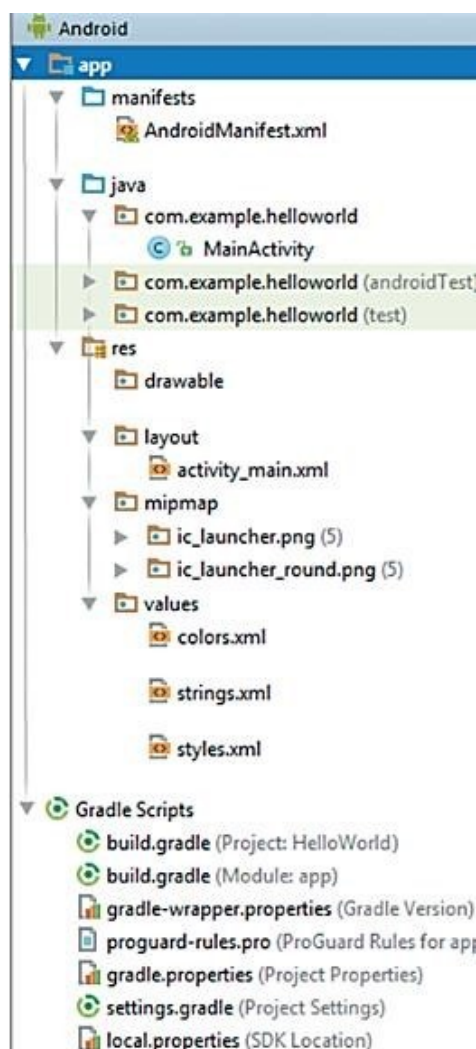


Рис. 1

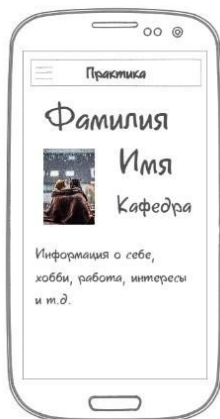
Папка/файл	Название	Что внутри
java	Файл стилей	Файлы манифестов, содержащие основную настроечную информацию о приложении
mipmap	Папка скриптов Gradle	В файле манифеста можно изменить имя приложения, версию, логотип, основную тему, структуру, назначить стартовый экран и т.д.
values	Папка значений ресурсов	Разметка интерфейса всех экранов приложения, описанная на языке XML
drawable	Файл build приложения	Файл Java-класса главной Activity приложения. В нем описывается все, что должно происходить при загрузке Activity, а также при нажатии на ее элементы и т.д.
app	Файл строк	Все ресурсы приложения: изображения, разметка интерфейса, цвета, строковые ресурсы, стили
color.xml	Папка значений	Копия логотипа приложения и его круглая вариация

	ресурсов	
ic_launcher	Папка изображений	Все классы Java, относящиеся к работе приложения
strings.xml	Папка круглого логотипа приложения	Все файлы, относящиеся к приложению
manifests	Папка логотипа приложения	Все картинки, которые планируется использовать в интерфейсе приложения
activity_main.xml	Папка исходных файлов приложения	В файле можно изменять основные цвета приложения и добавлять новые
ic_launcher_round	Папка манифестов	В файле можно изменить минимальную и требуемую версию SDK для приложения, а также прописать поддержку необходимых библиотек и сервисов
MainActivity	Файл цветов	Описанные на языке Groovy скрипты системы сборки Gradle, с помощью которой можно собирать и запускать наши приложения
Gradle Scripts	Файл манифеста	В файле можно изменять установленные для компонентов приложения стили, а также прописывать новые стили
styles.xml	Папка классов Java	В файле можно сохранять длинные текстовые содержания элементов и различные заголовки
AndroidManifest	Папка макетов	Описание интерфейса главной Activity на языке разметки XML
res	Файл MainActivity.java	Копия логотипа приложения в круглой вариации
build.gradle	Папка иконок	Описания ресурсов приложения (цветов, строк, стилей) на языке XML для более удобного доступа к ним
layout	Файл макета	Логотип приложения во всех необходимых системе Android разрешениях

Раздел 2. Языки программирования для разработки мобильных приложений (Java, Kotlin и др.)

Задание 1

1. Создайте два разных макета экрана приложения, используя языки программирования Java и Kotlin в среде Android Studio, согласно образцу ниже.



2. Добавьте рассказ о себе Пример 1: Я студент кафедры ПИМ ЗабГУ, учусь на 3-ем курсе и т.д.

Раздел 3. Разработка мобильных приложений для Android и др. платформ

Задание 1

1. Создайте приложение с меню, согласно образцу ниже. Главный экран приложения можно оформить на свой вкус или по образцам предыдущих работ.



2. В нем расскажите о себе в разделах:
 - Обо мне (образование, параллельно с учебой/работой я увлекаюсь тем-то и тем-то)
 - Моя учеба/работа/хобби (как проходит моя учеба/работа/хобби, лучшие моменты и прочее...)
 - Моя работа/хобби (работа/хобби посвящена ...)
 - Мои планы (я планирую работать/уже работаю в сфере, но если меня пригласит JetBrains, Google, NetCracker и т.д., я пойду туда)
 - Контакты (если вам интересны мои знания и умения, и вы хотите супер сотрудника, то пишите-звоните)

3.2. Оценочные средства промежуточной аттестации

а) Перечень теоретических вопросов (для оценки знаний) – традиционная сдача зачета:

1. Дайте обзор мобильных платформ, технологий, средств и сред разработки мобильных приложений.
2. Базовый синтаксис языков программирования, используемых для разработки мобильных приложений (Java, Kotlin и др.). Объектно-ориентированное программирование в мобильной разработке.
3. Среда разработки Android Studio, ее возможности, структура проекта, компоненты приложения. Git - распределённая система управления версиями, ее функционал для работы в Android Studio.
4. Эмулятор в Android Studio, тестирование приложения, отладка, подключение по USB.
5. Структура приложения в ОС Android и его жизненный цикл.
6. Проект, его написание, правило построения сборки, возможности отладки и тестирования.
7. Модель MVC в мобильной разработке и другие паттерны проектирования.
8. Activity. Intent. Основы создания интерфейса и его элементы.
9. Layout. Фрагменты. Многопоточность, разрешения.
10. AdapterViews. RecyclerView. Сервисы. API. Сетевое взаимодействие. JSON.
11. Ресурсы. Адаптеры и списки. Основы архитектуры клиент-сервер.
12. Работа с файловой системой. Работа с базами данных SQLite. Создание, открытие БД и работа с таблицами. Взаимосвязь структур данных, баз данных с интерфейсом приложения. Способы хранения данных в Android. SharedPreferences.
13. Content Provider. BroadcastReceivers. Стили и темы. CustomView.
14. Анимация. Публикация приложения. MaterialDesign.
15. Основы разработки мобильных приложений для iOS.
16. Основы кроссплатформенной разработки мобильных приложений.
17. Описание технического задания на разработку мобильного приложения.
18. Применение языка разметки XML для описания интерфейса приложения. Типы тегов XML и способы динамического создания интерфейса.
19. Стандартные библиотеки, используемые при разработке приложений для платформы Android.
20. Шифрование в Android, безопасность данных, защита мобильного приложения, API-ключи, регистрация прав на мобильное приложение. Публикация мобильного приложения в магазинах приложений.

Тема 1.3.2 Создание и тестирование модулей для мобильных приложений

Практических заданий:

1. Требуется разработать приложение с графическим пользовательским интерфейсом, поддерживающее

создание/редактирование/удаление/поиск заметок. Два варианта хранения заметок: А) в базе SQLite. Б) С использованием файловой системы.

2. Создать приложение с графическим пользовательским интерфейсом с функциями: Определение местоположения пользователя на карте Google Map; Определение скорости и направления движения пользователя; Масштабирование карты. Программа должна быть конфигурируемой. Настройки: Режим определения местоположения (через GPS либо по сотам); Включение/отключение режима поиска.

3. Разработать приложение-таймер с использованием датчика ориентации в виде песочных часов. Каждый раз для того чтобы активировать таймер, необходимо перевернуть экран мобильного устройства вверх ногами. Используйте анимацию для показа «перетекающего песка» и переворота песочных часов. Для задания времени перетекания песка требуется разработать push-notification сервер. Через форму ввода на сервере можно отправлять на клиент (приложение-таймер) указанное время (числовой ввод).

4. Разработка программы для обмена мгновенными сообщениями. Требуется разработать приложение для обмена мгновенными сообщениями через Wi-Fi/Bluetooth. Поддерживаемые режимы: а. Активный режим. Приложение занимает весь экран, содержит поля для отправки сообщений и список принятых сообщений. б. Режим уведомлений. Приложение через уведомления показывает принятые сообщения.

5. Разработка мобильного сайта с адаптивным дизайном. Требуется разработать сайт, пригодный для просмотра на смартфонах и планшетах, с использованием принципов адаптивного дизайна: 1) Гибкая сетка. 2) Пропорциональные шрифты. 3) Масштабируемые изображения. 4) Медиа-запросы

Вопросы для экзамена:

1. Понятие мобильного устройства. Основные типы мобильных устройств.
2. Понятие коммуникационных технологий.
3. Технология GSM: 1G, 2G.
4. Технология GSM: 3G, 4G.
5. Технология GSM: 5G.
6. Технология WiFi.
7. Технология Bluetooth.
8. Виды организации беспроводных сетей.
9. Понятие программной платформы.
10. Платформа Android.
11. Платформа iOS.
12. Платформа WindowsPhone.
13. Язык программирования Java.
14. Типы мобильных приложений.
15. Архитектура мобильных приложений.
16. Типы соединений.
17. Конфигурации и профили J2ME. Профиль Foundation.

18. Конфигурации и профили J2ME. Профиль Personal.
19. Конфигурации и профили J2ME. Профиль RMI.
20. Конфигурации и профили J2ME. Профиль Mobile Information Device.

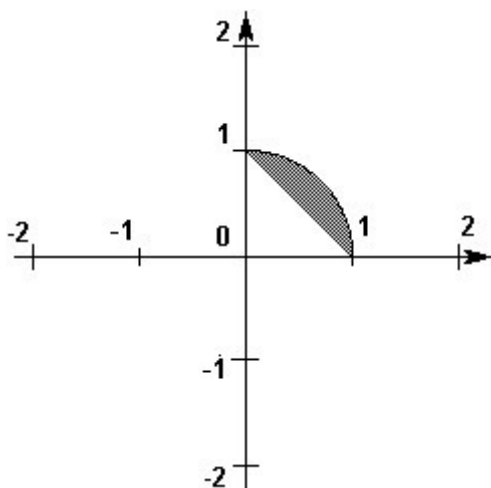
Раздел 4. МДК. 01.04. Системное программирование

Тема 1.4.1 Программирование на языке низкого уровня

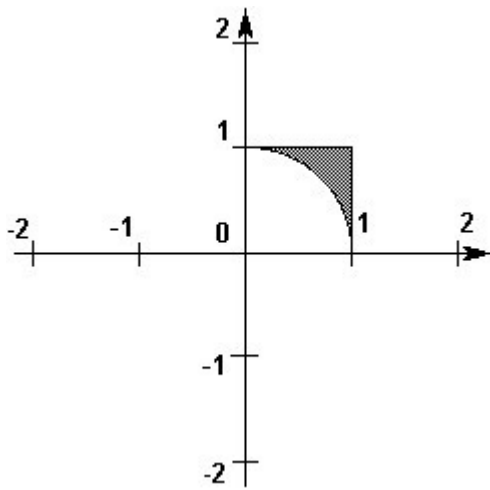
Практические задания

1. Написать программу, которая удаляет из введенной с клавиатуры строки первые пробелы (если их нет, то оставить как есть)
2. Подсчитать количество слов в строке
3. Выделить первое слово из строки
4. Найти максимальное из трех введенных целых чисел
5. Найти сумму цифр трехзначного числа
6. Решить квадратное уравнение $ax^2+bx+c=0$, коэффициенты уравнения вводятся с клавиатуры
7. Определить, кратна ли трем сумма цифр трехзначного числа. Число вводится с клавиатуры
8. Найти площадь прямоугольного треугольника по известным катету и гипотенузе. Исходные данные вводятся с клавиатуры
9. Переписать все символы строки в обратном порядке
10. Написать программу, которая вводит координаты точки (x, y) и определяет, попадает ли точка в заштрихованную область на рисунке.

Попадание на границу области считать попаданием в область.



11. Построить программу, которая вводит координаты точки (x, y) и определяет, попадает ли точка в заштрихованную область на рисунке. Попадание на границу области считать попаданием в область.



12. Написать программу для сортировки одномерных массивов
13. Вывести на экран все трехзначные числа, в которых хотя бы две цифры одинаковые
14. Вывести на экран все трехзначные числа, которые начинаются и заканчиваются на одну и ту же цифру
15. Вывести на экран таблицу значений функции $y=x^3$ при $x=-4\dots4$. Шаг изменения x вводится с клавиатуры
16. Найти сумму ряда с точностью 0.0001

$$a_n = (-1)^n \frac{n+1}{n^3+2}$$

17. В одномерном массиве, состоящем из 10 случайных чисел от -10 до 10 найти сумму положительных элементов
18. В одномерном массиве, состоящем из 10 случайных чисел от -10 до 10 количество отрицательных элементов
19. В одномерном массиве, состоящем из 10 случайных чисел от -10 до 10 найти среднее арифметическое четных элементов
20. В одномерном массиве, состоящем из 10 случайных чисел от -10 до 10 заменить все отрицательные элементы нулями
21. В одномерном массиве, состоящем из 10 случайных чисел от -10 до 10 переставить местами минимальный и максимальный элементы
22. Определить среднее арифметическое элементов главной диагонали квадратной матрицы
23. Определить максимальный элемент побочной диагонали квадратной матрицы
24. Найти сумму нечетных элементов прямоугольного массива, состоящего из пяти строк и семи столбцов
25. В прямоугольном массиве, состоящем из 7 строк и 10 столбцов заменить все отрицательные элементы их абсолютными значениями
26. Определить, является ли сумма элементов матрицы 3×2 нечетной
27. В прямоугольном массиве, состоящем из 7 строк и 10 столбцов заменить все отрицательные элементы их абсолютными значениями
28. Определить, является ли сумма элементов матрицы 3×2 нечетной
29. В прямоугольном массиве, состоящем из 7 строк и 10 столбцов заменить все отрицательные элементы их абсолютными значениями
30. Определить, является ли сумма элементов матрицы 3×2 нечетной

Перечень тестовых заданий по теме: **“Программирование на языке низкого уровня”**

1. Выберите неверно составленные инструкции
 - out dx, ax
 - mov ebx, [esp+4]
 - mov ebx, cr0
 - jz 0008:0000h
2. Какие утверждения верны для модели памяти *comrast*?
 - Адресация данных ближняя, адресация кода дальняя
 - Адресация данных ближняя, адресация кода ближняя
 - Адресация данных дальняя, адресация кода ближняя
 - Ничего из приведённого
3. Что делает следующая инструкция
 - lea eax, [eax*8][eax]
 - Умножение eax на 16
 - Загрузка содержимого памяти по адресу ds:[8*eax] в eax
 - Умножение eax на 9
4. В начале процедуры регистры были занесены в стек в таком порядке: eax, ebx, ecx, edx. В каком порядке их нужно восстанавливать в конце процедуры?
 - edx, ecx, eax, ebx
 - eax, ebx, ecx, edx
 - ebx, ecx, edx, eax
 - eax, ecx, ebx, edx
 - edx, ecx, ebx, eax
5. Можно ли работать с 32-х битными регистрами в реальном (16-ти битном) режиме процессора i386
 - Да
 - Нет
6. Какой флаг будет проверяться при выполнении инструкции условного перехода *je*?
 - af
 - cf
 - bf
 - ef
 - zf
7. Какой модели организации памяти из перечисленных не существует?
 - Сплошная модель защищённого режима
 - Сплошная модель памяти реального режима
 - Сегментированная модель памяти защищённого режима
 - Сегментированная модель памяти реального режима
8. Какой флаг устанавливается в 1 если результат арифметической команды требует коррекции?
 - cf
 - pf
 - af
 - df

- sf

9. Что называется физическим адресом?

- Адрес памяти хранящийся в сегментном регистре
- Адрес памяти используемый

10. Почему дополнительный код используется для представления целых отрицательных чисел практически во всех микропроцессорах и микроконтроллерах ? Укажите все подходящие варианты

- Дополнительный код легко читабелен и поэтому удобен для отладки
- Используя дополнительный код представления отрицательных чисел операцию вычитания можно заменить на операцию сложения и таким образом реализовать эти две операции на одном сумматоре.
- Дополнительный код не является распространённым представлением отрицательных чисел. В большинстве случаев используется обратный код

11. В чём различия инструкций ret и retf

- ret использует дальний адрес (селектор/смещение и смещение) из стека
- Эти инструкции идентичны
- retf восстанавливает регистр флагов из стека
- retf использует дальний адрес (сегмент/селектор и смещение) из стека

12. В чём заключается разница (если она есть) между двумя следующими инструкциями? Выберите все верные утверждения.

`and eax, 0FFh`

и

`test eax, 0FFh`

- Разница отсутствует
- Первая не изменяет состояния флагов, вторая изменяет
- Первая выполняет операцию «побитовое И», вторая «Логическое И».
- Первая сохраняет результат операции, вторая нет.

13. Что произойдёт в результате выполнения команды

`Jmp $`

- Переход в защищённый режим
- Данный код вызовет ошибку трансляции
- Перезагрузка компьютера
- Зацикливание программы

14. Одно ли и то же выполняют два следующих куска кода

1)

`mov ax, 1020h`

`mov dx, 0377h`

`out dx, ax`

2)

`mov ax, 1020h`

`mov dx, 0377h`

`out dx, al`

`inc dx`

`out dx, ah`

- Да

- Нет
- Зависит от устройства

15. Возможна ли прямая передача данных между ячейками памяти средствами набора команд 8086?

- Да
- Нет

16. Какое семейство процессоров первым стало поддерживать полноценный 32-х битный защищенный режим?

- 80486
- Pentium (80586)
- 80286
- 80386
- 8086/8088

17. Какой способ адресации имеет наиболее компактный код?

- Регистровый
- Непосредственный
- Прямой
- регистровый относительный

18. Регистр `ax = 0`, какое значение будет содержать `ax` после выполнения инструкции `"dec al"`?

- 0x0
- 0xf0f0
- 0xffff
- 0xff00
- Процессор выведет ошибку на экран
- 0x00ff
- 0x0f0f

19. Какой размер адресной шины в 8086?

- 36 бит
- 64 бита
- 16 бит
- 32 бита
- 20 бит

20. Какое значение будет находится в регистре `ax` после выполнения следующих команд?

```
clc
```

```
mov ax, ff00h
```

```
mov cl,4
```

```
rcl ax, cl
```

- f000h
- f00fh
- f007h
- 700fh

21. Выберите правильные утверждения:

- ESI - индекс источника, EDI - индекс приемника
- ESP, EBP - сегментные адресные регистры

- EAX, EBX, ECX, EDX - регистры общего назначения
- CS, DS - сегментные адресные регистры
- EDI - индекс источника, ESI - индекс приемника

22. Удастся ли в 32-х битном защищённом режиме получить доступ к памяти выше 4 ГиБ, если создать сегмент с базой большей нуля и пределом в 4 ГиБ?

- Да, но только при включенном PAE.
- Да, это работает всегда.
- Да, но только при выключенном PAE.
- Нет, даже при включенной 36-битной адресации (PAE) все процессы по прежнему смогут адресовать только 4 ГиБ.

5.2. Примерный перечень вопросов для проведения зачёта

Теоретическая часть

- 1) Архитектура процессора Intel Pentium.
- 2) Сегменты, принцип сегментации памяти.
- 3) Использование стека.
- 4) Идентификаторы, переменные, метки, имена, ключевые слова.
- 5) Типы данных.
- 6) Директивы определения данных.
- 7) Структура программы на языке ассемблера.
- 8) Директива ASSUME и INCLUDE.
- 9) Сегментные регистры по умолчанию. Адресация.
- 10) Команды для работы со стеком.
- 11) Команды арифметического сложения ADD и ADC.
- 12) Команды арифметического вычитания SUB и SBB.
- 13) Команды умножения MUL и IMUL.
- 14) Команды деления DIV и IDIV.
- 15) Команды выполняющие логические операции.
- 16) Команды выполняющие операции сдвигов.
- 17) Команды сравнения и передачи управления.
- 18) Команды организации циклов.
- 19) Организация подпрограмм.

Практическая часть

- 1) Разработать программу преобразования строки в верхний регистр, вывести строку на экран.
- 2) Разработать программу преобразования числа в двоичную форму представления, вывести число на экран.
- 3) Разработать подпрограмму вычисления факториала числа. Вывести результат на экран.
- 4) Разработать программу удаления во введённой строке пробелов, результат вывести на экран.
- 5) Разработать программу определения чётности числа, результат вывести на экран.
- 6) Разработать программу копирования символьных строк, заканчивающихся нулём.

- 7) Разработать программу сцепления символьных строк, заканчивающихся нулём.
- 8) Разработать программу объединения целочисленных массивов.
- 9) Разработать программу сравнения двух символьных строк, заканчивающихся нулём. Программа должна выдавать один из следующих вариантов: равны, не равны, меньше, больше.
- 10) Разработать программу замены во введённой строке первых строчных букв слов на прописные.
- 11) Разработать программу которая используя подпрограмму производит реверсирование элементов массива. Адрес массива и количество элементов в нём передаётся в подпрограмму через стек.
- 12) Написать программу которая позволяет возводить целое число в целую степень и выводить результат на экран. Операцию возведения числа в степень оформить в виде процедуры.
- 13) Разработать программу, которая используя подпрограмму находит количество чётных элементов в массиве чисел (двойных слов). Адрес первого элемента массива и количество элементов в нём передаётся в качестве параметров в подпрограмму через стек.
- 14) Разработать программу, которая используя подпрограмму находит наибольший элемент в массиве чисел (двойных слов). Адрес первого элемента массива и количество элементов в нём передаётся в качестве параметров в подпрограмму через стек.
- 15) Разработать программу, которая используя подпрограмму находит количество чётных элементов массива чисел (двойных слов). Адрес первого элемента массива и количество элементов в нём передаётся в качестве параметров в подпрограмму через стек.
- 16) Разработать программу, которая используя подпрограмму находит количество слов в строке. Адрес первого символа строки передаётся в качестве параметров в подпрограмму через стек. Строка заканчивается нулевым байтом.
- 17) Разработать программу, которая заменяет в массиве чисел (двойных слов) числа меньше 100 на 0.
- 18) Разработать программу, которая заменяет в массиве чисел (двойных слов) числа меньше 100 на 0.
- 19) Разработать программу, которая осуществляет реверсирование строки, выводит результат на экран.
- 20) Разработать программу, которая осуществляет сложение двух однобайтовых чисел в ASCII представлении.
- 21) Разработать программу, которая осуществляет деление двух чисел в ASCII представлении.
- 22) Разработать программу, которая осуществляет сравнение массивов, содержащих числа с плавающей точкой.

Тема 1.4.2. Программирование в Windows

Задание 1. Функции получения системной информации

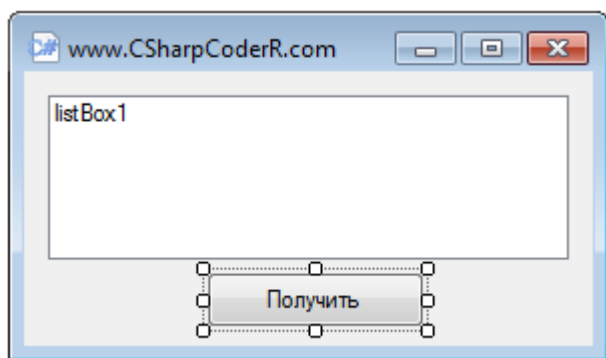
Для получения информации о температуре центрального процессора и системной платы воспользуемся одной из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows - Windows Management Instrumentation (WMI). Так как WMI построен по объектно-ориентированному принципу, то все данные операционной системы представлены в виде объектов и их свойств, и методов. Все классы группируются в пространства имен, которые иерархически упорядочены и логически связаны друг с другом по определенной технологии или области управления. В WMI имеется одно корневое пространство имен Root, которое в свою очередь имеет 4 подпространства: CIMv2, Default, Security и WMI. Воспользуемся последним подпространством «WMI» и его классом «MSAcpi_ThermalZoneTemperature». Данный класс содержит данные об текущей температуре системной платы и центрального процессора.

Создайте проект Windows Form в Microsoft Visual Studio и добавьте на главную форму два компонента:

ListBox

Button

У вас получится примерно такой вариант:



Сделайте двойной клик левой клавишей мыши по компоненту «Button» и перейдите в автоматически созданный метод «button1_Click», добавьте в него приведенный ниже листинг.

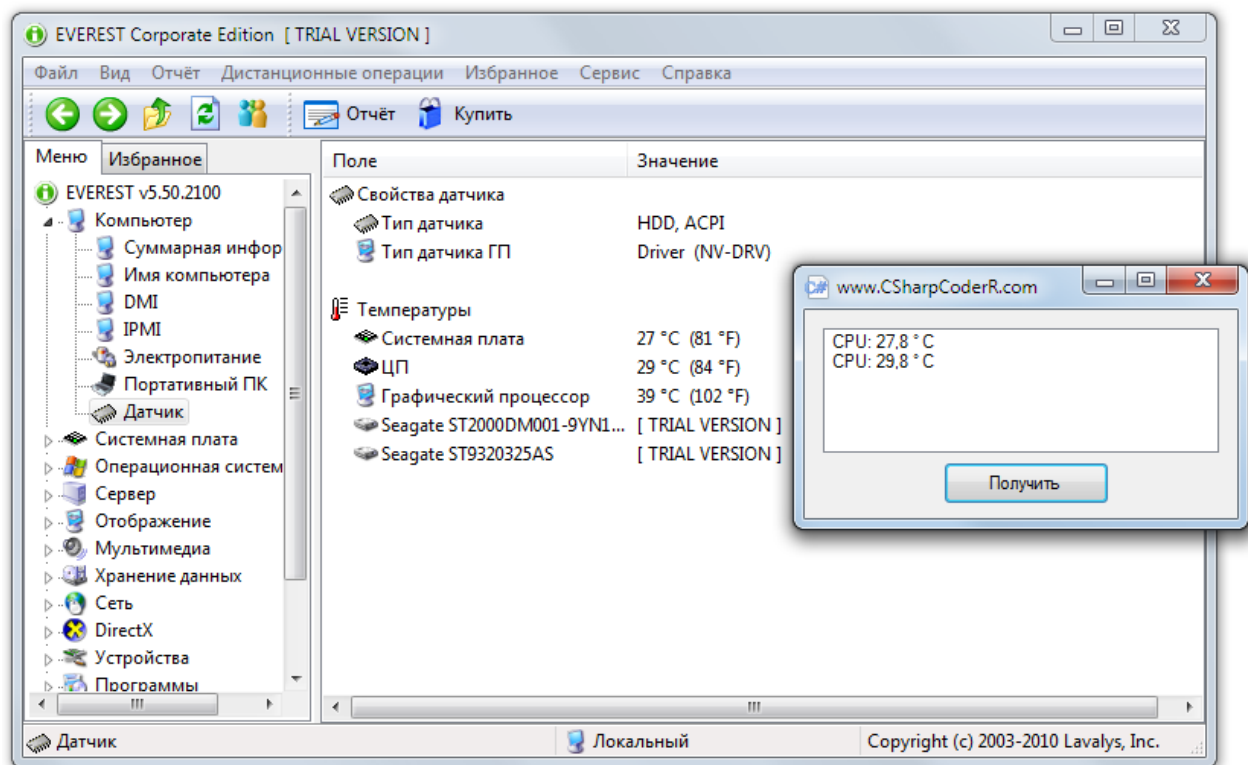
```
listBox1.Items.Clear();  
  
try  
{  
    Double CPUtpmt = 0;  
  
    System.Management.ManagementObjectSearcher MOS = new  
    System.Management.ManagementObjectSearcher("root\\WMI",  
    "SELECT * FROM MSAcpi_ThermalZoneTemperature");  
    foreach (System.Management.ManagementObject Mo in MOS.Get())  
    {
```

```

CPUtprt =
Convert.ToDouble(Convert.ToDouble(Mo.GetPropertyValue("CurrentTemperature").ToString())) - 2732) / 10.0;
listBox1.Items.Add(" CPU: " + CPUtprt.ToString() + " ° C");
}
}
catch (ManagementException ex)
{
MessageBox.Show("Ошибка получения данных " + ex.Message);
}

```

Запустите ваш проект и нажмите на единственную кнопку на главной форме. Для проверки работы программы воспользуйтесь любым приложением, предоставляющим информацию с температурных датчиков. Например, вы можете воспользоваться программой Everest.



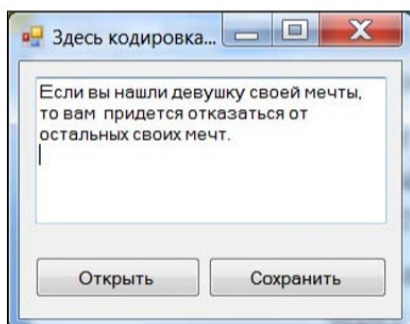
Задание 2. Чтение, запись текстовых и бинарных файлов, текстовый редактор
Чтение/запись текстового файла в кодировке Unicode . Обработка исключений
try...catch

Очень распространенной задачей является сохранение данных на диске в текстовом (не двоичном) формате. Понятно, что такое деление условно, поскольку и текстовые, и нетекстовые файлы на самом деле являются двоичными файлами. Но если нетекстовые файлы открыть, например, Блокнотом, мы увидим то, что называют "нечитаемым месивом" — отсюда такая классификация. Часто данные сохраняют на диск именно в текстовом формате, поскольку в этом случае

сохраненные данные можно читать, редактировать любым текстовым редактором, например, Блокнотом, или TextEdit, или другими. Следует также уметь и читать текстовые данные в своей пользовательской программе. Казалось бы, это очень простая задача. Например, чтение текстового файла сводится буквально к нескольким строчкам:

```
// Создание экземпляра StreamReader для чтения из файла: var Читатель = new
System.IO.StreamReader(@"D:\Text1.txt"); // Непосредственное чтение всего файла
в текстовое поле: textBox1.Text = Читатель.ReadToEnd(); Читатель.Close(); //
```

заккрытие файла Однако есть некоторые серьезные нюансы. Напишем программу, содержащую на экранной форме текстовое поле и две командные кнопки. При щелчке мышью на первой кнопке происходит чтение текстового файла в текстовое поле в кодировке Unicode. При щелчке на второй кнопке отредактированный пользователем текст в текстовом поле сохраняется в файл на диске. Запустим систему Visual Studio 12 и в окне New Project выберем шаблон Windows Forms Application Visual C#. Новую программу назовем TxtUnicode. Далее в форму из панели Toolbox перенесем текстовое поле и две командные кнопки. Для текстового поля в окне Properties сразу укажем для свойства Multiline значение True, чтобы текстовое поле имело не одну строку, а столько, сколько поместится в растянутом указателем мыши поле. Одна кнопка предназначена для открытия файла, а другая — для сохранения файла на машинном носителе. В листинге приведен текст этой программы. Листинг программы.



```

// Программа для чтения/записи текстового файла в кодировке Unicode
using System;
using System.Drawing;
using System.Windows.Forms;
// Другие директивы using удалены, поскольку они не используются в данной
// программе
namespace TxtUnicode
{
    public partial class Form1 : Form
    {
        // Объявляем ИмяФайла вне всех подпрограмм, чтобы эта переменная
        // была "общей" во всех процедурах обработки событий:
        String ИмяФайла;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            // Установка начальных значений:
            textBox1.Multiline = true; textBox1.Clear();
            textBox1.Size = new Size(268, 112);
            button1.Text = "Открыть"; button1.TabIndex = 0;
            button2.Text = "Сохранить";
            this.Text = "Здесь кодировка Unicode";
            ИмяФайла = @"D:\Text1.txt";
        }
        private void button1_Click(object sender, EventArgs e)
        {
            // Щелчок на кнопке Открыть.
            // Русские буквы будут корректно читаться,
            // если открыть файл в кодировке UNICODE:
            try
            {
                // Создание объекта StreamReader для чтения из файла:
                var Читатель = new System.IO.StreamReader(ИмяФайла);
                // Непосредственное чтение всего файла в текстовое поле:
                textBox1.Text = Читатель.ReadToEnd();
                Читатель.Close(); // закрытие файла
                // Чтение текстовый файл в кодировке UNICODE в массив
                // строк можно также таким образом (без Open и Close):
                // var МассивСтрок = System.IO.File.
                // ReadAllLines(@"D:\Text1.txt");
            }

            catch (System.IO.FileNotFoundException Ситуация)
            {
                // Обработка исключительной ситуации:
                MessageBox.Show(Ситуация.Message + "\n" +
                    "Нет такого файла", "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
            }
            catch (Exception Ситуация)
            {
                // Ответ о других ошибках:
                MessageBox.Show(Ситуация.Message, "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
        private void button2_Click(object sender, EventArgs e)
        {
            // Щелчок на кнопке Сохранить:
            try
            {
                // Создание объекта StreamWriter для записи в файл:
                var Писатель = new System.IO.
                    StreamWriter(ИмяФайла, false);
                Писатель.Write(textBox1.Text);
                Писатель.Close();
                // Сохранить текстовый файл можно также таким образом
                // (без Close), причем если файл уже существует,
                // то он будет заменен:
                // System.IO.File.WriteAllText(
                //     @"D:\tmp.tmp", textBox1.Text);
            }
            catch (Exception Ситуация)
            {
                // Ответ обо всех возможных ошибках:
                MessageBox.Show(Ситуация.Message, "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }
}

```

Несколько слов о блоках `try`, которые, как можно видеть, присутствуют в данном программном коде. Логика использования `try` следующая: попытаться (`try`) выполнить некоторую задачу, например прочитать файл. Если задача решена некорректно (например, файл не найден), то "перехватить" (`catch`) управление и обработать возникшую (исключительную, `Exception`) ситуацию. Как видно из текста программы, обработка исключительной ситуации свелась к информированию пользователя о недоразумении.

При обработке события "щелчок на кнопке Открыть" организован ввод файла `D:\Text1.txt`. Обычно в такой ситуации для выбора файла применяют элемент управления `OpenFileDialog`. Мы не стали использовать этот элемент управления для того, чтобы не "заговорить" проблему, а также свести к минимуму программный код. Далее создаем объект (поток) Читатель для чтения из файла. Для большей выразительности операций с данным объектом мы написали его название русскими буквами. Затем следует чтение файла `ИмяФайла` методом `ReadToEnd()` в текстовое поле `TextBox1.Text` и закрытие файла методом `Close()`. При обработке события "щелчок на кнопке Сохранить" аналогично организована запись файла на диск через объект Писатель. При создании объекта Писатель первым аргументом является `ИмяФайла`, а второй аргумент `false` указывает, что данные следует не добавить (`append`) к содержимому файла (если он уже существует), а перезаписать (`overwrite`). Запись на диск производится с помощью метода `Write()` из свойства `Text` элемента управления `TextBox1`. На рисунке приведен пример работы программы.

ВНИМАНИЕ! Запись текстового файла с помощью данной программы будет осуществляться в формате (кодировке) `Unicode`, как и чтение из файла. То есть, вы сможете читать эти файлы Блокнотом, редактировать их, но каждый раз при сохранении файлов из Блокнота вам требуется следить, чтобы кодировка была (оставалась) `Unicode`.

По умолчанию в редакторах обычно используется кодировка `ANSI` (кодировку `ANSI`, включающую русские буквы, называют еще `Windows 1251`). Некоторые редакторы, например, `RPad` ("русский" Блокнот), вообще не работают с `Unicode`. Таким образом, на сегодняшний день пока что записывать тексты в кодировке `Unicode` — это, как бы, эксклюзив. Возможно, даже наверняка, в ближайшее время многое изменится в пользу `Unicode`, поскольку информационные технологии развиваются очень быстро.

Задание 3. Чтение/запись текстового файла в кодировке Windows 1251

В данном примере также на экранной форме разместим текстовое поле и две командные кнопки (назначение этих элементов управления аналогично предыдущему примеру). Однако чтение и запись текстового файла здесь происходит в кодировке `Windows 1251 (ANSI)`. Поскольку структура этой программы аналогична предыдущей, сразу обратимся к ее коду в листинге.

Листинг

```

// Программа для чтения/записи текстового файла в кодировке Windows 1251
using System;
using System.Drawing;
using System.Windows.Forms;
// Другие директивы using удалены, поскольку они не используются в данной
// программе
namespace TXT_1251
{
    public partial class Form1 : Form
    {
        String ИмяФайла;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            textBox1.Multiline = true; textBox1.Clear();
            textBox1.Size = new Size(268, 112);
            button1.Text = "Открыть"; button1.TabIndex = 0;
            button2.Text = "Сохранить";
            this.Text = "Здесь кодировка Windows 1251";
            ИмяФайла = @"D:\Text2.txt";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Нажатие на кнопку Открыть
            try
            {
                // Чтобы русские буквы читались корректно, объявляем
                // объект Кодировка:
                var Кодировка = System.Text.Encoding.GetEncoding(1251);
                // Создаем экземпляр StreamReader для чтения из файла
                var Читатель = new System.IO.
                    StreamReader(ИмяФайла, Кодировка);
                textBox1.Text = Читатель.ReadToEnd();
                Читатель.Close();
                // Чтение текстового файла в кодировке Windows 1251 в массив
                // строк можно также таким образом (без Open и Close):
                var МассивСтрок = System.IO.File.
                    ReadAllLines(@"D:\Text2.txt", Кодировка);
            }
            catch (System.IO.FileNotFoundException Ошибка)
            {
                // Обработка исключительной ситуации:
                MessageBox.Show(Ошибка.Message + "\nНет такого файла",
                    "Ошибка", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
            }
            catch (Exception Ошибка)
            {
                // Ответ о других ошибках
                MessageBox.Show(Ошибка.Message, "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
        private void button2_Click(object sender, EventArgs e)
        {
            // Нажатие на кнопку Сохранить:
            try
            {
                var Кодировка =
                    System.Text.Encoding.GetEncoding(1251);
                // Создаем экземпляр StreamWriter для записи в файл:
                var Писатель = new System.IO.StreamWriter(
                    ИмяФайла, false, Кодировка);
                Писатель.Write(textBox1.Text);
                Писатель.Close();
                // Сохранить текстовый файл можно также таким образом (без
                // Close), причем если файл уже существует, то он будет
                // заменен:
                // System.IO.File.WriteAllText(@"D:\tmp.tmp",
                //                                     textBox1.Text, Кодировка);
            }
        }
    }
}

```

```

        catch (Exception Ситуация)
        {
            // Отчет обо всех возможных ошибках:
            MessageBox.Show(Ситуация.Message, "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
}

```

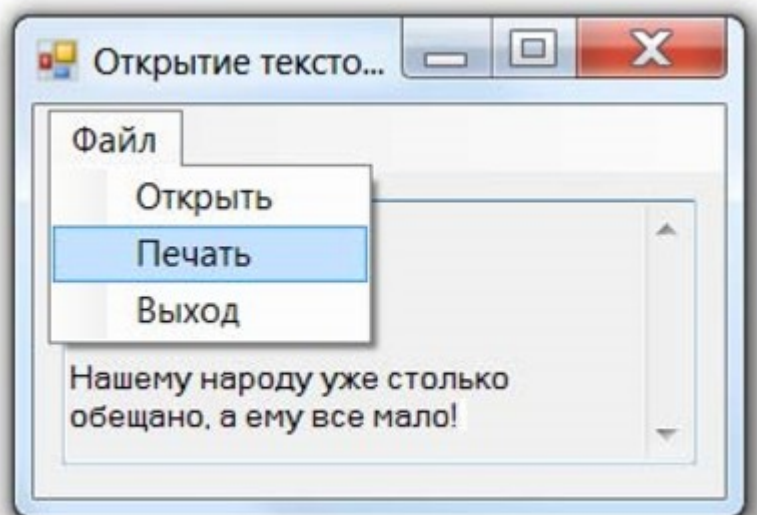
Текст этой программы отличается от предыдущей лишь тем, что здесь введен новый объект — Кодировка. Метод `GetEncoding(1251)` устанавливает кодовую страницу Windows 1251 для объекта Кодировка. Можно убедиться в этом, если распечатать свойство `Кодировка.HeaderName`. При создании объекта Читатель используются уже два аргумента: имя файла и объект Кодировка, указывающий, в какой кодировке (для какой кодовой страницы) читать данные из текстового файла. А при создании объекта Писатель используются три аргумента: имя файла, установка `false` (для случая, если файл уже существует, нужно будет не добавлять новые данные, а создавать новый файл) и объект Кодировка, указывающий, в какой кодировке писать данные в файл. Заметьте, что при обработке события "щелчок на второй кнопке" мы объявили переменные Кодировка и Писатель просто как `var`, т. е. типы этих переменных выводятся из выражения инициализации.

Задание 4. Печать текстового документа

Любой текстовый редактор (и не только текстовый) должен иметь возможность печати на принтере. Мы сознательно не добавляли такую возможность в текстовые редакторы, приведенные в предыдущих разделах, чтобы не запутать читателя. Понятно, что чем больше функциональности имеет программа, тем сложнее ее программный код и тем труднее текст программы для понимания. А наша задача — выразительно и ярко демонстрировать технологии в максимально простой форме. Программа, представленная в этом примере, имеет следующие скромные возможности: открыть в стандартном диалоговом окне Windows текстовый файл, просмотреть его в окне программы (в текстовом поле) без возможности изменения текста (`ReadOnly`) и при желании пользователя вывести этот текст на принтер. Эту программу назовем `TxtPrint`.

Чтение, запись текстовых и бинарных файлов, текстовый редактор

Запустив Visual Studio и выбрав шаблон Windows Forms Application C#, в дизайнере формы из панели элементов перенесем следующие элементы управления: текстовое поле `TextBox` со свойством `MultiLine = true`, меню `MenuStrip` с пунктами меню: Открыть, Печать и Выход (рис.), а также элементы управления `OpenFileDialog` и `PrintDocument`. Текст программы представлен в листинге



Листинг программы

```
// Программа позволяет открыть в стандартном диалоговом окне текстовый файл,
// посмотреть его в текстовом поле без возможности изменения текста
// (ReadOnly) и при желании пользователя вывести этот текст на принтер
using System;
using System.Drawing;
using System.Windows.Forms;
// Другие директивы using удалены, поскольку они не используются в данной
// программе
namespace TextPrint
{
    public partial class Form1 : Form
    {
        System.IO.StreamReader Чтение;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Открытие текстового файла и его печать";
            textBox1.Multiline = true; textBox1.Clear();
            textBox1.Size = new Size(260, 112);
            textBox1.ScrollBars = ScrollBars.Vertical;
            textBox1.ReadOnly = true;
            // До тех пор, пока файл не прочитан в текстовое поле,
            // не должен быть виден пункт меню "Печать..."
            печатьToolStripMenuItem.Visible = false;
            openFileDialog1.FileName = null;
        }
    }
}
```

```

private void открытьToolStripMenuItem_Click(
    Object sender, EventArgs e)
{
    // Клик по пункту меню "Открыть":
    openFileDialog1.Filter =
        "Текстовые файлы (*.txt)|*.txt|All files (*.*)|*.*";
    openFileDialog1.ShowDialog();
    if (openFileDialog1.FileName == null) return;
    try
    {
        // Создание потока StreamReader для чтения из файла
        Читатель = new System.IO.StreamReader(
            openFileDialog1.FileName,
            System.Text.Encoding.GetEncoding(1251));
        // - здесь заказ кодовой страницы Win1251 для русских букв
        textBox1.Text = Читатель.ReadToEnd();
        Читатель.Close();
        печатьToolStripMenuItem.Visible = true;
    }
    catch (System.IO.FileNotFoundException Ситуация)
    {
        MessageBox.Show(Ситуация.Message +
            "\nНет такого файла", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
    catch (Exception Ситуация)
    {
        // Ответ о других ошибках:
        MessageBox.Show(Ситуация.Message, "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}

private void печатьToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Пункт меню "Печать"
    try
    {
        Читатель = new System.IO.StreamReader(
            openFileDialog1.FileName,
            System.Text.Encoding.GetEncoding(1251));
        // - здесь заказ кодовой страницы Win1251 для русских букв
        try
        {
            printDocument1.Print();
        }
        finally
        {
            Читатель.Close();
        }
    }
}

```

```

        catch (Exception Ошибка)
        {
            MessageBox.Show(Ошибка.Message);
        }
    }

    private void printDocument1_PrintPage(object sender,
        System.Drawing.Printing.PrintPageEventArgs e)
    {
        // Событие вывода на печать страницы (PrintPage)
        // Чтобы получить пустой обработчик этого события, можно дважды
        // кликнуть на значке printDocument1 в дизайнера формы
        Single СтрочкаСтраницы = 0.0F;
        Single Y = 0;
        var ЛевыйКрай = e.MarginBounds.Left;
        var ВерхнийКрай = e.MarginBounds.Top;
        var Строка = String.Empty;
        var Шрифт = new Font("Times New Roman", 12.0F);
        // Вычисляем количество строк на одной странице
        СтрочкаСтраницы = e.MarginBounds.Height /
            Шрифт.GetHeight(e.Graphics);

        // Печатаем каждую строку файла
        var i = 0; // - счет строк
        while (i < СтрочкаСтраницы)
        {
            Строка = Читатель.ReadLine();
            if (Строка == null) break; // выход из цикла
            // Для VB: If Строка Is Nothing Then Exit While
            Y = ВерхнийКрай + i *
                Шрифт.GetHeight(e.Graphics);
            // Печать строки
            e.Graphics.DrawString(Строка, Шрифт, Brushes.Black,
                ЛевыйКрай, Y, new StringFormat());
            i = i + 1; // или i += 1 - счет строк
        }

        // Печать следующей страницы, если есть еще строки файла
        if (Строка != null) e.HasMorePages = true;
        // Для VB: If Строка <> Null Then ...
        else e.HasMorePages = false;
    }

    private void выходToolStripMenuItem_Click(
        object sender, EventArgs e)
    {
        // Выход из программы
        this.Close();
    }
}

```

Здесь при обработке события загрузки формы `Form1_Load` запрещаем пользователю редактировать текстовое поле: `ReadOnly = true`. Также назначаем свойству `ПечатьToolStripMenuItem.Visible = false` (пункт меню Печать), т. е. в начале работы программы пункт меню Печать пользователю не виден (поскольку пока распечатывать нечего — необходимо вначале открыть текстовый файл). Остальные присваивания при обработке события `Form1_Load` очевидны. При обработке события "щелчок на пункте меню Открыть" вызываем стандартный диалог `openFileDialog` и организуем чтение файла через создание потока `StreamReader`. Эти процедуры мы уже рассматривали подробно в разделах о текстовых редакторах, поэтому оставим их без комментария. Замечу только, что после чтения файла в текстовое поле назначаем видимость пункту меню Печать: `печатьToolStripMenuItem.Visible = true`, поскольку уже есть, что печатать на принтере (файл открыт). Представляет интерес обработка события "щелчок на пункте меню Печать". Здесь во вложенных блоках `try...finally...catch` программа еще раз создает поток `StreamReader`, а затем запускает процесс печати документа `printDocument1.Print`. Если ничего более не программировать, только метод `printDocument1.Print`, то принтер распечатает лишь пустую страницу. Чтобы принтер распечатал текст, необходимо обработать событие `PrintPage` (см. текст программы), которое создает объект `PrintDocument`. То есть роль метода `Print` — это создать событие `PrintPage`. Обратите внимание на обработку события `printDocument1_PrintPage`. Пример обработки этого события приведен в MSDN. Вначале перечислены объявления переменных, значения некоторых из них получаем из аргументов события `e`, например, `ЛевыйКрай` — значение отступа

от левого края, и т. д. Назначаем шрифт печати — Times New Roman , 12 пунктов . Далее в цикле while программа читает каждую строку Строка из файла — Читатель . ReadLine (), а затем распечатывает ее командой (методом) DrawString . Здесь используется графический объект Graphics , который получаем из аргумента события e . В переменной i происходит счет строк. Если количество строк оказывается большим, чем число строк на странице, то происходит выход из цикла, поскольку страница распечатана. Если есть еще страницы, а программа выясняет это, анализируя содержимое переменной Строка, — если ее содержимое отличается от значения null (Строка != null), то аргументной переменной e.HasMorePages назначаем true , что инициирует опять событие PrintPage , и процедура printDocument1 _ PrintPage начинает свою работу вновь. И так, пока не закончатся все страницы e.HasMorePages = false для печати на принтере.

Задание 5. Чтение/запись бинарных файлов с использованием потока данных

Обычно программа либо что-то читает с диска в оперативную память, либо что-то пишет на диск. Писать/читать можно либо в бинарный (двоичный) файл, либо в текстовый (литерный, строковый) файл. Разница между ними состоит в том, что текстовый файл можно прочитать текстовым редактором, например, Блокнотом, а бинарный — нет. Название "бинарный" (двоичный) — условное, поскольку, по сути, и текстовый, и бинарный файлы являются двоичными файлами.

Операции с двоичным файлом гораздо более скоростные, и такой файл занимает существенно меньшее пространство на диске. Зато текстовый файл можно читать и редактировать любым текстовым редактором. Обычно программист выбирает компромисс между этими двумя преимуществами. Приведем пример самого простейшего случая записи на диск бинарного файла с данными об успеваемости одного студента. Понятно, что эта программа может быть маленькой частью большой программы по обработке успеваемости студентов в вузе. Данная программа принимает от пользователя сведения только об одном студенте в текстовые поля формы. При нажатии кнопки Сохранить программа записывает введенные сведения в двоичный файл, а при нажатии кнопки Читать читает эти сведения из двоичного файла в текстовые поля формы. Итак, в форме мы должны иметь три текстовых поля, куда пользователь может записать соответственно номер студента по порядку, фамилию студента и его средний балл успеваемости. Поэтому в форму из панели Toolbox перенесем три текстовых поля TextBox , три метки Label и две командные кнопки: Читать и Сохранить. Таким образом, получим пользовательский интерфейс, показанный на рис. 4.10. Назовем программу ReadWriteBin . Текст ее приведен в листинге.

Успеваемость с...

Номер п/п 7

Фамилия И.О. Зиборов В.В.

Средний балл 4.97

Читать Сохранить

```
// Программа для чтения/записи бинарных файлов с использованием потока данных
using System;
using System.Windows.Forms;
// Другие директивы using удалены, поскольку они не используются в данной
// программе
namespace ReadWriteBin
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```

private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Успеваемость студента";
    label1.Text = "Номер п/п"; label2.Text = "Фамилия И.О.";
    label3.Text = "Средний балл";
    textBox1.Clear(); textBox2.Clear(); textBox3.Clear();
    button1.Text = "Читать"; button2.Text = "Сохранить";
}

private void button1_Click(object sender, EventArgs e)
{
    // ЧТЕНИЕ БИНАРНОГО ФАЙЛА
    // Если такого файла нет
    if (System.IO.File.Exists(@"D:\student.usp") == false) return;
    // Создание потока Читатель
    var Читатель = new System.IO.BinaryReader(
        System.IO.File.OpenRead(@"D:\student.usp"));

    try
    {
        var Номер_пп = Читатель.ReadInt32();
        var ФИО = Читатель.ReadString();
        var СредБалл = Читатель.ReadSingle();
        textBox1.Text = Convert.ToString(Номер_пп);
        textBox2.Text = Convert.ToString(ФИО);
        textBox3.Text = Convert.ToString(СредБалл);
    }
    finally { Читатель.Close(); }
}

private void button2_Click(object sender, EventArgs e)
{
    // ЗАПИСЬ БИНАРНОГО ФАЙЛА
    // Создаем поток Писатель для записи байтов в файл
    var Писатель = new System.IO.BinaryWriter(
        System.IO.File.Open(@"D:\student.usp",
            System.IO.FileMode.Create));

    try
    {
        var Номер_пп = Convert.ToInt32(textBox1.Text);
        var ФИО = Convert.ToString(textBox2.Text);
        // Разрешим в качестве разделителя целой и дробной
        // части как запятую, так и точку:
        textBox3.Text = textBox3.Text.Replace(".", ",");
        var СредБалл = Convert.ToSingle(textBox3.Text);
        Писатель.Write(Номер_пп);
        Писатель.Write(ФИО);
        Писатель.Write(СредБалл);
    }
    finally { Писатель.Close(); }
}
}

```

Как можно видеть, в процедуре обработки события загрузки формы организована инициализация (присвоение начальных значений) элементов формы: текстовых полей, меток и кнопок. Запись файла на диск происходит при обработке события `button2 . Click`, т. е. щелчка мышью на кнопке Сохранить (рис.). С этой целью создаем поток байтов Писатель для открытия файла `student.usp`. Если такого файла не существует, то он создается (`Create`), а если файл уже есть, то он перезаписывается. Как видно из текста программы, для ее упрощения мы не использовали элемент управления `OpenFileDialog` открытия файла в диалоге. Далее преобразуем записанное в текстовых полях в более естественные типы данных. Номер по порядку `Номер_пп` — это тип целых чисел, преобразование в целый тип может быть реализовано операцией `Convert.ToInt32` (можно использовать другие функции преобразования), для переменной `СредБалл` (средний балл) больше всего подходит тип с плавающей точкой `Single`, при этом преобразование осуществляется операцией `Convert.ToSingle`. Преобразование для строковой переменной `ФИО` является необязательным и приведено для симметрии записей. Операторы `Писатель.Write` записывают эти данные в файл. После блока `finally` происходит обязательное закрытие (`Close`) файла. Чтение файла выполняется при обработке события "щелчок мышью на кнопке Читать". Как уже упоминалось, для максимального упрощения в данной программе не предусмотрено открытие файла через стандартное диалоговое окно, поэтому в начале процедуры выясняем, существует ли такой файл. Если файла `D:\student.usp` нет, то программируем выход (`return`) из обработчика данного события. Заметьте, чтобы программисту было максимально легко отслеживать ветви оператора `if`, мы написали: "Если файла нет, то `return`". При этом длинная ветвь логики "если файл

есть" не включена непосредственно в оператор `if`. Поэтому такой фрагмент программного кода читается (воспринимается) программистом легко — это типичный пример хорошо написанного программного кода. Далее создается поток байтов `Читатель` из файла `student.usp`, открытого для чтения. Чтение из потока в каждую переменную реализовано с помощью функции `ReadInt32` — читать из потока `Читатель` в переменную целого типа, аналогично функциям `ReadString` и `ReadSingle`. Затем осуществлено конвертирование этих переменных в строковый тип `Convert.ToString`. Как видно из программы, можно было изначально все текстовые поля записывать в файл без конвертирования, но при дальнейшем развитии этой программы значения полей все равно пришлось бы преобразовывать в соответствующий тип. После блока `finally` происходит закрытие (`Close`) файла. Блок `finally` выполнится всегда, даже если перед ним была команда `return`. Дальнейшее развитие этой программы может быть осуществлено путем добавления в файл сведений о других студентах. В таком случае при чтении файла количество студентов будет неизвестно. Тогда следует обработать ситуацию достижения конца файла:

`catch (EndOfStreamException e)` а затем закрыть файл.

Типовые задания для экзамена квалификационного по ПМ.01

Задание 1

Разработать приложение с пользовательским графическим интерфейсом по предложенному заданию

Задание 2

Провести отладку и тестирования разработанного приложения

Задание 3

Создать мобильное приложение, реализующее часть функционала созданного приложения