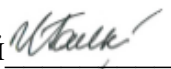


ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ
ФЕДЕРАЦИИ»
ПЕРМСКИЙ ФИНАНСОВО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ –
ФИЛИАЛ ФИНУНИВЕРСИТЕТА

РАССМОТРЕНО
на заседании кафедры
информационных систем и
программирования

Протокол № 6 от «24» января 2023 г.

Зав. кафедрой  /И.В. Галкин

УТВЕРЖДАЮ

Зам. директора по УМР
Пермского филиала
Финуниверситета, к.э.н.

 /Н.В.Галкина/
«25» января 2023 г.

**КОМПЛЕКТ
КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ
ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ
ПМ.01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ**

программы подготовки специалистов среднего звена по специальности
09.02.07 Информационные системы и программирование
квалификаций Администратор баз данных, Программист

Пермь, 2023

Комплект контрольно-оценочных средств разработан на основе Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 Информационные системы и программирование, утвержденного Министерством образования и науки Российской Федерации от 9 декабря 2016 г. № 1547; профессионального стандарта Разработчик веб и мультимедийных приложений, утвержденного приказом Министерства труда и социальной защиты Российской Федерации от 18 января 2017 г. № 44н.; а также в соответствии с Примерными программами по специальности 09.02.07 Информационные системы и программирование.

Разработчик:

Ставицкая Е.А., преподаватель Пермского филиала Финуниверситета

Одобрено Научно-методическим советом филиала
Протокол № 3 от «12» февраля 2023 г.

Содержание

I. ПАСПОРТ КОМПЛЕКТА КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ

1.1 Общие положения

1.2. Формы контроля и оценивания элементов профессионального модуля

II. РЕЗУЛЬТАТЫ ОСВОЕНИЯ МОДУЛЯ, ПОДЛЕЖАЩИЕ ПРОВЕРКЕ НА ЭКЗАМЕНЕ (КВАЛИФИКАЦИОННОМ)

2.1 Профессиональные и общие компетенции

III. ОЦЕНКА ОСВОЕНИЯ ТЕОРЕТИЧЕСКОГО КУРСА ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

3.1. Типовые задания для оценки освоения МДК 01.01 Разработка программных модулей

3.2. Типовые задания для оценки освоения МДК01.02 Поддержка и тестирование программных модулей

3.3 Типовые задания для оценки освоения МДК 01.03 Разработка мобильных приложений

3.4 Типовые задания для оценки освоения МДК 01.04 Системное программирование

IV ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ

4. 1 Промежуточная аттестация по ПМ01

4.2 Промежуточная аттестация МДК 01.01

4.3 Промежуточная аттестация МДК 01.02

4.4 Промежуточная аттестация МДК 01.03

4.5 Промежуточная аттестация МДК 01.04

4.6 Контрольные вопросы к экзамену по МДК.01.01.

4.7 Контрольные вопросы к экзамену по МДК 01.02

4.8 Контрольные вопросы к экзамену по МДК 01.03

4.9 Контрольные вопросы к экзамену по МДК 01.04

V ПЕРЕЧЕНЬ ВИДОВ РАБОТ ДЛЯ ПРОВЕРКИ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОГРАММЫ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ НА ПРАКТИКЕ

5.1. Требования к дифференцированному зачёту по учебной практике

5.2. Требования к дифференцированному зачёту по производственной практике

5.3.Производственная практика

5.4.Форма аттестационного листа

VI СТРУКТУРА КОНТРОЛЬНО-ОЦЕНОЧНЫХ МАТЕРИАЛОВ ДЛЯ ЭКЗАМЕНА (КВАЛИФИКАЦИОННОГО)

6.1.Задания экзаменуемого

6.2 Условия выполнения заданий

6.3 Структура контрольно-оценочных материалов для экзамена (квалификационного)

6.4 Общая оценка результата освоения вида профессиональной деятельности

6.5.Пакет экзаменатора

6.6 Перечень материалов, оборудования и информационных источников, используемых при аттестации

ПРИЛОЖЕНИЕ 1 ФОРМА ОЦЕНОЧНОЙ ВЕДОМОСТИ

І ПАСПОРТ КОМПЛЕКТА КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ

1.1 Общие положения

Комплект контрольно-оценочных средств предназначен для оценки результатов освоения профессионального модуля **ПМ 01 Разработка модулей программного обеспечения для компьютерных систем**

Перечень общих компетенций

Код	Наименование общих компетенций
ОК 1.	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 2.	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.
ОК 3	Планировать и реализовывать собственное профессиональное и личностное
ОК 4	Планировать и реализовывать собственное профессиональное и личностное
ОК 5	Планировать и реализовывать собственное профессиональное и личностное
ОК 6	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей
ОК 7	Содействовать сохранению окружающей среды, ресурсосбережению, 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	Использовать информационные технологии в профессиональной деятельности.
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языках

Перечень профессиональных компетенций

Код	Наименование видов деятельности и профессиональных компетенций
ВД 1	Разработка модулей программного обеспечения для компьютерных систем
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

Контроль и оценка результатов освоения ПМ осуществляется преподавателем в процессе текущего (рубежного) контроля, промежуточной аттестации в форме дифференцированного зачета, экзамена и экзамена квалификационного.

Код и наименование профессиональных и общих компетенций, формируемых в рамках модуля	Критерии оценки	Методы оценки
Раздел модуля 1. Разработка программных модулей		
ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием	Оценка «отлично» - техническое задание проанализировано, алгоритм разработан, соответствует техническому заданию и оформлен в соответствии со стандартами, пояснены его основные структуры. Дополнительно для квалификаций "Программист" указаны использованные стандарты в области документирования; выполнена оценка сложности алгоритма Оценка «хорошо» - алгоритм разработан, оформлен в соответствии со стандартами и соответствует заданию, пояснены его основные структуры. Дополнительно для квалификации "Программист": выполнена оценка сложности алгоритма Оценка «удовлетворительно» - алгоритм разработан и соответствует заданию.	Экзамен/зачет в форме собеседования: практическое задание по построению алгоритма в соответствии с техническим заданием, защита отчетов по практическим и лабораторным работам.
ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием	Оценка «отлично» - программный модуль разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно-ориентированного/структурного программирования и полностью соответствует техническому заданию, соблюдены и пояснены основные этапы разработки; документация на модуль оформлена и соответствует стандартам. Оценка «хорошо» - программный модуль разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно-ориентированного/структурного программирования и практически соответствует техническому заданию с незначительными отклонениями,	Экзамен/зачет в форме собеседования: практическое задание по разработке программного модуля в соответствии с техническим заданием Защита отчетов по практическим и лабораторным работам. Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики.

	пояснены основные этапы разработки; документация на модуль оформлена и соответствует стандартам. Оценка «удовлетворительно» - программный модуль: разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно-ориентированного/структурного программирования и соответствует техническому заданию; документация на модуль оформлена без существенных отклонений от стандартов.	
Раздел модуля 2. Поддержка и тестирование программных модулей		
ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств	Оценка «отлично» - выполнена отладка модуля (Дополнительно для квалификации "Программист": с использованием инструментария среды проектирования); с пояснением особенностей отладочных классов; сохранены и представлены результаты отладки. Оценка «хорошо» - выполнена отладка модуля (Дополнительно для квалификации "Программист": с использованием инструментария среды проектирования); сохранены и представлены результаты отладки. Оценка «удовлетворительно» - выполнена отладка модуля, пояснены ее результаты.	Экзамен/зачет в форме собеседования: практическое задание по выполнению отладки предложенного программного модуля. Защита отчетов по практическим и лабораторным работам. Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики.
ПК 1.4 Выполнять тестирование программных модулей	Оценка «отлично» - выполнено тестирование модуля, в том числе с помощью инструментальных средств, и оформлены результаты тестирования в соответствии со стандартами. Оценка «хорошо» - выполнено тестирование модуля, в том числе с помощью инструментальных средств, и оформлены результаты тестирования. Оценка «удовлетворительно» - выполнено тестирование модуля и оформлены результаты тестирования.	Экзамен/зачет в форме собеседования: практическое задание по выполнению заданных видов тестирования программного модуля. Защита отчетов по практическим и лабораторным работам. Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики.

К1.5Осуществлять рефакторинг и оптимизацию программного кода	Оценка «отлично» - определены качественные характеристики программного кода с помощью инструментальных средств; выявлены фрагменты некачественного кода; выполнен рефакторинг на уровнях переменных, функций, классов, алгоритмических структур; проведена оптимизация и подтверждено повышение качества программного кода. Оценка «хорошо» - определены качественные характеристики программного кода с помощью инструментальных средств; выявлены фрагменты некачественного кода; выполнен рефакторинг на нескольких уровнях; проведена оптимизация и выполнена оценка качества полученного программного кода. Оценка «удовлетворительно» - определены качественные характеристики программного кода частично с помощью инструментальных средств; выявлено несколько фрагментов некачественного кода; выполнен рефакторинг на нескольких уровнях; проведена оптимизация и выполнена оценка качества полученного программного кода.	Экзамен/зачет в форме собеседования: практическое задание по оценке качества кода предложенного программного модуля, поиску некачественного программного кода, его анализу, оптимизации методами рефакторинга. Защита отчетов по практическим и лабораторным работам. Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики.
---	--	---

Раздел модуля 3. Разработка мобильных приложений

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием	Оценка «отлично» - разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно- ориентированного/ структурного программирования и полностью соответствует техническому заданию, соблюдены и пояснены основные этапы разработки; документация на модуль оформлена и соответствует стандартам. Оценка «хорошо» - разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно- ориентированного/ структурного программирования и практически соответствует техническому заданию с незначительными отклонениями, пояснены основные этапы разработки; документация на модуль оформлена и соответствует стандартам.	Экзамен/зачет в форме собеседования: практическое задание по разработке программного модуля в соответствии с техническим заданием Защита отчетов по практическим и лабораторным работам Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики
--	---	--

	Оценка «удовлетворительно» - тестовый модуль разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно-ориентированного/структурного программирования и соответствует техническому заданию; документация на модуль оформлена без существенных отклонений от стандартов	
ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.	Оценка «отлично» - разработан модуль для заданного мобильного устройства с соблюдением основных этапов разработки на одном из современных языков программирования; при проверке работоспособности модуля на устройстве или эмуляторе установлено его соответствие спецификации. Оценка «хорошо» - разработан модуль для заданного мобильного устройства с учетом основных этапов разработки на одном из современных языков программирования; при проверке работоспособности модуля на устройстве или эмуляторе установлено соответствие выполняемых функций спецификации с незначительными отклонениями. Оценка «удовлетворительно» - разработан модуль для заданного мобильного устройства на одном из современных языков программирования; при проверке работоспособности модуля на устройстве или эмуляторе установлено соответствие основных выполняемых функций спецификации.	Экзамен/зачет в форме собеседования: практическое задание по созданию модуля для заданного мобильного устройства на основе спецификации Защита отчетов по практическим и лабораторным работам Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики
Раздел модуля 4. Системное программирование		
ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием	Оценка «отлично» - разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно-ориентированного/структурного программирования и полностью соответствует техническому заданию, соблюдены и пояснены основные этапы разработки; документация на модуль оформлена и соответствует стандартам. Оценка «хорошо» - разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) ме-	Экзамен/зачет в форме собеседования: практическое задание по разработке программного модуля в соответствии с техническим заданием. Защита отчетов по практическим и лабораторным работам. Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики.

	тодами объектно- ориентированного/ структурного программирования и практически соответствует техническому заданию с незначительными отклонениями, пояснены основные этапы разработки; документация на модуль оформлена и соответствует стандартам. Оценка «удовлетворительно» - разработан по имеющемуся алгоритму в среде разработки (Дополнительно для квалификации "Программист" на указанном языке программирования) методами объектно- ориентированного структурного программирования и соответствует техническому заданию; документация на модуль оформлена без существенных отклонений от стандартов.	
ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств	Оценка «отлично» - выполнена отладка модуля (Дополнительно для квалификации "Программист" с пояснением особенностей отладочных классов; сохранены и представлены результаты отладки. Оценка «хорошо» - выполнена отладка модуля (Дополнительно для квалификации "Программист" с использованием инструментария среды проектирования); сохранены и представлены результаты отладки. Оценка «удовлетворительно» - выполнена отладка модуля, пояснены ее результаты.	Экзамен/зачет в форме собеседования: практическое задание по выполнению отладки предложенного программного модуля. Защита отчетов по практическим и лабораторным работам. Интерпретация результатов наблюдений за деятельностью обучающегося в процессе практики
ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.	- обоснованность постановки цели, выбора и применения методов и способов решения профессиональных задач; - адекватная оценка и самооценка эффективности и качества выполнения профессиональных задач.	Экспертное наблюдение за выполнением работ.
ОП 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.	- использование различных источников, включая электронные ресурсы, медиаресурсы, Интернет-ресурсы, периодические издания по специальности для решения профессиональных задач.	
ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.	- демонстрация ответственности за принятые решения. - обоснованность самоанализа и коррекция результатов собственной работы.	

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.	- взаимодействовать с обучающимися, преподавателями и мастерами в ходе обучения, с руководителями учебной и производственной практик. - обоснованность анализа работы членов команды (подчиненных).
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.	Демонстрировать грамотность устной и письменной речи, - ясность формулирования и изложения мыслей.
ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.	- соблюдение норм поведения во время учебных занятий и прохождения учебной и производственной практик.
ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.	- эффективное выполнение правил ТБ во время учебных занятий, при прохождении учебной и производственной практик; - демонстрация знаний и использование ресурсосберегающих технологий в профессиональной деятельности.
ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.	- эффективность использовать средств физической культуры для сохранения и укрепления здоровья при выполнении профессиональной деятельности.
ОК 09. Использовать информационные технологии в профессиональной деятельности.	- эффективность использования информационно-коммуникационных технологий в профессиональной деятельности согласно формируемым умениям и получаемому практическому опыту.
ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.	- эффективность использования в профессиональной деятельности необходимой технической документации, в том числе на английском языке.

В результате оценки осуществляется проверка следующих объектов:

1.2. Формы контроля и оценивания элементов профессионального модуля

	Форма контроля и оценивания
--	------------------------------------

Элемент модуля, профессиональный модуль	Промежуточная аттестация	Текущий контроль
МДК.01.01 Разработка программных модулей	Экзамен	- выполнение лабораторно-практических заданий; -текущие контрольные работы; -Тестирование (зачёт); -ситуационные задачи; -самостоятельная работа;
МДК.01.02 Поддержка и тестирование программных модулей	Экзамен	
МДК.01.03 Разработка мобильных приложений	Экзамен	
МДК.01.04 Системное программирование	Экзамен	
УП.01. Учебная практика по разработке модулей программного обеспечения для компьютерных систем	Дифференцированный зачет	Выполнение лабораторных работ
ПП.01. Производственная практика по разработке модулей программного обеспечения для компьютерных систем	Дифференцированный зачет	
ПМ.01. Разработка модулей программного обеспечения для компьютерных систем	Экзамен (квалификационный)	Положительная аттестация по МДК.01.01, МДК.01.02, МДК.01.03, МДК.01.04, учебной и производственной практике

К экзамену по МДК допускаются студенты, полностью выполнившие все практические задания, и, имеющие положительные оценки по результатам текущего контроля.

К экзамену (квалификационному) по профессиональному модулю допускаются студенты, успешно прошедшие промежуточную аттестацию по МДК.01.01, МДК.01.02, МДК.01.03, МДК.01.04, учебной и производственной практике в рамках данного профессионального модуля.

Итогом экзамена является однозначное решение: «вид профессиональной деятельности освоен/ не освоен».

Форма проведения экзамена: решение профессиональных заданий.

II. РЕЗУЛЬТАТЫ ОСВОЕНИЯ МОДУЛЯ, ПОДЛЕЖАЩИЕ ПРОВЕРКЕ НА ЭКЗАМЕНЕ (КВАЛИФИКАЦИОННОМ)

2.1 Профессиональные и общие компетенции

В результате аттестации по профессиональному модулю осуществляется комплексная проверка следующих профессиональных и общих компетенций:

Профессиональные и общие компетенции, которые возможно сгруппировать для проверки

Показатели оценки результата

Объекты оценивания	Показатели оценки результата по каждому объекту оценивания	Критерии признака, на основе которого производится оценка по показателю
ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием. ОК 1 Выбирать способы решения задач профессиональной	Составление формализованных описаний решений поставленных задач в соответствии с требованиями технического задания или других принятых в организации нормативных документов	– Использование методов и приемов формализации задач – Использование методов и приемов алгоритмизации поставленных задач

деятельности, применительно к различным контекстам. ОК 4 Планировать и реализовывать собственное профессиональное и личностное развитие. ОК5	Разработка алгоритмов решения поставленных задач в соответствии с требованиями технического задания или других принятых в организации нормативных документов	– Использование программных продуктов для графического отображения алгоритмов – Применение стандартных алгоритмов в соответствующих областях
ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием. ОК 1 Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. ОК 4 Планировать и реализовывать собственное профессиональное и личностное развитие. ОК 8 Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности	Создание программного кода в соответствии с техническим заданием (готовыми спецификациями) Оптимизация программного кода с использованием специализированных программных средств	– Применение выбранных языков программирования для написания программного кода – Использование выбранной среды программирования и средства системы управления базами данных – Использование возможности имеющейся технической и/или программной архитектуры
ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств. ОК 3 Планировать и реализовывать собственное профессиональное и личностное развитие. ОК 7 Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.	Анализ и проверка исходного программного кода Отладка программного кода на уровне программных модулей Отладка программного кода на уровне межмодульных взаимодействий и взаимодействий с окружением	– Выявление ошибок в программном коде – Применение методов и приемов отладки программного кода – Интерпретирование сообщения об ошибках, предупреждения, записи технологических журналов – Применение современных компиляторов, отладчиков и оптимизаторов программного кода
ПК 1.4 Выполнять тестирование программных модулей. ОК 4 Планировать и реализовывать собственное профессиональное и личностное развитие. ОК 6 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное	Подготовка тестовых наборов данных в соответствии с выбранной методикой	– Разработка и оформление контрольных примеров для проверки работоспособности программного обеспечения – Разработка процедур генерации тестовых наборов данных с

поведение на основе традиционных общечеловеческих ценностей		заданными характеристиками – Подготовка наборов данных, используемых в процессе проверки работоспособности программного обеспечения
ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода ОК 7 Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. ОК10 Пользоваться профессиональной документацией на государственном и иностранном языках	Работа с системой контроля версий	– Регистрация изменений исходного текста программного кода в системе контроля версий – Слияние, разделение и сравнение исходных текстов программного кода – Сохранение сделанных изменений программного кода в соответствии с регламентом контроля версий
ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ ОК 2 Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности. ОК 9 Использовать информационные технологии в профессиональной деятельности. ОК10 Пользоваться профессиональной документацией на государственном и иностранном языках	Разработка кода программного модуля для мобильных платформ	– Осуществление разработки кода программного модуля на языках низкого уровня и высокого уровней в том числе для мобильных платформ. – Владение первичным опытом разработки мобильных приложений

III. Оценка освоения теоретического курса профессионального модуля

3.1. Типовые задания для оценки освоения МДК 01.01 Разработка программных модулей

Практикум по решению задач

Задача 1.Злоумышленник поменял местами действия в алгоритме вычисления среднего арифметического из квадратного корня трёх чисел:

Присвоить а значение $(a^2+b^2+c^2)/3$.

Вести а,в,с

Сообщить “Среднее арифметическое квадратов равно”

Сообщить а.

Восстановите правильный порядок действий.

Задача 2. Исправьте следующий алгоритм решения уравнения $(x-2)(x+2)=0$:

Присвоить x значение $+2$.

Сообщить “Корни уравнения равны”.

Сообщить первое значение x .

Сообщить второе значение x .

Задача 3. Автомобиль проехал три участка пути разной длины с разными скоростями. Составьте алгоритм нахождения средней скорости автомобиля.

Задача 4. Запишите в виде алгоритмов правила определения знака:

А) произведения двух действительных чисел;

Б) суммы двух действительных чисел.

6. В записи алгоритма вычисления значения выражения $(x^2 - 5x + 5) / (x^6 - 4x^2 + 3)$

Злоумышленник одно действие поставил не на свое место. Вот как стал выглядеть алгоритм:

ввести x

если $x^6 - 4x^2 + 3 = 0$, то:

сообщить “При таком x значение выражения не определено”.

иначе:

присвоить y значение $(x^2 - 5x + 5) / (x^6 - 4x^2 + 3)$.

конец ветвления.

сообщить y .

Верните действие на свое место.

Задача 5. Вывести на экран 3 строки текста:

Сказка Репка; 2) Посадил дед Репку); 3) Выросла Репка большая-пребольшая.

Задача 6. Вычислить значения корней квадратного уравнения:

$$x^2 + px + q = 0$$

Электронный тест №1 в тестовой оболочке Moodle

1. Алгоритм — это:

а) правила выполнения определенных действий;

б) ориентированный граф, указывающий порядок исполнения некоторого набора команд;

в) понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленных целей;

г) набор команд для компьютера;

д) протокол вычислительной сети.

2. Укажите наиболее полный перечень способов записи алгоритмов:

а) словесный, графический, псевдокод, программный;

б) словесный;

в) графический, программный;

г) словесный, программный;

д) псевдокод.

3. Суть такого свойства алгоритма как результативность заключается в том, что:

а) алгоритм должен иметь дискретную структуру (должен быть разбит на последовательность отдельных шагов);

б) записывая алгоритм для конкретного исполнителя, можно использовать лишь те команды, что входят в систему его команд;

в) алгоритм должен обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа;

- г) при точном исполнении всех команд алгоритма процесс должен прекратиться за конечное число шагов, приведя к определенному результату;
- д) исполнитель алгоритма не должен принимать решения, не предусмотренные составителем алгоритма.

4. Суть такого свойства алгоритма как массовость заключается в том, что:

- а) алгоритм должен иметь дискретную структуру (должен быть разбит на последовательность отдельных шагов);
- б) записывая алгоритм для конкретного исполнителя, можно использовать лишь те команды, что входят в систему его команд;
- в) алгоритм должен обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа;
- г) при точном исполнении всех команд алгоритма процесс должен прекратиться за конечное число шагов, приведя к определенному результату;
- д) исполнитель алгоритма не должен принимать решения, не предусмотренные составителем алгоритма.

5. Суть такого свойства алгоритма как дискретность заключается в том, что:

- а) алгоритм должен иметь дискретную структуру (должен быть разбит на последовательность отдельных шагов);
- б) записывая алгоритм для конкретного исполнителя, можно использовать лишь те команды, что входят в систему его команд;
- в) алгоритм должен обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа;
- г) при точном исполнении всех команд алгоритма процесс должен прекратиться за конечное число шагов, приведя к определенному результату;
- д) исполнитель алгоритма не должен принимать решения, не предусмотренные составителем алгоритма.

6. Суть такого свойства алгоритма как понятность заключается в том, что:

- а) алгоритм должен иметь дискретную структуру (должен быть разбит на последовательность отдельных шагов);
- б) записывая алгоритм для конкретного исполнителя, можно использовать лишь те команды, что входят в систему его команд;
- в) алгоритм должен обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа;
- г) при точном исполнении всех команд алгоритма процесс должен прекратиться за конечное число шагов, приведя к определенному результату;
- д) исполнитель алгоритма не должен принимать решения, не предусмотренные составителем алгоритма.

7. Суть такого свойства алгоритма как детерминированность заключается в том, что:

- а) алгоритм должен иметь дискретную структуру (должен быть разбит на последовательность отдельных шагов);
- б) записывая алгоритм для конкретного исполнителя можно использовать лишь те команды, что входят в систему его команд;
- в) алгоритм должен обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа;
- г) при точном исполнении всех команд алгоритма процесс должен прекратиться за конечное число шагов, приведя к определенному результату;
- д) исполнитель алгоритма не должен принимать решения, не предусмотренные составителем алгоритма.

8. Алгоритм называется линейным:

- а) если он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
- б) если ход его выполнения зависит от истинности тех или иных условий;
- в) если его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий;
- г) если он представим в табличной форме;
- д) если он включает в себя вспомогательный алгоритм.

9. Алгоритм называется циклическим:

- а) если он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
- б) если ход его выполнения зависит от истинности тех или иных условий;
- в) если его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий;
- г) если он представим в табличной форме;
- д) если он включает в себя вспомогательный алгоритм.

10. Алгоритм включает в себя ветвление, если:

- а) если он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
- б) если ход его выполнения зависит от истинности тех или иных условий;
- в) если его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий;
- г) если он представим в табличной форме;
- д) если он включает в себя вспомогательный алгоритм.

11. Системы программирования:

- а) обеспечивают непосредственное решение пользовательских задач;
- б) позволяют создавать новые программы на языках программирования;
- в) обеспечивают работу всех аппаратных устройств компьютера и доступ пользователя к ним;
- г) представляют собой совокупность программ, используемых для различных операций с документами;
- д) предназначены для уничтожения компьютерных вирусов и зараженных ими файлов.

12. Характерным признаком линейной программы является:

- а) строго последовательное выполнение операторов в порядке их записи;
- б) наличие в каждой программной строке только одного операторов;
- в) использование в ней исключительно операторов присваивания;
- г) присутствие в ней операторов условного и безусловного перехода;
- д) присутствие в ней операторов цикла.

13. Подпрограммой называют:

- а) часть программы, содержащую неоднократно выполняемые команды;
- б) независимый программный модуль;
- в) произвольный фрагмент программы;
- г) набор операторов, следующий в программе за оператором GOSUB;
- д) часть программы, служащей для решения некоторой вспомогательной задачи.

14. Важнейший принцип структурного программирования базируется на утверждении:

- а) любой алгоритм имеет дискретную структуру;
- б) алгоритм любой сложности можно построить с помощью следующих базовых структур: линейной, ветвящейся, циклической;

- в) современный компьютер — это единство аппаратных средств и программного обеспечения;
- г) сущность формализации решаемой задачи заключается в составлении алгоритма;
- д) в качестве обязательного этапа создания программы выступает ее тестирование и отладка.

15. Переменная в программировании полностью характеризуется:

- а) именем;
- б) именем, значением и типом;
- в) именем и типом;
- г) именем и значением;
- д) значением.

Практическая работа №1,2

Тема: Работа с символьными строками

Цель работы: получение практических навыков в работе с массивами и указателями языка С, обеспечение функциональной модульности

Темы для самостоятельного изучения

- Указатели в языке С.
- Представление строк.
- Функции и передача параметров.

Постановка задачи

По индивидуальному заданию создать функцию для обработки символьных строк. За образец брать библиотечные функции обработки строк языка С, но не применять их в своей функции. Предусмотреть обработку ошибок в задании параметров и особые случаи. Разработать два варианта заданной функции - используя традиционную обработку массивов и используя адресную арифметику.

Пример решения задачи

Индивидуальное задание:

Функция - *substr(s,n,l)*. Назначение - выделение из строки *s* подстроки, начиная с позиции *n*, длиной *l*.

Описание метода решения

1. Символьная строка в языке С представляется в памяти как массив символов, в конце которого находится байт с кодом 0 - признак конца строки. Строку, как и любой другой массив можно обрабатывать либо традиционным методом - как массив, с использованием операции индексации, либо через указатели, с использованием операций адресной арифметики. При работе со строкой как с массивом нужно иметь в виду, что длина строки заранее неизвестна, так что циклы должны быть организованы не со счетчиком, а до появления признака конца строки.

2. Функция должна реализовывать поставленную задачу - и ничего более. Это означает, что функцию можно будет, например, перенести без изменений в любую другую программу, если спецификации функции удовлетворяют условиям задачи. Это также означает, что при ошибочном задании параметров или при каких-то особых случаях в их значениях функция не должна аварийно завершать программу или выводить какие-то сообщения на экран, но должна возвращать какое-то прогнозируемое значение, по которому та функция, которая вызвала нашу, может сделать вывод об ошибке или об особом случае.

Определим состав параметров функции:

`int substr (src, dest, num, len);`

где	src -	строка, с которой выбираются символы;
	dest -	строка, в которую записываются символы;
	num -	номер первого символа в строке src, с которого начинается подстрока (нумерация символов ведется с 0);
	len -	длина выходной строки.

Возможные возвращаемые значения функции установим: 1 (задание параметров правильное) и 0 (задание не правильное). Эти значения при обращениях к функции можно будет интерпретировать как "истина" или "ложь".

Обозначим через `Lsrc` длину строки `src`. Тогда возможны такие варианты при задании параметров:

- `num+len <= Lsrc` - полностью правильное задание;
- `num+len > Lsrc`; `num < Lsrc` - правильное задание, но длина выходной строки будет меньше, чем `len`;
- `num >= Lsrc` - неправильное задание, выходная строка будет пустой;
- `num < 0` или `len <= 0` - неправильное задание, выходная строка будет пустой.

Заметим, что интерпретация конфигурации параметров как правильная / неправильная и выбор реакции на неправильное задание - дело исполнителя. Но исполнитель должен строго выполнять принятые правила. Возможен также случай, когда выходная строка выйдет большей длины, чем для нее отведено места в памяти. Однако, поскольку нашей функции неизвестен размер памяти, отведенный для строки, функция не может распознать и обработать этот случай - так же ведут себя и библиотечные функции языка C.

Описание логической структуры

1. Программа состоит из одного программного модуля - файл `LAB1.C`. В состав модуля входят три функции - `main`, `substr_mas` и `subs_ptr`. Общих переменных в программе нет. Макроконстантой `N` определена максимальная длина строки - 80.

2. Функция `main` является главной функцией программы, она предназначена для ввода исходных данных, вызова других функций и вывода результатов. В функции определены переменные:

- `ss` и `dd` - входная и выходная строки соответственно;
- `n` - номер символа, с которого должна начинаться выходная строка;
- `l` - длина выходной строки.

Функция запрашивает и вводит значение входной строки, номера символа и длины. Далее функция вызывает функцию `substr_mas`, передавая ей как параметры введенные значения. Если функция `substr_mas` возвращает 1, выводится на экран входная и выходная строки, если 0 - выводится сообщение об ошибке и входная строка. Потом входная строка делается пустой и то же самое выполняется для функции `substr_ptr`.

3. Функция `substr_mas` выполняет поставленное задание методом массивов. Ее параметры: - `src` и `dest` - входная и выходная строки соответственно, представленные в виде массивов неопределенного размера; `num` и `len`. Внутренние переменные `i` и `j` используются как индексы в массивах.

Функция проверяет значения параметров в соответствии со случаем 4, если условия этого случая обнаружены, в первый элемент массива `dest` записывается признак конца строки и функция возвращает 0.

Если случай 4 не выявлен, функция просматривает `num` первых символов входной строки. Если при этом будет найден признак конца строки, это - случай 3, при этом в первый элемент массива `dest` записывается признак конца строки и функция возвращает 0.

Если признак конца в первых `num` символах не найден, выполняется цикл, в котором индекс входного массива начинает меняться от 1, а индекс выходного - от 0. В каждой итерации этого цикла один элемент входного массива пересылается в выходной. Если пересланный элемент является признаком конца строки (случай 2), то функция немедленно заканчивается, возвращая 1. Если в цикле не встретится конец строки, цикл завершится после `len` итераций. В этом случае в конец выходной строки записывается признак конца и Функция возвращает 1.

4. Функция `substr_ptr` выполняет поставленное задание методом указателей. Ее параметры: - `src` и `dest` - входная и выходная строки соответственно, представленные в виде указателей на начала строк; `num` и `len`.

Функция проверяет значения параметров в соответствии со случаем 4, если условия этого случая выявлены, по адресу, который задает `dest`, записывается признак конца строки и функция возвращает 0, эти действия выполняются одним оператором.

Если случай 4 не обнаружен, функция пропускает num первых символов входной строки. Это сделано циклом while, условием выхода из которого является уменьшение счетчика num до 0 или появление признака конца входной строки. Важно четко представлять порядок операций, которые выполняются в этом цикле:

- выбирается счетчик num;
- счетчик num уменьшается на 1;
- если выбранное значение счетчика было 0 - цикл завершается;
- если выбранное значение было не 0 - выбирается символ, на который указывает указатель src;
- указатель src увеличивается на 1;
- если выбранное значение символа было 0, т.е., признак конца строки, цикл завершается, иначе - повторяется.

После выхода из цикла проверяется значение счетчика num: если оно не 0, это означает, что выход из цикла произошел по признаку конца строки (случай 3), по адресу, который задает dest, записывается признак конца строки и функция возвращает 0.

Если признак конца не найден, выполняется цикл, подобный первому циклу while, но по счетчику len. В каждой итерации этого цикла символ, на который показывает src переписывается по адресу, задаваемому dest, после чего оба указателя увеличиваются на 1. Цикл закончится, когда будет переписано len символов или встретится признак конца строки. В любом варианте завершения цикла по текущему адресу, который содержится в указателе dest, записывается признак конца строки и функция завершается, возвращая 1.

Данные для тестирования

Тестирование должно обеспечить проверку работоспособности функций для всех вариантов входных данных. Входные данные, на которых должно проводиться тестирование, сведены в таблицу:

вариант	1	2	3	4
src	012345 012345 012345	012345 012345 012345	012345	012345 012345 012345
num	2 0 0	5 2 0	8	-1 5 5
len	2 1 6	3 6 7	2	2 0 -1
dest	23 0 012345	5 2345 012345	пусто	пусто пусто пусто

Текст программы

```
#include <stdio.h>
#include <conio.h>
#define N 80
int substr_mas(char src[N],char dest[N],int num,int len){
int i, j;
/* проверка случая 4 */
if ( (num<0)||len<=0 ) {
dest[0]=0; return 0;
}
/* выход на num-ый символ */
for (i=0; i<=num; i++)
/* проверка случая 3 */
if ( src[i]=='\0') {
dest[0]=0; return 0;
```

```

}
/* перезапись символов */
for (i--, j=0; j<len; j++, i++) {
dest[j]=src[i];
/* проверка случая 2 */
if ( dest[j]=='\0') return 1;
}
/* запись признака конца в выходную строку */
dest[j]='\0';
return 1; }
int substr_ptr(char *src, char *dest, int num, int len) {
/* проверка случая 4 */
if ( (num<0)|| (len<=0) ) return dest[0]=0;
/* выход на num-ый символ или на конец строки */
while ( num-- && *src++ );
/* проверка случая 3 */
if ( !num ) return dest[0]=0;
/* перезапись символов */
while ( len-- && *src ) *dest++=*src++;
/* запись признака конца в выходную строку */
*dest=0;
return 1; }
Int main()
{
char ss[N], dd[N];
int n, l;
int C;
/* clrscr();*/
printf("Vvodite stroky:\n");
gets(ss);
printf("Nachalo=");
scanf("%d",&n);
printf("Dlina=");
scanf("%d",&l);
printf("Massiv:\n");
if (substr_mas(ss,dd,n,l)) printf(">>%s<<\n>>%s<<\n",ss,dd);
else printf("Oshibka! >>%s<<\n",dd);
dd[0]='\0';
printf("Adresnaya arifmetika:\n");
if (substr_ptr(ss,dd,n,l)) printf(">>%s<<\n>>%s<<\n",ss,dd);
else printf("Oshibka! >>%s<<\n",dd);
C=getch(); }

```

Практическая работа №3

Тема: Проверка оборудования

Цель работы: Получение практических навыков в определении конфигурации и основных характеристик компьютера.

Постановка задачи

Для компьютера на своем рабочем месте определить:

- тип компьютера;
- конфигурацию оборудования;
- объем оперативной памяти;
- наличие и объем расширенной памяти;
- наличие дополнительных ПЗУ;
- версию операционной системы.

Пример решения задачи

Структура данных программы

Программа использует, так называемый, список оборудования — 2-байтное слово в области данных BIOS по адресу 0040:0010. Назначение разрядов списка оборудования такое:

- `u0` установлен в 1, если есть НГМД
- `u1` установлен в 1, если есть сопроцессор
- `u2,3` число 16-Кбайтных блоков ОЗУ на системной плате
- `u4,5` код видеоадаптера: 11 — MDA, 10 — CGA, 80 — колонки, 01 — CGA, 40 — колонки, 00 — другой
- `u6,7` число НГМД-1 (если в разряде 0 единица)
- `u8 9`, если есть канал ПДП
- `u9,10,11` число последовательных портов RS-232
- `u12 1`, если есть джойстик
- `u13 1`, если есть последовательный принтер
- `u14,15` число параллельных принтеров

Структура программы

Программа состоит только из основной функции **main()**. Выделения фрагментов программы в отдельные процедуры не требуется, потому что нет таких операций, которые во время работы программы выполняются многократно.

Описание переменных

Переменные, применяемые в программе:

- **type_PC** — байт типа компьютера, записанный в ПЗУ BIOS по адресу FF00:0FFE;
- **a, b** — переменные для определения объема extended-памяти ПЭВМ, **a** — младший байт, **b** — старший байт;
- **konf_b** — 2-байтное слово из области данных BIOS, которое содержит список оборудования;
- **type** — массив символьных строк, представляющих типы компьютера;
- **typ1A** — массив байт, содержащий коды типов дисплеев;
- **types1A[]** — массив строк, содержащий названия типов дисплеев;
- **j** — вспомогательная переменная, которая используется для идентификации типа дисплея;
- **seg** — сегмент, в котором размещено дополнительное ПЗУ;
- **mark** — маркер ПЗУ;
- **bufVGA[64]** — буфер данных VGA, из которого (при наличии VGA) мы выбираем объем видеопамяти;
- **rr** и **sr** — переменные, которые используются для задания значения регистров общего назначения и сегментных регистров, соответственно, при вызове прерывания.

Описание алгоритма программы

Алгоритм основной программы может быть разбито на 5 частей.

Часть 1 предназначена для определения типа компьютера. Для этого читаем байт, записанный в ПЗУ BIOS по адресу FF00:0FFE. В зависимости от значения этого байта сделаем вывод о типе ПЭВМ. Так, например, компьютеру типа AT соответствует код 0xFC.

Часть 2 предназначена для определения конфигурации ПЭВМ. Для этого прочитаем из области данных BIOS список оборудования. Для определения количества дисководов (если бит 0 установлен в 1) необходимо выделить биты 6 и 7 (маска 00C0h) и сдвинуть их вправо на 6 разрядов, а потом добавить 1.

Для определения количества 16-Кбайтных блоков ОЗУ на системной плате необходимо выделить биты 2 и 3 с помощью маски 000Ch, сдвинуть вправо на 2 разряда и добавить 1.

Для определения количества последовательных портов RS-232 выделить с помощью маски 0Eh биты 9-11 и сдвинуть вправо на 9 разрядов.

Для определения наличия математического сопроцессора — проверить установку бита 1 маской 0002h.

Для определения наличия джойстика — бит 12 с помощью маски 1000h.

Определить количество параллельных принтеров можно, выделив биты 14 и 15 маской C000h и сдвинув их вправо на 14 разрядов.

Поскольку список оборудования содержит недостаточно информации про дисплейный адаптер, то для уточнения типа адаптера выполним дополнительные действия.

Видеоадаптер обслуживается прерыванием BIOS 10h. Для новых типов адаптеров список его функций расширяется. Эти новые функции и используются для определения типа адаптера.

Функция 1Ah доступна только при наличии расширения BIOS, ориентированного на обслуживание VGA. В этом случае функция возвращает в регистре AL код 1Ah — свою «визитную карточку», а в BL — код активного видеоадаптера. В случае, если функция 1Ah поддерживается, обратимся еще к функции 1Bh — последняя заполняет 70-байтный блок информации про состояние, из которого мы выбираемо объем видеопамати.

Если 1Ah не поддерживается, это означает, что VGA у нас нет, в этом случае можно обратиться к функции 12h — получение информации про EGA. При наличии расширения, ориентированного на EGA, эта функция изменяет содержимое BL (перед обращением он должен быть 10h) на 0 (цветной режим) или на 1 (монохромный режим) а в BH возвращает объем видеопамати.

Если же ни 1Ah, ни 12 не поддерживаются, то список оборудования BIOS содержит достаточную информацию про видеоадаптер и, выделив биты 4, 5 мы можем сделать окончательный вывод про тип адаптера, который у нас есть.

В 3 части программы определим объем оперативной памяти, наличие и объем extended-памяти. Объем оперативной памяти для AT может быть прочитан из регистров 15h (младший байт) и 16h (старший байт) CMOS-памяти или из области памяти BIOS по адресу 0040:0013 (2-байтное слово). Кроме того, в ПЭВМ может быть еще и дополнительная (expanded) память свыше 1 Мбайта. Ее объем можно получить из регистров 17h (младший байт) и 18h (старший байт) CMOS-памяти. Для чтения регистра CMOS-памяти необходимо выдать в порт 70h байт номера регистра, а потом из порта 71h прочитать байт содержимого этого регистра.

В 4 части программы определим наличие и объем дополнительных ПЗУ. В адресном пространстве от C000:0000 по F600:0000 размещаются расширения ПЗУ (эта память не обязательно присутствует в ПЭВМ). Для определения наличия дополнительного ПЗУ будем читать первое слово из каждые 2 Кбайт, начиная с адреса C000:0000 в поисках маркера расширения ПЗУ: 55AAh. Если такой маркер найден, то следующий байт содержит длину модуля ПЗУ.

В 5 части программы определим версию DOS, установленную на ПЭВМ. Для этого воспользуемся функцией DOS 30h, которая возвращает в регистре AL старшее число номера версии, а в регистре AH — младшее число.

Текст программы

```
/*----"Проверка состава оборудования"-----*/
/* Подключение стандартных заголовков */
#include<dos.h>
#include <conio.h>
#include <stdio.h>
/*-----*/
void main()
```

```

{
unsigned char type_PC, /* Типкомпьютера */
a,b; /* Переменные для определения */
/* характеристик памяти ПЭВМ */
unsigned int konf_b; /* Байт конфигурации из BIOS */
char *type[]={"АТ","PCjr","XT","IBM PC","unknown"};
unsigned char typ1A[]={0,1,2,4,5,6,7,8,10,11,12,0xff};
char *types1A[]={"немадисплею","MDA, моно","CGA, цв.",
"EGA, цв.", "EGA, моно", "PGA, цв.",
"VGA, моно, анал.", "VGA, кол., анал.",
"MCGA, кол., цифр.", "MCGA, моно, анал.",
"MCGA, кол., анал.", "неизвестный тип",
"непредусмотренный код"};
unsigned int j; /* Вспомогательная переменная */
unsigned int seg; /* СегментПЗУ */
unsigned int mark=0xAA55; /* МаркерПЗУ */
unsigned char bufVGA[64]; /* Буферданных VGA */
union REGS rr;
struct SREGS sr;
textbackground(0);
clrscr();
textattr(0x0a);
printf("Лабораторная работа N5");
printf("\nПроверка состава оборудования");
/* Определения типа компьютера */
type_PC=peekb(0xF000,0xFFFE);
if( (type_PC==0xFC)>4)
type_PC=4;
textattr(0x0b);
printf("\nТипкомпьютера: ");
textattr(0x0f);
printf("%s\n\r",type[type_PC]);
/* Конфигурация */
konf_b=peek(0x40,0x10); /* Чтение байта оборудования */
/* изпамятиBIOS */
textattr(0x0b);
printf("Конфигурация:\n\r");
/* Количество дисководов */
textattr(0x0e);
printf(" Дискководов ГМД: ");
textattr(0x0f);
if(konf_b&0x0001)
printf("%d\n\r",((konf_b&0x00C0)>>6)+1);
else
printf("нет\n\r");
textattr(0x0e);
printf(" Математич. сопроцессор: ");
textattr(0x0f);
if(konf_b&0x0002)
printf("есть\n\r");
else
printf("нет\n\r");
textattr(0x0e);
printf(" Типдисплейногоадаптера: ");

```

```

textattr(0x0f);
/* Определение активного адаптера */
/* Предположим наличие VGA */
rr.h.ah=0x1a;
rr.h.al=0;
int86(0x10,&rr,&rr);
if(rr.h.al==0x1a) /* Поддерживается функция 1Ah */
{
    /* прерывания 10h */
    for(j=0;j<12;j++)
    if(rr.h.bl==typ1A[j])
    break;
    cprintf("%s",types1A[j]);

    if(j>0 && j<12)
    {
        rr.h.ah=0x1b;
        rr.x.bx=0;
        sr.es=FP_SEG(bufVGA);
        rr.x.di=FP_OFF(bufVGA);
        int86x(0x10,&rr,&rr,&sr);
        cprintf(", %d Кбайт\n\r",((int)bufVGA[49]+1)*64);
    }
    else
    cprintf("\n\r");
}
else
{
    /* Предположим наличие EGA */
    rr.h.ah=0x12;
    rr.h.bl=0x10;
    int86(0x10,&rr,&rr);
    if(rr.h.bl!=0x10) /* Поддерживается функция 12h */
    {
        /* прерывания 10h */
        cprintf("EGA");
        if(rr.h.bh)
        cprintf(" моно");
    }
    else
    cprintf(" кол.");
    cprintf(", %d Кбайт\n\r",((int)rr.h.bl+1)*64);
}
else
{
    /* CGA или MDA */
    switch(konf_b&0x0030)
    {
        case 0: cprintf("EGA/VGA\n\r");break;
        case 0x10: cprintf("CGA,40\n\r");break;
        case 0x20: cprintf("CGA,80\n\r");break;
        case 0x30: cprintf("MDA");break;
    }
}
}
/* Блоки ОЗУ на системной плате */
textattr(0x0e);

```

```

    printf("\n\r Первичный блок памяти: ");
    textattr(0x0f);
    switch (konf_b&0x000C)
    {
    case 0:printf("16 Кбайт\n\r");break;
    case 4:printf("32 Кбайт\n\r");break;
    case 8:printf("48 Кбайт\n\r");break;
    case 12:printf("64 Кбайтилибольше\n\r");break;
    }
    /* Количество последовательных портов RS-232 */
    textattr(0x0e);
    printf(" Портов RS232:    ");
    textattr(0x0f);
    printf("%d\n\r",(konf_b&0x0E00)>>9);
    /* Наличиеджойстика */
    textattr(0x0e);
    printf(" Джойстик:      ");
    textattr(0x0f);
    if(konf_b&0x1000 )
    printf("есть\n\r");
    else
    printf("нет\n\r");
    /* Количество параллельных принтеров */
    textattr(0x0e);
    printf(" Принтеров:      ");
    textattr(0x0f);
    printf("%d\n\n\r",(konf_b&0xC000)>>14);
    /* Объем оперативной памяти */
    textattr(0x0e);
    printf("Объем оперативной памяти: ");
    textattr(0x0f);
    printf("%d Кбайт\n\r",peek(0x40,0x13));
    textattr(0x0e);
    /* Наличие и объем extended-памяти */
    outportb(0x70,0x17);
    a=inport(0x71);
    outportb(0x70,0x18);
    b=inport(0x71);
    printf("Объем extended-памяти: ");
    textattr(0x0f);
    printf("%d Кбайт\n\n\r",(b<<8)|a);
    /* Наличие дополнительных ПЗУ */
    for( seg=0xC000;seg<0xFFB0;seg+=0x40)
    /* Просмотр памяти от C000:0 с шагом 2 К */
    if(peek(seg,0)==mark) /* Маркернайден */
    {
    textattr(0x0a);
    printf("АдресПЗУ =");
    textattr(0x0f);
    printf(" %04x",seg);
    textattr(0x0a);
    printf(". Длина модуля = ");
    textattr(0x0f);
    printf("%d",512*peekb(seg,2));

```

```

textattr(0x0a);
cprintf(" байт\n\r",peekb(seg,2));
}
/* Определение версии операционной системы */
rr.h.ah=0x30;
intdos(&rr,&rr);
textattr(0x0c);
cprintf("\n\rВерсия MS-DOS ");
textattr(0x0f);
cprintf("%d.%d\n\r",rr.h.al,rr.h.ah);
textattr(0x0a);
gotoxy(30,24);
cprintf("Нажмите любую клавишу");
textattr(0x07);
getch();
clrscr();
}

```

Результаты работы программы

В процессе работы программы на экран будет выведена на пример такая информация:

Проверка состава оборудования

Тип компьютера: АТ

Конфигурация:

Дисководов ГМД: 2

Математич. сопроцессор: есть

Тип дисплейного адаптера: VGA, кол., анал., 256 Кбайт

Первичный блок памяти: 16 Кбайт

Портов RS232: 2

Джойстик: нет

Принтеров: 1

Объем оперативной памяти: 639 Кбайт

Объем extended-памяти: 384 Кбайт

Адрес ПЗУ = c000. Длина модуля = 24576 байт

Версия MS-DOS 6.20

Практическая работа №4

Тема: Управление клавиатурой

Цель работы: Изучение организации и принципов работы клавиатуры и закрепление практических навыков управления ею, а также практических навыков создания собственных программ обработки прерываний.

Постановка задачи

Разработать программу обработки прерывания от клавиатуры, которая должна:

- распознавать нажатие «горячей» комбинации клавиш и реагировать на него звуковым сигналом;
- при первом нажатии «горячей» комбинации переходить в режим блокировки ввода заданной клавиши, при втором — отменять этот режим;
- системная обработка всех других клавиш нарушаться не должна.

Индивидуальные задания

Пример решения задачи

Индивидуальное задание:

- комбинация клавиш **LeftCtrl+RightShift+F3**;
- блокирование ввода клавиши 3.

Разработка алгоритма

Структура программы

Программа состоит из основной программы и трех функций.

- **void *readvect(int in)** — функция читает вектор прерывания с номером **in** и возвращает его значение.
- **void writevect (int in, void *h)** — функция устанавливает новый вектор прерывания **in** на новый обработчик этого прерывания по адресу **h**.
- **void interrupt new9()** — процедура нового обработчика прерывания **9h**.

Описание переменных

Глобальные переменные программы: **old9** — адрес старого обработчика прерывания **9h**; **F3_code** — скан-код клавиши «F3», которая входит в комбинацию «горячих» клавиш; **key3_code** — скан-код клавиши «3», которая будет блокироваться/разблокироваться при каждом нажатии «горячей» комбинации клавиш; **f** — флаг, который при каждом нажатии «горячей» комбинации клавиш переключается из состояния 0 в 1 или из 1 в 0 (состояние 1 означает, что клавиша «3» заблокирована); **rr** и **sr** — переменные, которые используются для задания значений регистров общего назначения и сегментных регистров соответственно при вызове прерывания.

В главной программе использует символьный массив **string** для проверки работы программы.

Переменные процедуры обработки прерывания **9h**:

- **c** — переменная, которая используется для подтверждения приема из клавиатуры, в случае, если была нажата клавиша «3», а флаг **f** показывал, что эта клавиша заблокирована;
- **x, y** — переменные, которые используются для сохранения координат курсора на экране в момент вызова процедуры обработки прерывания;
- **byte17** — байт флага состояния клавиатуры в области данных BIOS по адресу 0040:0017;
- **byte18** — байт флага состояния клавиатуры в области данных BIOS по адресу 0040:0018;
- **mask** — маска, которая используется для определения нажатия клавиши левый **Shift** (в этом случае бит 1 в **byte17** установлен в 1);
- **mask17** — маска, которая используется для определения нажатия клавиши **Ctrl** (в этом случае бит 2 в **byte17** установлен в 1);
- **mask18** — маска, которая используется для определения нажатия клавиши левый **Ctrl** (в этом случае бит 0 в **byte18** установлен в 1);

Описание алгоритма программы

Главная программа выполняет такие действия:

- Запоминает адрес старого обработчика прерывания 9h, вызывая функцию readvect(in) с параметром in=9.
- Записывает в таблицу векторов прерываний адрес нового обработчика прерывания с помощью функции writevect().
- Вводом строки символов дает возможность проверить работу программы и ее реакцию на нажатие «горячей» комбинации клавиш и блокирование/разблокирование ввода клавиши «3».
- В конце работы восстанавливает в таблице векторов прерываний адрес старого обработчика.

Для решения задачи процедура обработки прерывания от клавиатуры new9() должна действовать по такому алгоритму:

- Прочитать флаги состояния клавиатуры (статус клавиш-переключателей), которые находятся в области данных BIOS (два байта по адресам 0040:0017 и 0040:0018).
- Выделить бит 1 в флаге по адресу 0040:0017 (если он равен 1, то нажата клавиша левый Shift).
- Выделить бит 2 в этом же флаге (если он равен 1, то нажата левый или правый Ctrl).
- Выделить бит 0 в флаге состояния клавиатуры по адресу 0040:0018 (если он равен 1, то нажата клавиша левый Ctrl).
- Из порта 60h прочитать скан-код нажатой клавиши.
- Если нажата комбинация клавиш левый Shift, правый Ctrl (нажата клавиша Ctrl, но это не правый Ctrl) и клавиша F3, то выполнить п.7. Иначе — перейти к п.8.
- Сигнализировать о нажатии «горячей» комбинации клавиш звуковым сигналом, переключить значение флага блокирования ввода клавиши «3» на обратное и вызвать старый обработчик прерывания от клавиатуры.
- Прочитав байт из порта 60h, определить, нажата ли клавиша «3» и если, кроме этого, еще и флаг блокирования указывает на то, что она заблокирована (f=1), то выполнить п.п. 9 и 10, иначе — вызвать старый обработчик прерывания.
- Послать подтверждение приема в клавиатуру. Для этого в порт 61h на короткое время выставить «1» по шине старшего разряда.
- Сбросить контроллер прерываний, посылая код 20h в порт 20h.

Функция readvect() читает вектор заданного прерывания. Для чтения вектора используется функция 35h DOS (прерывания 21h):

Вход: AH = 35h;

AL = номер вектора прерывания.

Выход: ES:BX = адрес программы обработки прерывания

Функция writevect() устанавливает новый вектор прерывания на заданный адрес. Для записи вектора используется функция 25h DOS:

Вход: AL = номер вектора прерывания;

DS:BX = 4-байтный адрес нового обработчика прерывания.

Текст программы

```
/*-----Управление клавиатурой-----*/
/* Подключение стандартных заголовков */
#include <dos.h>
void interrupt (*old9)(); /* Старый обработчик прерывания 9h */
void interrupt new9(); /* Новый обработчик прерывания 9h */
void *readvect (int in); /* Чтение вектора */
void writevect (int in, void *h); /* Запись вектора */
unsigned char F3_code=61; /* scan-code "F3" */
unsigned char key3_code=4; /* scan-code "3" */
char f=0; /* Флаг */
union REGS rr;
struct SREGS sr;
/*-----*/
```

```

void main()
{
char string[80]; /* Буфер для ввода текста */
textbackground(0);
clrscr();
textattr(0x0a);
printf("-----");
printf(" Лабораторная работа N5   ");
printf("-----");
printf("-----");
printf(" Управление клавиатурой   ");
printf("-----");
old9=readvect(9);
writevect(9,new9);
textattr(0x0c);
printf("\n\n\r\"горячая\" комбинация: ");
textattr(0x0a);
printf("Left Shift, Right Ctrl, F3\n\r");
textattr(0x0b);
printf("Клавиша, которая блокируется: ");
textattr(0x0f);
printf("3");
textattr(0x07);
printf("\r\nВводите строку символов>");
scanf("%s",string);
writevect(9,old9);
}
/*-----*/
/* Чтение вектора */
void *readvect(int in)
{
rr.h.ah=0x35;
rr.h.al=in;
intdosx(&rr,&rr,&sr);
return(MK_FP(sr.es,rr.x.bx));
}
/*-----*/
/* Запись вектора */
void writevect(int in,void *h)
{
rr.h.ah=0x25;
rr.h.al=in;
sr.ds=FP_SEG(h);
rr.x.dx=FP_OFF(h);
intdosx(&rr,&rr,&sr);
}
/*-----*/
/* Новый обработчик 9-го прерывания */
void interrupt new9()
{
unsigned char c,x,y;
unsigned char byte17,byte18;
unsigned char mask=0x02;
unsigned char mask17=0x04;

```

```

unsigned char mask18=0x01;

byte17=peekb(0x40,0x17);
byte18=peekb(0x40,0x18);
if((inportb(0x60)==F3_code)&&(byte17&mask)&&
    (byte17&mask17)&&!(byte18&mask18)))
{
cputs("\7");
x=wherex();
y=wherey();
gotoxy(55,3);
textattr(0x1e);
if(f==0)
{
f=1;
cprintf("Клавиша \"3\" заблокирована ");
}
else
{
f=0;
cprintf("Клавиша \"3\" разблокирована");
}
gotoxy(x,y);
textattr(0x07);
(*old9)();
}
if( (f==1) && (inportb(0x60)==key3_code) )
{
c=inportb(0x61);
outportb(0x61,c|0x80);
outportb(0x61,c);
outportb(0x20,0x20);
}
else
(*old9)();
}

```

Результаты работы программы

Во время программы при первом нажатии комбинации клавиш **LeftCtrl+RightShift+F3** программа переходит в режим блокирования вводу клавиши 3, при втором — отменяет этот режим.

Практическая работа №5

Тема: Управление таймером

Цель работы: Изучение функций системного таймера и закрепление практических навыков работы с ним.

Постановка задачи

Построить модель аналого-цифрового преобразователя (АЦП), которая работает в реальном времени. Процесс, который дискретизируется, моделируется программой (программным блоком), который выполняет циклическое вычисление функции $y=F(x)$, где x — номер итерации. Преобразователь моделируется программой, которая выполняет с заданной частотой (в реальном времени) прерывание процесса, считывание и запоминание текущего значения функции. Запомнить не меньше 80 значений функции. Обеспечить наглядное представление результатов работы «АЦП».

Индивидуальные задания

Для получения более наглядного представления «процесса» допускается подбирать другие коэффициенты функции. Частоту дискретизации выдерживать с точностью до 1 Гц.

Пример решения задачи

Индивидуальное задание

функция — $y=50*(\sin(x/10)+\cos(x/8))+R+150$;

R — в диапазоне 0 — 10;

частота — 36.4 Гц.

Разработка алгоритма решения

Структура программы

Программа состоит из основной программы и трех функций.

- **void *readvect(int in)** — функция читает вектор прерывания с номером in и возвращает его значение.
- **void writevect (int in, void *h)** — функция устанавливает новый вектор прерывания in на новый обработчик этого прерывания по адресу h .
- **void interrupt newtime()** — процедура нового обработчика прерывания таймера.

Описание переменных и констант

В этой программе применяются две константы:

- **TIMEINT=8** — номер прерывания таймера;
- **NN=100** — максимальное число показаний АЦП.

Переменные, глобальные для всей программы:

- **y** — массив показаний АЦП;
- **ny** — текущий индекс в массиве показаний;
- **yc** — текущее значение функции;
- **kf** — счетчик вызовов **oldtime** (**oldtime** вызывается каждые второй раз);
- **rr** и **sr** — переменные, которые используются для задания значений регистров общего назначения и сегментных регистров, соответственно при вызове прерывания.

Переменные главной программы:

- **oldtic** — старый коэффициент деления;
- **newtic** — новый коэффициент деления (применяется для увеличения частоты вызова прерываний таймера);
- **x** — аргумент заданной функции **F(x)**;
- **dd** — тип графического адаптера;
- **m** — режим графики;
- **errorcode** — код результата инициализации графики.

Описание алгоритма программы

Программу можно назвать моделью процесса АЦП. Главная программа постоянно вычисляет значения заданной функции **F(x)** при переменном аргументе, что имитирует непрерывный сигнал, а обработчик прерывания 8 имитирует преобразователь с постоянным шагом дискретизации по времени. Перед началом работы канал 0 таймера программируется на частоту в 2 рази большую обычной (записью в порт 43h управляющего байта 00110110b=36h, а

потом посылкой в порт 40h нового значения коэффициента деления), таким образом, «частота дискретизации» составляет около 36.4 Гц. При поступлении следующего прерывания запоминается текущее значение функции **F(x)**, старый обработчик прерывания oldtime вызывается не при каждом прерывании, а лишь один раз из двух (переменная **kf** — счетчик по модулю 2), когда **oldtime** не вызывается, наш обработчик сам сбрасывает контроллер прерываний посылкой значения 20h в порт 20h. После набора 100 «показаний АЦП» восстанавливается старый вектор обработчика таймера, а результат аналого-цифрового преобразование выводится на терминал в графическом режиме в виде решетчатой функции.

Функция **readvect()** читает вектор заданного прерывания. Для чтения вектора применяется функция 35h DOS (прерывания 21h):

Вход: AH = 35h;

AL = номер вектора прерывания.

Выход: ES:BX = адрес программы обработки прерывания.

Функция **writevect()** устанавливает новый вектор прерывания по заданному адресу. Для записи вектора применяется функция 25h DOS:

Вход: AH = 25h;

AL = номер вектора прерывания;

DS:BX = 4-байтный адрес нового обработчика прерывания.

Текст программы

```
/*-----"Управление таймером"-----*/
/* Подключение стандартных заголовков */
#include <dos.h>
#include <math.h>
#include <stdlib.h>
#include <graphics.h>
#include <time.h>
#include <conio.h>
#define TIMEINT 8 /* Прерывание таймера */
#define NN 100 /* Максимальное количество показаний */
void interrupt (*oldtime)(); /* Новый обработчик прерываний таймера */
void interrupt newtime(); /* Старый обработчик прерываний таймера */
static int y[NN]; /* Накопитель показаний */
static int ny; /* Индекс в массиве y */
static int yc; /* Текущее значение */
static int kf; /* Счетчиквызовов oldtime */
union REGS rr; /* Запись нового вектора */
struct SREGS sr;
void *readvect(int in); /* Получениестароговектора */
void writevect(int in, void *h); /* Записьновоговектора */
/*-----*/
void main()
{
    unsigned oldtic=65535u; /* Старыйкоэфф. деления */
    unsigned newtic=32768u; /* Новыйкоэфф. деления */
    int dd, /* Графический драйвер */

        m, /* Графический режим */
        errorcode; /* Код ошибки */
    double x; /* Аргумент функций sin и cos */
    textbackground(0);
    clrscr();
    textattr(0x0a);
    printf(" Лабораторная работа N6 ");
    printf("\n Управление таймером ");
```

```

textattr(0x8e);
gotoxy(35,12);
cprintf("Please wait");
/* Программирование канала 0 */
outportb(0x43,0x36); /* Управляющий байт */
outportb(0x40,newtic&0x00ff); /* Младший байт счетчика */
outportb(0x40,newtic>>8); /* Старший байт счетчика */
ny=-1; /* Признак того, что АЦП еще не началось */
kf=15;
/* Подключение к вектору */
oldtime=readvect(TIMEINT);
writevect(TIMEINT,newtime);
/* Запуск "непрерывного процесса" */
randomize();
for (x=ny=0; ny<NN; x+=1)
yc=(int)(50*(sin(x/10)+cos(x/8))+random(11)+150);
/* Восстановление вектора */
writevect(TIMEINT,oldtime);
/* Восстановление канала 0 */
outportb(0x43,0x36); /* Управляющий байт */
outportb(0x40,oldtic&0x00ff); /* Младший байт счетчика */
outportb(0x40,oldtic>>8); /* Старший байт счетчика */
/* Вывод запомненных результатов */
dd=3; /* EGA, 16 цветов */
m=1; /* Режим 640*350 */
initgraph(&dd,&m,"");
/* проверка результата инициализации */
errorcode = graphresult();
if (errorcode != grOk) /* ошибка графического режима */
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1); /* аварийное завершение */
}
setcolor(10);
settextstyle(0,0,2);
outtextxy(15,10,"Результаты аналого-цифрового преобразования:");
setcolor(9);
rectangle(15,40,624,330);
setcolor(11);
for(ny=0; ny<NN; ny++)
{
circle(22+ny*6,330-y[ny]*1,2);
line(22+ny*6,330,22+ny*6,330-y[ny]*1);
}
setcolor(12);
settextstyle(0,0,1);
outtextxy(260,340,"Нажмите любую клавишу ...");
getch();
closegraph();
}
/* Новый обработчик прерываний таймера */
void interrupt newtime()

```

```

{
if (--kf<0) {
/* Виклик oldtime — на 2-йраз */
(*oldtime)();
kf=1;
}
else /* иначе — сбросконтроллера */
outportb(0x20,0x20);
if ((ny>=0) /* Если АЦП началось, */
&&(ny<NN)) /* и NN показаний еще не набрано, */
y[ny++]=ус; /* запоминание очередного показания */
}
/* Получение старого вектора */
void *readvect(int in)
{
rr.h.ah=0x35; rr.h.al=in;
intdosx(&rr,&rr,&sr);
return(MK_FP(sr.es,rr.x.bx));
}
/* Записьновоговектора */
void writevect(int in, void *h)
{
rr.h.ah=0x25;
rr.h.al=in;
sr.ds=FP_SEG(h);
rr.x.dx=FP_OFF(h);
intdosx(&rr,&rr,&sr);
}

```

Результаты работы программы

Результат работы представляется в графическом режиме в виде решетчатой функции на экране терминала.

Практическая работа №6

Тема: Управление видеоадаптером

Цель работы: Изучение особенностей функционирования видеосистемы в текстовом режиме и получение практических навыков работы с видеомонитором в этом режиме.

Постановка задачи

Применяя прямую запись в видеопамять получить на экране оригинальный, желательно динамический видеоэффект. Возможны (но не обязательны) такие варианты видеоэффектов:

- «теннисный мячик» — шарик, который летает по экрану и отражается от рамок и границ экраны;
- «сухой лист» — опадание букв с экрана;
- «жук-пожиратель» — фигурка, которая перемещается по экрану по случайной траектории и «съедает» буквы;
- «удав» — то же, что и «жук», но к тому же он увеличивается в размерах, по мере «поедания» букв;

Пример решения задачи

Индивидуальное задание:

- весь экран (80x25 символов) условно делится на прямоугольники размером (10x5 символов).
- текущий прямоугольник инвертирует экран под собой.
- управлять положением текущего прямоугольника на экране можно с помощью клавиш управления курсором.
- при нажатии клавиши «пробел» текущий прямоугольник обменивается содержимым с левым верхним прямоугольником.
- при нажатии клавиши Enter содержимое прямоугольников экрана начинает перемешиваться случайным образом между собой до нажатия любой клавиши.
- после этого, используя клавиатуру, можно восстановить начальный экран или выйти из программы (клавиша Esc).

Разработка алгоритм решения

Структура программы

Программа состоит из основной функции **main()** и семи вспомогательных функций.

- **byte GetSym(x1,y1)** — функция читает символ с заданной позиции экрана дисплея.
- **byte GetAtr(x1,y1)** — функция читает атрибут символа с заданной позиции экрана дисплея.
- **void PutSym(x1,y1,sym)** — функция выводит на экран дисплея символ в заданную позицию (x1,y1).
- **void PutAtr(x1,y1,atr)** — функция меняет на экране дисплея атрибут символа в заданной позиции (x1,y1).
- **void Invert(x1,y1)** — функция инвертирует участок на экране размером (10x5), координаты (x1,y1) задают один из участков на экране.
- **void Change(x,y)** — функция обменивает содержимое текущего участка с содержимым левого верхнего участка на экране. Координаты (x,y) задают положение текущего участка.
- **void RandText(void)** — функция псевдослучайным образом перетасовывает все участки на экране.

Описание переменных

Переменные, глобальные для всей программы:

- **xk** — координата X текущего участка;
- **yk** — координата Y текущего участка;
- Координаты участка задаются в пределах: **X** — [0..7], **Y** — [0..4] .

Описание алгоритма программы

Основная функция **main()** проверяет, был ли в командной строке дополнительный параметр. Если нет, программа не очищает старый экран. Если какой-нибудь параметр был, то экран

очищается и выводится инструкция по управлению программой. Далее в основной программе выполняется бесконечный цикл, в котором обрабатываются коды нажатых клавиш и, в зависимости от них, вызываются вспомогательные функции. Выход из цикла — по клавише **Esc**.

Функции **GetSym(x1,y1)**, **GetAtr(x1,y1)** читают непосредственно из видеопамати дисплея символ и атрибут соответственно.

Функции **PutSym(x1,y1,sym)**, **PutAtr(x1,y1,atr)** выводят непосредственно в видеопамать дисплея символ и атрибут соответственно.

Во всех этих четырех функциях координаты задаются в квадрате 79x24 символов (нумерация начинается с нуля).

Функция **Invert(x1,y1)** использует функции **GetAtr** и **PutAtr** для инверсии прямоугольника экрана размером 10x5 по маске 0x7f, при этом независимо выполняется инверсия фона и цвета символа.

Функция **Change(x,y)** обменивает содержимое текущего участка с содержимым левого верхнего участка путем последовательного побайтного обмена атрибутов и символов. Она использует функции **GetSym**, **GetAtr**, **PutSym**, **PutAtr**.

Функция **RandText(void)** — псевдослучайным образом перетасовывает все участки на экране, при этом она в цикле увеличивает на единицу локальные в данной функции координаты текущего участка **xk**, **yk** и обращается к функции **Change**. Таким образом достигается эффект перемешивания. Функция работает, пока не будет нажата любая клавиша.

Текст программы

```
/*-----Управление видеоадаптером.-----*/
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
/*-----Константы----- */
#define VSEG 0xb800 /* Сегментный адрес видеопамати */
#define byte unsigned char

#define word unsigned int
#define Esc 27
#define Spase 32
#define Enter 13
#define Up 0x48
#define Down 0x50
#define Left 0x4b
#define Right 0x4d
#define Home 0x47
int xk,yk;
/*----Чтение символа из видеопамати-----*/
byte GetSym(x1,y1)
int x1,y1;
{
return(peekb(VSEG,y1*160+x1*2));
}
/*---Чтение атрибута из видеопамати-----*/
byte GetAtr(x1,y1)
int x1,y1;
{
return(peekb(VSEG,y1*160+x1*2+1));
}
/*---Запись символа в видеопамать-----*/
void PutSym(x1,y1,sym)
```

```

int x1,y1;
byte sym;
{
pokeb(VSEG,y1*160+x1*2,sym);
}
/*---Запись атрибута в видеопамять-----*/
void PutAtr(x1,y1,atr)
int x1,y1;
byte atr;
{
pokeb(VSEG,y1*160+x1*2+1,atr);
}
/*--Инверсия квадрата на экране-----*/
void Invert(x1,y1)
int x1,y1;
{
byte b;
int i,j;
for (j=0;j<5;j++)
for (i=0;i<10;i++)
{
b=GetAtr(x1*10+i,y1*5+j);
PutAtr(x1*10+i,y1*5+j,(b^0x7f));
}
}
/*--Замена текущего квадрата на левый верхний--*/
void Change(x,y)
byte x,y;
{
int i,j;
byte ba,bs;
if ((x!=0)||(y!=0))
for (j=0;j<5;j++)
for (i=0;i<10;i++)
{
bs=GetSym(x*10+i,y*5+j);
ba=GetAtr(x*10+i,y*5+j);
PutSym(x*10+i,y*5+j,GetSym(i,j));
PutAtr(x*10+i,y*5+j,GetAtr(i,j));
PutSym(i,j,bs);
PutAtr(i,j,ba);
}
}
/*--Перемешивание квадратов до нажатия клавиши-*/
void RandText(void)
{
Invert(xk,yk);
xk=5;
yk=1;
while(!kbhit())
{
Change(xk,yk);
xk++;
if (xk>7) xk=0;
}
}

```

```

yk++;
if (yk>4) yk=0;
}
Invert(xk,yk);
}
/*-----Начало основной программы-----*/
main(int argn,char *argc[])
{
int i;
xk=0;
yk=0;
if (argn>1){}
else /* Если параметров нет, вывод инструкции */
{
textattr(10);
clrscr();
printf("-----");
printf(" Лабораторная работа N7 ");
printf("-----");
printf("-----");
printf(" Управление видеоадаптером. ");
printf("-----");
textattr(15);
gotoxy(23,4);printf("Демонстрация работы с видеопамятью.");
textattr(12);
gotoxy(30,6);printf("<< М О З А И К А >>");
textattr(14);
gotoxy(30,8);printf("Клавиши управления:");
gotoxy(7,10);printf("<Left, Right, Up, Down> — ");
printf("управление выделенным квадратом.");
gotoxy(7,11);printf("<Space Bar> — Обмен содержимым ");
printf("между выделенным квадратом");
gotoxy(7,12);printf(" и левым верхним");
printf(" квадратом.");
gotoxy(7,13);printf("<Enter> — перемешивание квадратов");
printf(" до нажатия любой клавиши.");
gotoxy(7,14);printf("<Esc> — вихід.");
textattr(11);
gotoxy(28,16);printf("З А Д А Ч А И Г Р Ы :");
gotoxy(14,17);printf("Собрать при помощи клавиш ");
printf("управления начальный экран.");
textattr(12);
gotoxy(27,19);printf("Ж е л а е м у с п е х а !");
textattr(7);
gotoxy(1,21);printf("Примечание:При запуске с ");
printf("параметром<->");
gotoxy(13,22);printf("начальным экраном для игры ");
printf("является текущий.");
}
Invert(xk,yk);
for(i=0;i==0;)
switch(getch())
{ /* Обработка нажатых клавиш */
case Esc: i++; break;

```

```
case Enter:RandText();break;
case Spase:Invert(xk,yk);
```

```
Change(xk,yk);
Invert(xk,yk);
break;
case 0:
switch (getch()) {
case Left:Invert(xk,yk);
xk--;
if(xk<0) xk=7;
Invert(xk,yk);
break;
case Right:Invert(xk,yk);
xk++;
if(xk>7) xk=0;
Invert(xk,yk);
break;
case Up:Invert(xk,yk);
yk--;
if(yk<0) yk=4;
Invert(xk,yk);
break;
case Down:Invert(xk,yk);
yk++;
if(yk>4) yk=0;
Invert(xk,yk);
break;
}
}
Invert(xk,yk);
}
```

Результаты работы программы

Результаты работы программы выводятся на экран терминала и меняются интерактивно.

Практическая работа №7

Тема: Двухпросмотровый алгоритм

Цель работы: Изучение работы двухпросмотрового алгоритма

Постановка задачи

Макропроцессор, как и язык Ассемблера, просматривает и обрабатывает строки текста. Но в языке все строки связаны адресацией — одна строка может ссылаться на другую при помощи адреса или имени, которое должно быть «известно» Ассемблеру.

Более того, адрес присваиваемый каждой отдельной строке зависит от содержимого, количества и адресов предшествующих строк. Если рассматривать макроопределение, как единый объект, то можно сказать, что строки нашего макроопределения не так сильно взаимосвязаны.

Макроопределения не могут ссылаться на объекты вовне этого макроопределения.

Предположим, что в теле макроопределения есть строка INCR X, причем перед этой командой параметр X получил значение 10. Макропроцессор не производит синтаксический анализ, а производит простую текстовую подстановку вместо «X» подставляется «10».

Наш алгоритм будет выполнять 2 систематических просмотра входного текста. В первый проход будут детерминированы все макроопределения, во второй проход будут открыты все ссылки на макросы.

Так же, как и язык Ассемблера не может выполнить ссылку на символ до того момента, как он встретит этот символ, язык макрокоманд не может выполнить расширение до тех пор, пока не встретит соответствующее макроопределение.

Во время первого просмотра проверяется каждый код операции, макроопределения запоминаются в таблице макроопределений, а копия исходного текста без макроопределений запоминается во внешней памяти, для использования ее при втором проходе.

Помимо таблицы макроопределений во время первого прохода будет также таблица имен, во второй проход она будет использоваться для выделения макроопераций и расширения их до текста соответствующего макроопределения.

Пример решения задачи

Данные для первого просмотра

1. ВХТ — Входной текст
2. ВЫХ1 — Выходная копия текста для использования во второй проход.
3. МДТ — таблица макроопределений, в которой хранятся тела макроопределений
4. МНТ — таблица имен, необходимая для хранения имен макрокоманд, определенных в МНТ
5. МДТС — счетчик для таблицы МДТ
6. МНТС — счетчик для таблицы МНТ
7. АЛА — массив списка параметров для подстановки индексных маркеров вместо формальных параметров, перед запоминанием определения.

Данные для второго просмотра

1. ВЫХ1 — Выходная копия текста после первого прохода
2. ВЫХ2 — Выходная копия текста после второго прохода
3. МДТ — таблица макроопределений, в которой хранятся тела макроопределений
4. МНТ — таблица имен, необходимая для хранения имен макрокоманд, определенных в МНТ
5. МДТС — счетчик для таблицы МДТ
6. МНТС — счетчик для таблицы МНТ
7. АЛА — массив списка параметров для подстановки индексных маркеров вместо формальных параметров, перед запоминанием определения.

Алгоритм

Приведем формальную запись соответствующих алгоритмов обработки макроопределений двухпросмотровым способом.

Каждый из алгоритмов осуществляет построчный просмотр входного текста.

Первый просмотр — макроопределения

Алгоритм первого просмотра проверяет каждую строку входного текста. Если она представляет собой псевдооперацию MACRO, то все следующие за ней строки запоминаются в ближайших свободных ячейках МДТ.

Первая строка макроопределения — это имя самого макроса. Имя заносится в таблицу имен МНТ с индексом этой строки в МДТ. При этом происходит также подстановка номеров формальных параметров, вместо их имен.

Если в течение просмотра встречается команда END, то это означает, что весь текст обработан, и управление можно передавать второму просмотру для обработки макрокоманд.

Второй просмотр — расширение макрокоманд

Алгоритм второго просмотра проверяет мнемонический код каждого предложения. Если это имя содержится в МНТ, то происходит обработка макропредложения по следующему правилу: из таблицы МНТ берется указатель на начало описания макроса в МДТ.

Макропроцессор готовит массив списка АЛА содержащий таблицу индексов формальных параметров и соответствующих операндов макрокоманды. Чтение производится из МДТ, после чего в прочитанную строку подставляются необходимые параметры, и полученная таким образом строка записывается в ВЫХТ2. Когда встречается директива END, текст полученного кода передается для компиляции Ассемблеру.

Первый просмотр

Начало алгоритма

МДТС = 0

МНТС = 0

ФЛАГ ВЫХОДА = 0

цикл пока (ФЛАГ ВЫХОДА == 0) {

чтение следующей строки ВХТ

если !(операция MACRO) {

вывод строки в ВЫХТ1

если (операция END) ФЛАГ ВЫХОДА = 1

}

иначе {

чтение идентификатора

запись имени и индекса в МНТ

МНТС ++

приготовить массив списка АЛА

запись имени в МДТ

МДТС ++

цикл {

чтение следующей строки ВХТ

подстановка индекса операторов

добавление в МДТ

МДТС ++

} пока !(операция MEND)

}

}

переход ко второму проходу

конец алгоритма

Второй просмотр

Начало алгоритма

ФЛАГ ВЫХОДА = 0

цикл пока (ФЛАГ ВЫХОДА == 0) {

чтение строки из ВЫХТ1

НАЙДЕНО = поиск кода в МНТ

если !(НАЙДЕНО) {

запись в ВЫХТ2 строки

если (операция END) {

```

ФЛАГ ВЫХОДА = 1
}
}
иначе {
УКАЗАТЕЛЬ = индекс из МНТ
Заполнение списка параметров АЛА
цикл {
УКАЗАТЕЛЬ ++
чтение след.строки из МДТ
подстановка параметров
вывод в ВЫХТ2
} пока !(операция MEND)
}
}
переход к компиляции
конец алгоритма

```

Практическая работа №8

Тема: Однопросмотровый алгоритм

Цель работы: Изучение работы однопросмотрового алгоритма

Постановка задачи

Предположим, что мы допускаем реализацию макроопределения внутри макроопределений. Основная проблема здесь заключена в том, что внутреннее макро определено только после того, как выполнен вызов внешнего.

Для обеспечения использования внутреннего макро нам придется повторять как просмотр обработки макроопределений, так и просмотр обработки макрокоманд. Однако существует и еще одно решение, которое позволяет произвести распознавание и расширение в один просмотр.

Рассмотрим аналогию с Ассемблером. Макроопределение должно обрабатываться до обработки макрокоманд, поскольку макро должны быть определены для процессора раньше, чем макрокоманды обращения к ним.

Однако, если мы наложим ограничение, что каждое макроопределение должно быть определено до того, как произойдет обращение к нему, мы устраним основное препятствие для однопросмотровой обработки.

Заметим, что то же самое может быть верно и для символических имен в Ассемблере, но такое требование было бы неоправданным ограничением для программиста.

В случае же макрорасширения может быть вполне естественно потребовать, чтобы объявления макро предшествовали вызовам. Это не накладывает очень существенных ограничений на использование аппарата макрокоманд.

Этот механизм даже не запрещает обращение макро к самому себе, поскольку обращение ведется в тот момент, когда имя макроса уже определено.

Расширение же макроса идет не в процессе разбора макроса, а в процессе последующего вызова.

Предложенный ниже алгоритм объединяет два вышеприведенных алгоритма для двупросмотрового макроассемблера в один.

Пример решения задачи

Алгоритм

Однопросмотровый макроассемблер

Начало алгоритма

```

МТДС = 0
МНТС = 0
ФЛАГ ВЫХОДА = 0
цикл пока !(ФЛАГ ВЫХОДА) {
чтение следующей строки ВХТ
НАЙДЕНО = поиск кода в МНТ

```

```

если (НАЙДЕНО) {
МДИ = 1
УКАЗАТЕЛЬ = индекс из МНТ
Заполнение списка параметров АЛА
цикл {
УКАЗАТЕЛЬ ++
чтение след.строки из МДТ
подстановка параметров
вставка во ВХТ
} пока !(операция MEND)
иначе если !(операция MACRO) {
вывод строки в ВЫХТ1
если (операция END) ФЛАГ ВЫХОДА = 1
}
иначе {
чтение идентификатора
запись имени и индекса в МНТ
МНТС ++
приготовить массив списка АЛА
запись имени в МДТ
МДТС ++
цикл {
чтение следующей строки ВХТ
подстановка индекса операторов
добавление в МДТ
МДТС ++
} пока !(операция MEND)
}
}
конец алгоритма

```

Описание алгоритма

Данный алгоритм является упрощением алгоритма приведенного раньше. Различие состоит в том, что современные средства интеллектуализации программирования дают нам возможность осуществлять вставки и удаления из крупных массивов с минимальными затратами процессорного времени, что было невозможно при использовании перфокарт.

Кроме того, скорость работы современных процессоров настолько велика, что позволяет производить прямые вставки и удаления в массивах данных средней величины (скажем, до 64 килобайт) в режиме реального времени.

Таким образом, расширение исходного макроса может быть напрямую вставлено в массив исходного текста и обработано в расширенном виде. Такая технология позволяет значительно упростить алгоритм обработки макроязыка.

Практическая работа №9

Тема: Реализация внутри Ассемблера

Цель работы: Изучение реализации внутри Ассемблер. Создание первой программы.

Постановка задачи

Макропроцессор, описанный нами предназначался для обработки текста в режиме препроцессора, то есть он выполнял полный просмотр входного текста, до того, как передать управление Ассемблеру.

Но макропроцессор также может быть реализован внутри первого прохода Ассемблера. Такая реализация позволяет исключить промежуточные файлы, и позволяет достичь на порядок большей интеграции макропроцессора и Ассемблера путем объединения сходных функций.

Например, возможно объединение таблиц имен макросов и имен кода операции; специальный признак может указывать на то макро это или встроенная операция.

Основные преимущества включения макропроцессора в первый просмотр состоят в следующем:

- Многие функции не надо реализовывать дважды (например, функции ввода-вывода, проверки на тип, и т.п.)
- В процессе обработки отпадает необходимость создавать промежуточные файлы или массивы данных.
- У программиста появляются дополнительные возможности по совмещению средств Ассемблера (например, команды EUQ) совместно с макрокомандами.

Основные недостатки:

- Программа должна требовать больше оперативной памяти, что критично на некоторых типах ЭВМ, не имеющих много оперативной памяти.
- Реализация подобного типа задачи может оказаться на порядок сложнее, чем отдельная реализация Ассемблера и макропроцессора.

Отдельно от рассмотрения реализации аппарата макросредств в Ассемблер лежит рассмотрение дополнительного просмотра, используемого многими программами для выявления определенных характеристик исходной программы, таких как типы данных.

Располагая таким макропроцессором, можно использовать команды условной компиляции, позволяющие поставить расширение макрокоманд в зависимость от определенных характеристик программы.

Создание программы на языке ассемблера

В отличие от языков программирования высокого уровня, язык ассемблера поставляется без среды разработки программ. Поэтому разработчику приходится самостоятельно контролировать весь процесс создания программы. Для языка ассемблера этот процесс включает в себя следующие этапы:

1. **Постановка задачи** (точное и подробное описание функциональности будущей программы, а также описание всех входных и выходных данных и способа их передачи программе);
2. **Разработка алгоритма программы** (построение блок-схемы, граф-схемы или текстовое или математическое описание решения);
3. **Формализация алгоритма** (запись алгоритма на языке программирования). Создание текстового файла программы с расширением .asm (например my.asm). Отсутствие среды разработки позволяет программисту самостоятельно выбрать текстовый редактор для написания кода программы. Для этой цели подойдет любой текстовый редактор с нумерацией строк, мы рекомендуем редактор, встроенный в оболочку "FAR Manager".
4. **Компиляция программы**; Компиляция - процесс перевода программы из текстового вида в машинный код. При использовании компилятора фирмы Borland необходимо выполнить: `tasm my.asm`, т.е. запускаем компилятор `tasm` и передаем с командной строки имя файла, содержащего программу. Если программа имеет синтаксические ошибки, компилятор выдаст сообщение об ошибке с указанием номера строки и описанием для каждой ошибки (нужно вернуться на этап №3 и исправить синтаксические ошибки). В случае успешной компиляции будет создан файл,

содержащий объектный код программы `my.obj`, который ещё не является исполняемым модулем.

5. **Компоновка программы;** Компоновка - создание из файла объектного кода исполняемого модуля. `link my.obj` В качестве параметра компоновщик `link` принимает имя файла содержащего объектный код программы (в нашем случае - `my.obj`). В случае успешной компоновки будет создан исполняемый модуль `my.exe`
6. **Запуск и тестирование** исполняемого модуля программы. На данном этапе необходимо проверить, соответствует ли написанная программа постановке задачи, сделанной нами на этапе №1. Неправильная работа программы говорит об алгоритмической ошибке (семантическая ошибка), поэтому для успешного её устранения нужно вернуться на этап разработки алгоритма (этап №2).

Написание первой программы на языке ассемблера

1. **Постановка задачи.** Написать программу, которая выводит на экран строку "Привет!".
2. **Разработка алгоритма программы.** Алгоритм линейный, разработки не требует.
3. **Формализация (запись) алгоритма**
В текстовом редакторе создаем файл `privet.asm` и записываем в него следующий код (без номеров строк) :

1		<code>data segment</code>	<code>;описание сегмента данных</code>
2		<code>mes db 'Привет!\$'</code>	<code>;строка для вывода на экран. '\$' - признак конца строки</code>
3		<code>data ends</code>	<code>;конец сегмента данных</code>
4			
5		<code>code segment</code>	<code>;начало сегмента кода</code>
6		<code>start:</code>	<code>;метка start - начало нашей программы</code>
7		<code>assume cs:code, ds: data</code>	<code>;директива компилятора</code>
8		<code>mov ax, data</code>	<code>;настройка сегмента данных</code>
9		<code>mov ds, ax</code>	
10			
11		<code>mov ah, 9</code>	<code>;функция №9 - вывод строки на экран</code>
12		<code>lea dx, mes</code>	<code>;берём адрес строки</code>
13		<code>int 21h</code>	<code>;вызов прерывания для вывода строки</code>
14			
15		<code>mov ax, 4c00h</code>	<code>;функция завершения программы</code>
16		<code>int 21h</code>	<code>;завершаем программу</code>
17		<code>code ends</code>	<code>;конец сегмента кода</code>
18		<code>end start</code>	<code>;конец программы с точкой входа start</code>

4. **Компиляция программы** `asm privet.asm`
5. **Компоновка программы** `link privet.obj`
6. **Запуск и тестирование** `privet.exe`

Описание программы `privet.asm`

Строки 1 - 3 программы `privet.asm` содержат описание сегмента данных. Сегмент данных - область памяти, в которой будут храниться данные для наших программ. Строки 5 - 17 — это код программы, её исполняемая часть.

В 8 и 9 строках выполняется настройка сегмента данных программы.

Строки 11 - 13 — вывод строки на экран при помощи функции №9 прерывания `21h` (подробнее о функциях и работе с ними на следующей лабораторной работе).

15 и 16 строки — стандартное завершение программы.

После символа ';' пишутся комментарии, они не обрабатываются компилятором.

Переход на новую строку

Для организации перехода на новую строку достаточно вывести на экран символы перевода строки и возврата каретки (CR/LF). Эти символы имеют коды 10 и 13. Если в нашей программе

необходимо после вывода строки перейти на новую, то для этого достаточно переписать вторую строку программы:

- `mes2 db 'Выводим строку и переходим на новую...', 10, 13, '$'`
- Переход на новую строку можно выполнить и до вывода сообщения на экран:
- `mes3 db 10, 13, 'Выводим с новой строки...$'`

Задание для выполнения

Написать программу, которая выводит одно под другим следующие сообщения:

Привет!

Меня зовут компьютер!

До свидания!

Практическая работа №10

Тема: Главная загрузочная запись

Цель работы: Получение практических навыков в работе с Главной Загрузочной Записью жесткого диска.

Постановка задачи

Прочитать и выполнить форматный вывод на экран Главной Загрузочной Записи жесткого диска на своем рабочем месте.

Порядок выполнения

Порядок выполнения работы и содержание отчета определены в общих указаниях.

Пример решения задачи

Разработка алгоритма решения

Программа состоит из основной программы **main()**, которая реализует все действия для чтения Главной Загрузочной Записи.

Описание переменных

Переменные в основной программе:

- `x, y` — экранные координаты;
- `head` — номер головки (0);
- `Sect_Trk` — номер дорожки и сектора (0,1);
- `ndrive=0` — номер логического диска;
- `EndList` — указатель на подпись.

Кроме того, в программе есть такие структуры:

Структура элемента раздела:

```
struct Part {  
    byte ActFlag; /* признак активного раздела */  
    /* физический адрес начала раздела */  
    byte Begin_Hd; /* # головки */  
    wordBegin_SecTrk; /* # сектора дорожки */  
    byte SysCode; /* код системы */  
    /* физический адрес конца раздела */  
    byte End_Hd; /* # головки */  
    word End_SecTrk; /* # сектора и дорожки */  
    dword RelSec; /* # сектора початку */  
    dword Size; /* количество секторов */  
};
```

Структура главной загрузочной записи.

```
struct MBR {  
    char LoadCode[0x1be]; /* программа загрузки */  
    struct Part trt[4]; /* 4 элементарных разделов */  
    word EndFlag; /* подпись MBR */  
};
```

Описание алгоритма программы

Эта программа демонстрирует разделение логического диска.

Начальный адрес для чтения задается: 0,0,1. При помощи прерывания 0x13 программа считывает сектор по заданному адресу, далее происходит поэлементный анализ таблицы разделов — пока не встретится признак конца таблицы или раздел нулевого размера. Значения полей элемента таблицы выводятся на экран. Манипуляции, которые описываются макросами TRK и SECT, обеспечивают распаковку номера дорожки и сектора. Если в поле SysCode содержится признак расширенного раздела, то устанавливается новый дисковый адрес, считывается новый сектор и анализируется новая таблица.

Текст программы

```
/*---"Главная загрузочная запись"-----*/
/* Стандартные заголовки */
#include<dos.h>
#include <conio.h>
/* Типыданных */
#define byte unsigned char
#define word unsigned int
#define dword unsigned long
void read_MBR(void); /* Чтение MBR */
/* Получение из упакованного SecTrk # сектора */
#define SECT(x) x&0x3f
/* Получение из упакованного SecTrk # дорожки */
#define TRK(x) (x>>8)|((x<<2)&0x300)
/* структура элемента раздела */
struct Part {
    byte ActFlag; /* признак активного раздела */
    /* физический адрес начала раздела */
    byte Begin_Hd; /* # головки */
    word Begin_SecTrk; /* # сектора и дорожки */
    byte SysCode; /* код системы */
    /* физический адрес конца раздела */
    byte End_Hd; /* # головки */
    word End_SecTrk; /* # сектора и дорожки */
    dword RelSec; /* # сектора початку */
    dword Size; /* количество секторов */
};
/* структура главной загрузочной записи */
struct MBR {
    char LoadCode[0x1be]; /* программа загрузки */
    struct Part trt[4]; /* 4 эл-та разделов */
    word EndFlag; /* подпись MBR */
} mbr;
/* дополнительные переменные */
int x=10,y; /* экранные координаты */
byte head=0; /* номер головки (0) */
word Sect_Trk=1; /* номер дорожки и сектора (0,1) */
int ndrive=0; /* номер логического диска */
word *EndList; /* указатель на подпись */
union REGS rr;
struct SREGS sr;
word i;
/*-----*/
main()
{
    textbackground(0);
    clrscr();
```

```

textattr(0x0a);
cprintf("    Лабораторная работа N8");
gotoxy(1,2);
cprintf("    Главная загрузочная запись");
textattr(12);
gotoxy(30,4);
cprintf("Разделы жесткого диска:\n");
gotoxy(1,6);
textattr(11);
cprintf("Лог.диск ----> \n\r");
cprintf("Признак -----> \n\r");
cprintf("Код системы --> \n\r");
cprintf("Начало: гол.--> \n\r");
cprintf("    дор.--> \n\r");
cprintf("    сект.-> \n\r");
cprintf("Конец: гол.-->\n\r");
cprintf("    дор. -> \n\r");
cprintf("    сект.-> \n\r");
cprintf("Нач.сектор ---> \n\r");
cprintf("Размер -----> \n\r");
textcolor(11);
NEXT:
    read_MBR();
    for (EndList=(word *)&mbr.rt[(i=0)];
        (*EndList!=0xaa55)&&(mbr.rt[i].Size>0L);
        EndList=(word *)&mbr.rt[++i])
    {
        /* координаты курсора */
        y=6;
        x+=7;
        gotoxy(x,y++);
        if (mbr.rt[i].SysCode==5)
            {textattr(13);
            cprintf("Ext ");
            }
        else
            textattr(12);
        cprintf("%-7c",'C'+ndrive++);

        gotoxy(x,y++); textattr(14);
        cprintf("%02xH  ",mbr.rt[i].ActFlag);
        gotoxy(x,y++); textattr(15);
        cprintf("%-7d",mbr.rt[i].SysCode);
        gotoxy(x,y++); textattr(14);
        cprintf("%-7d",mbr.rt[i].Begin_Hd);
        gotoxy(x,y++); textattr(15);
        cprintf("%-7u",TRK(mbr.rt[i].Begin_SecTrk));
        gotoxy(x,y++); textattr(14);
        cprintf("%-7u",SECT(mbr.rt[i].Begin_SecTrk));
        gotoxy(x,y++); textattr(15);
        cprintf("%-7d",mbr.rt[i].End_Hd);
        gotoxy(x,y++); textattr(14);
        cprintf("%-7u",TRK(mbr.rt[i].End_SecTrk));
        gotoxy(x,y++); textattr(15);

```

```

cprintf("%-7u",SECT(mbr.rt[i].End_SecTrk));
gotoxy(x,y++); textattr(14);
cprintf("%-7lu",mbr.rt[i].RelSec);
gotoxy(x,y++); textattr(15);
cprintf("%-7lu",mbr.rt[i].Size);
if (mbr.rt[i].SysCode==5)
{
    /* если код системы 5, раздел содержит свою таблицу разделов; определяется ее дисковый
адрес, и новая таблица считывается в память */
    head=mbr.rt[i].Begin_Hd;
    Sect_Trk=mbr.rt[i].Begin_SecTrk;
    goto NEXT;
}
}
gotoxy(x,y++);
textattr(10+128);
gotoxy(29,18);
cprintf("Нажмите любую клавишу...");
getch();
}
/*-----Читання MBR-----*/
void read_MBR(void)
{
    rr.h.ah=2;    /* Чтение */
    rr.h.al=1;    /* Секторов 1 */
    rr.h.dl=0x80; /* Жесткий диск */
    rr.h.dh=head; /* Головка */
    rr.x.cx=Sect_Trk; /* Дорожка, сектор */
    sr.es=FP_SEG(&mbr); /* Адрес буфера в ОП */
    rr.x.bx=FP_OFF(&mbr);
    int86x(0x13,&rr,&rr,&sr);
    /* Проверка ошибок чтения */
    if (rr.x.cflag)
    {
        printf("Ошибка чтения: %x. ",rr.h.ah);
        printf("Нажмите любую клавишу...\n\7");
        getch();
        exit();
    }
}

```

Результаты работы программы

В процессе работы программы на экран выводится информация такого вида:

Главная загрузочная запись

Разделы жесткого диска:

Лог.диск ----->CExtEExtG

Признак -----> 80H 00H 00H 00H 00H

Код системы --> 1 5 4 5 0

Начало: гол.--> 1 0 1 0 1

дор.--> 0 121 121 724 724

сект.-> 1 1 1 1 1

Конец: гол.--> 4 4 4 4 4

дор. -> 120 975 723 975 975

сект.-> 17 17 17 17 17

Нач.сектор ---> 17 10285 17 51255 17

Размер -----> 10268 72675 51238 21420 21403

Нажмите любую клавишу...

Практическая работа №11

Тема: Использование компонентов TLabel, TButton

Цель работы: изучить интерфейс системы программирования Delphi, рассмотреть технологию ввода программного кода, алгоритм сохранения программ.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Общие положения

Delphi – интегрированная среда разработки (**IntegratedDevelopmentEnvironment – IDE**).

Delphi дает возможность создавать программы в стиле визуального конструирования формы, разместив на ней какие-либо визуальные элементы.

В **Delphi** имеются 10 окон, но после загрузки появляются четыре окна (рис.1).

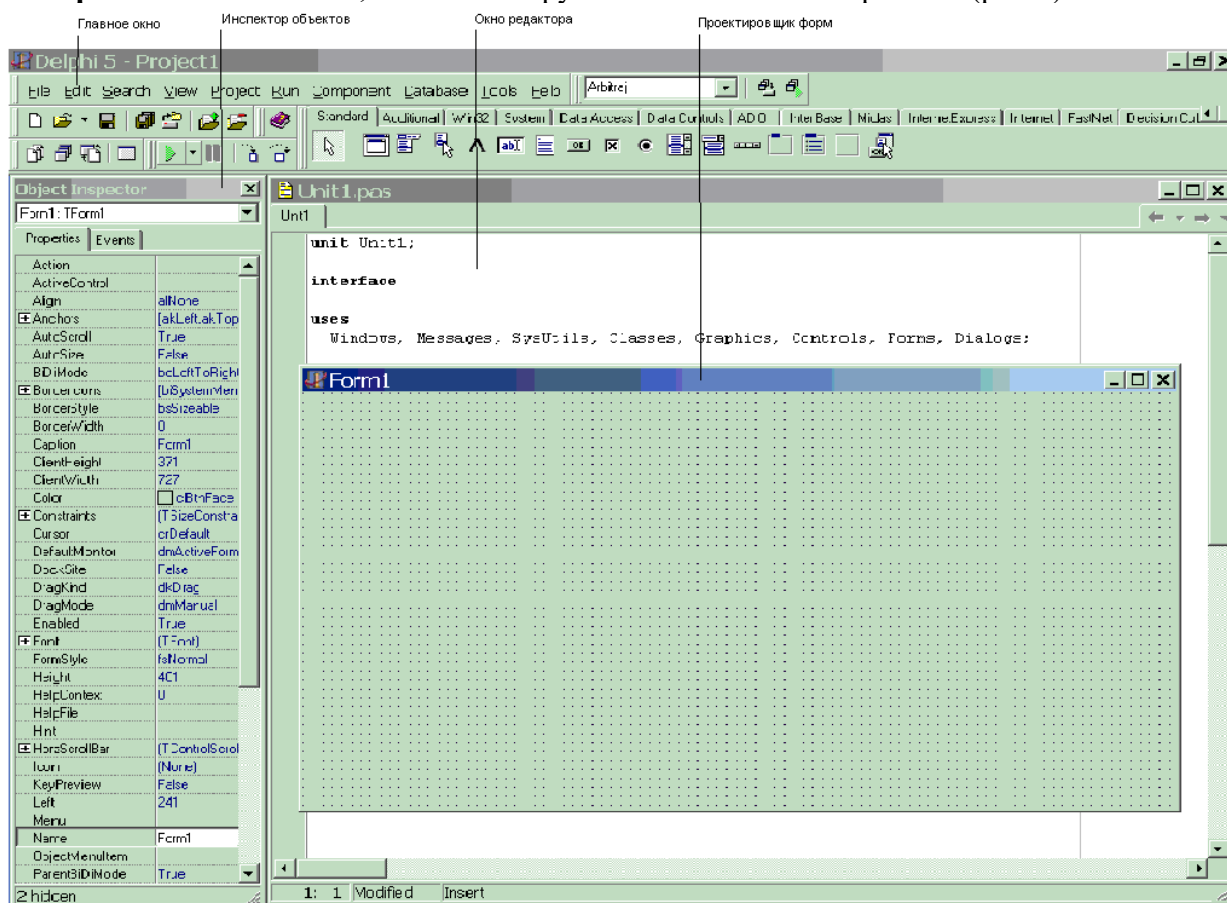


Рис. 1. Интерфейс программы

- Главное окно Delphi - <имя проекта>;
- окно с формой для проектирования приложения **Form1** (окно проектировщика формы);
- окно инспектора объектов **ObjectInspector**;
- окно редактора кода **Unit1.pas**.

В главном окне реализуются основные функции управления проектом создаваемой программы.

Это окно содержит (рис. 2):

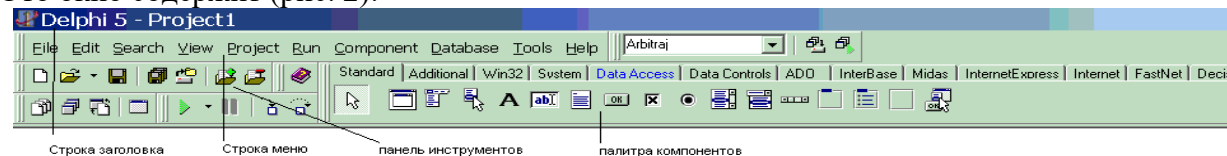


Рис. 2. Функции управления проектом

- Строку заголовка;
- Строку меню;
- Панель инструментов;
- Палитру компонентов.

Строка заголовка главного окна отображает имя открытого в данный момент проекта.

По заголовку главного окна можно определить, в каком режиме открыт проект:

Текст заголовка	Режим работы
Delphi – Project1	Редактирование
Delphi – Project1 [Running]	выполнение

Строка меню содержит команды, необходимые для разработки и тестирования приложений, и используется так же, как любое стандартное меню **Windows**.

Панель инструментов предназначена для выполнения некоторых команд, реализуемых главным меню. На этой панели есть, в частности, кнопка сохранения проекта на диске ,

кнопка открытия проекта , кнопка запуска программы на выполнение .

Палитра компонентов устроена в виде набора пиктограмм. Совокупность наборов составляет библиотеку визуальных компонентов (**VisualComponentLibrary – VCL**). Имеется несколько категорий компонентов, каждая из которых расположена на своей вкладке. С помощью палитры компонентов мы будем создавать экземпляры компонент (объекты) на форме.

Окно проектировщика формы – главное место, где происходит сборка программы из компонентов, содержащихся в палитре компонентов. Сама форма – это уже готовая к выполнению программа. В указанное место формы будет вставляться объект – экземпляр компонента выбранного типа.

Окно инспектора объектов отображает свойства активизированного щелчком мышью какого-либо компонента или самой формы. Имя активизированного компонента находится под заголовком окна. Это окно имеет две вкладки – **Properties** (Свойства) и **Events** (События). Свойства определяет атрибут компонента, например размер кнопки или шрифт метки. Событие означает, например, такие действия, как щелчок мышью на кнопке или закрытия окна.

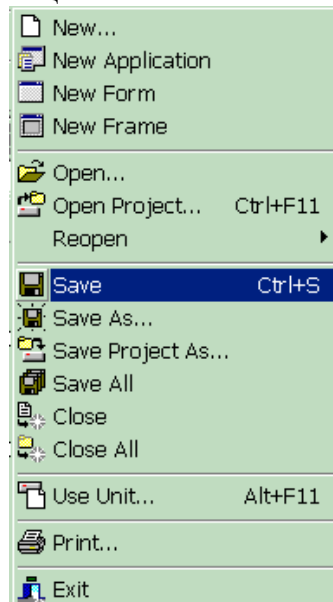


Рис. 3. Меню файл

По ходу работы система формирует в **окне редактора кода** текст программы на языке Паскаль, связанной с формой. После загрузки **Delphi** это окно скрыто за окном формы, и его

можно увидеть, щелкнув на кнопке **ToggleForm/Unit**  на панели инструментов.

Пользователь может дополнять текст программы самостоятельно или по предложению системы в момент размещения объектов на форме.

Создание и сохранение проекта и файла модуля

Каждый пункт меню содержит набор команд. На рис. 3 изображены пункты меню **File**.

Рассмотрим некоторые команды (файл проекта – комплекс форм, библиотек и программного кода, .dpr, команда **Project/ViewSource**; файл модуля объектно-ориентированный код, .pas, **F12**; в файле формы описываются компоненты и их свойства, .dfm)

Команда **New** открывает окно **NewItems**, при помощи которого можно создать новую форму, шаблон проекта формы, которые затем могут быть использованы.

Команда **NewForm** открывает окно новой формы и соответствующего ей модуля и добавляет их к активному проекту.

Команда **NewApplication** создает новый проект, состоящий из файла проекта project1.dpr, файла модуля unit1.pas и файла формы unit1.dfm. При этом IDE отображает окно проектировщика формы и редактора кода. Окно проектировщика является активным. После этого рекомендуется сохранить новый проект, присвоив ему имя.

Команда **OpenProject** открывает диалоговое окно **OpenProject**, в котором выбирается открываемый файл.

Команда **Save** сохраняет активные файлы под их именами. Если файл не был ранее сохранен и ему не было присвоено имя, **Delphi** открывает диалоговое окно **SaveAs**, в котором нужно указать имя файла.

Команда **SaveProjectAs** позволяет сохранить проект под другим именем и в случае необходимости в другом каталоге. После выбора этой команды появляется диалоговое окно для сохранения файла кода программы. По умолчанию файлу дается расширение pas, указанное в окне **Тип файла**. Далее следует ввести имя файла и щелкнуть на кнопке **OK**. Появится следующее диалоговое окно для сохранения файла проекта (расширение .dpr – **DelphiProject**). Надо ввести имя проекта и щелкнуть на кнопке **OK**.

Примечание. Для сохранения ранее созданного проекта достаточно выполнить команду **Save**.

Задание 1. Разработка проекта «Первый проект»

1. Создайте новый проект. Для этого выполните команду **File, NewApplication**.
2. Щелкните мышью на кнопке **Run** – выполните «пустую» программу, созданную в п.1. На экране исчезнут все вспомогательные окна, у формы пропадет координатная сетка.
3. Для возвращения в среду Delphi нажмите комбинацию клавиш ALT+F4.
4. Составьте программу, выполняющую следующие действия. После запуска программы на экране видно сообщение «НЕ ЗАЧЕТ», при нажатии на кнопку с надписью «ДА» выходит сообщение «СДАЛ»

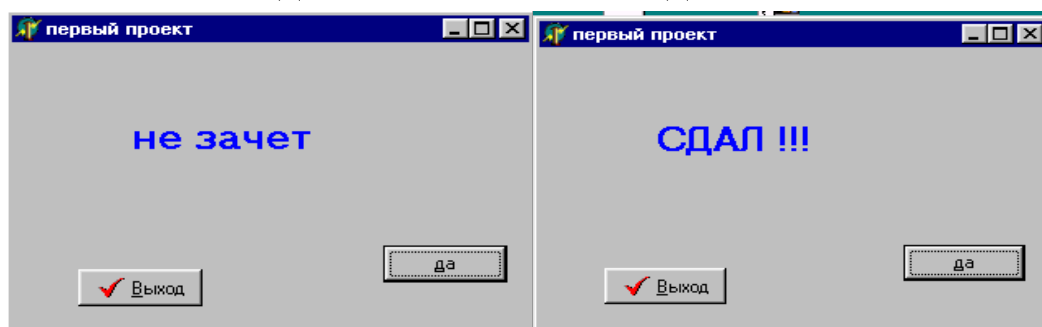


Рис. 4. Окно проекта

5. Создайте новый проект **File/New Application**.
6. Настройте компоненты согласно рис. 4.

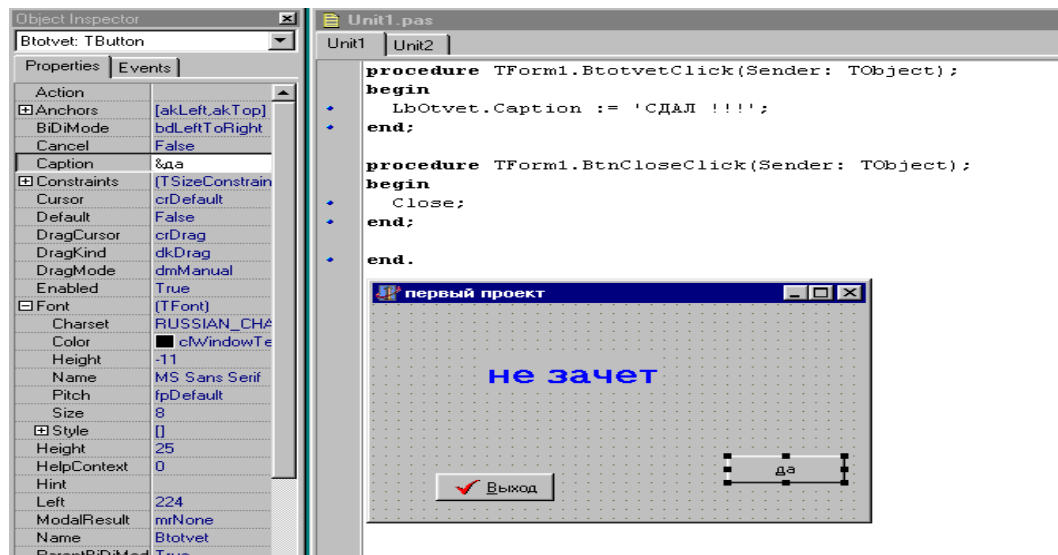


Рис. 5. Настройка вкладок «Свойства», «События»

7. Создайте обработчик событий (рис. 5):

//Обработчик события на кнопку Button1 (дважды нажмите на кнопку Button1)

```
begin
Label1.Caption := 'СДАЛ !!!';
end;
```

8. Создайте обработчик события на кнопку BitBtn1:

//Обработчик события на кнопку BitBtn1

```
begin
Close;
end;
```

9. Запустите программу на выполнение, нажав на панели инструментов .

10. Сохраните проект. Для этого выполните команду **File, SaveProjectas**, предварительно создав папку, в которую вы будите записывать свои проекты.

11. Закройте **Delphi**.

12. Загрузите **Delphi**.

13. Загрузите свой проект, выполнив команду **File, OpenProject** и указав правильно имя.

Объект – это структура содержащая переменные и подпрограммы. Программа в среде Windows, представляет собой в конце концов совокупность объектов, которые взаимодействуют друг с другом через эти переменные и подпрограммы.

Компонент – это специальным образом оформленный объект, который является самостоятельным элементом и может использоваться в различных программах, т.е. он не принадлежит, только одной программе. При работе с компонентами оперируем таким понятием как свойства.

Свойства – это атрибуты, которые составляют индивидуальность компоненты и помогают описать её. Свойства могут влиять, как на внешний вид компоненты так и на её поведение (**PROPERTIES**).

Событие – в визуальном программировании применяется программирование основанное на событиях. События – это реально происходящий отклик на одно или более внешних воздействий. Суть управляемого событием программирования, заключается в отслеживании таких действий, которые требуют реакции программы (**EVENTS**).

Для каждого компонента **Delphi** включая форму, существует свой список событий на который он может реагировать. Этот список доступен на вкладке **EVENTS** Инспектора Объектов.

Двойное нажатие на каком-либо объекте переносит нас в окно с процедурой, соответствующей этой кнопке.

Для реакции на событие в приложениях Delphi используются специальные подпрограммы, которые называются **обработчиками событий**. Создание **обработчика** происходит

автоматически при двойном щелчке левой клавишей мыши напротив названия события в **Инспекторе Объектов** (например: ONCLICK – событие на нажатие кнопки BUTTON).

Компонеты TLabel, TButton,

TLabel (Вкладка **Standart**)  - статический текст **TLabel** используется когда необходимо отображать на форму текст (**Properties/Caption**) который не может быть изменен пользователем.

Основные свойства компонентов

Caption – заголовок содержит текст, который будет размещаться в поле компоненты.

Alignment – управляет расположением текста, содержащегося в свойстве **Caption** (влево, вправо, по центру)

Height – высота компоненты в пикселях

Width – ширина компоненты в пикселях

Left – расстояние от левого края формы до компоненты в пикселях

Top – расстояние от верхнего края формы до компоненты в пикселях.

Visible – **TRUE** (компонента видна в программе), **FALSE** (невидна)

Font – задает размер шрифта, цвет, стиль шрифта

Помимо этого компонент **TLabel** может реагировать на событие **OnClick**, которое возникает тогда, когда пользователь однократно щелкает мышкой на компоненте.

TButton (Вкладка **Standart**)  – стандартная кнопка **Button** обычно используется в качестве управляющего элемента диалоговых панелей. Нажатие кнопки, как правило приводит к незамедлительному выполнению каких либо действий. Основные свойства кнопки соответствуют основным свойствам компонента **Label**, за исключением свойства **Aligment**.

Компоненты нумеруются в порядке возрастания, например **Button1, Button2,...ButtonN**

Это название по умолчанию, их можно переименовывать и желательно это делать таким образом, чтобы название компоненты отражало действие на событие, пример **Btotchet**.

TBitBtn (вкладка **Additional**)  - компонент представляет обычную кнопку, которая помимо текста может содержать графическое изображение. Графическое изображение загружается с помощью свойства **Glyphic** расширением ***.bmp**.

Рисунки находятся: **ProgramFiles\CommandFiles\ BorlandShared\Images\Button**

Свойство **Kind** – с помощью которого можно установить для кнопки некоторое предопределенное значение: **bkcancel, bkkall,...** при этом на кнопке появиться рисунок соответствующий значению, а также определяется поведение приложения при нажатии на эту кнопку. Например, при значении **bkclose** нажатие на кнопку вызывает закрытие приложения.

Задание 2. Разработка проекта «Приветствие»

Постановка задачи

Создайте программу, выполняющую следующие действия.

После запуска программы по щелчку мышью на кнопке «Приветствие» появляется сообщение «Первые успехи!». Для выхода из программы щелкнуть мышью на кнопке «Выход».

Пояснение. Необходимо повторить использование компонентов **TLabel** и **TButton** палитры компонентов **Standart** и обработка события **OnClick** – нажатия кнопки.

План разработки программы

1. Откройте новый проект.
2. Разместите на форме экземпляры компонентов: метку **TLabel** и две кнопки **TButton** (рис. 6).

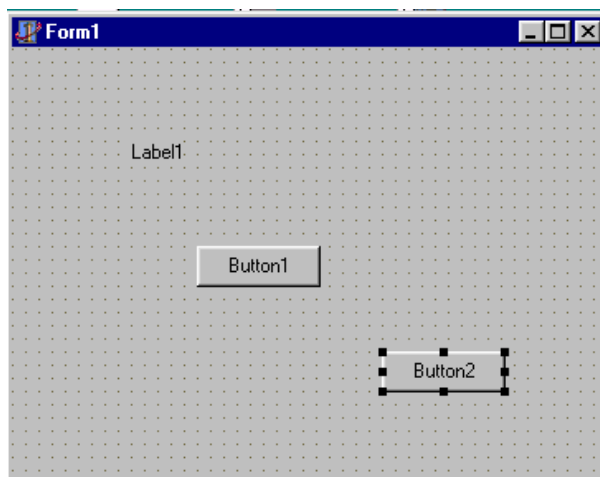


Рис. 6. Форма

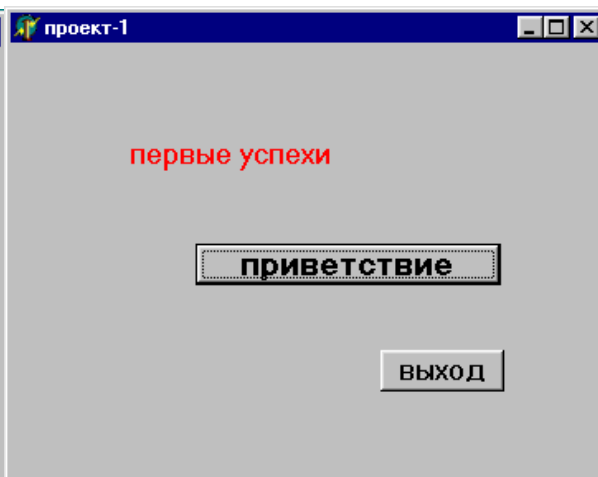


Рис. 7. Проект

1. Выполните следующие действия (рис. 7):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Button2	Properties	Caption	Установка имени кнопки: «Выход»
	Events	OnClick	Close

2. Сохраните код программы и проект под именами, например, `povtorenie.pas` и `povtorenie.dpr`.
 3. Запустите программу, затем закройте окно проекта, щелкнув на кнопке «Выход».
 4. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Установка имени формы: «Проект-1»
Button1	Properties	Caption	Установка имени кнопки: «Приветствие»
	Events	OnClick	Label1.Caption:='Первые успехи';

5. Сохраните, запустите и протестируйте проект.

Задание 3.

Постановка задачи

- Сделайте шрифт выводимой реплики «Первые успехи!» отличным от стандартного по виду, цвету и размеру.
- Подсказка.** В **ObjectInspector** дважды щелкните справа от названия свойства **Font**.
- Замените вид кнопки «Выход» на более привлекательный.
- Подсказка.** Для замены кнопки надо удалить существующую, а другую найти в палитре компонентов на вкладке **Additional**. Она самая левая, называется **BitBtn**. Затем измените ее вид с помощью свойства **Kind**.
- Необходимо сделать так, чтобы после нажатия кнопки «Приветствие» на экране появилось сообщение «Первые и не последние!».
- Подсказка.** Измените значение свойства **Caption** метки **Label1** при реакции кнопки **Button1** на событие **OnClick**.
- Запустите исполняемый файл `povtorenie.exe` в среде не **Delphi**, а **Windows**.

Листинг программы

```
Unit Unit1;
```

```
Interface
```

```
Uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons;
```

```
type
```

```
  TFrom1 = class(TForm)
```

```

Label1: TLabel;
Button1: TButton;
Button2: TButton;
Procedure Button1Click (Sender: TObject);
Private
    {Private declarations}
Public
    {Public declarations}
end;
var
Form1: TForm1;
Implementation
procedure TForm1.Button1Click (Sender: TObject);
begin
Label1.Caption:='Первые шаги'
End;
End.

```

Контрольные вопросы:

1. Назовите элементы интерфейса программы
2. Сформулируйте технологию ввода кода программы
3. Перечислите этапы алгоритма сохранения и запуска проекта
4. Сформулируйте назначение вкладок «Свойства», «События»
5. Назовите основные свойства компонентов «LABEL», «BUTTON»

Практическая работа №12

Тема: Применение простых типов данных

Цель работы: рассмотреть типы данных и использовать алгоритм применения изученных типов при решении задач.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

В **Delphi** все типы данных являются либо стандартными (имеющимися – присутствующими в **Delphi**) или пользовательскими, т.е. определяемые пользователем. Типы данных могут быть простыми или сложными. Переменная простого типа данных в каждый момент времени может содержать только какое-либо одно значение.

Целые или целочисленные типы данных (Переменные данного типа предназначены для хранения только целых чисел): **ShortInt** (-127...128), **SmallInt**, **Integer** (-32тыс ... 32 тыс), **LongInt**. Для целых типов данных определены три арифметических операции «+», «-», «*», а также операции целочисленного деления: **mod** (получение остатка от деления двух целых чисел), **div** (получение целой части от деления двух целых чисел). Кроме того над целыми числами определены операции отношения.

Вещественный тип (Предназначены для описания переменных способных хранить вещественные, т.е. дробные числа. В памяти компьютера вещественные числа хранятся вместе с точкой): **Real**, **Single**, **Double**, **Extended**, **Comp**. Над переменными вещественного типа возможно выполнение операций: «*», «+», «-», «/», а также операции отношения.

Денежный тип данных: Currency. Используется для описания переменных, предназначенных для хранения больших денежных величин с точностью до 4 цифр после запятой.

Вариантный тип данных: Variant. Пользоваться вариантным типом данных нужно осторожно. Обрабатываются данные вариантного типа очень долго и может произойти искажение данных. Это тип который совместим со всеми типами данных **Delphi**.

Символьный тип данных: Char, WideChar (Предназначен для хранения символов кодовой страницы ASCII). Переменной символьного типа значения можно присвоить используя графическое изображение символов либо его код. Код символа обозначается с помощью символа # (C:='C' или C:=#99, для ENTER #13). При работе с символами для них допускаются операции отношения, причем из двух символов, тот символ больше, чей код больше 'c'(99)<'d'(100). Для

преобразования кодов символов используются функции: CHR – преобразует целое число в символ с кодом равным этому числу; ORD – преобразует символ в число соответствующее кода символа.

Перечисляемый тип данных. Этот тип данных определяется, как упорядоченный набор имен заданных путем их перечисления через запятую. Перечисляемый тип данных всегда вводит новый тип данных и поэтому описывается в разделе **TYPE** (typetday=(Monday, Yuesday,Wednesday); vard: tday.

Интервальный тип данных – тип диапазон. Представляет подмножество значений, некоторого базового типа. Переменная интервального типа может принимать значения в интервале от минимального до максимального, нельзя вводить вещественный тип (например: TypeTSInt=5..1000; VarSint: TSInt;)

Задание 1. Разработка проекта «Вычисление функций»

1. Откройте новый проект.
2. Создайте форму следующего вида, разместив на ней следующие компоненты (рис 1.):

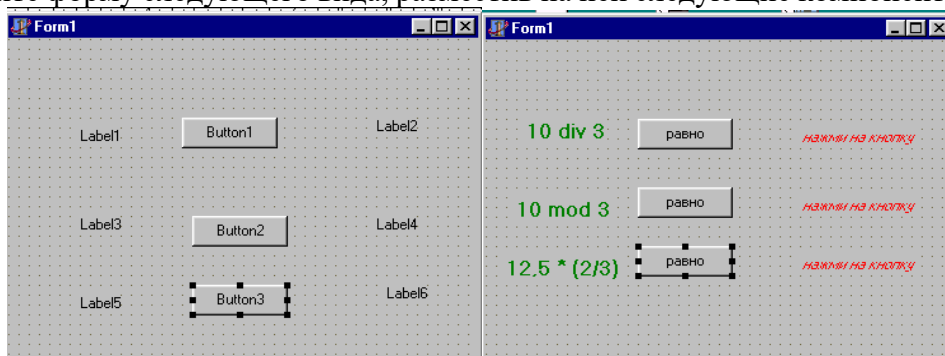


Рис. 1. Форма

Рис. 2. Проект

3. Выполните следующие действия (рис. 2):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Label1	Properties	Caption	Установка имени кнопки «10 div 3»
	Properties	Font	Установка стиля, цвета, размера
Button1	Properties	Caption	Установка имени кнопки «Равно»
	Events	OnClick	var vr: variant; consti=10; j=3; begin vr:=i div j; label2.Caption:=vr ; end;
Label2	Properties	Caption	Установка имени кнопки «Нажми кнопку»
	Properties	Font	Установка стиля, цвета, размера
Label3	Properties	Caption	Установка имени кнопки «10 mod 3»
	Properties	Font	Установка стиля, цвета, размера
Button2	Properties	Caption	Установка имени кнопки «Равно»
	Events	OnClick	var vr: variant; consti=10; j=3; begin vr:=i mod j; label4.Caption:=vr ; end;
Label4	Properties	Caption	Установка имени кнопки «Нажми кнопку»
	Properties	Font	Установка стиля, цвета, размера

Label5	Properties	Caption	Установка имени кнопки «15,2 *(2/3)»
	Properties	Font	Установка стиля, цвета, размера
Button3	Properties	Caption	Установка имени кнопки «Равно»
	Events	OnClick	<pre> var vr: variant; cr: currency; i: integer; j: real; begin j:=15.2; i:=2; cr:=j*(i/3); vr:=cr; label6.Caption:=vr; end; </pre>
Label6	Properties	Caption	Установка имени кнопки «Нажми кнопку»
	Properties	Font	Установка стиля, цвета, размера

4. Сохраните код программы и проект.

5. Запустите программу, (рис. 2) затем по очереди нажимайте на кнопки с названием «Нажмите кнопку»

Задание 2.

Постановка задачи

Пусть X - переменная типа INTEGER, а Y- переменная типа REAL. Составьте программу определения значения Y и вывода этого значения на экран компьютера, если Y определяется по формуле:

$$y = \frac{3 - 7,14}{8} + x * 9,67, \text{ где } x=3$$

План разработки программы

1. Создайте новый проект.
2. Разместите на форме экземпляры компонентов (рис. 3):

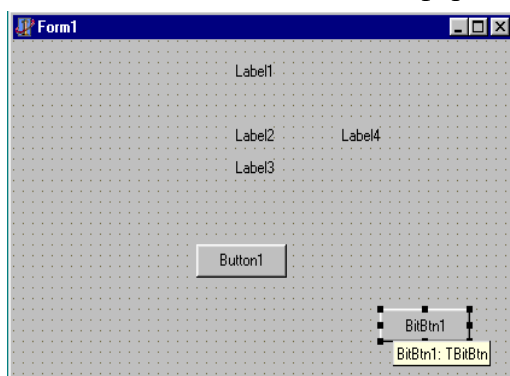


Рис. 3. Форма

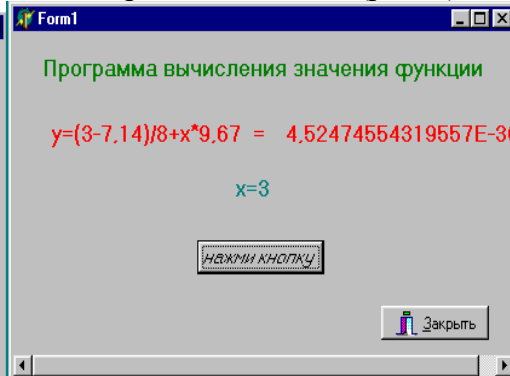


Рис. 4. Проект

3. Выполните следующие действия (рис. 4):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Label1	Properties	Caption	Имя кнопки «Программа вычисления значения функции»
		Font	12,Ж, красный
Label2	Properties	Caption	Имя кнопки « $y=(3-7,14)/8+x*9,67$ »
		Font	12,Ж, красный
		Visible	False
Label3	Properties	Caption	Имя кнопки « $x=3$ »
		Font	12,Ж, красный
Label4	Properties	Caption	

		Font	12,Ж, красный
<i>Button1</i>	Properties	Caption	Имя кнопки «Нажми кнопку»
		Font	12,Ж, красный
	Events	OnClick	Var X:integer; Y:real; Begin X:=3; Y:=(3-7.14)/8+x+9.67; Label4.Caption:=FloatToStr(y); Label2.Visible :=True; End;
<i>BitBtn1</i>	Properties	Kind	bkClose
		Caption	&Закреть

Контрольные вопросы:

1. Назначение целочисленных типов данных
2. Назначение вещественного типа данных
3. Назначение денежного типа данных
4. Назначение вариантного типа данных
5. Назначение символьного типа данных
6. Назначение интервального типа данных
7. Назначение перечисляемого типа данных

Практическая работа №13

Тема: Использование стандартных математических функций

Цель работы: рассмотреть стандартные математические функции и использовать алгоритм применения изученных функций при решении задач.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Наиболее часто используемые функции:

ABS(x) – модуль X

COS(x) – $\cos x$

SIN(x) – $\sin x$

ARCTAN(x) – $\arctan x$

EXP(x) – e^x

LN(x) – $\ln x$

SQR(x) – x^2

SQRT(x) – $\sqrt{x}$

ROUND(x) – округляет вещественное число x до ближайшего целого

RANDOM(x) – возвращает случайное число в диапазоне от 0 до x

ODD(x) – проверка четности числа x, если x нечетное результат **TRUE**.

Стандартные математические процедуры

INC(x) – увеличение числа x на единицу

DEC(x) – уменьшение числа x на единицу

RANDOMAZE – запуск генератора случайных чисел

Компонета TEdit (вкладка Standart)

TEdit – представляет собой редактирования, в которой пользователь может вводить с клавиатуры исходный текст.

Свойства

Основным свойством является **Text**, куда вводимый текст и помещается.

ReadOnly – только для чтения, свойство логического типа позволяет отключать режим редактирования, его параметры:

Задание 1. Разработка проекта «Вычисление функции»

Постановка задачи

Вычислите x^y (Используйте формулу $x^y = \exp(y \cdot \ln(x))$), не использовать «-» или «0»)

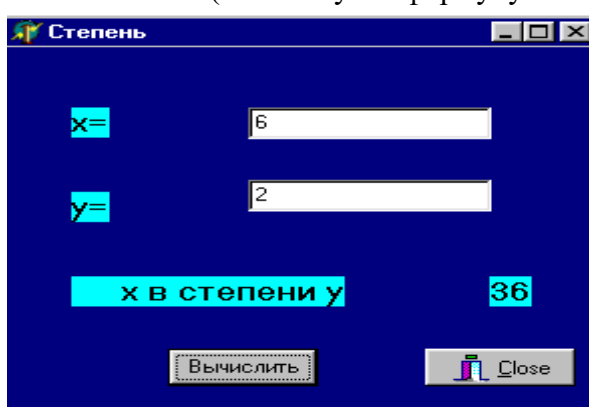


Рис. 1 Форма

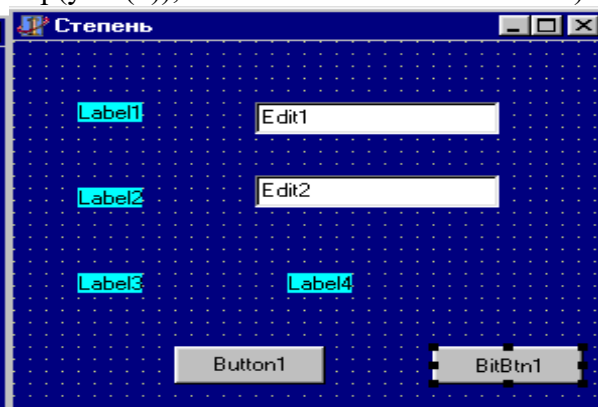


Рис. 2 Проект

План разработки программы

1. Откройте новый проект. Дайте форме название «Степень».
2. Разместите на форме экземпляры компонентов в соответствии с рис. 1.
3. Выполните следующие действия (рис. 2):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Степень
		Font	clHighlight
Label1	Properties	Caption	Имя кнопки «X=»

		Font	12,пж, синий
Label2	Properties	Caption	Имя кнопки «Y=»
		Font	12,пж, синий
Label3	Properties	Caption	Имя кнопки «x^y»
		Font	14, пж, красный
Label4	Properties	Caption	
Button1	Properties	Caption	Имя кнопки «Вычислить»
	Events	OnClick	var vr1,vr2: Real; begin vr1:=StrToFloat (Edit1.Text); vr2:= StrToFloat (Edit2.Text); vr2:=exp(vr2*ln(vr1)); label4.Caption:=FloatToStr(vr2); end;
BitBtn1	Properties	Caption	Имя кнопки «Очистить»
		Kind	bkClose

Задание 2. Составьте программу для вычисления $y = \sqrt[3]{x}$ ($y = x^{\frac{1}{3}} \rightarrow y = \exp(\frac{1}{3} * \ln(x))$)

Задание 3. С клавиатуры вводится значение переменных x1,x2,r1,r2 типа integer.

Определите значение $Y = \sqrt{\frac{R2 + R1}{\ln(R2)} + \frac{(x1 - x2)^2}{|x2|}}$

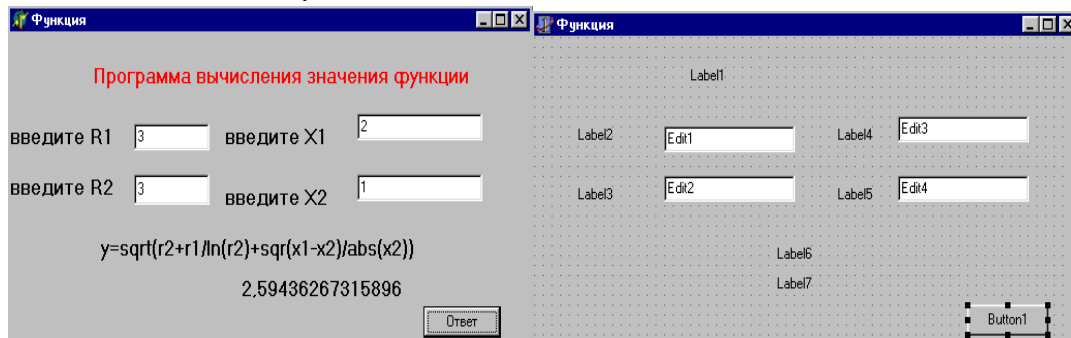


Рис. 3 Проект

Рис. 4 Форма

План разработки программы

1. Откройте новый проект. Дайте форме название «Функция».
2. Разместите на форме экземпляры компонентов в соответствии с рис. 3.
3. Выполните следующие действия (рис. 4):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Функция
		Font	clYellow
Label1	Properties	Caption	Имя кнопки «Программа вычисления значения функции»
		Font	14,пж, синий
Label2	Properties	Caption	Имя кнопки «введите R1»
		Font	12,пж, синий
Label3	Properties	Caption	Имя кнопки «введите R2»
		Font	12,пж, синий
Label4	Properties	Caption	Имя кнопки «введите x1»
		Font	12 пж, красный
Label5	Properties	Caption	Имя кнопки «введите x2»
		Font	12 пж, красный

Label6	Properties	Caption	Имя кнопки « $y=\sqrt{R2+r1/\ln(R2)+\sqrt{x1-x2}/\text{abs}(x2)}$ »
		Font	12 пж, красный
Label7	Properties	Caption	
		Font	12 пж, красный
Button1	Properties	Caption	Имя кнопки «Ответ»
	Events	OnClick	<pre> var x1,x2,R1,R2:integer; y:real; begin R1:=StrToInt(Edit1.Text); R2:=StrToInt(Edit2.Text); x1:=StrToInt(Edit3.Text); x2:=StrToInt(Edit4.Text); y:=sqrt(R2+r1/ln(R2)+sqrt(x1- x2)/abs(x2)); Label7.Caption:=FloatToStr(y); end; </pre>
Edit1	Properties	Text	
Edit2	Properties	Text	
Edit3	Properties	Text	
Edit4	Properties	Text	

Задание 4. С клавиатуры вводится значение переменных $x1, x2, r1, r2$ типа integer.

Определите значение $y = \ln(|\arctg(x1)|) * (\frac{x2-r2}{r1})^2$. Полученное значение необходимо вывести на экран.

Задание 5. С клавиатуры вводится значение переменных $x1, x2, r1, r2$ типа integer.

Определите значение $y = \frac{\sin(x) * x2 + r1 - 4}{\ln(r2) - (x2)^2}$. Полученное значение необходимо вывести на экран.

Задание 6. Создайте программу, выполняющую следующие действия. После запуска программы пользователь вводит два числа, в прямоугольники с мигающим текстовым курсором и нажимает кнопку «Умножить». На экране высвечивается ответ при умножении двух чисел (рис. 5).

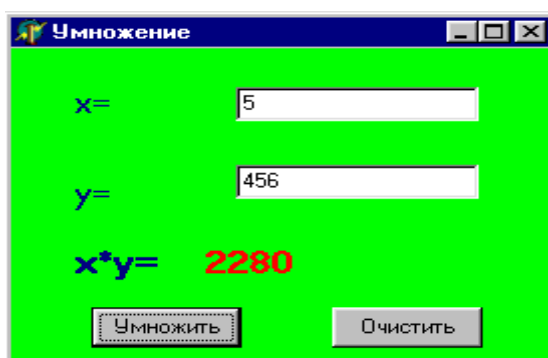


Рис. 5. Проект

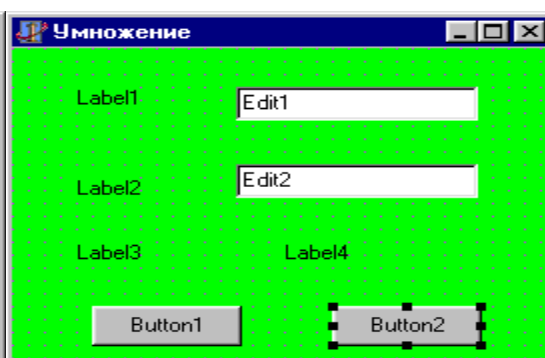


Рис. 6. Форма

План разработки программы

1. Откройте новый проект. Дайте форме название «Умножение».
2. Разместите на форме экземпляры компонентов в соответствии с рис. 5.
3. Выполните следующие действия (рис. 6):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Умножение
		Font	clLime (салатовый)
Label1	Properties	Caption	Имя кнопки «X=»

		Font	12, пж, синий
Label2	Properties	Caption	Имя кнопки «Y=»
		Font	12, пж, синий
Label3	Properties	Caption	Имя кнопки «X*Y»
		Font	14, пж, красный
Label4	Properties	Caption	
		Font	14, пж, красный
Button1	Properties	Caption	Имя кнопки «Умножить»
	Events	OnClick	var vr1,vr2:variant; begin vr1:=Edit1.Text; vr2:=Edit2.Text; vr2:=vr1*vr2; label4.Caption:=vr2; Edit1.ReadOnly:=true; Edit2.ReadOnly:=true; end;
Button2	Properties	Caption	Имя кнопки «Очистить»
	Events	OnClick	begin Edit1.Text:=""; Edit2.Text:=""; Edit1.ReadOnly:=true; Edit2.ReadOnly:=true; Label4.Caption:=""; End;

4. К предыдущему заданию добавить операции сложения, вычитания, деления двух чисел.

Контрольные вопросы:

1. Основные стандартные математические функции
2. Основные свойства компоненты «EDIT»

Практическая работа №14

Тема: Использование оператора условия

Цель работы: отработать навыки работы с условной конструкцией на языке ObjectPascal.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Процедуры и функции преобразования строковых типов

STR (IBR,ST) – процедура преобразует числовое значение целого или вещественного типа переменной IBR в строковый эквивалент результат помещается в строковую переменную ST.

VAL (ST,IBR,CODE) – процедура преобразует значение ST в величину целого или вещественного типа данных. Результат помещается в переменную IBR. CODE – целочисленная переменная, если при преобразовании ошибки не обнаружены, тогда значение переменной CODE равно нулю. Если ошибка обнаружена CODE будет содержать номер позиции первого ошибочного символа.

IntToStr (Int) – функция преобразует целочисленное значение в строковый компонент. Возвращает строку соответствующую целому числу Int.

StrToInt (St) – функция возвращает целое число полученное из строки St.

FloatToStr (Re) – функция преобразования вещественного числа в строковую переменную. Возвращает строку соответствующую вещественному числу Re.

StrToFloat (St) – функция возвращает вещественное число полученное из строки St.

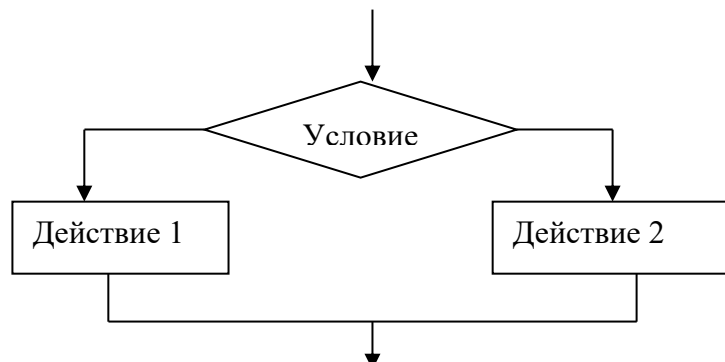
Составной оператор

Структурные операторы – это операторы которые могут состоять из нескольких операторов:

- Составной оператор – представляет собой группу из произвольного числа операторов отделенных друг от друга (;) и ограниченную операторными скобками **BEGIN – END**, между этими словами указываются произвольный список операторов. Составной оператор воспринимается, как единое целое и может находиться в любом месте программы, где допускается оператор.

Условный оператор

- Относится к составным, предназначен, для выбора к исполнению одного из возможных действий операторов, в зависимости от некоторого условия.



Оператор позволяет проверить некоторое условие и в зависимости от результата проверки выполнить то или иное действие.

Здесь «условие» - выражение логического типа. В качестве операторов 1 и 2 может выступать любой оператор **Delphi**. Оператор работает следующим образом: в начале определяется результат условия (**TRUE** или **FALSE**). Если условие не верно, выполняется оператор 2. в любом случае выполняется только один из операторов. После этого управление передается следующему за условным оператором оператору.

Формат описания оператора: IF<условие>THEN<оператор 1>ELSE<оператор 2>

Краткий формат: IF<условие>THEN<оператор 1>

Если в качестве оператора 1 или оператора 2 необходимо выполнить несколько операторов, то эти операторы нужно заключить в операторные скобки.

В этом случае формат оператора будет:

IF<условие>THEN

```

Begin
    <список операторов>
end
ELSE
begin
    <список операторов>;
end;

```

Задание 1. Разработка проекта «Диалог»

Постановка задачи

Создайте программу, выполняющую следующие действия:

- После запуска программы пользователь вводит свое имя, например Александр, в прямоугольник с мигающим текстовым курсором и нажимает клавишу ENTER.
- Появляется вопрос: «Александр, ты любишь читать?». Если пользователь щелкает на кнопке «Да», то появляется реплика «Молодец!», если на кнопке «Нет», то реплика «Почему же? Надо читать».
- Для выхода из программы необходимо щелкнуть на кнопке «Выход».

Пояснение. Новым в этой работе является использование строки ввода **Edit** (вкладка палитры компонентов **Standard**) и обработки события **OnKeyPress** – нажатия клавиши

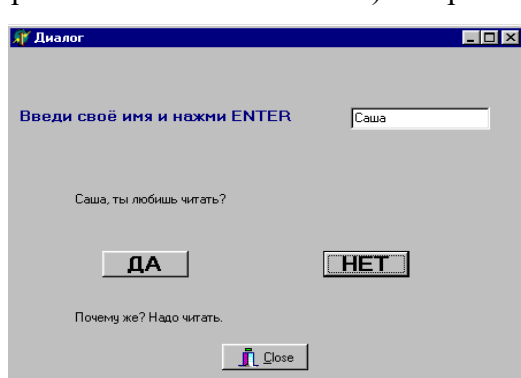


Рис. 1. Проект

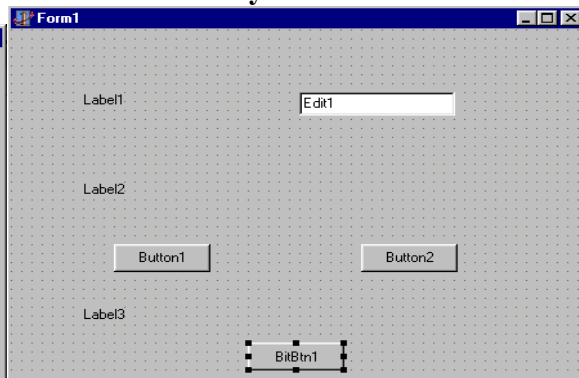


Рис. 2. Форма

План разработки программы

1. Откройте новый проект. Дайте форме название «Диалог».
2. Разместите на форме экземпляры компонентов в соответствии с рис. 2.
3. Выполните следующие действия (рис. 1):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
BitBtn1	Properties	Caption	Установка имени кнопки «Выход»
		Kind	bkClose
Label1	Properties	Caption	Ввод надписи «Введи свое имя и нажми ENTER»
Edit1	Events	OnKeyPress	<i>If key=#13 then</i> Begin Label2.Caption := Edit1.Text + ', ты любишь читать?'; End;
Button1	Properties	Caption	Установка имени кнопки «Да»
	Events	OnClick	Label3.Caption := 'Молодец'
Button2	Properties	Caption	Установка имени кнопки «Нет»
	Events	OnClick	Label3.Caption := 'Почему же? Надо читать'

4. Сохраните код программы и проект под именами, например, pr-2.pas и pr.dpr.
5. Запустите программу, затем закрыть окно проекта, щелкнув на кнопке «Выход».

6. Сохраните проект, запустите проект на выполнение.

7. Сделайте кнопки «да» и «нет» доступными только после ввода имени и нажатия клавиши enter.

Подсказка. Значение свойства **Enabled** кнопок «Да» и «Нет» установить равным **False**, а в процедуру **Edit1KeyPress** включить код:

```
Button1.Enabled := true;
```

```
Button2.Enabled := true;
```

8. Удалите имена объектов **Edit1**, **Label2**, **Label3** для обеспечения возможности повторить диалог.

Подсказка. Разместите на форме еще одну кнопку **BitBtn**. Установите значение **bkRetry** свойства **Kind** и значение «Повторить» свойства **Caption**. Введите программно по нажатию кнопки «Повторить» пустые строки вместо надписей и имен кнопок:

```
Label2.Caption := ''
```

```
Label3.Caption := ''
```

```
Edit1.Text := '';
```

9. Сделайте так, чтобы при повторении диалога строка ввода была снова активной.

Подсказка. Введите команду **Form1.ActiveControl := Edit1**

Листинг программы

```
Unit Unit2;
```

```
Interface
```

```
Uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
Buttons, StdCtrls;
```

```
Type
```

```
TForm1 = class (TForm)
```

```
Edit1: Tedit;
```

```
Button1: TButton;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
Button2: TButton;
```

```
Label3: TLabel;
```

```
BitBtn1: TbitBtn;
```

```
BitBtn2: TbitBtn;
```

```
Procedure Edit1KeyPress (Sender: TObject; var Key: Char);
```

```
Pocedure Button1Click (Sender: TObject);
```

```
Pocedure Button2Click (Sender: TObject);
```

```
Pocedure ButBtn2Click (Sender: TObject);
```

```
Private
```

```
{private declarations}
```

```
public
```

```
{public declarations}
```

```
end;
```

```
var
```

```
Form1: TForm;
```

```
Implementatuon
```

```
{$R *.DFM}
```

```
procedure TForm1.Edit1KeyPress (Sender: TObject; var Key: Char);
```

```
begin
```

```
if key=#13 then begin
```

```
label2.Caption := Edit1.Text+ ' , тылюбишьчитать?'
```

```
Button1.Enabled := true;
```

```
Button2.Enabled := true;
```

```
End;
```

```

End;
Procedure TForm1.Button1Click (Sender: Tobject);
Begin
Label3.Caption := 'Молодец !';
End;
Procedure TForm1.Button2Click (Sender: Tobject);
Begin
Label3.Caption := 'Почему же? Надочитать.';
End;
Procedure TForm1.BitBtn2Click (Sender: Tobject);
Begin
Edit1.Text := ' ';
Label2.Caption := '';
Label3.Caption := '';
Button1.Enabled := false;
Button2.Enabled := false;
Form1.ActiveControl := Edit1;
End;
End.

```

Задание 3. Определите значение функции y в зависимости от значения x (рис. 3)

$$y = \begin{cases} (x-3)^2, & x \leq -1 \\ -x+1, & -1 < x \leq 0 \\ \ln(x), & x > 0 \end{cases}$$

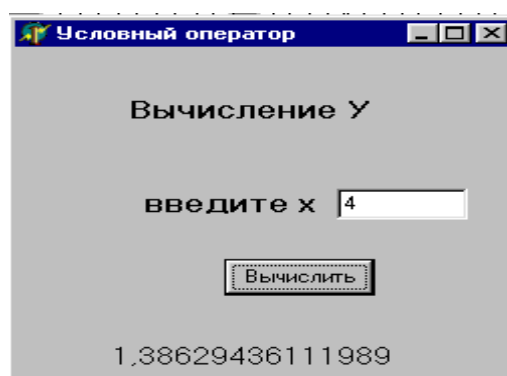


Рис. 3. Проект

План разработки программы

1. Откройте новый проект. Дайте форме название «Условный оператор».
2. Разместите на форме экземпляры компонентов в соответствии с рис. 4.

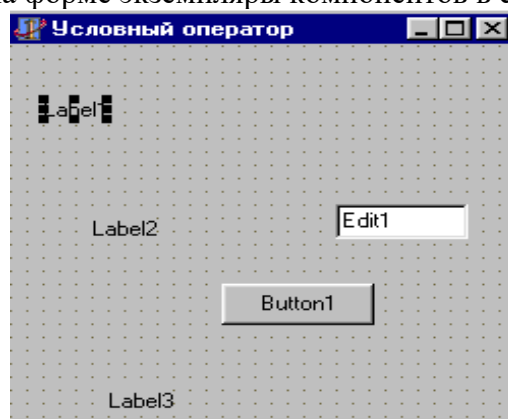


Рис. 4 Форма

3. Выполните следующие действия (рис. 3):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
-------------------	--	------------------------------	----------

Form1	Properties	Caption	Условный оператор
		Font	clYellow
Label1	Properties	Caption	Имя кнопки «Программа вычисления значения функции»
		Font	14,пж, синий
Label2	Properties	Caption	Имя кнопки «Значение Y»
		Font	12,пж, синий
Button1	Properties	Caption	Имя кнопки «Вычислить»
		OnClick	<pre> var x:integer; y:real; begin x:=StrToInt(Edit1.Text); if (x>-1) and (x<0) then y:=-x+1; if x<=1 then y:=sqr(x-3) else y:=ln(x); Label3.Caption:=FloatToStr(y); end;</pre>
Edit1	Properties	Text	

Задание 4. Определите значение функции у в зависимости от значения х

$$y = \begin{cases} x^4 - x^2 - 5, & x \leq 0 \\ -x^3, & 0 < x \leq 2 \\ \exp(x), & x > 2 \end{cases}$$

Контрольные вопросы:

1. Формат записи составного оператора
2. Формат записи условного оператора

Практическая работа №15

Тема: Применение оператора варианта

Цель работы: отработать навыки работы с формой записи оператора варианта на языке ObjectPascal.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Форма записи:

```
CASEвыражение OF  
Константа 1: оператор 1;  
Константа 2: оператор 2;  
.....  
Константа N: оператор N;  
ELSEоператор;  
END;
```

Тип выражения может быть одним из стандартных типов: целым числом, символьным типом, кроме действительного. Результат выражения будут поочередно сравниваться с каждым из значений в списках и при первом совпадении будет выполнено соответствующее действие, а все оставшиеся действия будут пропущены. В случае, когда ни одного совпадения результата выражения с заданными значениями не произошло, выполняется действие, указанное за словом **else** (если оно имеется), или не выполняется ничего, если слово **else** внутри оператора отсутствует.

Компонент **TListBox** «Список»



TListBox – компонента содержит список элементов, которые могут быть выбраны при помощи клавиатуры или мыши. Первоначально компонент **TListBox**, размещенный на форме, изображается в виде пустого квадрата. Его размеры можно настроить протягиванием мыши. Список может иметь несколько столбцов. Это не означает, что каждый столбец представляет собой отдельный список, просто при заполнении видимой части списка строками донизу, очередная строка отображается в следующем столбце. В остальном, работать такой список будет так же, как обычный список.

Число столбцов задается в свойстве **Columns**. Если оно больше 0, то на каждую колонку отводится часть общей ширины списка (значение свойства **Width**, деленное на число столбцов). Список может принадлежать к одному из трех возможных типов (свойство **Style**).

Значение	Стиль списка
lbStandard	Стандартный список (стиль по умолчанию)
lbOwnerDrawFixed	Каждый элемент списка имеет фиксированную высоту, но способ его отображения определяется программистом
lbOwnerDrawVariable	Кроме способа рисования в тексте программы необходимо явно задавать размер каждого элемента списка (что позволяет создавать списки с элементами разных размеров)

Свойства: **Item**, **ItemIndex**, **Sorted**, **Style**. В **TListBox** всех строки списка отображаются одновременно при этом для компонента можно установить режим выбора нескольких элементов списка одновременно. Это устанавливается с помощью свойства **MultiSelect** логического типа. При значении **True** групповой выбор разрешен.

Существует 2 способа группового выбора:

1. множественный выбор – выбор элементов осуществляется только с помощью мыши;
2. расширенный выбор – когда при выборе группы используются клавиши **CTRL/SHIFT** для выбора несложных и смежных значений. Расширенный выбор возможен, если свойство **Extended Select = True**.

Для списка с групповым выбором программа может получить информацию о количестве выбранных элементов из свойства **SelCount** и определить какие элементы выбраны, используя свойство – массив **Selected**. Это массив логических значений, содержащих такое же количество элементов, что и список.

Значение элемента свойства

Selected [I]=true, если i-й элемент выбран. Оба компонента могут обрабатывать событие **OnClick**, которое возникает тогда когда происходит выбор какого-либо элемента списка.

True– изменить содержимое невозможно.

False– изменить содержимое возможно.

Задание 1. Разработка проекта «Справочник»

Постановка задачи

Создайте программу, выполняющую следующие действия:

- После запуска программы пользователь выбирает с помощью мыши или стрелок название цвета и нажимает ENTER. На экране появляется название цвета в формате RGB.

Программа заканчивает свою работу по нажатию клавиши «Выход».

Пояснение. Новым в работе является использование компоненты **TListBox** (список) (вкладка палитры компонентов **Standart**, работа со встроенным редактором для ввода информации и алгоритм выбора (оператор **Case**)).

План разработки программы

1. Откройте новый проект. Дайте форме название «Справочник».

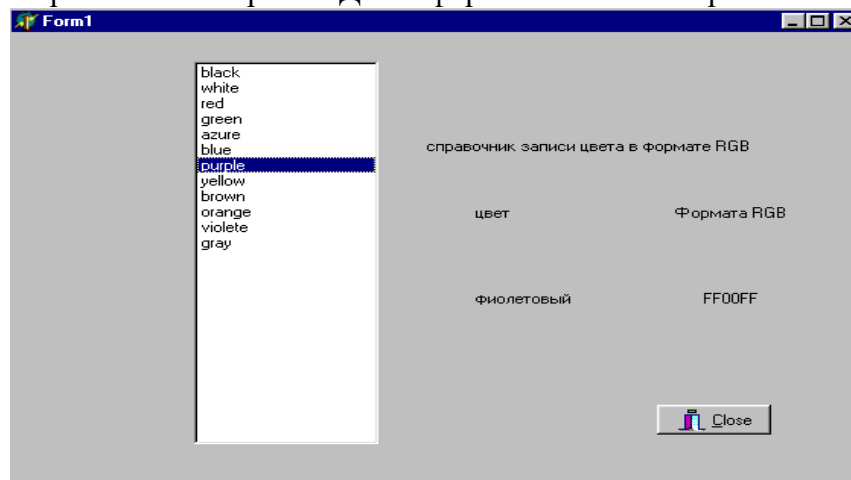


Рис. 1. Проект

2. Разместите на форме экземпляры компонентов в соответствии с рис. 2:

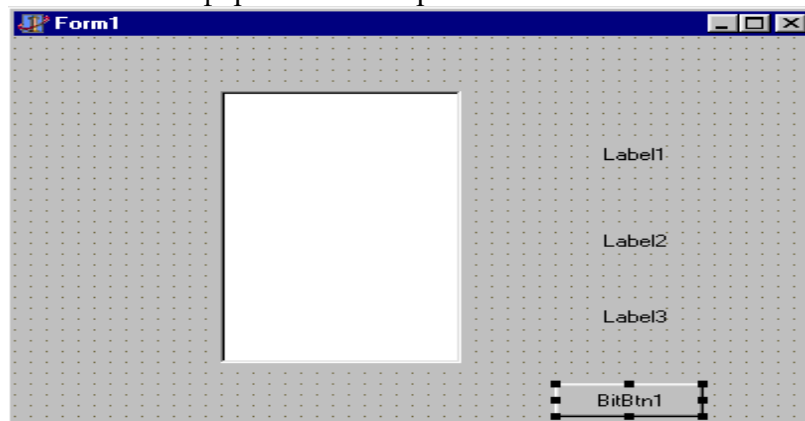


Рис. 2 Форма

3. Выполните следующие действия (рис. 1):

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
BitBtn1	Properties	Caption	Установка имени кнопки: «Выход»
		Kind	bk Close
Label1	Properties	Caption	Ввод надписи «Справочник записи цвета в формате RGB»
Label2	Properties	Caption	Ввод надписи «Цвет Формат RGB»

Label3	Properties	Caption	Восстановление названия и кода цвета
---------------	-------------------	----------------	--------------------------------------

4. Сохраните код программы и проект.
5. Запустите программу, затем закрыть окно проекта, щелкнув на кнопке «Выход».



6. Выделите объект **TListBox**, найдите свойство **Items**, щелкните на кнопке с тремя точками, расположенной справа от него. В появившемся окне встроенного редактора ввести названия цветов, каждый на новой строке.

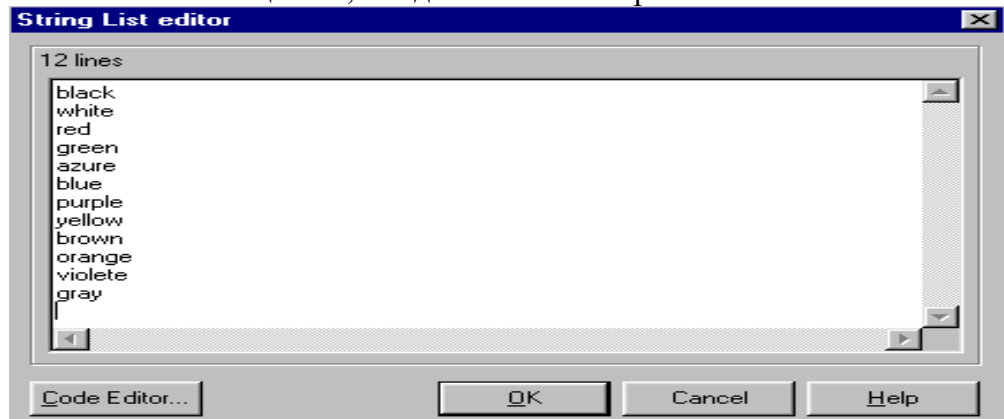


Рис. 3 Редактор

7. Сохраните набранный текст под именем color.txt. Для этого нажать правую кнопку мыши и выбрать сохранить режим **Save**. Для выхода из встроенного редактора щелкните на кнопке **OK**.
8. Выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
ListBox1	Events	OnKeyPress	<pre> If key=#13 then Case ListBox1.ItemIndex of 0: Label3.Caption:='черный 000000'; 1: Label3.Caption:='белый FFFFFFFF'; 2: Label3.Caption:='красный FF0000'; 3: Label3.Caption:='зеленый 00FF00'; 4: Label3.Caption:='бирюзовый 00FFFF'; 5: Label3.Caption:='синий 0000FF'; 6: Label3.Caption:='фиолетовый FF00FF'; 7: Label3.Caption:='желтыйFFFF00'; 8: Label3.Caption:='коричневый 996633'; 9: Label3.Caption:='оранжевыйFF8000'; 10: Label3.Caption:='лиловый 8000FF'; 11: Label3.Caption:='серый A0A0A0'; end; </pre>

9. Сохраните проект окончательно, запустите проект на выполнение.
10. Измените шрифты, цвет экрана и букв.

Подсказка. Возможно, придется в коде программы подкорректировать количество пробелов между названием цвета и его кодом.

11. Измените код программы таким образом, чтобы при установке курсора мыши в поле **TListBox** появлялась подсказка о том, что надо сделать.

Подсказка. Воспользуйтесь свойствами **Hint**(текст сообщения), **Showhint**(показывать ли сообщение) объекта **TListBox1**.

12. Внесите изменения в программу, чтобы для надписей «Цвет» и «Формат RGB» использовались два отдельных объекта **Label**.

13. Ответьте на вопрос: как изменится программа, если исходные данные о таблице цветов будут заданы в массиве?

Листинг программы

```
unit Pr3;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons;
type
TForm1 = class(TForm)
  ListBox1: TListBox;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  BitBtn1: TBitBtn;
procedure ListBox1KeyPress(Sender: TObject; var Key: Char);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.ListBox1KeyPress(Sender: TObject; var Key: Char);
begin
  If key=#13 then
  Case ListBox1.ItemIndex of
  0: Label3.Caption:='черный    000000';
  1: Label3.Caption:='белый      FFFFFFFF';
  2: Label3.Caption:='красный    FF0000';
  3: Label3.Caption:='зеленый   00FF00';
  4: Label3.Caption:='бирюзовый  00FFFF';
  5: Label3.Caption:='синий      0000FF';
  6: Label3.Caption:='фиолетовый FF00FF';
  7: Label3.Caption:='желтый     FFFF00';
  8: Label3.Caption:='коричневый 996633';
  9: Label3.Caption:='оранжевый  FF8000';
  10: Label3.Caption:='лиловый    8000FF';
  11: Label3.Caption:='серыйA0A0A0';
  end;
end;
end.
```

Задание 6. Разработка проекта «Выбор подарка на Рождество»

Постановка задачи

Создайте программу, выполняющую следующие действия:

- После запуска программы пользователь выбирает с помощью мыши название подарков для его и его друзей. Все эти выбранные данные переходят во второе окно. С помощью клавиши добавить можно добавить название подарков, которых нет в списке, а

с помощью другой кнопки удалить если вы передумали название подарка. Программа заканчивает свою работу по нажатию клавиши «Выход».

План разработки программы

1. Откройте новый проект. Дайте форме название «Подарок к Рождеству».

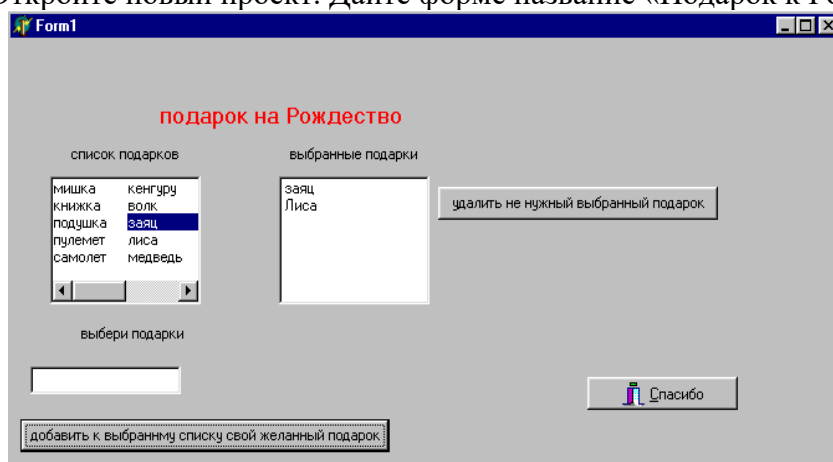


Рис. 4 Проект

2. Разместите на форме экземпляры компонентов в соответствии с рис. 5:

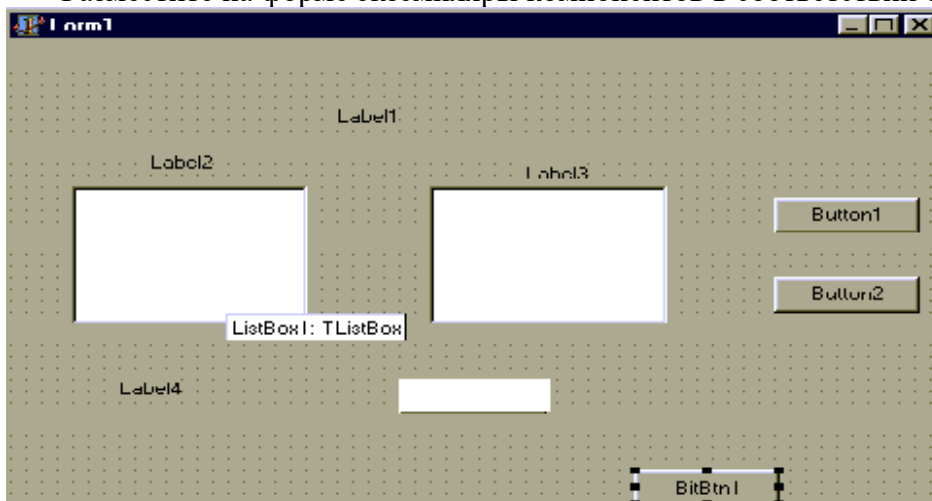


Рис. 5 Форма

3. Выполните следующие действия (рис. 4):

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
BitBtn1	Properties	Caption	Установка имени кнопки: «Спасибо»
		Kind	bk Close
Label1	Properties	Caption	Ввод надписи «подарок на Рождество»
Label2	Properties	Caption	Ввод надписи «список подарков»
Label3	Properties	Caption	Ввод надписи «выбранные подарки»
Label3	Properties	Caption	Ввод надписи «выбери подарки»
Edit1	Properties	Text	
Button1	Properties	Caption	«Удалить не нужный выбранный подарок»
	Events	OnClick	Listbox2.Items.Delete(Listbox2.ItemIndex);
Button2	Properties	Caption	«Добавить к выбранному списку свой желанный подарок»
	Events	OnClick	if Edit1.Text = " Then Exit else Listbox2.Items.Add(Edit1.Text);

ListBox1	Properties	Items	мишка книжка подушка пулемет самолет кенгуру волк заяц лиса медведь
	Events	OnClick	ListBox2.Items.Add(ListBox1.Items.Strings[ListBox1.ItemIndex]);

4. Сохраните код программы и проект.
5. Запустите программу, затем закройте окно проекта, щелкнув на кнопке «Спасибо».
6. Сохраните проект окончательно, запустите и протестируйте проект.

Контрольные вопросы:

1. Форма записи оператора варианта
2. Свойства компоненты «TListBox»

Практическая работа №16

Тема: Создание базовых элементов интерфейса Windows: компоненты TComboBox «Поле со списком», TCheckBox «Флажок»»

Цель работы: освоить приемы работы с компонентами TComboBox, TCheckBox и ее свойствами.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Компонент TComboBox «Поле со списком»

ComboBox  – компонент представляет собой комбинированный список – это интерфейсный элемент включающий в себя список строк и статический текст или строку редактирования. В каждый момент времени отображает только один элемент из списка строк, который помещается в свойства **Text**. Компонента **ComboBox** позволяет пользователю выбрать отображаемый элемент из списка. Список строк задается с помощью свойства **Items**. Свойство **Items** обладает методами **Add**, **Delete**, **Insert** (для редактирования списка строк из программного кода). Кроме этого в окне **Инспектора Объектов** можно вызвать редактор списка строк компоненты. Определить какой элемент списка выбран можно обратившись к свойству **ItemIndex**. Для сортировки строк списка по алфавиту необходимо установить значение свойства **Sorted=True**. С помощью свойства **Style** можно изменить стиль компоненты **ComboBox**.

Компонент Поле со списком может работать в трех разных режимах, определяемых значением свойства **Style**.

Значение	Механизм работы списка
csDropDown	В присоединенном поле можно указывать значения, отсутствующие в списке. Свойство MaxLength определяет максимально допустимое число символов, которое можно ввести в это поле (значение 0 указывает на отсутствие ограничений). Текст, введенный пользователем, доступен через свойство Text . Список раскрывающийся
csDropDownList	Допустим только выбор значений, уже имеющихся в списке. Список раскрывающийся
csSimple	Отличается от стиля csDropDown только тем, что список не является раскрывающимся

Вводимый текст может автоматически преобразовываться к верхнему регистру (если свойство **CharCase** имеет значение **ecUpperCase**), к нижнему (**ecLowerCase**) или никак не преобразовываться (**ecNormal**, по умолчанию).

Максимальное число элементов, одновременно отображаемых в видимой части списка, задается в свойстве **DropDownCount**. Чтобы открыть список из программы, свойству **DroppedDown** (Раскрыт) надо присвоить значение **TRUE**.

Понятия «выделенная строка» в раскрывающемся списке нет. В нем имеется только текущая выбранная строка (ее номер в списке хранится в свойстве **ItemIndex**). Соответственно, нельзя и выделять строки списка. Единственный метод, связанный с выделением данных, - это процедура **SelectAll**, которая выделяет весь текст, введенный пользователем в присоединенное поле. При работе раскрывающегося списка наиболее важными являются представленные события.

Название	Условия генерации
onChange	Пользователь изменил текст в присоединенном поле
onDropDown	Список раскрывается. Это событие необходимо обрабатывать, если содержимое списка может меняться во время работы программы. Тогда в обработчике этого

	события можно заново сформировать содержимое списка (свойство Items)
--	--

Компонент TCheckBox «Флажок»

Данный компонент  используется для фиксации включенного или выключенного состояния (одного из них).

После размещения компонента Флажок на форме подпись к этому элементу управления можно задать в свойстве **Caption**. Расположение этой подписи определяется свойством **Alignment**: значение **taLeftJustify** – слева. Главное свойство флажка называется **Checked**. Оно доступно для изменения и на этапе проектирования, и на этапе работы программы. Это свойство принимает значение **TRUE**, если флажок включен, и **FALSE**, если он сброшен.

Некоторые флажки могут находиться в третьем, «частично установленном» состоянии, когда признак установки флажка отображается приглушенным цветом. Такая возможность нужна, когда требуется сообщить пользователю о неполном соответствии указанному статусу (например, в ходе установки программ таким образом сообщается, что для установки выбраны не все доступные средства).

Если для свойства **AllowGrayed** задано значение **TRUE**, то флажок при последовательных щелчках на нем будет поочередно принимать состояния «сброшен», «установлен частично», «установлен». Определить текущее состояние или задать новое из числа доступных можно, проверив или изменив свойство **State**. Чтобы реагировать на изменение состояния флажка, надо создать обработчик события **onClick** (при щелчке).

Значение	Состояние флажка
cbUnchecked	Сброшен
cbGrayed	Установлен частично
cbChecked	Установлен

Стандартные панели сообщений

Панели сообщений – это специальные окна диалоговых панелей, предназначенных для выдачи коротких сообщений на экран ПК.

Панель сообщений вызывается на экран путем вызова соответствующей процедуры или функции.

Функция **MessageDlg (Msg,Dlgtype, Abutton,HelpCtx)** – обеспечивает выдачу на экран сообщения **Msg**. Параметр **DlgType** задает тип панели сообщения (возможны значения: **mtWarning** – внимание, **mtError** – ошибка, **mtInformation** – информационное сообщение, **mtConfirmation** – уточнение или вопрос). Параметр **Abuttons** позволяет указать какие кнопки должны присутствовать в панели сообщений. Например **[mbYes]** – должна присутствовать в панели сообщения **<Yes>**, **[mbOk, mbCancel]** –отображается две кнопки **<Ok>** и **<Cancel>**. Функция возвращает значение содержащее информацию, о том какая из кнопок была нажата на панели сообщений (например: **MessageDlg –mrOk, mrYes,mrCancel**).

Параметр **HelpCtx** – подключает к проекту файл-справки. Используется при наличии файла-справки в проекте, содержит идентификационный номер контекста справочного пособия. Если помощь не используется **HelpCtx=0**.

// Обработчик события **Button1**

begin

messageDlg ('Сообщение',mtConfirmation, [mbYes,mbCancel],0);

end;

Вывести сообщение в несколько строк:

Begin

MessageDlg ('Сообщение'+#10#13+'aaa', mtConfirmation,[mbYes,mbCancel],0);

Назначение - отображает диалоговое окно сообщений в центре экрана.

Вызов **MessageDlg** отображает диалоговое окно и ожидает ответа пользователя. Сообщение в окне задается параметром функции **Msg**.

Вид отображаемого окна задается параметром **AType**. Возможные значения этого параметра:

Значение	Описание
mtWarning	Окно замечаний, содержащее желтый восклицательный знак.
mtError	Окно ошибок, содержащее красный стоп-сигнал.
mtInformation	Информационное окно, содержащее голубой символ "i".
mtConfirmation	Окно подтверждения, содержащее зеленый вопросительный знак.
mtCustom	Заказное окно без рисунка. Заголовок соответствует имени выполняемого файла приложения.

Параметр **AButtons** определяет, какие кнопки будут присутствовать в окне. Тип **TMsgDlgBtns** параметра **AButtons** является множеством, которое включает различные кнопки. Возможные значения видов кнопок:

Значение	Описание
mbYes	Кнопка с надписью 'Yes'
mbNo	Кнопка с надписью 'No'
mbOK	Кнопка с надписью 'OK'
mbCancel	Кнопка с надписью 'Cancel'
mbHelp	Кнопка с надписью 'Help'
mbAbort	Кнопка с надписью 'Abort'
mbRetry	Кнопка с надписью 'Retry'
mbIgnore	Кнопка с надписью 'Ignore'
mbAll	Кнопка с надписью 'All'

Список необходимых кнопок заключается в квадратные скобки [], поскольку параметр **AButtons** является множеством. Если внутри скобок список отсутствует, в окне не будет ни одной кнопки и пользователю придется закрывать окно системными кнопками **Windows**.

Кроме множества значений, соответствующих отдельным кнопкам, в **Delphi** определены три константы, соответствующие часто используемым сочетаниям кнопок:

Значение	Описание
mbYesNoCancel	Включает в окно кнопки Yes, No и Cancel
mbOkCancel	Включает в окно кнопки OK и Cancel
mbAbortRetryIgnore	Включает в окно кнопки Abort, Retry и Ignore

Эти константы являются предопределенными множествами. Поэтому при их использовании их не надо заключать в квадратные скобки [].

Параметр **HelpCtx** определяет экран контекстной справки, соответствующий данному диалоговому окну. Этот экран справки будет появляться при нажатии пользователем клавиши **F1**. Если вы справку не планируете, при вызове **MessageDlg** надо задать нулевое значение параметра **HelpCtx**. Подробнее о контекстной справке см. в описании свойства **HelpContext**.

Функция **MessageDlg** возвращает значение, соответствующее выбранной пользователем кнопке. Возможные возвращаемые значения:

mrNonemrAbortmrYes
mrOkmrRetrymrNo
mrCancel mrIgnore mrAll

Функция **MessageDlg** очень полезна для быстрого создания прототипа приложения и проверки диалогового взаимодействия с пользователем. Но у нее есть заметный недостаток: в заголовках и надписях на кнопках тексты английские, так что при использовании русских сообщений получается смесь русского с английским. Поэтому после создания прототипа приложения лучше заменить стандартные диалоговые окна на спроектированные самим разработчиком.

1. **ShowMessage** - отображает окно сообщения с кнопкой ОК.

Задание 1. Разработка проекта «Словарь»

Постановка задачи

Разработайте программу для составления словаря новых терминов. Должна иметься возможность внесения в словарь изменений, дополнения или сокращения его. При выборе термина на контрольной панели должны появляться его номер и общее количество слов в словаре.

Пояснение. Новым в этой работе является использование комбинированного списка **ComboBox** (представляющего собой объединение строки ввода и компонента **ListBox**), компонента – выключателя **CheckBox**, а также функции **MessageDlg**.

План разработки проекта

1. Откройте новый проект.
2. Разместите на форме экземпляры компонентов в соответствии с рис. 1.

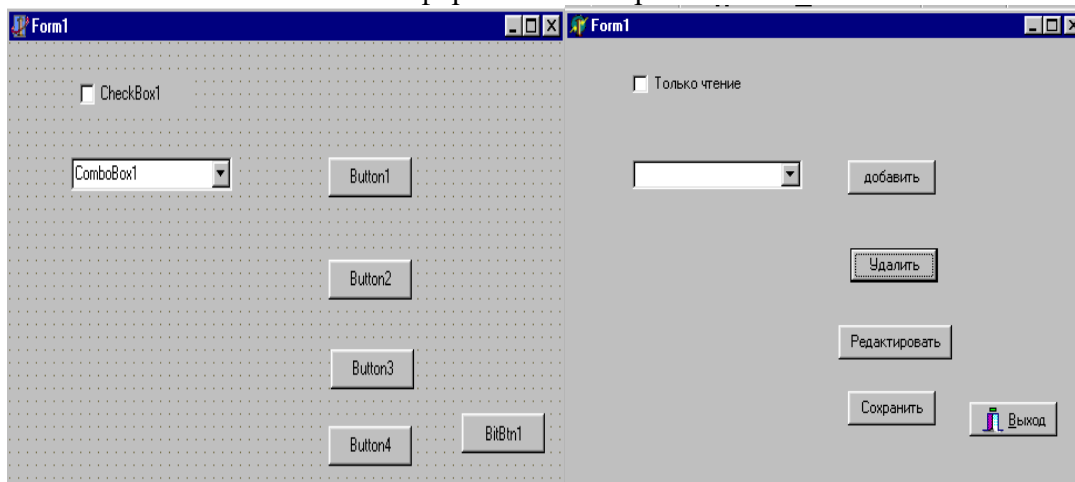


Рис. 1 Форма

Рис.2 Проект

Выполните следующие действия (рис. 2):

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
BitBtn1	Properties	Caption	Установка имени кнопки: «Спасибо»
		Kind	bk Close
ComboBox1	Properties	Text	Вставка пробела
		Items	Открытие двойным щелчком списка Stringlisteditor . Ввод нескольких терминов, сохранение их в файле glostext.txt, предварительно убедитесь что выбрана нужная директория.

3. Сохраните код программы и проект.
4. Запустите программу, затем закрыть окно проекта, щелкнув на кнопке «Выход».
5. Запустить программу, щелкните на стрелке объекта **ComboBox1**, выберите из открывавшегося списка любой термин, а затем убедиться, что текст, набираемый в строке ввода, в список не вносится. После этого закройте проект, щелкнув на кнопке «Выход».

6. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие

Button1	Properties	Caption	Установка имени кнопки: «Добавить»
	Events	OnClick	<pre>ComboBox1.Items.Add(ComboBox1.Text); if ComboBox1.ItemIndex = -1 then ComboBox1.Text := "";</pre> <i>примечание.</i> Последний оператор очищает строку ввода после того, как текст из нее попадет в список.

7. Запустите программу и убедитесь, что набранные в строке ввода слова после нажатия кнопки «Добавить» добавляются в список **ComboBox1**.

8. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
Button2	Properties	Caption	Установка имени кнопки: «Удалить»
	Events	OnClick	<pre>if MessageDlg ('Вы действительно хотите удалить запись?', mtWarning, [mbYes,mbNo],0) =mrYes then ComboBox1.Items.Delete (ComboBox1.ItemIndex);</pre>

Пояснение. Условие **if MessageDlg ...** означает, что при нажатии кнопки «Удалить» появиться диалоговое окно с вопросом «Вы действительно хотите удалить запись?» с двумя кнопками «Yes» и «No». Предварительно выбранный термин будет удален из списка только при нажатии кнопки «Yes».

MessageDlg является функцией с четырьмя аргументами:

- 1) сообщение, которое нужно отобразить, набирается в одинарных кавычках;
- 2) константы **mtWarning**, **mtError**, **mtInformation**, **mtConfirmation** определяют специальную пиктограмму и заголовок диалогового окна;
- 3) список констант в квадратных скобках определяют присутствующие в окне кнопки. Возможные значения – **mbYes**, **mbNo**, **mbCancel**, **mbHelp**;
- 4) число, с которым ассоциирована тема справки, отлично от нуля, если в окне присутствует кнопка **mbHelp**, а программа имеет связанный с ней файл справки Windows.

Функция **MessageDlg** возвращает константу, определяющую нажатую пользователем кнопку.

14. Запустите программу, добавьте в список и (или) удалите несколько строк, закрыв ее, а затем снова запустите. Обратите внимание, что внесенные изменения не сохранились. Это связано с тем, что в **ComboBox** загружается первоначальный список.

15. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
Button4	Properties	Caption	Установка имени кнопки: «Сохранить»
	Events	OnClick	<pre>ComboBox1.Items.SaveToFile ('glostext.txt');</pre>
Form1	Events	OnCreate	<pre>ComboBox1.Items.LoadFromFile ('glostext.txt');</pre>

Пояснение. Теперь при создании формы в список **ComboBox1** будет загружаться содержимое файла **glostext.txt**, а измененный список запомнится в нем при нажатии кнопки «Сохранить».

16. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
-------------------	--	-------------------------------	----------

Button3	Properties	Caption	Установка имени кнопки: «Редактировать»
	Events	OnClick	<pre> num :=0; ComboBox1.Items.Delete (num); ComboBox1.Items.Add (ComboBox1.Text); if ComboBox1.ItemIndex = -1 then ComboBox1.Text:= ""; </pre>
ComboBox1	Events	OnClick	<pre> num:=ComboBox1.ItemIndex; Предварительно надо описать в разделе Var целочисленную переменную num public { Public declarations } end; var Form1: TForm1; num:integer; implementation {\$R *.DFM} </pre>

Пояснение.

Для исправления ошибок, замеченных в строках списка, выделенную строку надо удалить из списка, а исправленную строку (т.е. содержимое строки ввода) добавить в список.

Новая переменная num необходима для сохранения номера выбранной строки. При внесении изменений выбранной строкой становится строка ввода, для которой **ItemIndex = -1**.

17. Для кнопок «Добавить», «Редактировать», «Удалить» для сохранения внесенных изменений необходимо выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Button1 Button2 Button3	Events	OnClick	ComboBox1.Items.SaveToFile ('glostext.txt');

18. Запустите программу и убедитесь, что если выбрать строку, исправив ее, а затем щелкнуть на кнопке «Редактировать», то измененная строка попадет в конец списка.

19. Предусмотрите режим работы со списком, допускающий только чтение. Для этого выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
CheckBox1	Properties	Caption	Установка заголовка кнопки: «Только чтение»
	Events	OnClick	<pre> if CheckBox1.Checked=False then ComboBox1.Style :=csDropDown else ComboBox1.Style:= csDropDownList; <i>примечание.</i>Свойство CheckBoxy выбранного выключателя имеет значение True, а в исходном состоянии – значение False </pre>

Пояснение. Комбинированные списки бывают трех типов:

- простые (simple) – список всегда открыт;

- раскрывающийся (dropdown) – список свернут;
- неизменяемые (dropdownlist) – список свернут и невозможна коррекция списка.
- В режиме «Только чтение» сделайте недоступной кнопку «Добавить»:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства /имя события	Действие
CheckBox1	Events	OnClick	if CheckBox1.Checked = false then Button1.Enabled := true else Button1.Enabled:= false;

Аналогично сделать недоступными кнопки «Редактировать», «Удалить», «Сохранить».

20. Сохраните проект, запустить и протестировать его.

21. Предусмотрите возможность добавления терминов в список не только при щелчке на кнопке «Добавить», но и при нажатии клавиши Enter.

Подсказка. Для объекта **ComboBox1** в обработчик события **onKeyPress** вставить код:
if Key = #13 then ...

22. Добавьте четыре метки и вывести на две из них общее количество терминов в списке и номер выбранного термина.

Подсказка. Для объекта **ComboBox1** в обработчик события **Change** вставить код, присваивающий заголовку панели значение **ComboBox1.Items.Count**, а в обработчик события **Click** – код, присваивающий заголовку другой панели значение **ComboBox1.ItemIndex+1**, имея в виду, что первый термин имеет индекс 0, второй – 1 и т.д.

23. Обеспечьте сохранение всех изменений словаря при закрытии формы.

Подсказка. Для объекта **Form1** в обработчик события **CloseQuery** вставить код, сохраняющий измененный список в файл **glostext.txt**.

24. Отсортируйте список.

Подсказка. Свойству **Sorted** объекта **ComboBox1** присвоить значение **True**.

Листинг программы

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Buttons, StdCtrls;
type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    ComboBox1: TComboBox;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure ComboBox1Change(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure CheckBox1Click(Sender: TObject);
  procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
  procedure ComboBox1Click(Sender: TObject);
  procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  end;
end.
```

```

private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
  num:integer;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
begin
  ComboBox1.Items.Add(ComboBox1.Text);
  if ComboBox1.ItemIndex = -1 then ComboBox1.Text := "";
  ComboBox1.Items.SaveToFile ('glostext.txt');
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  if
('Вы действительно хотите удалить запись?',mtWarning,[mbYes,mbNo],0)=mrYes
  then ComboBox1.Items.Delete (ComboBox1.ItemIndex);
  ComboBox1.Items.SaveToFile ('glostext.txt');
  if CheckBox1.Checked = false
  then Button2.Enabled := true
  else Button2.Enabled := false;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
  ComboBox1.Items.SaveToFile ('glostext.txt');
  if CheckBox1.Checked = false
  then Button4.Enabled := true
  else Button4.Enabled := false;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  ComboBox1.Items.LoadFromFile ('glostext.txt')
end;
procedure TForm1.ComboBox1Change(Sender: TObject);
begin
  num:=ComboBox1.ItemIndex;
  label3.Caption:=inttostr(combobox1.Items.count);
end;
procedure TForm1.Button3Click(Sender: TObject);
var num:integer;
begin
  num :=0;
  ComboBox1.Items.Delete (num);
  ComboBox1.Items.Add (ComboBox1.Text);
  if ComboBox1.ItemIndex = -1 then ComboBox1.Text:="";
  ComboBox1.Items.SaveToFile ('glostext.txt');
  if CheckBox1.Checked = false
  then Button3.Enabled := true
  else Button3.Enabled := false;
end;

```

MessageDlg

```

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked=False
  then ComboBox1.Style :=csDropDown
  else ComboBox1.Style := csDropDownList;
  if CheckBox1.Checked = false
  then Button1.Enabled := true
  else Button1.Enabled := false;
end;
procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  if key=#13 then combobox1.Items.add(combobox1.text);
end;
procedure TForm1.ComboBox1Click(Sender: TObject);
begin
  label4.caption:=inttostr(combobox1.Itemindex+1);
end;
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  combobox1.Items.SaveToFile ('glostext.txt');
end;
end.

```

Задание 2. Разработка проекта «Ваш вес»

Постановка задачи

Пусть оптимальный вес человека определяется как рост 100 см. если фактический вес человека меньше оптимального, то будем считать его худым, если больше, то полным.

Создайте программу, выполняющую следующие действия:

- введя свой рост и фактический вес и нажав кнопку «Расчет», учащийся может определить, худой он или полный и на сколько надо поправиться или похудеть.

Пояснение. В этой работе следует использовать:

- использование типов переменных целочисленного и действительного (INTEGER и REAL);
- преобразование строковых данных в числовые и числовых в строковые: с помощью функций StrToInt; StrToFloat; IntToStr; FloatToStr; Format();
- обработка исключительных ситуаций с помощью оператора **Tryexcept – end;**
- использование процедуры ShowMessage для вывода сообщения в отдельном окне.

План разработки программы

1. Откройте новый проект.

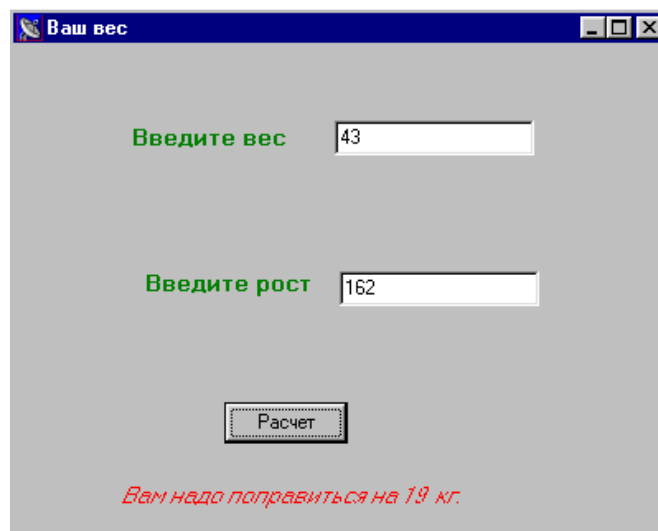


Рис. 3. Проект

2. Разместите на форме экземпляры компонентов в соответствии с рис. 3. В поле **Edit1** следует вводить вес в килограммах, а в **Edit2** – рост в сантиметрах.
3. Сохраните код программы и проект.
4. Введите в раздел Var переменные для сохранения значений:
 - фактического веса (factW),
 - оптимального веса (optW),
 - роста (Rost),
 - разницы между оптимальным весом и фактическим (Delta),

Вначале следует считать, что все данные целые числа:

Var

FactW, optW, Rost, Delta : integer;

5. Создайте процедуру, реагирующую на значок на кнопке **Button1**, и заполните ее следующим кодом:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
Button1	Events	OnClick	<pre>factW:=StrToInt(Edit1.Text); Rost:=strToInt(edit2.Text); optW:=Rost-100; Delta:=abs(factW-optW); if optW=factW then label3.Caption := 'Ваш вес идеален!' else if optW>factW then label3.caption := 'Вам надо поправиться на '+intToStr(delta)+' кг.' else label3.caption := 'Вам надо похудеть на '+IntToStr(delta)+' кг.'; end;</pre>

Пояснение. Функция **StrToInt** преобразует строку в целое число, функция **IntToStr** выполняет обратное действие – целое число преобразует в строку.

6. Сохраните код программы и проект.
7. Усовершенствуйте программу так, чтобы можно было вводить любые десятичные величины. Для этого необходимо использовать вещественный тип переменных *Real*:

var

factW, optW, Rost, Delta: real;

преобразование действительных чисел в строки и строк в действительные числа выполняется с помощью функций FloatToStr StrToFloat. Внести соответствующие изменения в обработку события **OnClick** компонента **Button1**.

8. Сохраните проект, запустите и протестируйте его.
9. Для формирования и преобразования действительных чисел вместо функции FloatToStr() применить функцию Format('%f',[переменная]).
Например: Format('%f',[Delta]).

10. Для форматирования и преобразования применить эту же функцию в виде: Format('Строка%f',[переменная]).

Например: Format('Вам надо похудеть % f',[Delta]).

11. Усовершенствуйте проект:
 - сделайте к программе заголовок;
 - сделайте шрифт выводимой реплики отличным от стандартного по виду, цвету и размеру;
 - вставьте кнопку выхода из программы;
 - предусмотреть возможность повторного запуска программы.

12. Измените код программы таким образом, чтобы в начале программы объекты **Label2** и **Edit2** были не видны и появлялись бы на экране только после того, как будет введен вес.

Подсказка. Свойству **Visible** нужно присвоить значение **False**.

13. В случае преобразования строкового типа в числовой может возникнуть ошибка, если ввести недоступные символы. Если функции **StrToInt** или **StrToFloat** обнаруживают ошибку в записи числа, они иницируются так называемую исключительную ситуацию (исключение), которая обычно приводит к аварийному завершению работы программы.

Обработчик исключений строится в виде таких конструкций:

Try

<защищенный блок операторов>

except

<обработка исключений>

end;

Если при выполнении операторов из защищенного блока возникнет исключение, управление будет передано в блок операторов, располагающийся между операторами **except** и **end**, но, если ошибка не возникнет, блок обработки исключений игнорируется и управление передается оператору, следующему за оператором **end**.

Пример использования обработки исключений для процедуры **Edit1KeyPressed** может выглядеть так:

Try

FactW:=StrToInt(Edit1.Text);

Except

ShowMessage('Ошибочная запись числа: '+Edit1.Text);

Edit1.SetFocus;

Exit;

End;

В результате выполнения оператора **FactW:=StrToInt(Edit1.Text)** на экране появляется окно с текстовым сообщением и кнопкой ОК. после этого работа программы приостанавливается в ожидании реакции пользователя. При вызове стандартной *Exit* активизируется редактор для исправления ранее введенного ошибочного текста.

Внести необходимые изменения для обработки исключительных ситуаций, возникающих при вводе чисел.

Примечание. При работе в среде Delphi эксперименты с исключениями плохо прослеживаются, так как при каждом исключении среда «перехватывает» управление программой. Для отмены этого не обязательно, выбрав команду **Tools, Debugger Options, Language Exceptions**, снять флажок **Stop on Delphi Exceptions**.

14. Измените алгоритм расчета с учетом индекса массы тела.

Вес: X;

Рост: Y;

Индекс массы тела: $A = X/Y^2$ (кг/м²).

Результат определяется по таблице:

№ п/п	Значение индекса	Результат (сообщение, которое надо вывести)
1	$A < 18$	Большой недовес
2	$18 \leq A \leq 20$	Маловато и небезопасно, можно получить истощение
3	$20 \leq A \leq 25$	Идеально
4	$26 \leq A \leq 30$	Легкий перебор
5	$30 < A$	Срочно нужно худеть

Листинг программы

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Buttons;

type

```
TForm1 = class(TForm)
  Label1: TLabel;
  Label2: TLabel;
  Edit1: TEdit;
  Edit2: TEdit;
  Button1: TButton;
  Label3: TLabel;
  BitBtn1: TBitBtn;
  BitBtn2: TBitBtn;
  procedure Button1Click(Sender: TObject);
  procedure BitBtn2Click(Sender: TObject);
  procedure Edit1Change(Sender: TObject);
  procedure FormActivate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

var

```
Form1: TForm1;
var a, factW, optW, Rost, Delta: real;
```

implementation

```
{ $R *.DFM }
```

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
```

```
  try
```

```
    factW:=strtoint(edit1.text);
```

```
  except
```

```
    showmessage ('ошибочная запись числа: '+edit1.text);
```

```
    Edit1.SetFocus;
```

```
  exit;
```

```
end;
```

```
  try
```

```
    rost:=strtoint(edit2.text);
```

```
  except
```

```
    showmessage ('ошибочная запись числа: '+edit2.text);
```

```
    edit2.SetFocus;
```

```
  end;
```

```
  factW:=StrToFloat(Edit1.Text);
```

```
  Rost:=strToFloat(edit2.Text);
```

```
  optW:=Rost-100;
```

```
  Delta:=abs(factW-optW);
```

```
  if optW=factW then label3.Caption := 'Ваш вес идеален!'
```

```
  else
```

```
    if optW>factW then label3.caption :=format('Вам надо поправиться на %f',[delta])
```

```
    else label3.caption := format('Вам надо похудеть на %f',[delta]);
```

```
    a:=(factW*10000)/(rost*rost);
```

```
    if a<18 then ShowMessage ('Жуткий вес');
```

```
  if (a>=18) and (a<20) then showMessage ('маловато и небезопасно, можно получить истощение');
```

```
  if (a>=20) and (a<=25) then Showmessage ('Идеально');
```

```

        if (a>=25) and (a<=30) then showmessage ('Легкийперебор');
        if a>30 then showmessage ('Нужнопохудеть');
    end;
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    edit1.text:='';
    edit2.text:='';
    label3.caption:='';
    label2.visible:=false;
    edit2.visible:=false;
end;
procedure TForm1.Edit1Change(Sender: TObject);
begin
    label2.Visible:=true;
    Edit2.Visible:=true;
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
    Edit1.SetFocus;
end;
end.

```

Контрольные вопросы:

1. Назначение компоненты TComboBox.
2. Режимы работы компоненты «Поле со списком».
3. Назначение компоненты TCheckBox.
4. Назначение стандартных панелей сообщений.

Практическая работа №17

Тема: Создание базовых элементов интерфейса Windows: компонент «Полоса прокрутки», «Панель»

Цель работы: освоить приемы работы с компонентами TPanel, TScrollBar и их свойствами.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Компонент TScrollBar «Полоса прокрутки»

Полосы прокрутки  обычно используется как вспомогательный инструмент в других элементах управления: списках, текстовых областях и прочих. Однако их можно довольно эффективно применять и в качестве самостоятельных элементов управления, например для приблизительной, грубой настройки значений с помощью ползунка.

После того как компонент **TScrollBar** размещен на форме, надо определить его вид, который задается значением свойства **Kind** (**sbHorizontal** – горизонтальная полоса, **sbVertical** – вертикальная). Диапазон значений, охватываемых полосой прокрутки, указывается в свойствах **Min** (минимальное значение) и **Max** (максимальное значение). Текущая позиция ползунка определяется свойством **Position** (значение должно лежать в диапазоне от Min до Max). При перемещении ползунка значение в этом свойстве изменяется автоматически. Но можно выполнить такое изменение и программно – тогда ползунок сам переместиться в нужную позицию.

Определить значение свойств **Min**, **Max** и **Position** можно «за один раз» с помощью метода **SetParams: ProsedureSetParams (Aposition, Amin, Amax: Integer);**

При щелчке на кнопках полосы прокрутки или при нажатии курсорных клавиш значение свойства **Position** изменяется на величину, указанную в свойстве **SmallChange**. Ползунок перемещается в соответствии с этим изменением. Если же щелчок выполняется на свободной области прокрутки или происходит нажатие клавиш **PAGEUP/PAGEDOWN** (листание страниц), то значение свойства **Position** (и соответствующее ему положение ползунка) изменяется на величину указанную в свойстве **LargeChange**.

При изменении значения свойства **Position** всегда генерируется событие **OnChange**.

Компонент TPanel «Панель»

Компонент **TPanel**  представляет собой контейнер общего назначения. Он не имеет заголовка и поэтому менее удобен для функционального группирования элементов. С другой стороны, его свойство **Caption** отображается в виде текстовой строки и может использоваться для вывода сообщений. Компоненты этого класса часто помещаются на форму для того, чтобы располагать вставленные в них дочерние компоненты вдоль одной из сторон окна независимо от изменения размеров этого окна.

После размещения панели на форме ее внешний вид можно настроить с помощью целого набора свойств:

- Свойства **BevelInner** и **BevelOuter** задают стили оформления соответственно внутренней и внешней рамок панели. Они могут принимать одно из четырех значений.

Значение	Вид рамки
bvNone	Отсутствует
bvLowered	«Вдавленная» рамка
bvRaised	«Выпуклая» рамка
bvSpace	«Плоская» рамка

- Свойство **BevelWidth** определяет расстояние между внутренней и внешней рамками (в пикселях).

Задание 1. Разработка проекта «Цвета в формате RGB»

Постановка задачи

Создайте программу, с помощью которой пользователь мог бы увидеть в зависимости от значений насыщенности красного, зеленого и синего результирующий цвет рис. 1.

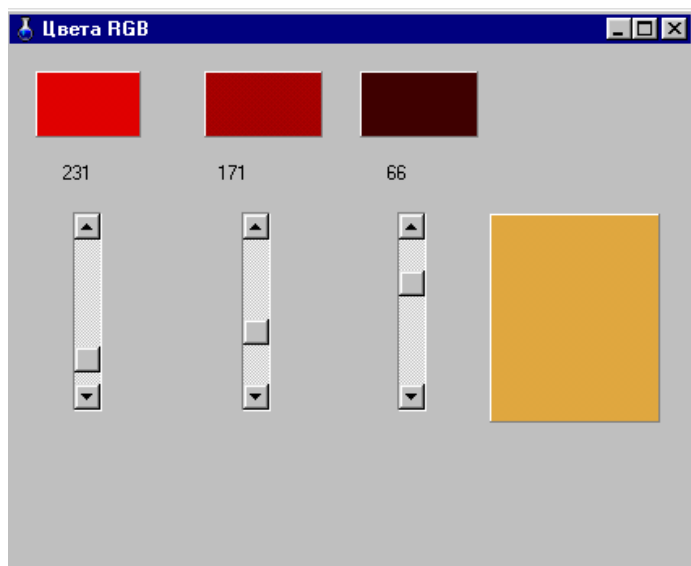


Рис. 1. Проект

Пояснение. Новым в этой работе будет использование:

- полос прокрутки **ScrollBar** для ввода данных,
- функции преобразования значений цветовых составляющих **TColorRef**.

План разработки программы

1. Откройте новый проект.
2. Разместите на форме экземпляры компонентов в соответствии с рис. 2.

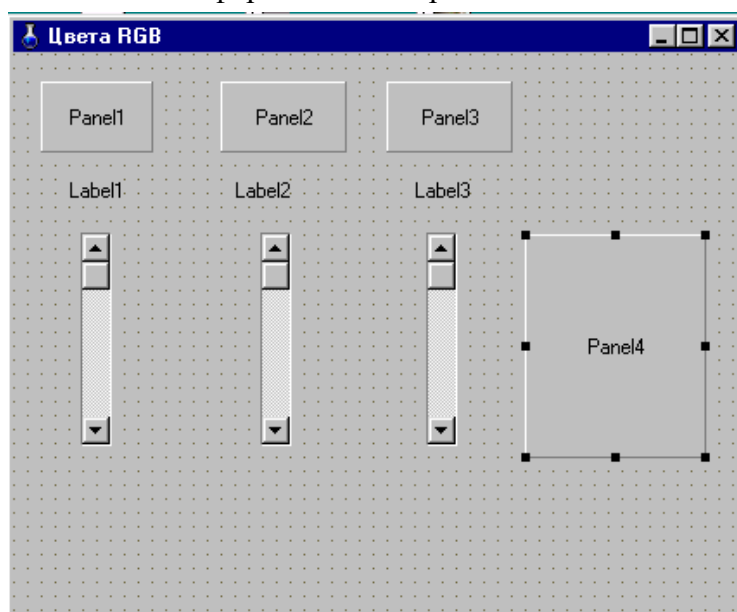


Рис. 2. Форма

3. Полоса прокрутки **ScrollBar** может быть горизонтальной (по умолчанию) или вертикальной. Это определяется свойством **Kind**. В нашем случае используется вертикальная полоса прокрутки.
4. Сохраните код программы и проект.
5. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
ScrollBar1	Properties	Name	Установка имени полосы прокрутки «RedBar», под которым компонент будет известен программе

		Max	Установка максимального количества градаций компонент RGB:255
		Position	Установка начального значения: 122

Аналогично задайте значения для **ScrollBar2**, **ScrollBar3**, присвоив им имена «**GreenBar**» и «**BlueBar**».

6. Для всех компонентов формы установите значение свойства Caption ‘’.

7. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
RedBar	Events	OnChange	Panel1.Color:=TcolorRef(rgb(redbar.position,0,0)); Label1.Caption:=IntToStr (redbar.position); Panel4.Color:=TColorRef(RGB(RedBar.position,GreenBar.Position,bluebar.Position));

Аналогично задать значения для **ScrollBar2**, **ScrollBar3**, проследив за правильностью записи параметров функции **RGB** и **IntStr**.

Пояснение. В зависимости от передвижения ползунка **ScrollBar1** будет меняться цвет **Panel1**, выводиться числовое значение кода на месте **Label1** и меняется цвет **Panel4**.

8. Сохраните, запустите и протестируйте его.

9. Поместить на форму кнопку выхода из программы.

10. Предусмотрите, чтобы после запуска программы установились начальные цвета панелей в зависимости от исходных значений ползунков.

Подсказка. Предусмотреть в реакции на событие **OnCreated** для формы **Form1** обработку значений **ScrollBar1**, **ScrollBar2**, **ScrollBar3**.

11. Внести в программу изменения, чтобы значение кода цвета выводилось не только в десятичной, но и в шестнадцатеричной системе счисления.

Листинг программы

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
```

```
ExtCtrls, StdCtrls, Buttons;
```

```
type
```

```
TForm1 = class(TForm)
```

```
redbar: TScrollBar;
```

```
greenbar: TScrollBar;
```

```
bluebar: TScrollBar;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
Label3: TLabel;
```

```
Panel1: TPanel;
```

```
Panel2: TPanel;
```

```
Panel3: TPanel;
```

```
Panel4: TPanel;
```

```
Label4: TLabel;
```

```

Label5: TLabel;
Label6: TLabel;
BitBtn1: TBitBtn;
procedure redbarChange(Sender: TObject);
procedure greenbarChange(Sender: TObject);
procedure bluebarChange(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.redbarChange(Sender: TObject);
begin
  panel1.Color:=TcolorRef(rgb(redbar.position,0,0));
  label1.caption:=inttostr (redbar.position);
  panel4.color:=tcolorref(rgb(redbar.position,greenbar.position,bluebar.Position));
  label4.caption:=format('0x%x',[redbar.position]);
end;
procedure TForm1.greenbarChange(Sender: TObject);
begin
  panel2.Color:=TcolorRef(rgb(greenbar.position,0,0));
  label2.caption:=inttostr (greenbar.position);
  panel4.color:=tcolorref(rgb(redbar.position,greenbar.position,bluebar.Position));
  label5.caption:=format('0x%x',[greenbar.position]);
end;
procedure TForm1.bluebarChange(Sender: TObject);
begin
  panel3.Color:=TcolorRef(rgb(bluebar.position,0,0));
  label3.caption:=inttostr (bluebar.position);
  panel4.color:=tcolorref(rgb(redbar.position,greenbar.position,bluebar.Position));
  label6.caption:=format('0x%x',[bluebar.position]);
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  label4.caption:=format('0x%x',[redbar.position]);
  label5.caption:=format('0x%x',[greenbar.position]);
  label6.Caption:=format('0x%x',[bluebar.position]);
  panel1.color:=rgb(redbar.position,0,0);
  label1.caption:=inttostr(redbar.position);

```

```

panel2.Color:=rgb(0,greenbar.position,0);
label2.caption:=inttostr(greenbar.position);
panel3.color:=rgb(0,0,bluebar.position);
label3.Caption:=inttostr(bluebar.position);
panel4.color:=rgb(redbar.position,greenbar.position,bluebar.position);
end;
end.

```

Задание 2. Разработка проекта «Тест по физике»

Постановка задачи

Создайте программу, выполняющую следующие действия.

После запуска программы появляется изображение на рис.3.

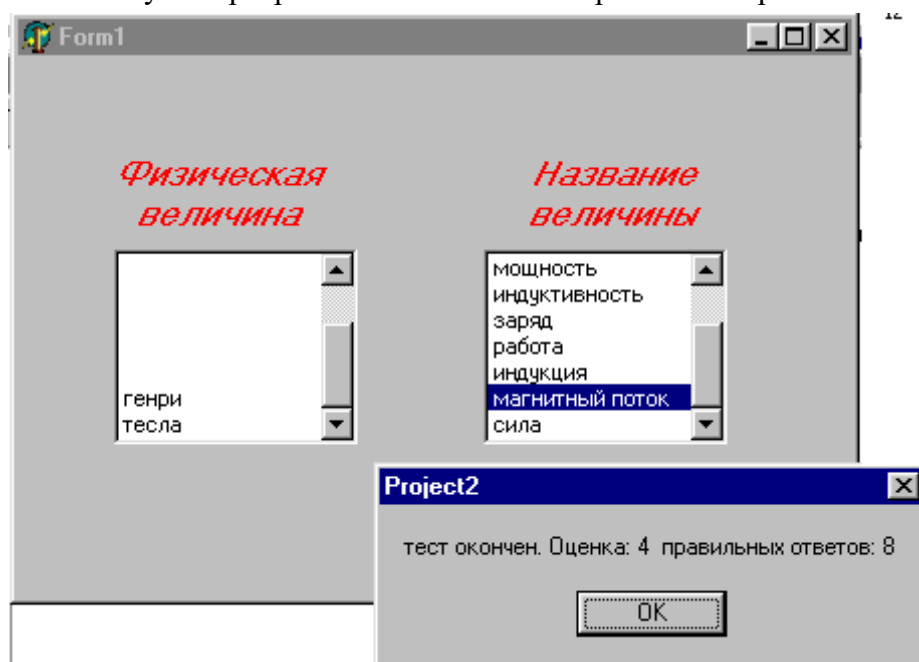


Рис. 3 Прект

Пользователь, перемещаясь с помощью клавиш-стрелок по списку «Физическая величина», выбирает любое слово, нажав клавишу Enter. Затем он переходит в список «Название величины» и выбирает соответствующее название выбранной физической величины. Если выбрано правильное название величины, то под словом «Оценка» появляется одобрительная реплика «Правильно», если неправильное слово, то – «Ошибка». Правильные названия ищутся до тех пор, пока не будет получено соответствие всех физических величин и их названий.

Пояснение. В этой работе будет обеспечиваться взаимодействие двух списков **ListBox** на основе свойств **Items** и **ItemIndex** и создание многострочных надписей в объекте **Label**.

План разработки программы

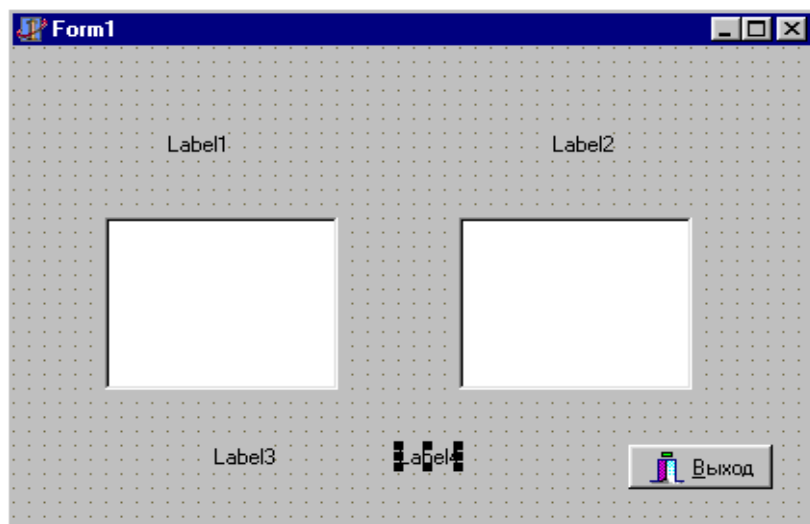


Рис. 4 Форма

1. Откройте новый проект.
2. Разместите на форме объекты в соответствии с рис. 4 и присвойте заголовки меткам. Обратите внимание, что заголовки меток «Физическая величина» и «Название величины» состоят из двух строк и отцентрированы.

Для вывода многострочных надписей в объектах **Label** выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Label1	Properties	AutoSize (изменение размера в зависимости от текста в Caption)	Установка значения False
		WordWrap (разрыв строки)	Установка значения True
		Height Width	Установка подходящих параметров
		Alignment (выравнивание текста)	Установка значения taCenter

3. Установите кнопку выхода из программы и выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
BitBtn1	Properties	Caption	Установка заголовка кнопки: «Выход»
		Kind	bkCancel
	Events	BitBtn1Click	Close;

4. Сохраните код программы и проект.
5. Запустите программу, затем закрыть окно проекта, щелкнув на кнопке «Выход».
6. Выделите объект **ListBox1**, справа от свойства **Items** щелкнуть на кнопке с тремя точками. В появившемся окне встроенного редактора ввести физические величины, каждую на новой строке: «Ватт», «Ом», «Вольт», «Ампер».

Сохраните набранный текст под именем fiz1.txt для этого щелкнуть правой кнопкой мыши и выберите режим **Save**. Для выхода из встроенного редактора щелкнуть на кнопке **OK**.

7. Выделите объект **ListBox2** и проделайте с ним аналогичную работу, введя названия физических величин: «Напряжение», «Соппротивление», «Сила тока», «Мощность». Сохраните набранный текст под именем fiz2.txt.

8. Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
ListBox1	Events	OnKeyPress	if key=#13 then num1:=ListBox1.ItemIndex ;
ListBox1	Events	OnKeyPress	if key=#13 then begin case ListBox2.ItemIndex of 0: Num2:=2; 1: Num2:=1; 2: Num2:=3; 3: Num2:=0; end; if Num1=Num2 then begin Label4.caption:='правильно'; CountR:=CountR+1 ; end else Label4.caption:='ошибка'; if CountR=5 then begin ShowMessage('тестокончен'); end

В разделе **Var** переменные **Num1**, **Num2** и **CountR** объявить как целочисленные.

9. Сохраните, запустите и протестируйте проект.

10. Списки **ListBox1** и **ListBox2** сделайте поочередно доступными после нажатия клавиши **ENTER**.

Подсказка. Установите значение **False** свойства **Enabled** объекта **ListBox2**, а в процедуру **KeyPressed**, относящуюся к **ListBox1**, включить строки:

```
ListBox2.Enabled := True;
ListBox1.Enabled := False;
Form1.ActiveControl := ListBox2;
```

11. Внести изменения в программу, чтобы при правильном выборе названия физической величины слово в левом списке исчезло.

Подсказка. В процедуру **KeyPressed**, относящуюся к **ListBox2**, включить:

```
ListBox1.Items.Delete (Num1);
ListBox1.Items.Insert (Num1,'');
```

12. Увеличить количество физических величин до 10.

13. Вывести сообщение о количестве правильных ответов и оценке за тест.

Листинг программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons;
type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    ListBox2: TListBox;
```

```

Label1: TLabel;
Label2: TLabel;
BitBtn1: TBitBtn;
procedure BitBtn1Click(Sender: TObject);
procedure ListBox1KeyPress(Sender: TObject; var Key: Char);
procedure ListBox2KeyPress(Sender: TObject; var Key: Char);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
  var num1, num2, countR, Rez: integer;
implementation
{$R *.DFM}
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
close;
end;
procedure TForm1.ListBox1KeyPress(Sender: TObject; var Key: Char);
begin
  if key=#13 then num1:=ListBox1.ItemIndex ;
  ListBox2.Enabled := true;
  ListBox1.Enabled :=False;
  Form1.ActiveControl :=ListBox2;
end;
procedure TForm1.ListBox2KeyPress(Sender: TObject; var Key: Char);
begin
  if key=#13 then
  begin
    case ListBox2.ItemIndex of
      0: Num2:=2;
      1: Num2:=1;
      2: Num2:=3;
      3: Num2:=0;
      4: Num2:=8;
      5: Num2:=6;
      6: Num2:=5;
      7: Num2:=9;
      8: Num2:=7;
      9: Num2:=4;
    end;
  end;
  if Num1=Num2 then

```

```

begin
  CountR:=CountR+1 ;
  Rez:=Rez+1 ;
  ListBox1.Items.Delete(Num1);
  ListBox1.Items.Insert(num1,"");
  ListBox1.Enabled:=True;
end
else
  begin
    CountR:=CountR+1 ;
    ListBox1.Enabled:=True;
  end;
  if CountR=10 then begin
    ShowMessage('тестокончен. Оценка: ' + IntToStr(rezdiv 2) + ' правильных ответов: '+
IntToStr(rez));
    exit;
  end;
end;
end;
end.

```

Контрольные вопросы:

1. Назначение компоненты TScrollBar и ее свойства
2. Назначение компоненты TPanel и ее свойства

Лабораторная работа 2 «Оптимизация программы»

Цели:

- приобретение практических навыков оптимизации программ, используя методы и законы оптимизации;
- способствовать формированию аналитического мышления, общих и профессиональных компетенций;
- формировать осознание принадлежности к конкретной профессиональной группе.

Задачи: Применение различных методов и приемов оптимизации для улучшения программ, написанных на различных языках программирования и оценка целесообразности оптимизации.

Теоретические сведения:

Оптимизация-это улучшение программы.

Оптимизация бывает двух видов: глобальная и локальная.

Глобальная оптимизация – это оптимизация, при которой экономия ресурсов достигается путём изменения алгоритма всей программы или большей ее части.

Локальная оптимизация – это оптимизация, при которой экономия ресурсов достигается путём изменения участка программы.

Способы локальной оптимизации:

- Чистка программы
- Разгрузка участков повторения
- Реализация действий
- Упрощение действий
- Экономия памяти
- Сокращение программы

Основные законы оптимизации:

1. Делать проверки всего того, что вводит пользователь.
2. Не делать проверки внутри цикла.

Так как циклы - это слабые места любой программы, и оптимизацию нужно начинать с них.

3. Надо знать принцип работы компьютера. Чем лучше будете знать как компьютер выполняет код программы, тем лучше можно оптимизировать этот код.

Расчет метрики 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 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): $H = D \times V$.

Пример вычисления метрики Холстеда:

Задание:

1. Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей!

2. Оптимизировать программу (можно использовать процедуры или функции).

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

3. Для созданных программ оценить метрические характеристики по Холстеду;
4. Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

Вариант задания №

Код программы: Код оптимизированной программы:

Расчет метрики: Расчет метрики:

Варианты:

Вариант 1. Даны натуральные числа $n, a_1, \dots, a_n$. Определить количество членов a_k последовательности $a_1, \dots, a_n$:

- а) являющихся нечетными числами;
- б) кратных 3 и не кратных 5;
- в) имеющих четные порядковые номера и являющихся нечетными числами.

Вариант 2. Даны натуральные числа $n, q_1, \dots, q_n$. Найти те члены q_i , последовательности $q_1, \dots, q_n$, которые

- а) являются удвоенными нечетными числами;
- б) при делении на 7 дают остаток 1, 2 или 5;
- в) делятся на 5 и не делятся на 7, найти количество и сумму тех членов последовательности.

Вариант 3. Даны целые числа $A_1, \dots, A_{80}$. Получить сумму тех чисел данной последовательности, которые

- а) кратны 5;
- б) нечетны и отрицательны;
- в) удовлетворяют условию $A[i] < i^2$.

Отчет о выполненной лабораторной работе должен содержать:

- цель лабораторной работы, вариант задания и условие задачи;
- таблицу, содержащую задачу, решенную двумя способами (два кода программы) и расчет соответствующих метрик;
- выводы о выполненной лабораторной работе.

Контрольные вопросы:

1. Какой процесс называют «Оптимизацией программы»?
2. Какие существуют способы оптимизации? Для чего они применяются?
3. Приведите примеры приемов алгоритмической оптимизации.

Практическая работа №18

Тема: Создание базовых элементов интерфейса Windows: компонент «Группа переключателей»

Цель работы: освоить приемы работы с компонентой TRadioGroup и ее свойствами.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения Windows; система программирования Delphi.

Компонент TRadioGroup «Группа переключателей»

Если в программе требуется использовать несколько групп переключателей можно использовать компонент TRadioGroup , который объединяет свойства и методы, обеспечивающие поддержку работы группы переключателей.

После размещения на форме компонента TRadioGroup  входящие в него переключатели задаются перечислением их названий. Эти названия вводятся в свойство Items, имеющие тип TString. Так как требуется ввести не одну строку, а несколько, для их ввода предусмотрен специальный редактор, который вызывается щелчком на кнопке с тремя точками, расположенный справа в строке, описывающий свойство Items. Большая текстовая область редактора предназначена для ввода названий переключателей, по одному в каждой строке. Не следует забывать о поддержке возможности управления программой с помощью клавиатуры, поэтому перед некоторыми буквами в названиях надо указать символ &, чтобы сделать эти буквы «горячими». Затем щелкнуть на кнопке ОК, и внешний вид объекта компонента RadioGroup измениться.

Свойство Caption определяет не подпись каждого переключателя (эти подписи теперь задаются в свойстве Items), а заголовок группы. Свойство Columns задает число столбцов (первоначально один), образованных переключателями. Свойство ItemIndex (исходное значение –1). Значение этого свойства изменяется автоматически, когда пользователь выбирает один из переключателей группы.

Задание 1. Разработка проекта «Тест по географии»

Постановка задачи

Создайте проект, состоящий из нескольких форм: титульной формы, формы меню, формы помощи и формы непосредственного текста. В зависимости от правильности ответов на тест, который состоит из нескольких разделов (в свою очередь в разделе может быть несколько вопросов), подсчитывается результат и выставляется оценка по каждому разделу.

Пояснение. Новым в этой работе является использование в проекте нескольких форм и организация вызова этих форм, использование шаблона формы Tabbedpagesиз окна NewItemsвкладки Forms, новых компонентов главного меню, многострочного текстового редактора, которые находятся на вкладке палитры компонентов Standard, компонента Image(графический объект) со вкладки Additionalпалитры компонентов и компонента PageControl (набор панелей с вкладками).

План разработки программы

Откройте новый проект.

Создайте титульную форму:

Дайте название форме «Тест по географии».

Разместите на форме метки в соответствии с рис. 1 и присвоить им заголовки.

Разместите на форме объект Image(графический объект), который находится на вкладке Additionalпалитры компонентов. Выделить Image, справа от свойства Pictureщелкнуть на кнопке с тремя точками. В появившемся окне встроенного редактора PictureEditorщелкнуть на кнопке Load, которая вызывает открытие окна для установки имени графического файла. Выбранная

картинка появится вначале в служебном окне, и если пользователь подтвердит (кнопка ОК), то она возникнет и в окне формы.

Разместите на форме кнопку «Выход» и дайте ей значение Close.

Разместите на форме кнопку «Продолжить». Пока не задавать действия, которые будут выполняться по нажатию этой кнопки.

Сохраните форму под именем, например, F_Titul .pas и проект – под именем Pr_Test.dpr

Создайте форму «Меню».

Командой FileNewFormсоздайте новую форму.

Выполните следующие действия:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства / имя события	Действие
Form2	Properties	Caption	Установка заголовка: «Меню»
		Name	Установка имени формы: U_Menu, под которым компонент будет известен программе

Создать меню с помощью компонента MainMenu, который находится на вкладке Standardпалитры компонентов. Теперь необходимо создать его опции (рис. 1).

Дважды щелкните на появившейся пиктограмме MainMenuили, выделив объект MainMenu, щелкните на кнопке в правой половине строки Items. Появиться окно с заголовком Form1.MainMenu1.

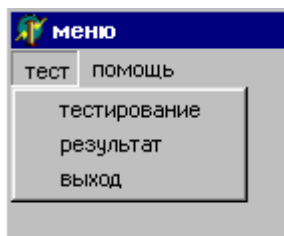


Рис. 1. Проект

Установите в качестве значения свойства Captionназвание первого пункта меню: «Тест». Оно запишется в синем прямоугольнике слева на полосе меню. Нажмите клавишу Enter. Теперь опция готова и можно переходить к созданию следующей.

Каждая опция главного меню может содержать список подопций. Для создания подопций щелкнуть мышью на строке ниже опции и ввести первую подопцию. Продолжить ввод, пока не будет создан весь список подопций.

В названиях опций, являющихся значениями свойства Caption, можно указать символ «&» перед тем символом, который будет использоваться для быстрого выбора опции. В примере для опции «Тест» можно выбрать комбинацию клавиш Alt+T. Для этого в строке Captionнадо указать «&Тест».

Опция «Тест» должна содержать подопции «Тестирование», «Результат», «Выход».

Опция «Помощь» не должна содержать подопций.

Для реализации подопции «Выход» выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Выход	Events	OnClick	Close;

Сохраните форму под именем F_Menu.pas.

Создайте переход с титульной формы на форму «Меню». Для этого выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Кнопка «Продолжить»	Events	OnClick	U_Menu.ShowModal; Кроме того, описать в модуле F_Titul.pas внешнее имя F_Menu, разместив его в блоке реализации (implementation); Uses F_Menu

Сохраните выполненный проект (File, SaveAll). Запустите программу на выполнение и проверьте правильность ее работы:

перейти с титульной формы на форму «Меню»;

выйти из проекта.

Создайте форму «Помощь»:

командой File, Newоткрыть окно NewItems, при помощи которого можно создать новую форму.

На вкладке Forms выберите форму Tabbedpages. Появится стандартное окно новой формы TabbedNotebookDialog.

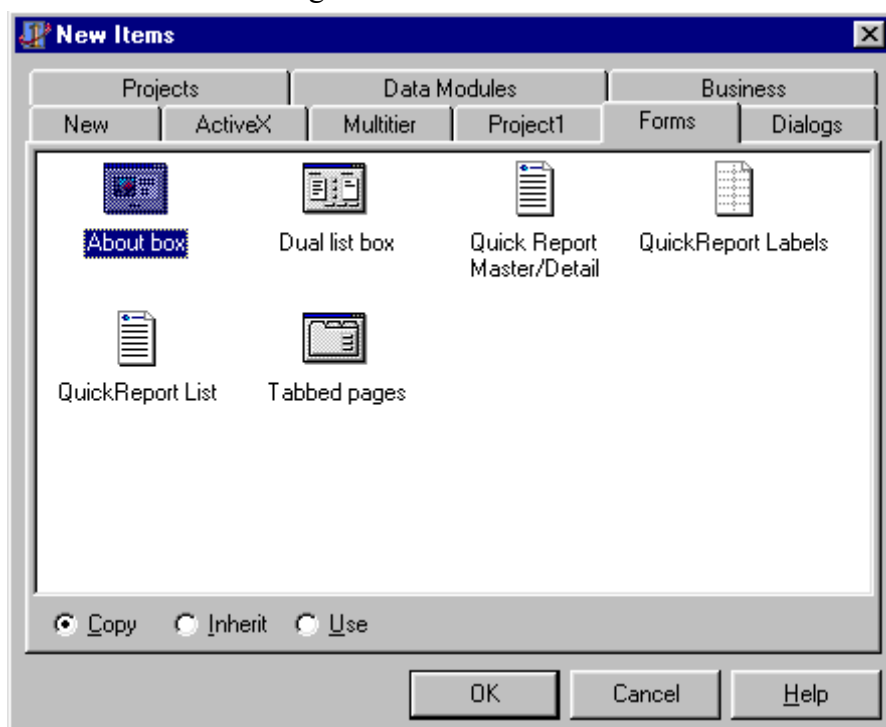


Рис. 2.

Вызовите и рассмотрите список объектов, содержащихся на форме. Форма построена на основе компоненты TPageControl и содержит несколько перекрывающих друг друга вкладок класса TTabSheet. Каждая вкладка может содержать свой набор помещенных на нее компонентов. Чтобы на этапе конструирования добавить новую вкладку или выбрать ранее вставленную, надо щелкнуть на объекте правой кнопкой мыши и выбрать NewPage(новая панель), NextPage(следующая панель) или PreviousPage(предыдущая панель). Смена панелей идет циклически.

Удалить объекты-кнопки OK и Help.

Заменить название кнопки Cancel на «Выход» и установите для нее действие Close.

Измените название формы, для чего поместите на форму объект PageDlg1 и выполните следующие действия:

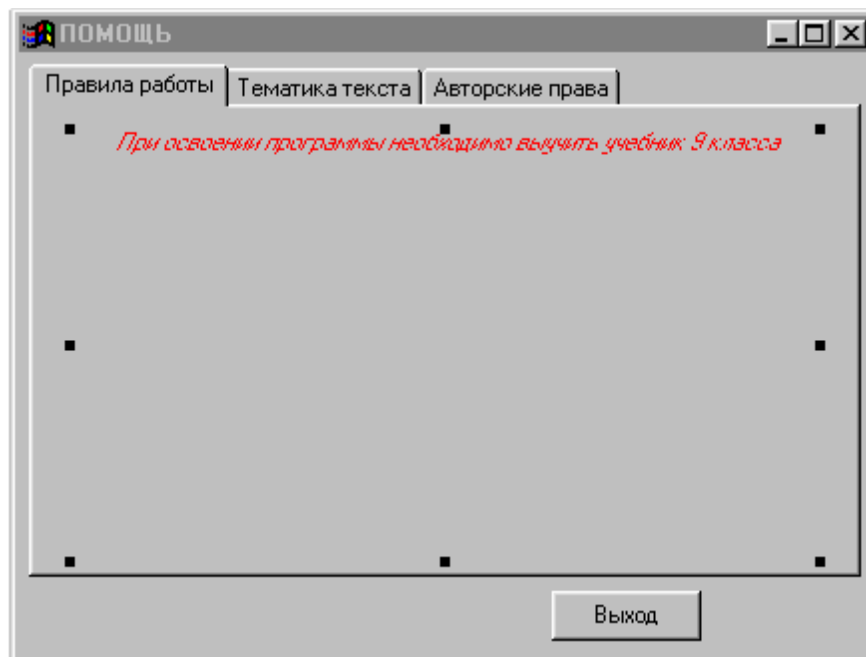
Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
PageDlg1	Properties	Caption	Установка заголовка формы: «Помощь»
		Name	Установка имени компоненты: U_Instr

Дайте вкладкам названия «Правила работы», «Тематика текста», «Авторские права», для этого в свойстве Caption заменить значения TabSheet1, TabSheet2, TabSheet3.

Заполните страницы формы TabbedNotebookDialog текстом инструкции, для чего на каждую страницу добавьте экземпляр компонента Memo из вкладки Standard палитры компонентов. Для ввода текста в объект Memo щелкните на кнопке с тремя точками свойства Lines. В появившемся окне встроенного редактора введите необходимый текст.

Для того чтобы цвет вставленного объекта совпадал с цветом формы, выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Memo	Properties	Color	Установка значения свойства clBtnFace для совпадения по цвету с фоном формы
		BorderStyle	Установка значения свойства bsNone – отмена рамки
		Font	Установка значения свойства – выбор нужного шрифта
		Alignment	Установка значения свойства – подходящее выравнивание текста



Сохраните созданную форму под именем F_Instr.pas.

Создайте переход с формы «Меню» на форму «Помощь». Для этого перейдите на форму «Меню», установите курсор на опцию «Помощь», щелкнув мышью и внесите изменения в процедуру обработки:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Команда меню «Помощь»	Events	OnClick	U_Instr.ShowModal; Кроме того, описать в модуле F_Menu.pas внешнее имя F_Instr, разместив его в блоке реализации (implementation): Uses F_Instr;

Сохраните выполненный проект (File, SaveAll). Запустите программу на выполнение и проверьте правильность ее работы:

перейдите с титульной формы на форму «Меню»;

перейдите с формы «Меню» на форму «Помощь»;

выйдите из проекта.

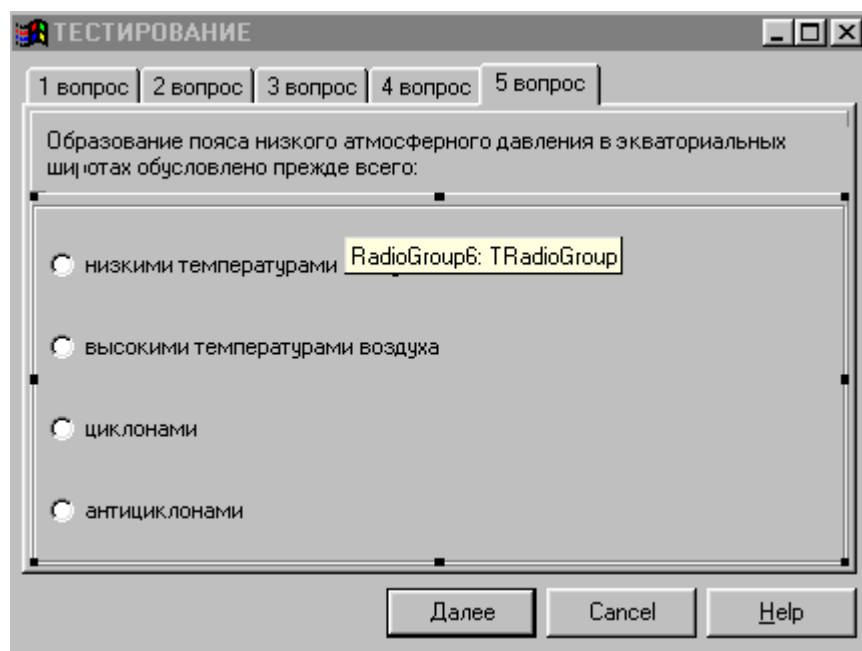
Создайте форму «Тестирование», дайте ей имя U_Test:

постройте форму «Тестирование» на основе стандартной формы TabbedNotebookDialog и включите в нее 5 вопросов (5 вкладок TTabSheet).

на первую вкладку поставьте TPanel1, Tpanel2, Label1, RadioGroup1.

Выделенный объект	Вкладка окна Object Inspector	Имя свойства / имя события	Действие
Panel1	Properties	Caption	
		Align	AlTop
Label1	Properties	WordWrap	True —для автоматического переноса слов

		Caption	Задать вопрос
Panel1	Properties	Caption	
		Align	alClient
RadioGroup1	Properties	Caption	
		Align	AlClient
		Items	Ввести ответы



Пример теста по географии

1) За начало отсчета географической долготы места принят:

экватор

меридиан 180 градусов

нулевой меридиан

северный полюс

Ответ: 3

2) Линии, соединяющие на климатических картах точки с одинаковым атмосферным давлением, называются:

горизонталями

изометрами

изобарами

изогиями

Ответ: 3

3) Артезианскими называются следующие воды:

почвенные

грунтовые

межпластовые ненапорные

Межпластовые напорные

Ответ: 2

4) Основным географическим следствием шарообразной формы Земли является:

смена дня и ночи

смена времен года

широтная зональность

высотная зональность

Ответ: 3

5) Образование пояса низкого атмосферного давления в экваториальных широтах обусловлено прежде всего:

низкими температурами воздуха

высокими температурами воздуха

циклонами

антициклонами

Ответ: 2

Создайте переход с формы «Меню» на форму «Тестирование».

Сохраните созданную форму под именем F_Test.pas.

Для правильного контроля выполнения тестирования сделать так, чтобы последовательно был доступен сначала первый вопрос, а затем только второй и т.д. Для этого предусмотрите дополнительную кнопку завершения тестирования каждого вопроса.

Выделенный объект	Вкладка Object Inspector	Имя свойства / имя события	Действие
Команда меню «Тестирование»	Events	OnClick	<pre>if Tag =0 Then Begin PagesDlg.TabSheet2.TabVisible := True; PagesDlg.TabSheet1.TabVisible := False; Tag :=1; exit; end; if Tag =1 Then Begin PagesDlg.TabSheet3.TabVisible := True; PagesDlg.TabSheet2.TabVisible := False; Tag :=2; exit; end; if Tag =2 Then Begin PagesDlg.TabSheet4.TabVisible := True; PagesDlg.TabSheet3.TabVisible := False; Tag :=3; exit; end;</pre>

			<pre> if Tag =3 Then Begin PagesDlg.TabSheet5.TabVisible := True; PagesDlg.TabSheet4.TabVisible := False; Tag := 4; exit; end; PagesDlg.Activate; if Tag=4 then Close; </pre>
--	--	--	---

Сохраните выполненный проект (File, SaveAll). Запустите программу на выполнение и проверьте правильность ее работы:

- перейдите с титульной формы на форму «Меню»;
- перейдите с формы «Меню» на форму «Тестирование»;
- выполните тестирование;
- выйдите из проекта.

Для подсчета результата введите переменную Ball, которую необходимо объявить в разделе Var формы «Результат». Внесите изменения в программу для подсчета баллов.

Создайте форму «Результат» (дать ей имя U_Res). Форма «Результат» должна содержать оценку тестирования и кнопку «Выход».

Самостоятельно создайте форму «Результат».

Сохраните созданную форму под именем F_Rez.pas.

Создайте переход с формы «Меню» и установите курсор на опции «Результат». Для этого перейдите на форму «Меню» и установите курсор на опции «Результат», щелкните мышью и внесите изменения в процедуру обработки:

Выделенный объект	Вкладка Object Inspector	Имя свойства / имя события	Действие
Команда меню «Результат»	Events	OnClick	<pre> U_Res.Label1.Caption:='Ваш результат: '+ IntToStr(Ball); U_Res.ShowModal; </pre> Кроме того, нужно описать в модуле F_Menu.pas внешнее имя F_Rez, разместив его в блоке реализации (implementation): uses F_rez;

Проверьте правильность работы программы.

Создайте тестирующую программу по предмету «Основы программирования».

Дополните программу разделом «Содержание изучаемой темы».

Внесите изменения в программу так, чтобы вначале запрашивалось: «Имя тестируемого», которое также выводилось бы на экран при выдаче результата прохождения теста.

Тема: Создание базы данных, состоящей из одной таблицы в СУБД ACCESS и подключение ее к среде DELPHI

Цель работы:

повторить основные понятия базы данных;

научиться создавать таблицу базы данных в режиме Конструктор в СУБД Microsoft Access;

освоить переход из режима Конструктор в режим таблицы;

освоить основные приемы заполнения и редактирования таблиц базы данных;

познакомиться с простой сортировкой значений таблицы;

познакомиться с поиском записей по образцу;

научиться сохранять и загружать базу данных;

научиться распечатывать таблицы;

научить подключать таблицы через ADO к системе DELPHI;

познакомиться с основными компонентами для управления элементами таблицы в системе DELPHI;

изучить основные свойства компонент для форматирования вида подключенной базы данных.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения WINDOWS; система программирования DELPHI.

Этапы работы

Проектирование структуры базы данных.

Предположим, что нам надо изготовить самый простой вариант базы данных, когда вся информация хранится в одной таблице.

Конструирование структуры будущих таблиц базы данных.

Таблицу будем создавать в режиме Конструктор. В ней будет 7 полей (код, фамилия, имя, отчество, год рождения, школа, класс) и 10 записей. Поле код – уникальный ключ записи (обычно используется для связи записей из разных таблиц).

Создание схемы базы данных.

При наличии одной таблицы схема базы данных тоже простая: состоит из одной этой таблицы, т.е. ее можно специально не делать.

Ввод данных в таблицы.

В данном варианте будет только одна таблица. Таблицу будем создавать в режиме Конструктор, а заполнять – в режиме таблицы, передвигаясь по ячейкам с помощью стрелок, клавиши табуляции или мышки.

Задание 1. Создайте новую базу данных и изготовьте структуру таблицы с информацией о студентах «Компьютерной школы».

Порядок работы:

Вызовите программу ACCESS. Откроется окно системы управления базами данных.

Включите мышкой переключатель Новая база данных, в поле Имя файла – укажите имя базы данных ADO_01.accdb и путь куда сохранится созданная база данных (C:/PracticumBD_Delphi).

Щелкните по кнопке Создать или нажмите ENTER.

Открывается вкладка Таблица и автоматически вам предлагается новая таблица, для того чтобы организовать структуру вашей таблицы, необходимо перейти в режим Конструктор (Вид/Конструктор).

Введите в появившееся диалоговое окно имя таблицы «Список»

Заполните поля в Конструкторе данными из таблицы 1. Общие свойства поля оставляем по умолчанию (какие задает сама программа).

Таблица 1

Имя поля	Тип данных
Код	счетчик
Фамилия	текстовый
Имя	текстовый
Отчество	текстовый
Год рождения	числовой
Школа	числовой
Класс	числовой

По умолчанию программа автоматически создает ключ на поле Код. Оставьте его без изменения – это наиболее удачный вариант создания ключевого поля (в нем никогда не будет повторяющихся записей, так тип поля у него счетчик, т.е. автоматическая нумерация)

Сохраните таблицу, щелкнув на кнопке Сохранить.

Перейдите в режим таблицы (Вид/Режим Таблицы)

Заполните базу данных значениями из таблицы 2.

Код	Фамилия	Имя	Отчество	Год рождения	Школа	Класс
1	Иванникова	Анна	Ивановна	1984	1	9
2	Баранова	Ирина	Алексеевна	1983	3	10
3	Корнилова	Ольга	Владимировна	1984	5	9
4	Воробьева	Александра	Петровна	1983	1	10
5	Воробьева	Александра	Ивановна	1984	3	9
6	Воробьева	Олег	Григорьевич	1985	5	8
7	Скоркин	Александр	Евгеньевич	1982	1	11
8	Володина	Анна	Алексеевна	1984	3	9
9	Новоселов	Александр	Антонович	1983	5	10

0	1 дрова	Алексан	Елена	Алексеев на	1 984	1	9
---	------------	---------	-------	----------------	----------	---	---

Сохраните таблицу, щелкнув на кнопке Сохранить.

Задание 2. Выполните редактирование ячеек

Порядок работы:

Замените фамилию Иванникова на Иванова. Для этого выделите редактируемую ячейку и наберите новую фамилию.

Замените год рождения на 1983.

Внимательно посмотрите таблицу и исправьте свои ошибки.

Сохраните таблицу, щелкнув на кнопке Сохранить.

Задание 3. Отсортируйте значения таблицы.

Порядок работы:

Чтобы произвести сортировку значений, нужно поставить курсор в любую ячейку сортируемого столбца и щелкнуть по кнопке (вкладка Главная/Группа Сортировка И Фильтр):



по возрастанию;



по убыванию.

Отсортируйте:

фамилии – по алфавиту (поставьте маркер на любую фамилию с названием Фамилия и щелкните мышкой по кнопке «по возрастанию»;

имя – по алфавиту (поставьте маркер на любое имя в столбце с названием Имя и щелкните мышкой по кнопке «по возрастанию»;

номер школы – по убыванию (поставьте маркер на любой номер школы в столбце с названием Школа и щелкните мышкой «по убыванию»);

год рождения – по убыванию (поставьте маркер на любой год рождения в столбце с названием Год рождения и щелкните мышкой по кнопке «по убыванию»);

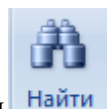
класс – по возрастанию (поставьте маркер на любой класс в столбце с названием Класс и щелкните мышкой по «по возрастанию»);

Сохраните таблицу, щелкнув на кнопке Сохранить.

Задание 4. Выполните поиск записей по образцу.

Порядок работы:

Установите текстовый курсор в поле Фамилия;



Щелкните по кнопке Найти (вкладка Главная/Раздел Найти), которая позволяет найти запись по введенному значению. В результате появится диалоговое окно, в поле Образец наберите фамилию Баранова и щелкните на кнопке НайтиДалее.

Сохраните таблицу, щелкнув на кнопке Сохранить.

Закройте таблицу и программу ACCESS.

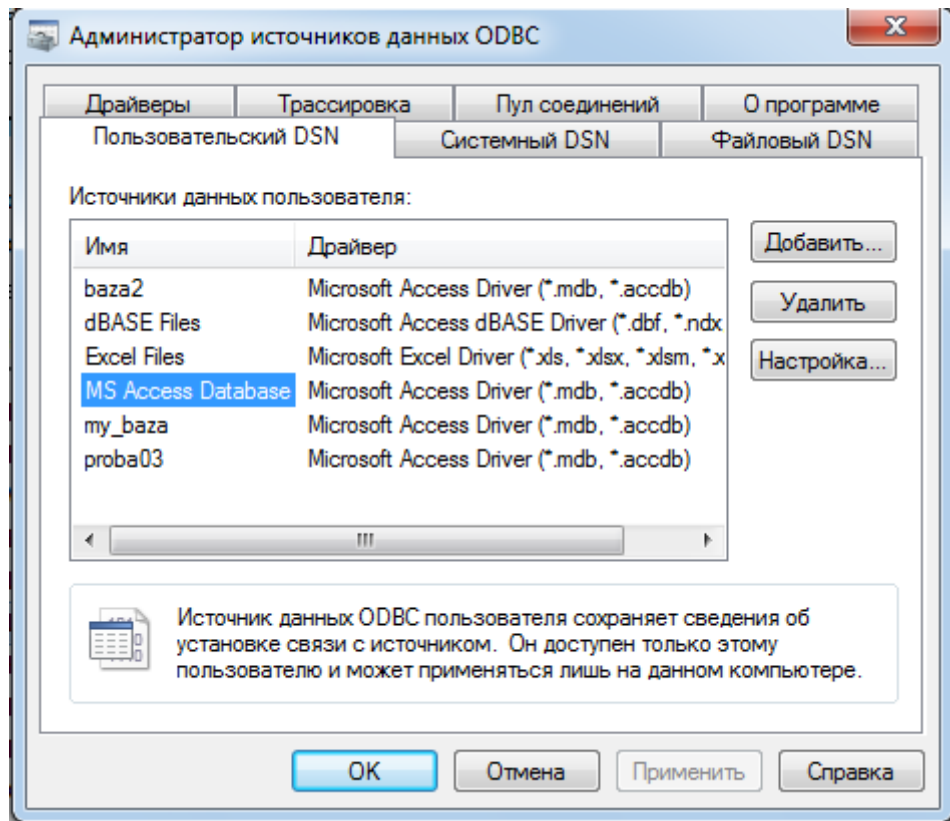
Задание 5. Подключение базы данных к системе DELPHI через драйвер Microsoft AccessDriver (*.mdb, *.accdb)

Выполните открытие команды Администрирования (Пуск/Панель Управления/Администрирование).

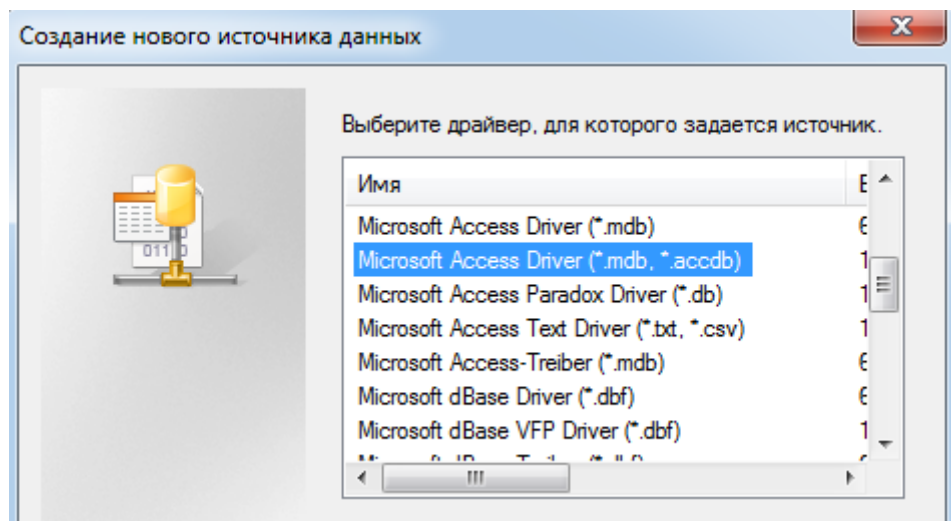
В открывшемся окне выберите Источник данных ODBC

	Microsoft .NET Framework 1.1 Configura...	09.04.2012 22:24	Ярлык	2 КБ
	Microsoft .NET Framework 1.1 Wizards	09.04.2012 22:24	Ярлык	2 КБ
	Windows PowerShell Modules	14.07.2009 10:52	Ярлык	3 КБ
	Брандмауэр Windows в режиме повы...	14.07.2009 10:41	Ярлык	2 КБ
	Инициатор iSCSI	14.07.2009 10:41	Ярлык	2 КБ
	Источники данных (ODBC)	14.07.2009 10:41	Ярлык	2 КБ
	Конфигурация системы	14.07.2009 10:41	Ярлык	2 КБ
	Планировщик заданий	14.07.2009 10:42	Ярлык	2 КБ
	Просмотр событий	14.07.2009 10:42	Ярлык	2 КБ
	Системный монитор	14.07.2009 10:41	Ярлык	2 КБ
	Службы компонентов	14.07.2009 10:46	Ярлык	2 КБ
	Службы	14.07.2009 10:41	Ярлык	2 КБ
	Средство проверки памяти Windows	14.07.2009 10:41	Ярлык	2 КБ
	Управление компьютером	14.07.2009 10:41	Ярлык	2 КБ

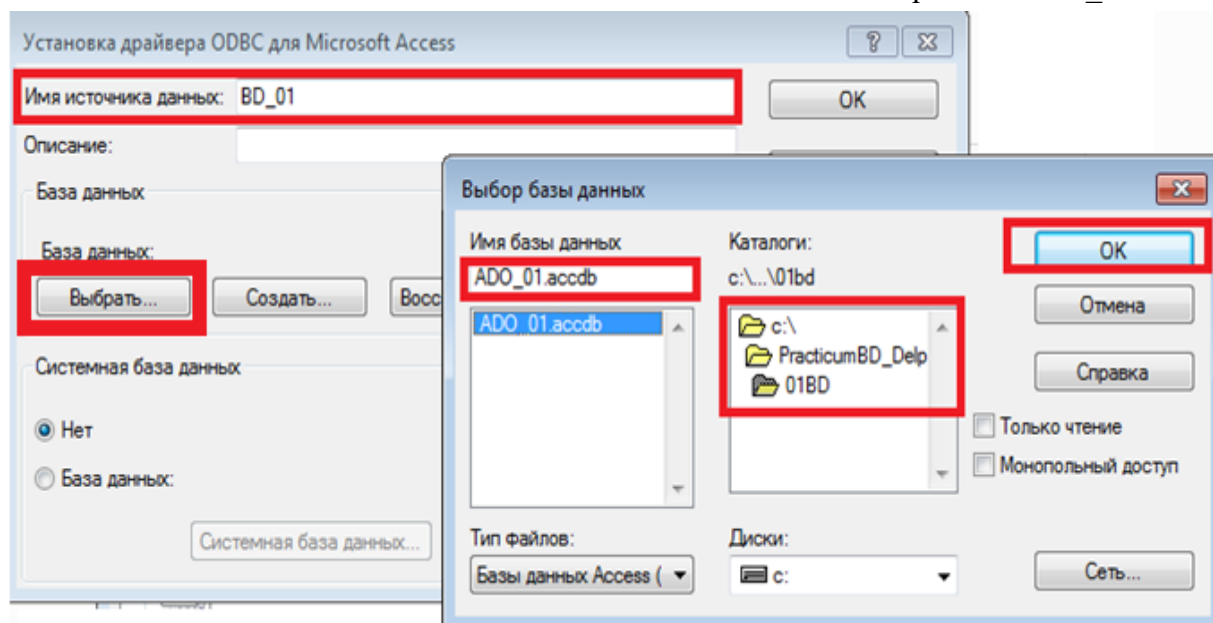
В появившемся диалоговом окне выберите MS Access Database (вкладка Пользовательский DSN) и нажмите кнопку ДОБАВИТЬ.



В следующем появившемся окне выбираем драйвер, для которого создается источник – Microsoft AccessDriver (*.mdb, *.accdb), так как созданная таблица в СУБД ACCESS имеет расширение *.accdb.



В появившемся диалоговом окне в поле ИМЯ ИСТОЧНИКА наберите имя BD_01:

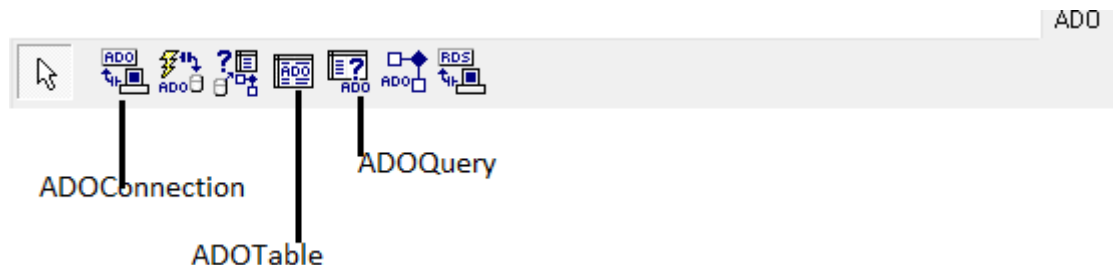


Нажав на кнопку Выбрать, необходимо указать путь к подключаемой базе данных в нашем случае это ADO_01.accdb. Заполнение диалогового окна закончилось можно нажать ОК.

Драйвер подключен, можно открыть систему DELPHI и сохранить новый проект File / Save Project As... в той же папке что база данных.

Задание 6. Доступ к данным

Для доступа к базе данных будем использовать технологию ADO. Механизм ADO доступа к информации базы данных является стандартом фирмы MICROSOFT. Использование этой технологии подразумевает использование настраиваемых провайдеров данных. Технология ADO обеспечивает универсальный механизм доступа из приложений к информации источников данных. Эта технология основана на стандартных интерфейсах COM, являющихся системным механизмом Windows. Это позволяет удобно распространять приложения баз данных без вспомогательных библиотек. Доступ к данным при использовании технологии ADO обеспечивают компоненты ADOConnection, ADOTable и ADOQuery, значки которых находятся на вкладке ADO.



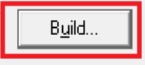
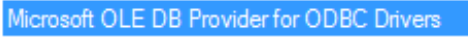
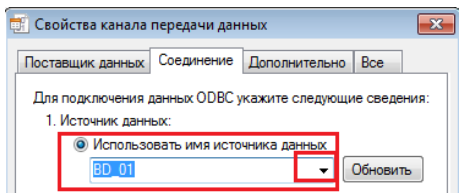
Компонент ADOTable – представляет данные, полученные из одной таблицы.

Компонент ADOQuery – данные, полученные из базы данных в результате выполнения SQL-команды, результатом которой являются не данные, а результат их обработки.

Предварительно разотметьте свою форму панелями или контейнерами на ваше усмотрение.

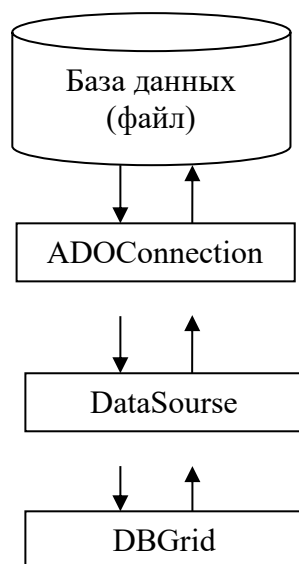
Выделенный объект	Вкладка окна Object Inspector	Имя свойства/имя события	Действие
Panel1	Properties	Align	alClient
		Caption	Пусто
Form1	Properties	Caption	Практическая работа 1

Разместите компонент ADOConnection на форме и установите следующие свойства:

Выделенный объект	Вкладка окна/меню	Имя свойства/имя события	Действие
 ADOConnection1	ADO		Дв. лкм на компоненте ADOConnection
			в диалоговом окне/ выбрать переключатель
			☒ Use Connection String / кнопка 
			в диалоговом окне Свойства канала передачи данных на вкладке Поставщик данных выбрать 
			в диалоговом окне Свойства канала передачи данных на вкладке Соединение выбрать имя источника данных
			

 ADOConnection1	es	Property	Connected	дв. лкм на false / Выходит диалоговое окно для ввода пароля/ пароль не вводим/ Нажать ОК (Без пароля) – значение параметра изменится на True
			LoginPrompt	False (не спрашивать пароль при запуске приложения)
 ADOTable1	es	Property	Connection	ADOConnection1
			TableName	Выбрать таблицу Список

Для связи между данными, в качестве которых может выступать компонент ADOTable и ADOQuery, и компонентом, обеспечивающим отображение данных, например DBGrid, используется компонент DataSource.

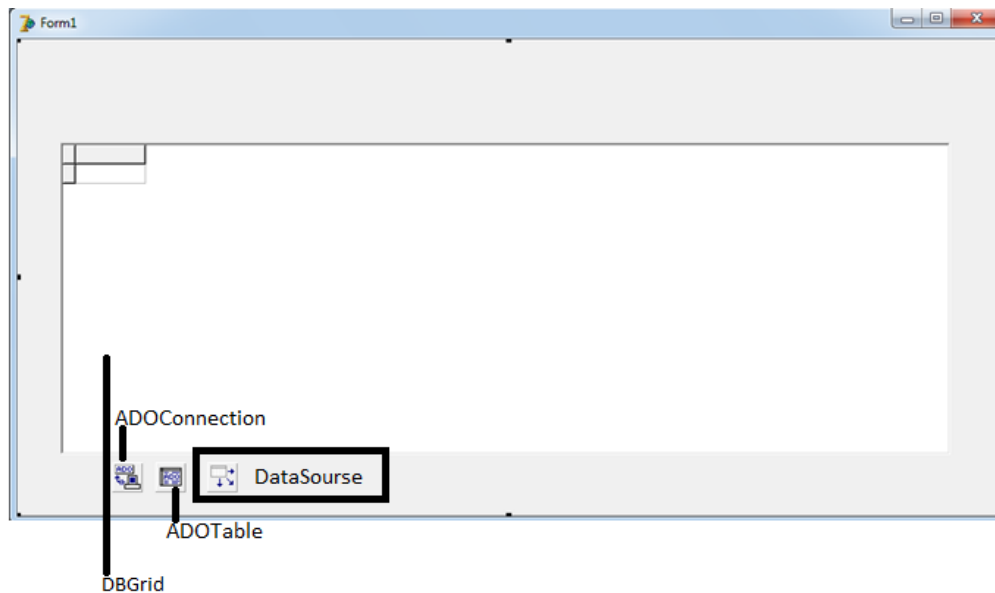


Добавьте на форму компонент DataSource (вкладка Data Access). Компоненты ADOConnection, DataSource, DBGrid являются не визуальными (в окне программы, во время ее работы, они не отображаются). Поэтому их можно поместить в любую область формы.

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/имя события	Действие
 DataSource1	Properties	DataSet	ADOTable1
		Name	DataSource1

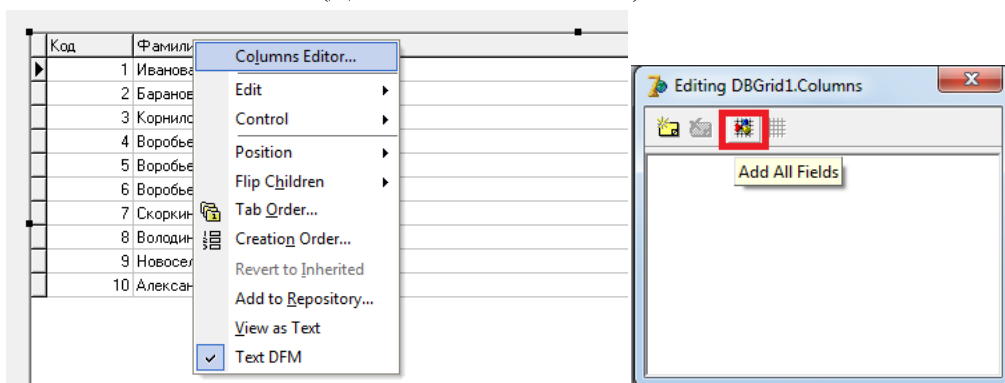
Задание 7. Отобразить данные базы на форме проекта.

Установите на форму компонент DBGrid (вкладка Data Controls). И определите ее свойства используя ниже представленную таблицу.



Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/имя события	Действие
DBGrid1	Properties	DataSource	DataSource1
ADOTable1	Properties	Active	True

Активизируйте на форме компонент DBGrid1 и вызовите контекстное меню для него – ColumnsEditor – AddAllFields (Добавить все столбцы)



Задание 8. Оформите внешний вид таблицы в вашем проекте.

Выбрав соответствующее поле (например, Код) установите на вкладке Properties его Width (Ширину)=50 пикселей

Имя поля	Width (Ширину)
Код	50
Фамилия	150
Имя	150
Отчество	150
Год рождения	100
Школа	50
Класс	50

Код	Фамилия	Имя	Отчество	Год рождения	Школа	Класс
1	Иванова	Анна	Ивановна	1983	1	9
2	Баранова	Ирина	Алексеевна	1983	3	10
3	Корнилова	Ольга	Владимировна	1984	5	9
4	Воробьев	Алексей	Петрович	1983	1	10
5	Воробьев	Алексей	Иванович	1984	3	9
6	Воробьев	Олег	Григорьевич	1985	5	8
7	Скоркин	Александр	Евгеньевич	1982	1	11
8	Володина	Анна	Алексеевна	1984	3	9
9	Новоселов	Алексей	Антонович	1983	5	10
10	Александрова	Елена	Алексеевна	1984	1	9

Используя, свойства компонента DBGrid поэкспериментируйте с внешним видом таблицы (будьте осторожны при смене свойств, изменяете обдуманно):

Таблица 3. Свойства компонента DBGrid

Свойство	Описание
DataSource	Ссылка на источник данных
Columns	Отображаемая информация (столбцы)
BorderStyle	Вид границы вокруг компонента
Options.dgEditing	Разрешает (True) изменение, добавление и удаление данных. Чтобы активизировать режим редактирования записи, надо нажать клавишу «F2»; чтобы добавить запись – «Insert»; чтобы удалить запись – «Ctrl»+ «Del» или «Del», если значение свойства Options.dbConfirmDelete равно False.
Options.dbConfirmDelete	Необходимость подтверждения удаления записи. Если значение свойства равно True, то, чтобы удалить запись, пользователь должен нажать комбинацию клавиш «Ctrl» + «Del» и подтвердить выполнение операции удаления щелчком на кнопке ОК в появившемся окне Confirm. Если значение свойства равно False, то текущая запись будет удалена в результате нажатия клавиши «Del»
Options.dgTitles	Разрешает вывод строки заголовка столбцов
Options.dgIndicator	Разрешает (True) отображение колонки индикатора. Во время работы с базой данных текущая запись помечается в колонке индикатора треугольником, новая запись – звездочкой, редактируемая – специальным значком
Options.dgColumnResize	Разрешает (True) менять во время работы программы ширину колонок таблицы
Options.dgColLines	Разрешает (True) выводить линии, разделяющие колонки таблицы

Options.dgRowLines	Разрешает (True) выводить линии, разделяющие строки таблицы
--------------------	---

Свойство Columns компонента DBGrid представляет собой коллекцию (массив) объектов типа TColumns. Свойства объекта TColumns определяют информацию, которая отображается в столбце (Таблица 4).

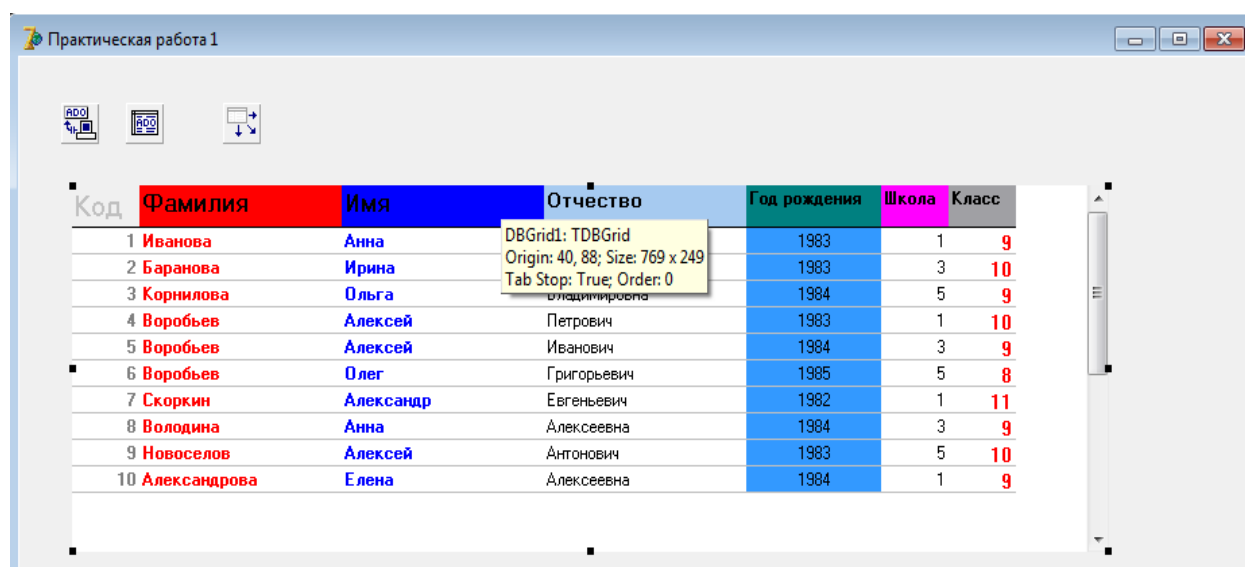
Таблица 4. Свойства объекта TColumns

Свойство	Описание
FieldName	Поле, содержимое которого отображается в колонке
Width	Ширина колонки в пикселах
Font	Шрифт, используемый для отображения текста в ячейках колонки
Color	Цвет фона
Alignment	Способ выравнивания текста в ячейках колонки. Текст может быть выровнен по левому краю (taLeftJustify), по центру (taCenter), по правому краю (taRightJustify)
Title.Caption	Заголовок колонки. По умолчанию в заголовке отображается имя поля
Title.Alignment	Способ выравнивания заголовка. Заголовок может быть выровнен по левому краю (taLeftJustify), по центру (taCenter), по правому краю (taRightJustify)
Title.Color	Цвет фона заголовка колонки
Title.Font	Шрифт заголовка колонки

Вам предлагается следующий дизайн таблицы, повторите его, выполнив настройку компонентов следующим образом:

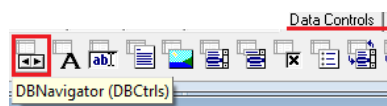
Выделенный объект	Вкладк а окна Object Inspector	Имясвойст ва/ имя события	Действие
DBGrid1	Properties	BorderStyle	bsNone
		Options.dgIndicator	False
		Options.dgColumnResize	False
		Options.dgColLines	False
TColumns (Код)	Properties	Font	8, Полужир, серый
TColumns (Фамилия)	Properties	Font	8, Полужир, красный
TColumns (Имя)	Properties	Font	8, Полужир, синий
TColumns (Класс)	Properties	Font	10, Полужир, красный

TColumns (Год рождения)	ies	Propert	Color Alignment	clMenuHighlight
TColumns (Код)	ies	Propert	Title.Font	14, обычный
TColumns (Фамилия)	ies	Propert	Title.Font	12, Полужир
TColumns (Имя)	ies	Propert	Title.Color	Красный
TColumns (Отчество)	ies	Propert	Title.Font	12, Полужир
TColumns (Год рождения)	ies	Propert	Title.Color	Синий
TColumns (Школа)	ies	Propert	Title.Font	10, Полужир
TColumns (Класс)	ies	Propert	Title.Color	Голубой
TColumns (Год рождения)	ies	Propert	Title.Font	8, Полужир
TColumns (Школа)	ies	Propert	Title.Color	Бирюзовый
TColumns (Класс)	ies	Propert	Title.Font	8, Полужир
TColumns (Класс)	ies	Propert	Title.Color	Сиреневый
TColumns (Класс)	ies	Propert	Title.Font	8, Полужир
TColumns (Класс)	ies	Propert	Title.Color	серый

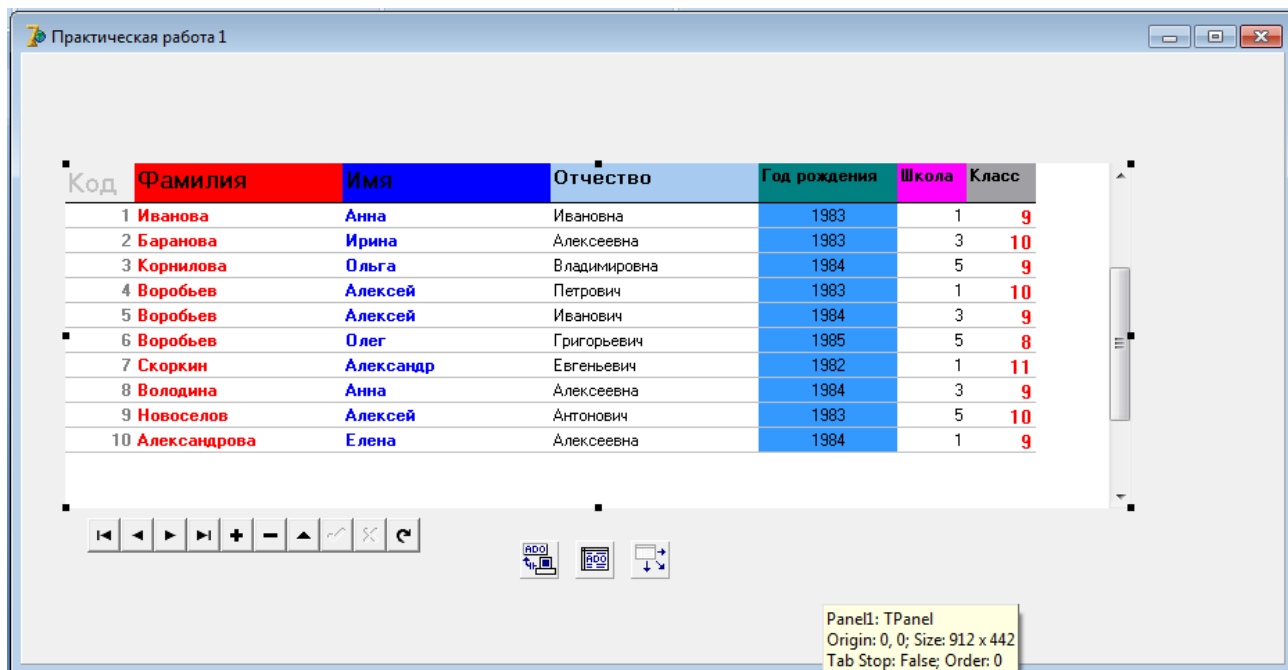


Задание 9. Подключите к таблице компонент добавления и удаления записей.

Добавьте компонент DBNavigator (вкладка DataControls) и обязательно привяжите его к компоненте DBGrid1:



Выделенный объект	Вкладка Object Inspector	Имя свойства/ имя события	Действие
DBNavigator1	Properties	DataSource1	DataSource1



Сохраните проект File/Project As...

Запустите на выполнение проект нажав кнопку Выполнить (Run).

Добавьте новую запись нажав на Навигаторе «+» и сохраните нажав «√».

Откройте свою базу данных созданную, в приложении ACCESS и убедитесь, что запись добавлена.

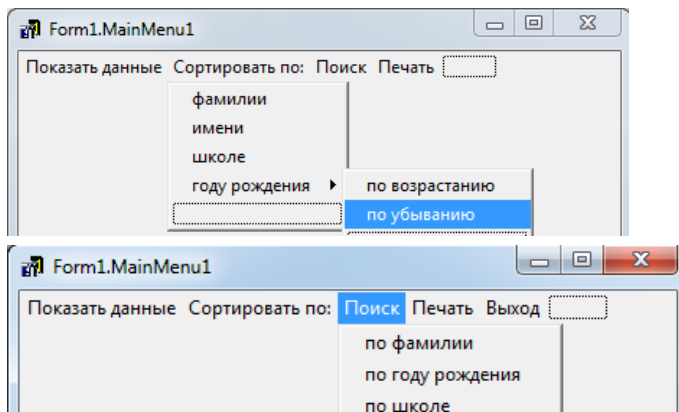
Задание 10. Добавьте на форму Главное меню и организуйте его действия.

В DELPHI имеется возможность создания главного меню: MainMenu (на странице Standard). Основное свойство компонента — Items. Его заполнение производится с помощью Конструктора Меню, вызываемого двойным щелчком на компоненте MainMenu или нажатием кнопки с многоточием рядом со свойством Items в окне Инспектора Объектов. В результате откроется окно, вид которого представлен на рис. В этом окне вы можете спроектировать все меню. При работе в конструкторе меню новые разделы можно вводить, помещая курсор в рамку из точек, обозначающую место расположения нового раздела. Если при этом раздел ввелся не на нужном вам месте, вы можете отбуксировать его мышью туда, куда вам надо. Создайте его и установите следующую структуру:

Показать данные Сортировать по: Поиск Печать Выход

Выделенный объект	Вкладка а окна ObjectInspector	Имя свойства/ имя события	Действие
 MainMenu	Properties	 Items	Показать данные Сортировать по: Поиск Печать

Организуйте подменю у пунктов «Сортировать по:» и «Поиск». Для этого дважды нажмите MainMenu и в свободные поля подменю введите нужные названия используя в свойстве Caption:



Организуйте подменю и для пункта «Году рождения», используя для него команду

Create Submenu

Ctrl+Right

из контекстного меню.

Для пункта меню «Показать данные» необходимо написать код программы таким образом, чтобы только при нажатии на данный пункт меню в таблице Список появлялись данные, а до этого их не было видно.

Выделенный объект	Вкладка Object Inspector	Имя свойства/имя события	Действие
 MainMenu (Показать данные)	Properties	OnClick	<pre> begin if ADOTable1.Active then begin MainMenu1.Items[0].Caption := 'Выключитьтаблицу'; ADOTable1.Active :=false; end else begin MainMenu1.Items[0].Caption :='Отключитьтаблицу'; ADOTable1.Active := true; end; end; </pre>

Запустите на выполнение приложение и проверьте данное действие.

Следующим шагом, организуем сортировку:

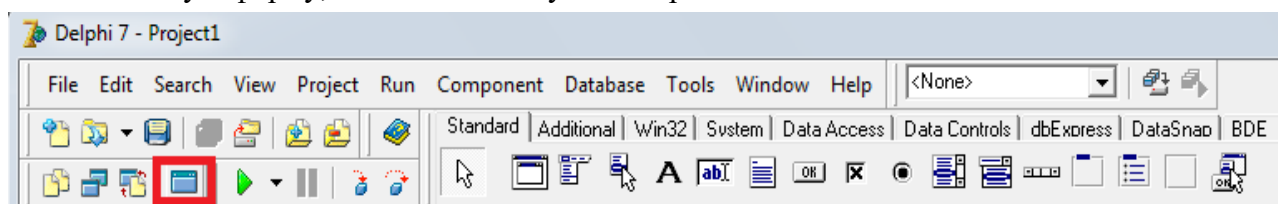
Выделенный объект	Вкладка Object Inspector	Имя свойства/имя события	Действие

 MainMen и (Сортировать по: фамилии))	ies	Propert	ck OnCli	begin ADOTABLE1.Sort := 'Фамилия'; end;
 MainMen и (Сортировать по: имени)	ies	Propert	ck OnCli	begin ADOTable1.sort:='Имя'; end;
 MainMen и (Сортировать по: школе)	ies	Propert	ck OnCli	begin ADOTable1.Sort :='Школа' end;
 MainMen и (Сортировать по: году рождения-по возрастаню)	ies	Propert	ck OnCli	begin ADOTable1.Sort :='[Годрождения] ASC'; end;
 MainMen и (Сортировать по: году рождения-по убыванию)	ies	Propert	ck OnCli	begin ADOTable1.Sort :='[Годрождения] DESC'; end;

Сохраните проект и запустите приложение на выполнение проверьте действие данных пунктов.

Для организации поиска будьте очень внимательны и следуйте шагам представленных ниже в таблице:

создайте новую форму, нажав на кнопку Стандартной панели New Form:



настройте свойства формы:

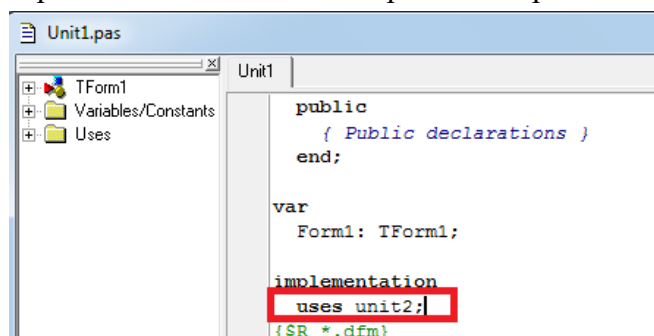
Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/имя события	Действие
Form2	Properties	Caption	Поиск по фамилии
		Height	150
		Width	600

на форме разместите компоненты Label1, Edit1 и Button1 и настройте их свойства:

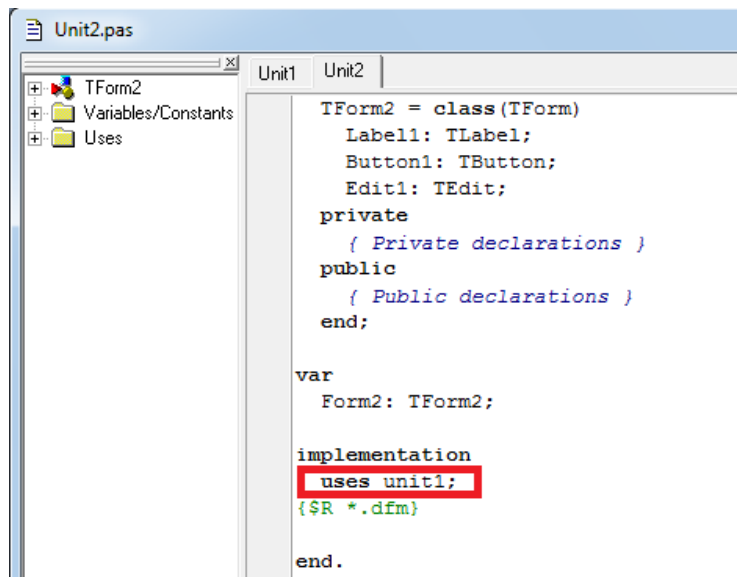
Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/имя события	Действие
Label1	Properties	Caption	ВВЕДИТЕ ФАМИЛИЮ
		Font	10, полужирный, красный
Button1	Properties	Caption	Искать
Edit1	Properties	Text	
Form2	Properties	Color	Небесно-голубой

сохраните Unit2;

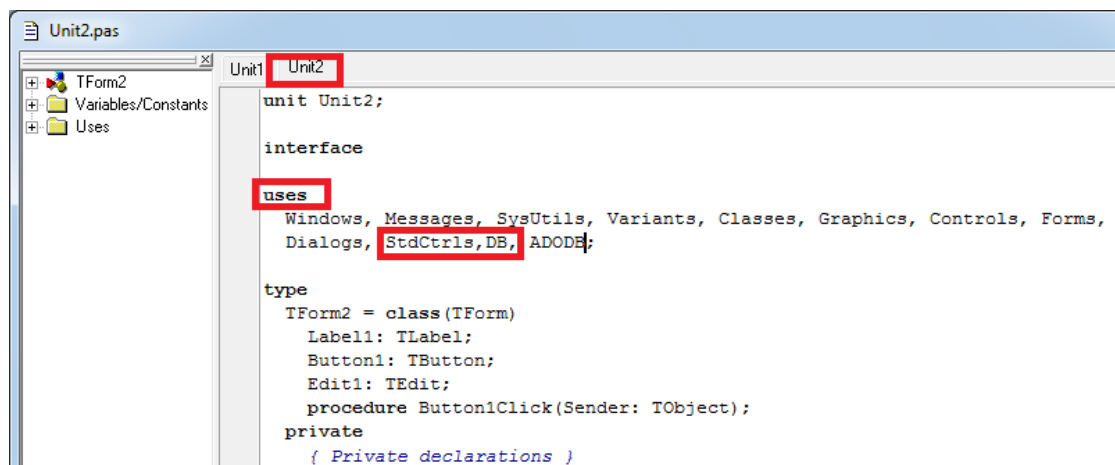
перейдите в окно Unit1 и в разделе implementation подключите форму Unit2:



перейдите в окно Unit2 (Ctrl+F12) и в разделе implementation подключите форму Unit1:



для Unit2 подключите библиотеки работы с базами данных DB и ADODB скопировав их из Unit1:



для Form2 (Shift+F12) организуйте событие на кнопку Button1:

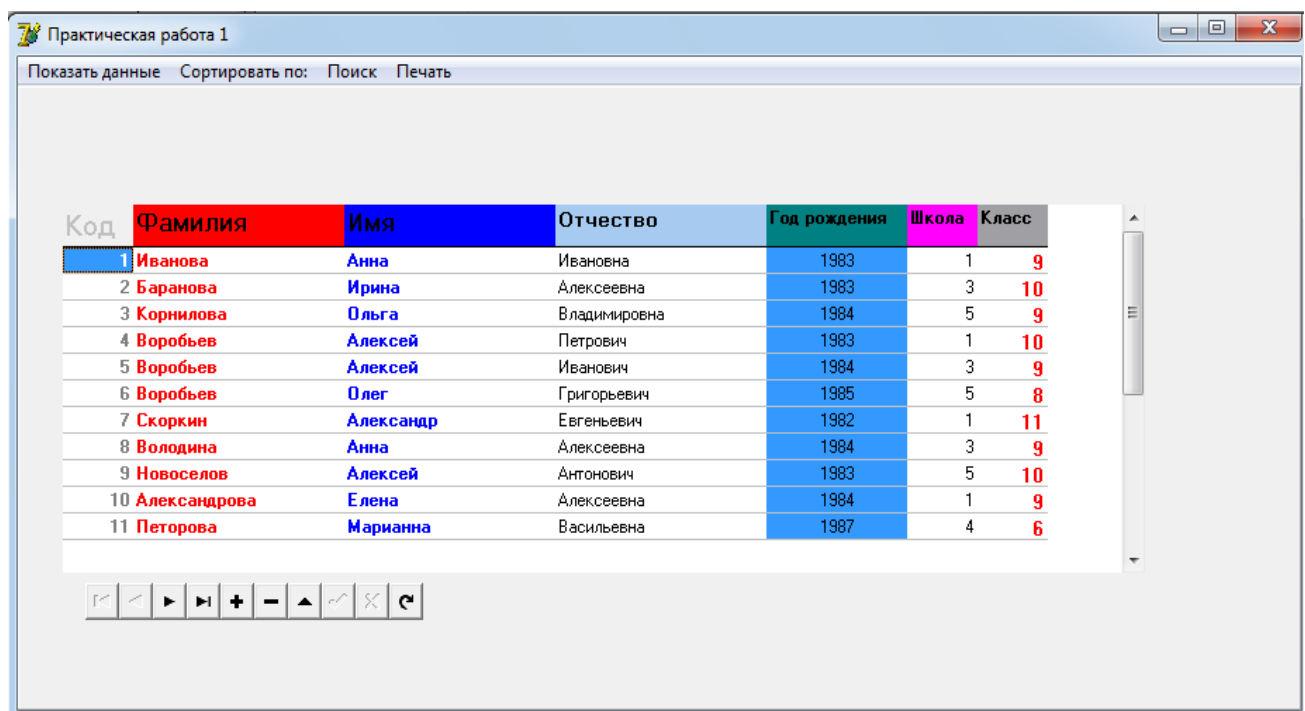
Выделенный объект	Вкладк а окна ObjectInspector	Имя свойства/ имя события	Действие

Button1	ies	ck	OnCli	<pre> var opts: TLocateOptions; begin opts := [loPartialKey]; if Form1.ADOTable1.locate('Фамилия',Edit1.text , opts) then MessageDlg('Запись найдена',mtInformation, [mbOK],0) else MessageDlg('Запись не найдена',mtInformation, [mbOK],0); end; </pre>
---------	-----	----	-------	--

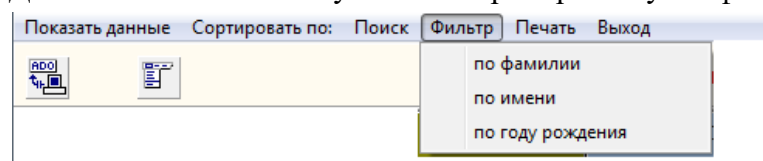
на Form1 открыть двойным щелчком MainMenu и выбрать пункт поиск - по фамилии, при этом вы попадаете в окно Unit1, где на данное событие нужно прописать код открытия модального окна Form2.ShowModal;

сохраните приложение и запустите на выполнение, отработав пункт Поиск – по фамилии.

Для остальных пунктов меню поиск «году рождения» и «школе» повторите аналогичные действия.



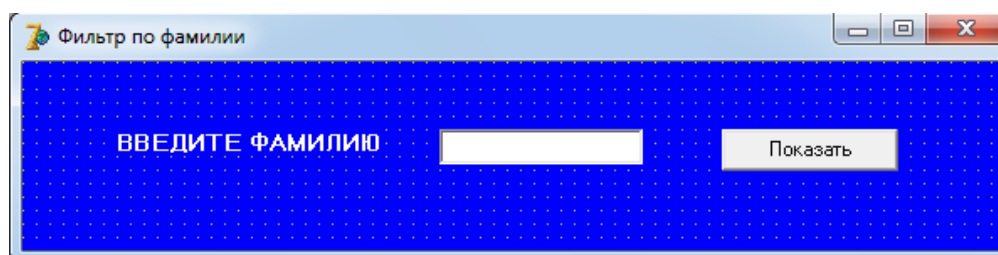
Добавьте в MainMenu пункт Фильтр и организуйте фильтрацию по:



создайте новую форму и сохраните Unit5.

на форме разместите компоненты Label1, Edit1 и Button1 и настройте их свойства:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Label1	Properties	Caption	ВВЕДИТЕ ФАМИЛИЮ
		Font	10, полужирный, белый
Button1	Properties	Caption	Показать
Edit1	Properties	Text	
Form5	Properties	Color	Синий
		Height	150
		Width	600
		Caption	Фильтр по фамилии



перейдите в окно Unit1 и в разделе implementation подключите форму unit5;
перейдите в окно Unit5 (Ctrl+F12) и в разделе implementation подключите форму unit1;
для Unit5 подключите библиотеки работы с базами данных DB и ADODB скопировав их из Unit1;

для Form5 (Shift+F12) организуйте событие на кнопку Button1:

Выделенный объект	Вкладка окна ObjectInspector	Имя свойства/ имя события	Действие
Button1	Properties	OnClick	<pre> with Form1.ADOTable1 do begin Filtered := False; Filter := 'Фамилия like "' + Edit1.Text + '%"'; Filtered := True; end; </pre>

на Form1 открыть двойным щелчком MainMenu и выбрать пункт фильтр - по фамилии, при этом вы попадаете в окно Unit1, где на данное событие нужно прописать код открытия модального окна Form5.ShowModal;

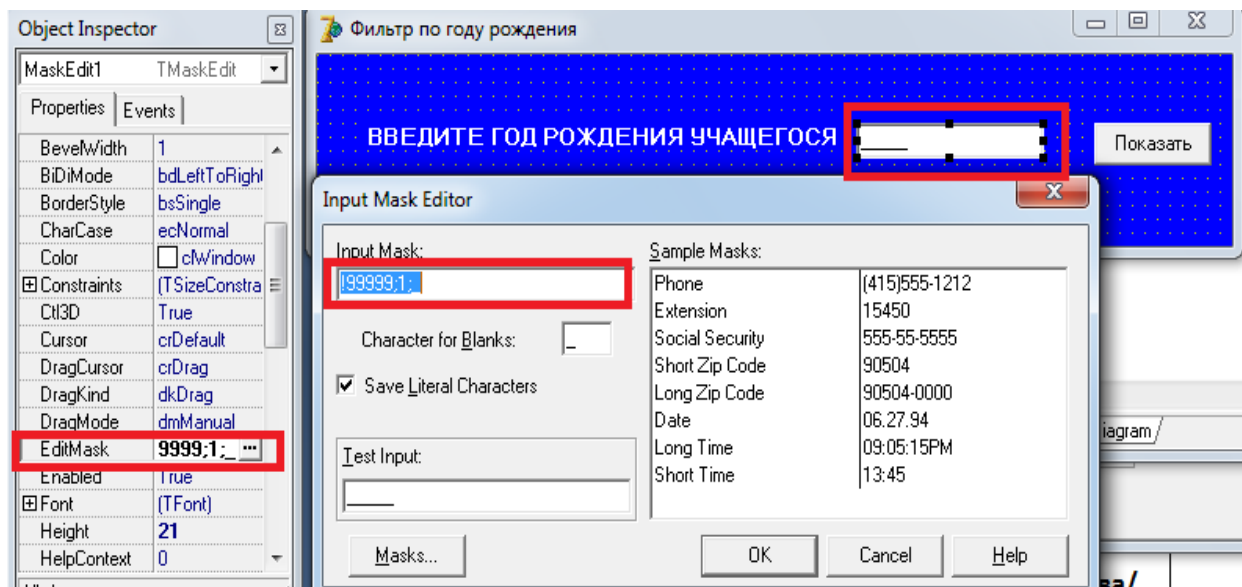
сохраните приложение и запустите на выполнение, отработав пункт Поиск – по фамилии.

Для пункта меню выполните фильтрацию по полю «Имя» повторив аналогичные действия.

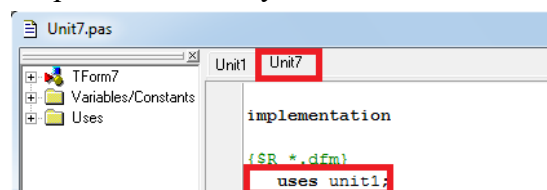
Фильтр в поле «Год рождения» имеет свои особенности:

создайте новую форму Form7. Установите на нее компоненты Label1 и Button1;

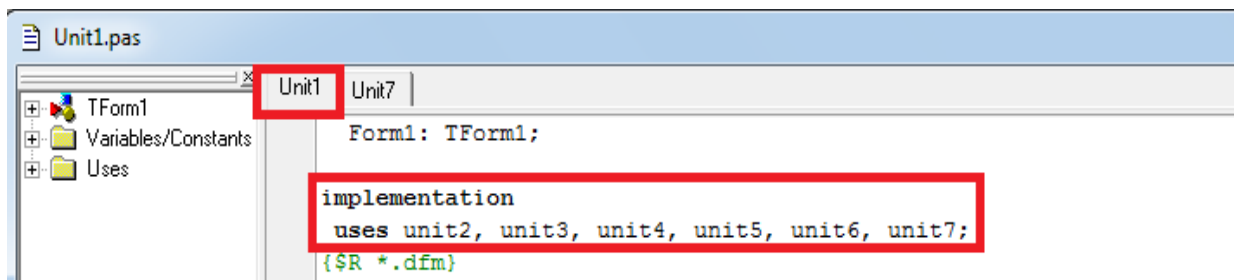
на Form7 размещается компонента MaskEdit (вкладка Additional) и указываются ее свойства:



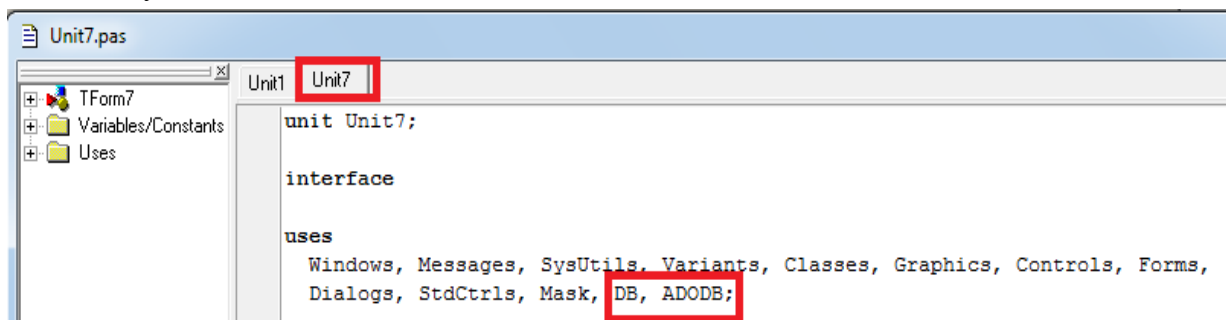
сохраните Unit7 и укажите для нее подключение Unit1:



для Unit1 укажите подключение Unit7:



для Unit7 укажите подключение библиотек DB, ADODB:

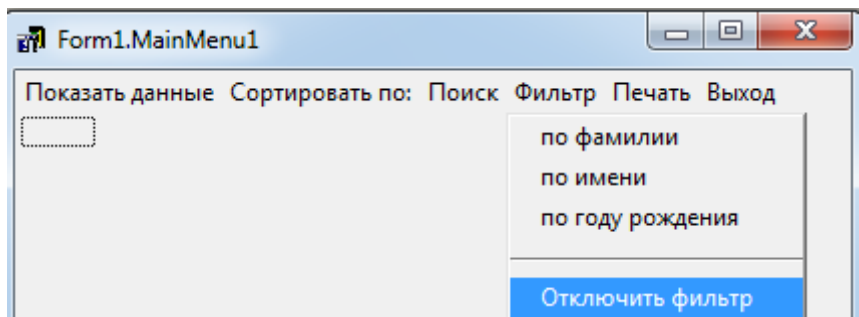


поле «Год рождения» имеет тип данных – числовой, поэтому в коде программы для него необходимо использовать следующий код:

Выделенный объект	Вкладок а окна	Имя свойства/	Действие
-------------------	----------------	---------------	----------

	ObjectInspector	имя события	
Button1	Properties	OnClick	<pre> var s: string; begin with Form1.ADOTable1 do begin Filtered := False; s := '[Годрождения] = ' + MaskEdit1.Text; Filter := s; Filtered := True; end; end; </pre>

В компонент MainMenu добавьте пункт «Отключить фильтр»:



для добавления разделительной линии достаточно в свободном пункте меню установить свойство Caption – «-» и нажать Enter.

активизируйте Form1 и нажмите на дважды на пункт «Отключить фильтр» и в окне Unit1 для данного события Onclick укажите ADOTable1.Filtered := False;

сохраните проект и выполните его запуск, протестируйте фильтрацию данных.

Организуйте действие при нажатии пункта меню «Печать»:

подключите библиотеки OleServer и ComObj:

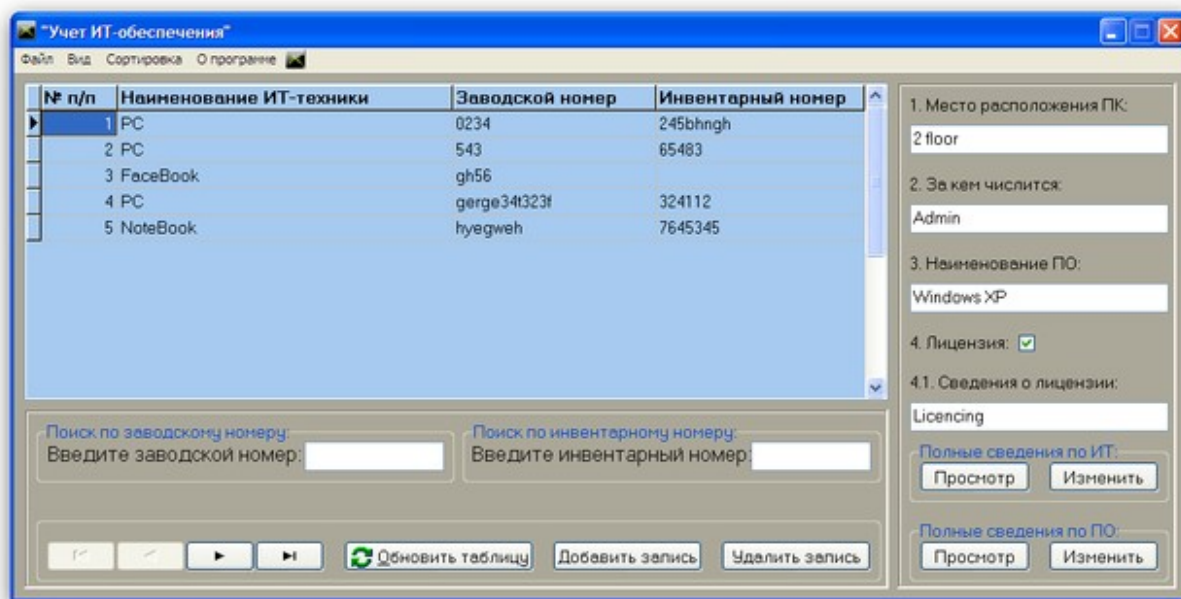
Выделенный объект	Вкладка ObjectInspector	Имя свойства/ имя события	Действие
-------------------	----------------------------	------------------------------------	----------

Form1.MainMenu1 (Печать)	Properties	OnClick	<pre> var col: integer; ExcelApp: OLEVariant; begin ExcelApp := CreateOleObject('Excel.Application'); ExcelApp.WorkBooks.Add; ExcelApp.WorkBooks[1].Worksheets[1].Name :='Список'; ExcelApp.Visible := true; //вывод названия колонок ExcelApp.Cells.Item[1,1].Value:= DBGrid1.Columns[1].Title.Caption; //вывод полей Grid DBGrid1.DataSource.DataSet.First; for col := 0 to DBGrid1.DataSource.DataSet.FieldCount-1 do begin ExcelApp.Cells.Item[col+2,1].Value:=ADOTa ble1.FieldByName('Фамилия').value; //записываю значение полей ExcelApp.Cells.Item[col+2,2].Value:=ADOTa ble1.FieldByName('Имя').value; ExcelApp.Cells.Item[col+2,3].Value:=ADOTa ble1.FieldByName('Отчество').value; DBGrid1.DataSource.DataSet.Next; end; end; </pre>
-----------------------------	------------	---------	--

Организуйте действие при нажатии пункта меню «Выход»:

Выделен ный объект	Вк ладка окна Ob jectInspect or	Имя свойства/ имя события	Действие
Form1.MainMenu1 (Выход)	Properties	OnClick	Close;

Бесплатная курсовая. База данных - Учет техники в организации.



Практическая работа №20

Тема: Создание базы данных, состоящей из двух таблиц в СУБД ACCESS и подключение их к среде DELPHI

Цель работы:

научиться создавать таблицы базы данных в режиме таблицы;

научиться создавать самостоятельное ключевое поле в приложении ACCESS;

научиться создавать формы для ввода данных в приложении ACCESS;

закрепить навыки по заполнению и редактированию таблиц базы данных;

познакомиться с технологией связывания таблиц и создания связи «один-ко-многим» в приложении ACCESS;

освоить приемы размещения и настройки окон редактирования элементов базы данных в среде DELPHI;

освоить приемы размещения и настройки таблицы для вывода данных в среде DELPHI;

освоить приемы формирования и вывода значений вычисляемого столбца в среде DELPHI.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения WINDOWS; система программирования DELPHI.

Этапы работы

Проектирование структуры базы данных.

Создадим более сложный вариант базы данных. В ней будут две таблицы: Список и Группы.

Конструирование пустых таблиц базы данных.

Воспользуемся новым способом изготовления таблиц. Таблицы будем создавать в режиме таблицы. В таблице Список будет 7 полей (код, фамилия, имя, отчество, год рождения, школа, класс, номер группы). Номера групп и фамилии преподавателей будут храниться в отдельной таблице Группы в виде двух столбцов.

Создание схемы базы данных.

В данном случае таблицы связаны связью «один-ко-многим». Это значит, что в таблице Группы каждое значение может встречаться только один раз, а в таблице Список – сколько угодно (несколько человек могут быть из одной группы). Связи следует устанавливать при пустых таблицах. Если таблицы заполнены, могут возникнуть проблемы при создании связей и свойств связей. Для связи в обеих таблицах должны быть ключевые поля. В таблице Список – поле Код, в таблице Группы – поле ID_kod.

Ввод данных в таблицы.

Создадим форму для ввода данных и воспользуемся ею. При наличии связанных таблиц имеет значение порядок заполнения их значениями.

Ход работы

Задание 1. Откройте учебную базу, изготовленную на прошлом занятии

Порядок работы

Откройте диск C: /папку PracticumBD_Delphi.

Создайте в исходной папке новую папку 02BD.

Задание 2. Создайте новую базу данных и изготовьте структуру таблицы с информацией о студентах «Компьютерной школы».

Порядок работы

Запустите приложение Microsoft Access.

Создайте новую базу данных ADO_02.accdb.

Открывается вкладка Таблица и автоматически предлагается новая таблица, для того чтобы организовать структуру вашей таблицы, необходимо перейти в режим Конструктор (Вид/Конструктор).

Введите в появившееся диалоговое окно имя таблицы «Список»

Сохраните таблицу, щелкнув на кнопке Сохранить.

Задание 3. Создайте таблицу Группы.

Порядок работы

Выберите вкладку Создание, группа Таблицы – пиктограмма Таблица.

Появится пустая таблица, поля которой не определены и не имеют названия. Тип поля будет выбран автоматически в зависимости от введенной информации.

Переименуйте Поле 1. Для этого поставьте курсор в ячейку заголовка столбца Поле 1. вызовите контекстное меню и выберите команду Переименовать столбец. Ячейка имени столбца окажется выделенной. Введите название поля ID_kod и нажмите клавишу Enter.

Переименуйте Поле 2. Для этого поставьте курсор в ячейку заголовка столбца Поле 2. вызовите контекстное меню и выберите команду Переименовать столбец. Ячейка имени столбца окажется выделенной. Введите название поля Группа и нажмите клавишу Enter.

Переименуйте Поле 3. Для этого поставьте курсор в ячейку заголовка столбца Поле 3. вызовите контекстное меню и выберите команду Переименовать столбец. Ячейка имени столбца окажется выделенной. Введите название поля Преподаватель и нажмите клавишу Enter.

Сохраните таблицу с именем Группы, щелкнув по кнопке Сохранить.

Перейдите в режим Конструктор, выбрав вкладку Главная/Вид/Конструктор. Сделайте поле ID_kod ключевым, поместив курсор на имя этого поля и щелкнув по кнопке Ключевое поле. Тип данных поля ID_kod задайте «счетчик», выбрав мышкой из ниспадающего списка.

Установите: поле Учебная группа – тип поля «числовой», поле Преподаватель – тип поля «текстовый»

Щелкните по кнопке Сохранить. Закройте таблицу (при сохранении таблицы вопросов не появится, так как имя таблицы уже задано).

Задание 4. Создайте таблицу Список.

Порядок работы

Выберите вкладку Создание группы Таблица и нажмите на пиктограмму Таблица.

Появится пустая таблица, поля которой не определены и не имеют названия.

Переименуйте поля по аналогии с предыдущим заданием. Данные возьмите из таблицы 1.

Таблица 1

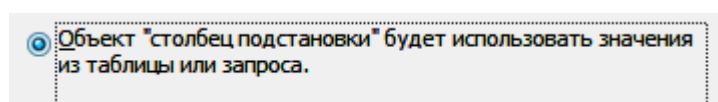
Старое название	Новое название
Поле 1	Код
Поле 2	Фамилия
Поле 3	Имя
Поле 4	Отчество
Поле 5	Год рождения
Поле 6	Школа
Поле 7	Класс

Сохраните таблицу с именем Список, щелкнув на кнопке Сохранить.

Перейдите в режим Конструктор и посмотрите, как заданы поля. Сделайте поле Код ключевым, поместив курсор на имя этого поля и нажав кнопку Ключевое поле. Тип поля Код – счетчик, полей Фамилия, Имя, Отчество – текстовые, полей Год рождения, Школа, Класс – числовые.

Создайте новое поле Учебная группа, тип поля – Мастер подстановок. Следуя, шагам мастера установите:

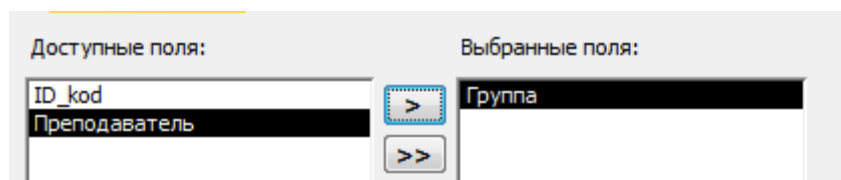
Шаг 1



Шаг 2



Шаг 3



Шаг 4

Порядок сортировки элементов – не меняйте.

Шаг 5

Ширину столбца подстановки – не меняйте

Шаг 6

Название столбца – оставьте Учебная группа

Шаг 7

Нажмите Готово

Шаг 8

На вопрос создать связь – ответьте утвердительно

Сохраните изменения, щелкнув по кнопке Сохранить.

Закройте таблицу.

В результате этой работы вы получите две связанные таблицы. Нужно создать определить вид связи.

Задание 5. Создайте схему данных.

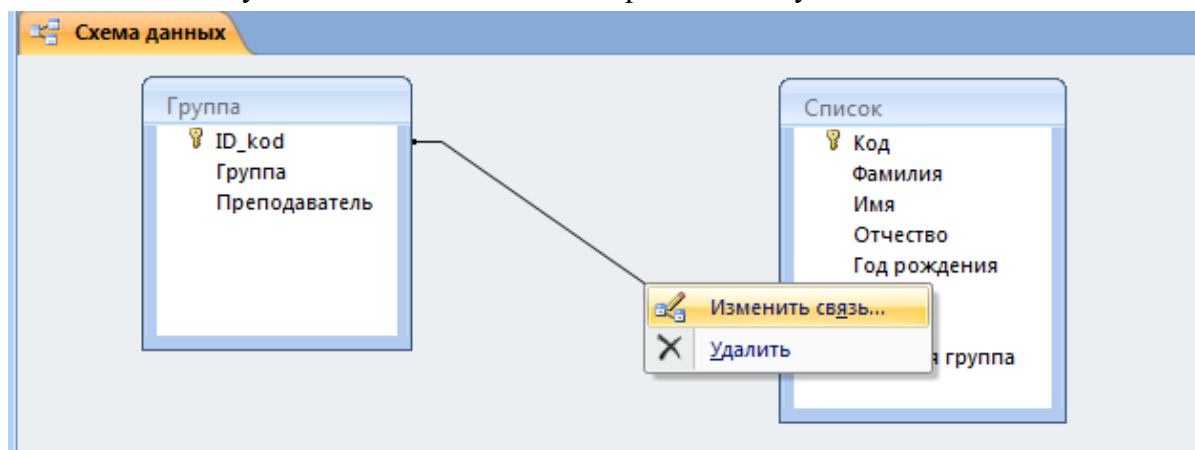
Порядок работы

Щелкните по кнопке Схема данных (вкладка Работа с базами данных).

В появившемся окне Добавление таблицы выделите таблицу Группы и щелкните по кнопке Добавить. Выделите таблицу Список и щелкните по кнопке Добавить. В окне Схема данных появится условный вид этих таблиц и между ними неопределенная связь. Щелкните по кнопке Закройте окна Добавление таблицы.

Увеличьте окно таблицы Список так, чтобы были видны все поля.

Нажмите ПКМ на условном виде связи и выберите команду Изменить связь.



Появится диалоговое окно Связи, представленное на рис. 2.

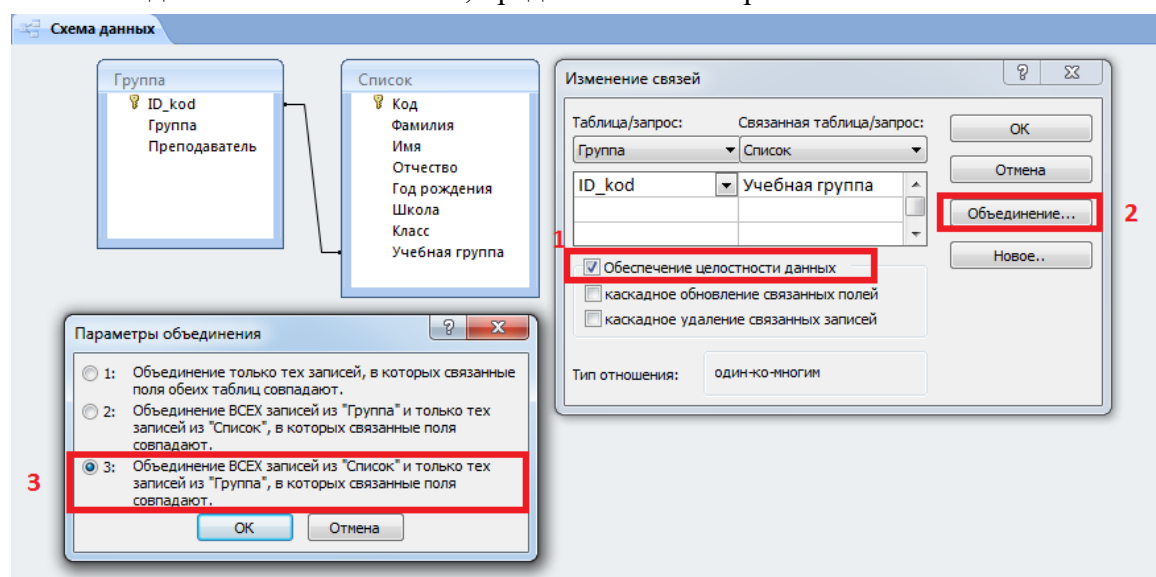


Рис. 2.

Включите значок Обеспечение целостности данных. Это невозможно будет сделать, если типы обоих полей заданы не одинаково.

Включите значок Каскадное обновление связанных полей. Это приведет к тому, что при изменении номера группы в таблице Группы автоматически изменится соответствующий номер в таблице Список.

Включите значок Каскадное удаление связанных полей. Это приведет к тому, что при удалении записи с номером группы в таблице Группы будут удалены все записи из таблицы Список, в которой стояли соответствующие номера групп.

Нажмите кнопку Объединение, откроется диалоговое окно Параметры объединения установите для связи «один-ко-многим» переключатель в положение 3. Нажмите ОК.

Щелкните по кнопке Создать. Появится связь «один-ко-многим». Схема данных представлена на рис. 3.

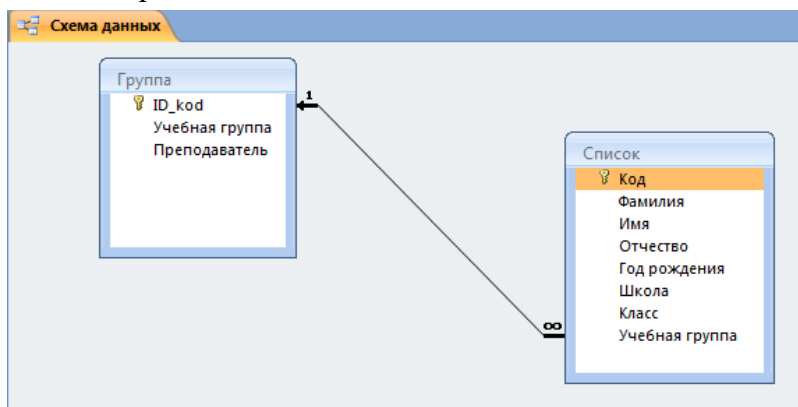


Рис. 3.

Закройте схему данных, утвердительно ответив на вопрос о сохранении схемы данных.

Задание 6. Заполните таблицу Группы значениями

Порядок работы:

Откройте таблицу Группы в режиме таблицы

Заполните ее записями из таблицы 2.

Таблица 2.

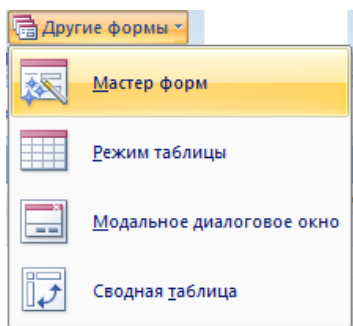
Учебная группа	Преподаватель
101	Верзаков С.А.
102	Белоусов А.И.
103	Масалова В.А.
104	Новикова Е.В.
105	Зачесова Т.П.

Сохраните таблицу и закройте ее.

Задание 7. Создайте формы для ввода данных.

Порядок работы

Выберите вкладку Создание/ группу Формы /раскрывающийся список Другие формы/ Мастер форм



Появится диалоговое окно, в котором следует выбрать в поле Таблицы и запросы источник данных – Таблица: Список.

Перенести из поля Доступные поля все поля в правую часть окна Выбранные поля.

Перейдя в следующий шаг Мастера выберите внешний вид формы «в один столбец».

В следующем шаге Мастера выберите требуемый стиль «Бумажная».

Далее задайте имя формы «Список» и выберите действие «Открыть форму для просмотра и ввода данных».

Нажмите кнопку «Готово». Построение формы на этом закончено.

Задание 8. Добавьте в базу данных записи, используя форму.

Порядок работы:

Заполните базу данных, представленную в таблице 3.

Таблица 3.

од	Фамилия	Имя	Отчество	Дата рождения	Школа	Класс	Учебная группа
	Чернов	Кристина	Ивановна	1984	1	9	101
	Терещенко	Инна	Алексеевна	1983	3	10	103
	Истратов	Максим	Владимирович	1984	5	9	101
	Бондарь	Ольга	Петровна	1983	1	10	104
	Новоселов	Алексей	Иванович	1984	3	9	105

Сохраните введенные данные. Имя формы – Список. Закройте форму.

Откройте таблицу Список. Убедитесь, что в таблице появились новые записи.

Сохраните текущую таблицу. Нажав на кнопку Сохранить.

Закройте текущую таблицу.

Задание 9. Проверьте каскадное обновление связанных полей.

Порядок работы:

Откройте таблицу Группы.

Исправьте учебные группы на 201, 202, 203, 204, 205.

Сохраните таблицу.

Закройте таблицу Группы.

Откройте таблицу Список.

Убедитесь, что значения групп изменились.

Закройте таблицу Список.

Задание 10. Проверьте каскадное удаление связанных полей.

Порядок работы:

Откройте таблицу Группы.

Удалите первую запись (всю строку выделите и нажмите на клавишу Delete).

Согласитесь с проверочным вопросом.

Закройте таблицу Группы.

Откройте таблицу Список.

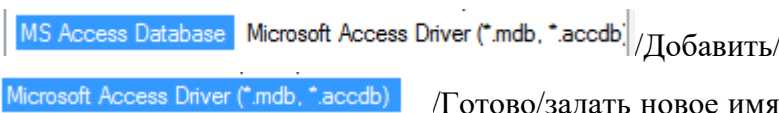
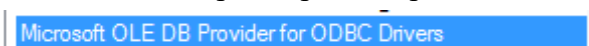
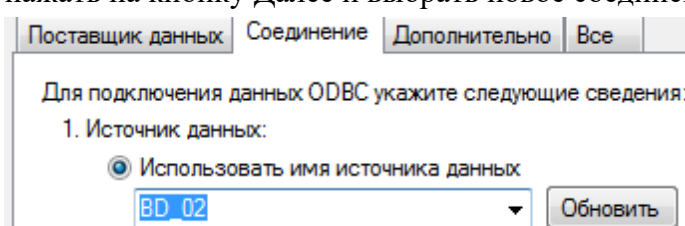
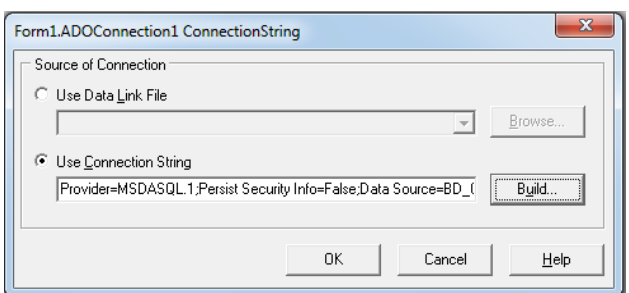
Убедитесь, что исчезли записи с номером группы 201.

Закройте таблицу Список.

Закройте базу данных ADO.accdb.

Задание 11. Настройте соединение с новой таблицей Список

Порядок работы:

шаг 1	Пуск / Панель управления /Администрирование / Источники данных (ODBC)  исходную базу ADO_02. accdb	
шаг 2	Открыть новый проект в среде Delphi и сохранить в той же папке, где хранятся данные: C:/ PracticumBD_Delphi/02BD	
шаг 3	Дайте название объекту Form.Caption=Практическая работа 2 Установите необходимые панели и контейнеры для новой базы данных (Panel1 для Label1 названия нового проекта Компьютерная школа, Panel2 – для DBGrid1, Panel3 для вывода дополнительных параметров)	
шаг 4	 ADOConnection1	Для этого необходимо дважды на него нажать. В появившемся окне выбрать провайдера  нажать на кнопку Далее и выбрать новое соединение  Проверить соединение Появится окно 

		Нажать OK		
		Properties	LoginPrompt	False
		Properties	Connected	True
5 шаг	 ADOTable1	Properties	Connection	ADODConnection
			TableName	Список
6 шаг	 DataSource1	Properties	DataSet	ADOTable1
7 шаг	DBGid1	Properties	DataSource	DataSource1
8 шаг	 ADOTable1	Properties	Active	True
9 шаг	DBGid1	Вызовите контекстное меню для компонента DBGid1, выберите команду Columns Editor, нажмите на пиктограмму AddAllFields. В компоненте DBGid1 появятся ваши данные		
10 шаг	 ADOTable1	Properties	Active	True

Настройте ширину столбцов компонента DBGid1:

Название полей	Свойство Width
Код	50
Фамилия	150
Имя	150
Отчество	150
Год рождения	100
Школа	50
Класс	50
Учебная группа	85

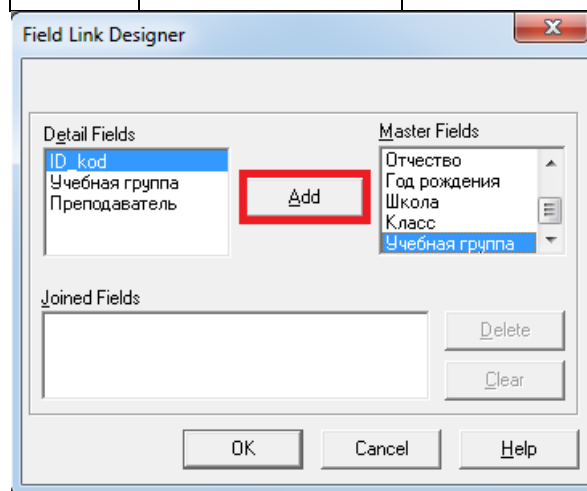
Практическая работа 2							
Компьютерная школа "Вундеркинд"							
Код	Фамилия	Имя	Отчество	Год рождения	Школа	Класс	Учебная группа
1	Чернова	Кристина	Ивановна	1984	1	9	1
2	Терещенко	Инна	Алексеевна	1983	5	9	3
3	Истратов	Максим	Владимирович	1984	5	9	1
4	Бондарь	Ольга	Петровна	1983	1	10	4
5	Новоселов	Алексей	Иванович	1984	3	9	5

Задание 12. Подключите таблицу Группы к таблице Список

Порядок работы:

шаг		es	Properti	Connection	ADODConnection
				TableName	Группа

		ADOTabl			
	2	e2			
шаг	2	 DataSours	Properti	DataSet	ADOTable2
		e2	es		
шаг	3	 ADOTabl	Properti	MasterSour	DataSource1
		e2	es	se	Active
					True
			ds	MasterFiel	ID_kod (таблицы Группа) – Учебная группа (таблицы Список)

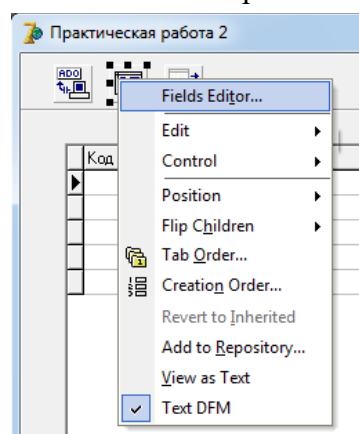


Связь между таблицами создана, необходимо создать вычисляемое поле, чтобы в таблицы Список мастером подстановки из таблицы Группы добавлялись, менялись записи.

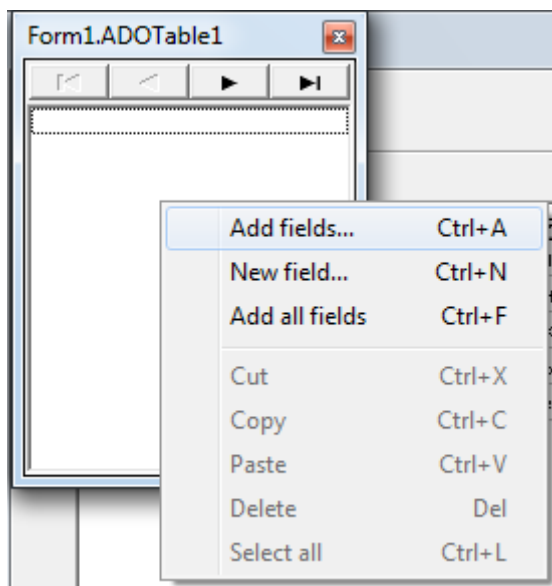
Задание 13. Создайте вычисляемое поле для таблицы Список

Порядок работы:

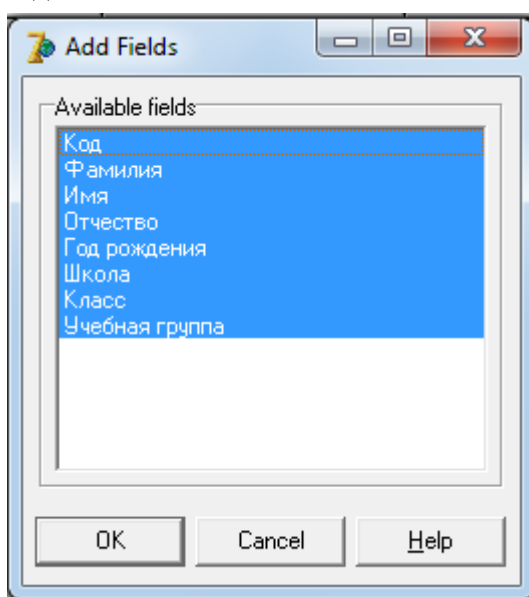
Активизируйте компонент ADOTable1 и вызовите для него контекстное меню, в появившемся окне выберите команду Fields Editor...:



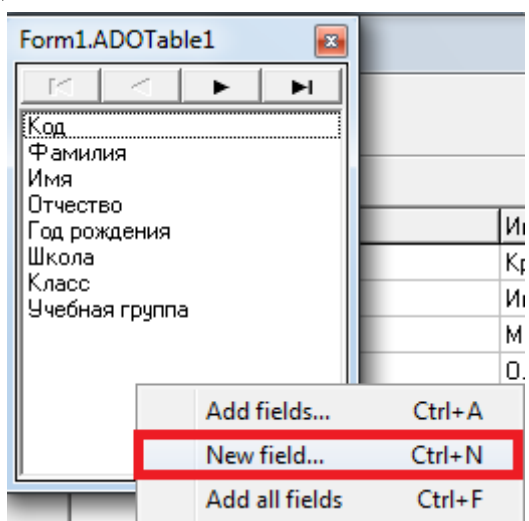
Вызовите контекстное меню для появившегося окна и выберите команду Addfields:



В диалоговом окне Available fields выберите все поля и нажмите ОК.



Для данного диалогового окна Available fields вызовите снова контекстное меню и выберите команду New fields... - это даст вам возможность создать новое вычисляемое поле в таблице Список.



В диалоговом окне New Fields задать новое поле группа, указав связи:

New Field

Field properties:
Name: группа Component: ADOTable1Field
Type: String Size: 85

Field type:
☐ Data ☐ Calculated ☒ Lookup

Lookup definition:
Key Fields: Учебная группа Dataset: ADOTable2
 Lookup Keys: ID_kod Result Field: Учебная группа

OK Cancel Help

Обновите соединение таблиц

1 шаг	 ADOTable1	Properties	Active	True
2 шаг	 ADOTable2	Properties	Active	True
3 шаг	DBGid1	Обновить столбцы	Вызвать контекстное меню, удалить столбцы и добавить обновленные	

Добавьте компонент Navigator и присоедините к таблице Список:

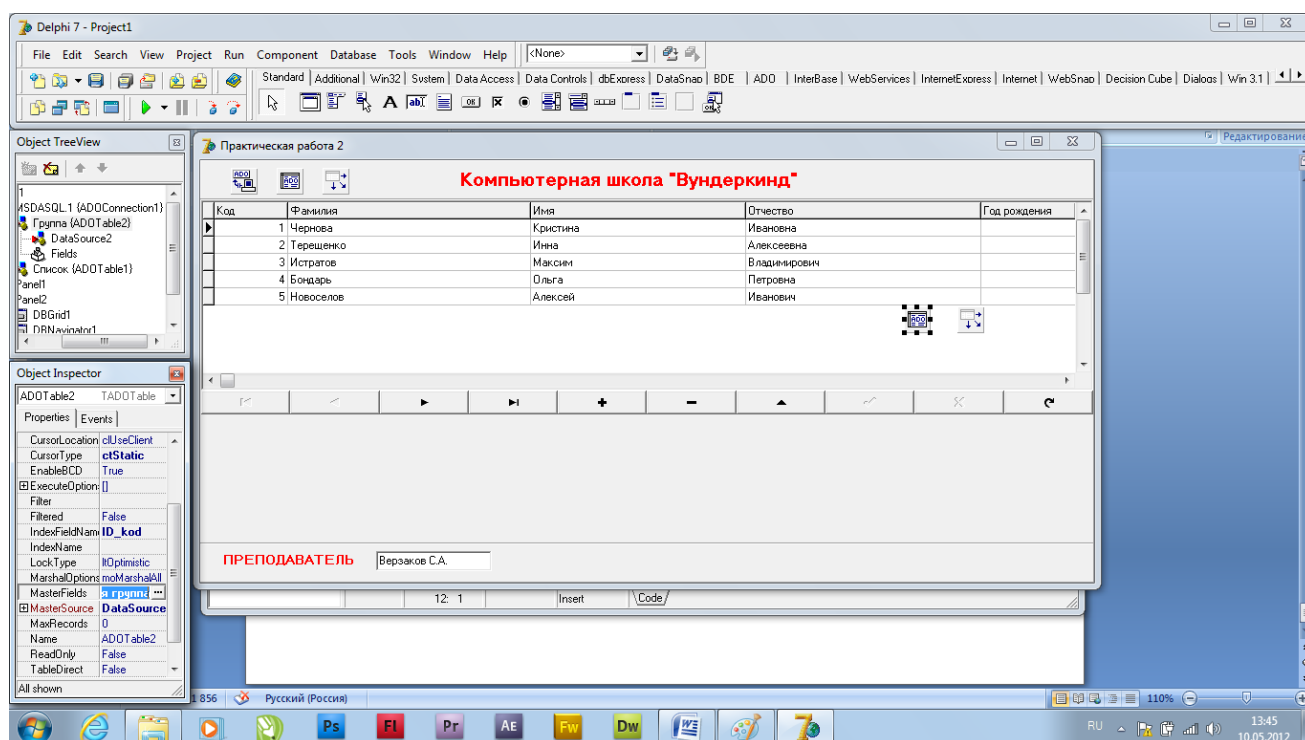
Выделенный объект	Вкладка окна Object Inspector	Имя свойства/имя события	Действие
DBNavigator1	Properties	DataSource	DataSource1
DBGrid1	Properties	Align	alTop
Form1	Properties	WindowState	Maximized
DBGrid1 (Учебная группа)	Properties	Visible	False
DBNavigator1	Properties	Align	alTop

Запустите приложение на выполнение и проверьте возможность создания и корректировки текстовых элементов записей базы данных. Создайте несколько записей.

Добавьте компоненты Label1 и DDEdit1 для вывода отдельно фамилии преподавателя из таблицы Группы при выборе фамилии ученика.

Label1	Properties	Caption	Преподаватель
		Font	10, полужирный, красный

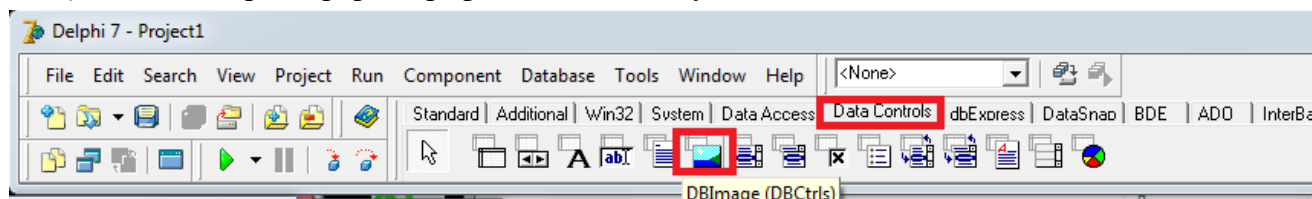
DBEdit1	s	e	DataSours
			DataFields
			Font
 ADOTabl e2	s	rse	DataSource2
			Преподаватель
		ds	10, полужирный, красный
			DataSource1
			ID_kod – Учебная группа



Задание 14. Добавьте к таблице Список поле Фотография и выведите ее на форму проекта. Откройте базу данных ADO_02.accdb.

В таблице Список в режиме Конструктора создайте дополнительное поле Фотография (тип данных - Поле объекта OLE).

Откройте свой проект (если он закрыт) и установите компонент DBImage (вкладка DataControl) на Panel2 в размер фотографии и компоненту Button1 для показа объекта:



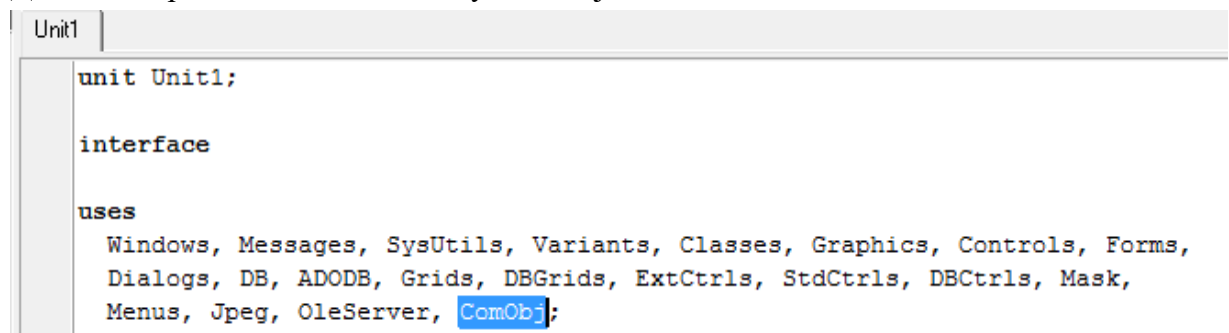
Выделенный объект	Вкладка а окна Object Inspector	Имя войства/ имя события	Действие
-------------------	--	-----------------------------------	----------

Button1	es	ck	if OpenFileDialog1.Execute then begin ADOTable1.Edit; ADOTable1DSDesigner8.LoadFromFile(OpenDialog1.FileName); TBlobField(ADOTable1.FieldByName('Фотография')).LoadFromFile(OpenDialog1.FileName); ADOTable1.Post; end;
---------	----	----	---

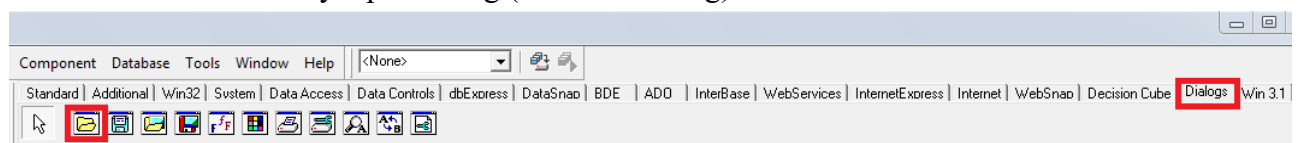
Указать компоненты DBImage1 следующие свойства:

Выделенный объект	Вкладка Object Inspector	Имя свойства/ имя события	Действие
DBImage1	Properties	DataField	Фотография
		DataSource	DataSiourse1

Добавьте в раздел uses библиотеку ComObj для окна Unit1:



Установите компоненту OpenFileDialog (вкладка Dialog)



Запустите приложение на выполнение и для добавления фотографии нажмите на кнопку Добавить фотографию. Не забывайте что в проект системы Delphi добавляются только графические файлы с расширением .bmp. Укажите путь к графическому объекту.

Если размеры графических файлов не подходят, подгоните размер фотографии с помощью программы Paint.

Практическая работа №21

Тема: Создание базы данных, состоящей из трех таблиц

Цель работы:

- научиться создавать таблицу базы данных с помощью Мастера таблиц;
- закрепить навыки по добавлению и удалению записей;
- закрепить навыки по заполнению и редактированию таблиц базы данных;

научиться использовать фильтр в таблице.

Необходимое программное обеспечение: установка и задание необходимых параметров программного обеспечения WINDOWS; система программирования **DELPHI**.

Этапы работы

Проектирование структуры базы данных.

Создадим более сложный вариант базы данных. В ней будет три таблицы: Список, Группы и Личные данные.

Конструирование пустых таблиц базы данных.

Создадим новую таблицу Личные данные. Структуру таблиц Список и Группы скопируем из базы прошлого занятия.

Создание схемы базы данных.

В данном случае таблицы Группы и Список объединены связью «один-к-одному». Таблицы Группы и Личные данные прямо не связаны.

Ввод данных в таблицы.

Создадим форму для ввода данных и воспользуемся ею.

Использование базы данных для практических задач.

Задание 1. Откройте учебную базу данных, созданную на прошлом занятии.

Порядок работы:

Вызовите программу Access. Перед вами откроется окно системы управления базами данных, в котором появится меню.

В разделе окна Открыть последнюю базу данных выделить мышкой необходимую базу. Вы мгновенно попадаете в нужную структуру базы данных.

Задание 2. Откройте данные в таблице Группы

Порядок работы:

Откройте таблицу Группы, нажав на ней дважды лкм.

Добавьте недостающие записи. Исправьте существующие. Таблица должна выглядеть так, как представлено в таблице 1.

Закройте таблицу предварительно сохранив ее.

Учебная группа	Преподаватель
101	Верзаков С.А.
102	Белоусов А.И.
103	Масалова В.А.
104	Новикова Е.В.
105	Зачесова Т.П.

Задание 3. Удалите все записи таблицы Список, оставив ее структуру.

Порядок работы:

Откройте таблицу Список.

Выделите все записи.

Нажмите клавишу Delete. Щелкнув по кнопке Да в вопросе о подтверждении удаления.

Закройте таблицу, сохранив ее.

Задание 4. Создайте таблицу Личные данные с ключевым полем.

Порядок работы:

Создайте таблицу Личные данные используя режим Конструктор со следующими полями:

Название поля	Тип данных
КодСтудента	счетчик
Адрес	текстовый
Номер телефона	текстовый
Word	числовой
Excel	числовой
Access	числовой

Поля Word, Excel, Access – поля в которых будут находиться семестровые оценки по этим темам.

Щелкните по кнопке Сохранить.

Закройте таблицу. В результате вы получите три таблицы, две из которых связаны, а третья нет.

Задание 5. Исправьте схему данных.

Порядок работы:

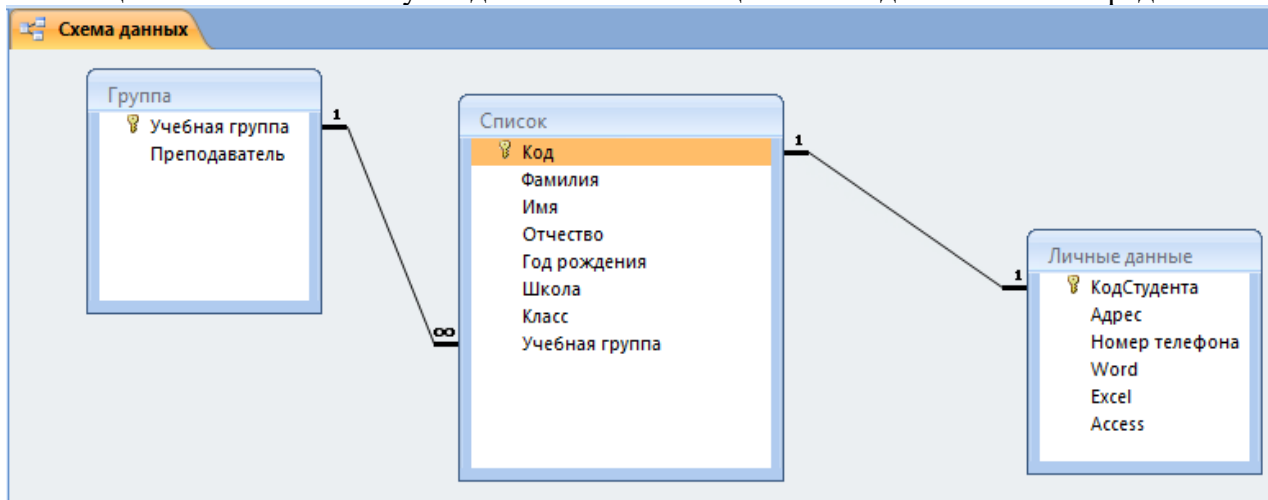
Щелкните на кнопке Схема данных (вкладка Работа с базами данных). Появится диалоговое окно Схема данных.

Вызовите контекстное меню для свободного места окна. Появится окно в котором щелкните по команде Добавить таблицу. В появившемся окне Добавление таблицы выделите таблицу Личные данные и щелкните по кнопке Добавить, а затем – по кнопке Закрывать окна Добавления таблицы.

Поставьте мышку на имя поля Код в таблице Список и, не отпуская кнопку мышки, перетащите ее на КодСтудента в таблице Личные данные поле. Отпустите мышку. Появится диалоговое окно Связи.

Включите флажок Обеспечение целостности данных. Это невозможно сделать, если типы обоих полей заданы неодинаково.

Щелкните по кнопке Создать. Появится связь «один-к-одному». Это значит, что одной записи в таблице Список соответствует одна запись в таблице Личные данные. Схема представлена:



Закройте схему данных сохранив ее.

Теперь встает задача ввода записей одновременно в разные таблицы: Список и Личные данные. ФИО мы храним в одной таблице, а адрес и номер телефона – в другой. Можно конечно, попробовать ввести значения в каждую таблицу отдельно, но удобнее видеть клетки обеих таблиц для ввода данных одновременно. Эту задачу легко решить вводом значений через специально созданную форму, в которой присутствуют поля всех необходимых таблиц. Данные вводим в форму, в результате заполняются таблицы.

Задание 6. Создайте форму для ввода данных.

Порядок работы:

В окне базы данных выберите вкладку Создание, группу Формы, раскрывающийся список Другие формы, команда Мастер форм.

Появится окно создания форм. Выберите все поля из таблицы Список и все поля из таблицы Личные данные. Для этого выберите имя таблицы Список в поле Таблицы и запросы. В результате появляется список полей в окне Доступные поля. Щелкните по кнопке, которая переносит все поля из списка. Затем выберите имя таблицы Личные данные в поле Таблицы и запросы и перенесите все поля данной таблицы.

Щелкните по кнопке Далее.

Оставьте внешний вид формы «в один столбец» (выбран по умолчанию).

Щелкните по кнопке Далее.

Выберите требуемый стиль (например, Обычный). Щелкните по кнопке Далее.

Задайте имя формы: Общая форма. Щелкните по кнопке Готово. В результате вы получите форму, в которой можно менять существующие данные и вводить новые значения. Эти значения будут попадать в ту таблицу, в которую нужно (часть значений – в одну, часть – в другую).

Задание 7. Заполните таблицы данными.

Внесите данные таблицы Список в форму Общая форма (Поля Код и КодСтудента заполняются автоматически):

Код	Фамилия	Имя	Отчество	Год рождения	Школа	Класс	Учебная группа
1	Иванова	Анна	Ивановна	1984	1	9	101
2	Баранова	Ирина	Алексеевна	1983	3	10	102
3	Корнилова	Ольга	Владимировна	1984	5	9	103
4	Воробьев	Алексей	Петрович	1983	1	10	101
5	Воробьев	Алексей	Иванович	1984	3	9	104
6	Воробьев	Олег	Григорьевич	1985	5	8	105
7	Скоркин	Александр	Евгеньевич	1982	1	11	101
8	Володина	Анна	Алексеевна	1984	3	9	102
9	Новоселов	Алексей	Антонович	1983	5	10	103
10	Александров а	Елена	Алексеевна	1984	1	9	101

Внесите данные таблицы Личные данные в форму «Общая форма» (Поля Код и КодСтудента заполняются автоматически):

КодСтудента	Адрес	Номер телефона	Word	Excel	Access
1	Центральная 11-5	51-17-22	5	5	5
2	Солнечная 8-117	51-18-22	4	4	4
3	Сиреневый 7-16	51-19-22	3	4	5
4	Центральная 14-81	51-20-22	5	5	4
5	Сиреневый 7-16	51-21-22	5	4	4
6	Солнечная 2-121	51-22-22	4	4	4
7	Школьная 5-34	51-23-22	5	5	5
8	Центральная 30-7	51-24-22	3	3	3
9	Сиреневый 7-16	51-25-22	3	4	4
10	Солнечная 6-34	51-26-22	5	5	5

Закройте форму, предварительно сохранив ее.

Откройте таблицу Список и убедитесь, что в них появились данные. Закройте таблицу.

Откройте таблицу Личные данные и убедитесь, что в них появились данные. Закройте таблицу.

Задание 8. Добавьте новое поле Портрет (рисунки или фотографии) в таблицу Список

Порядок работы:

Откройте таблицу Список в режиме Конструктор.

Добавьте еще одно поле (вводите имя поля ниже поля Учебная группа), имя которого Портрет, тип данных – Поле объекта OLE, общие свойства поля оставить по умолчанию.

Щелкнуть по кнопке Сохранить.

Перейдите в режим таблицы, щелкните ПКМ по клетке, где должно быть значение поля Портрет

Выполните команду Вставить объект/Создать из файла/Обзор и найдите файл с фотографией.

Для других студентов также вставьте фотографию в поле Портрет.

Задание 9. Научитесь использовать фильтр.

Порядок работы:

Выражения в фильтре могут состоять из точечных значений, которые Access использует для сравнения в том виде, в котором они вводятся. Числа вводятся без ограничителей, например 22.

Текст должен быть заключен в кавычки, например «Александров». Даты ограничиваются символами #, например, #10/01/99#. Элементы выражения могут быть связаны операторами:

арифметическими: *, +, -, /, ^

сравнения: <, <=, >, >=, =, <>

логическими: and (и), not (не), or (или)

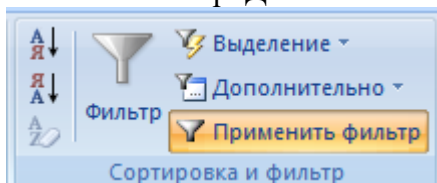
Like – для использования логики замены в выражениях;

In – для определения, содержится ли элемент данных в списке значений

Between...And – для выбора значений из определенного интервала

Откройте таблицу Список. В заголовке таблицы представлены поля и около каждого поля имеется значок раскрывающегося списка. Раскройте стрелку выбора для поля Год рождения. Выберите Год рождения 1984 сняв другие галочки с полей и нажмите ОК. вы автоматически попадаете в таблицу, в которой будут только выбранные записи.

Отмените выбор. Для этого необходимо отжать кнопку Применить фильтр.



Измените фильтр так, чтобы в таблице были видны только ученики школы №5, 10-го класса (одновременный запрос в двух полях – Школа и Класс).

Измените фильтр. В режиме таблицы в раскрывающемся списке выбора щелкните на команду Числовые фильтры и выберите строку больше наберите в появившемся окне >1984.

Чтобы получить записи учеников, у которых фамилии начинаются на «В», в раскрывающемся списке выбора поля Фамилия выберите команду Текстовые фильтры/Начинается С и введите в открывшееся окно ту букву на которую вы хотите вывести фамилии, например В (в данном случае русская буква, так как фамилии набраны в базе русским алфавитом).

Выведите все записи, кроме записей начинающихся на букву «В»: Текстовые фильтры/Не начинается с...В.

Выберите учеников всех школ, кроме школы №3.

Выберите всех учеников школы №1, фамилии которых начинаются на букву «А».

Выберите учеников, год рождения которых 1982 или 1983 (запрос: 1982 or 1983).

Отмените все запросы.

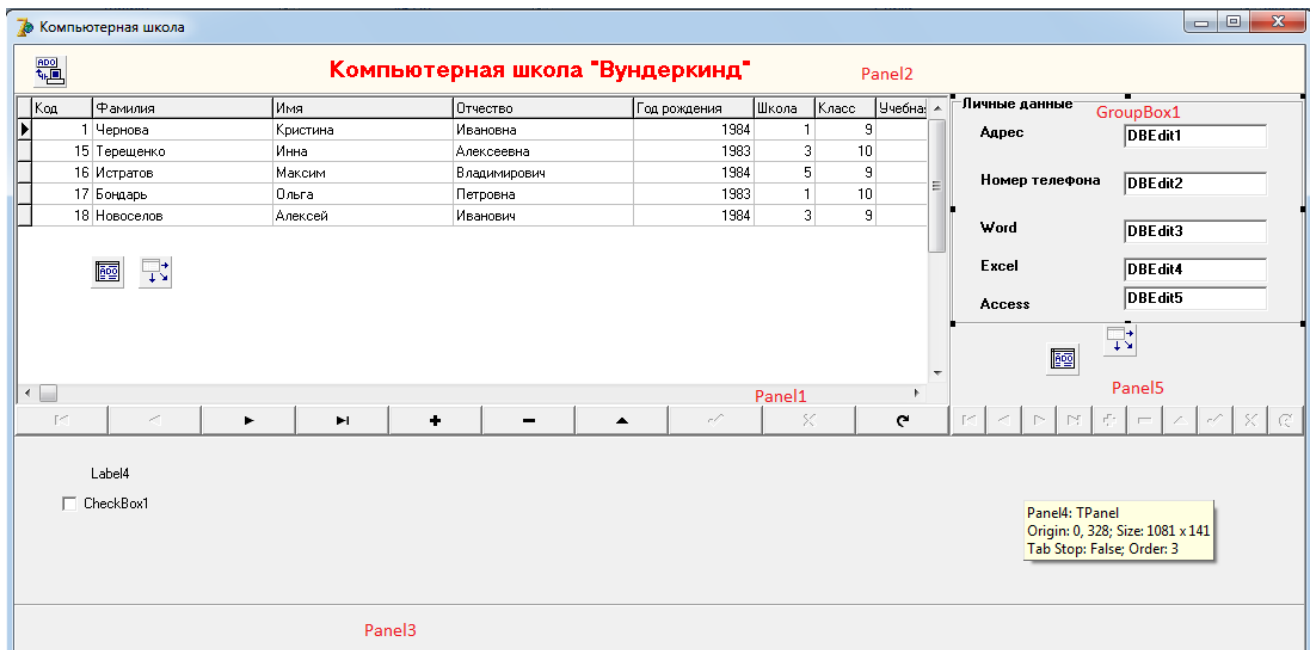
Закройте все таблицы и приложение Access.

Задание 10. Добавьте на форму новые компоненты

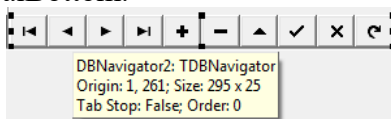
Порядок работы:

Установить необходимые панели и назначить им свойства:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Panel1	Properties	Align	alNone
		Caption	Пусто
Panel3	Properties	Align	alBottom
		Caption	Пусто
Panel4	Properties	Align	alBottom
		Caption	Пусто
Panel5	Properties	Align	alRigth
		Caption	Пусто
Panel1	Properties	Align	alClient
		Caption	Пусто
DBGrid1	Properties	Align	alClient



Установите компонент DBNavigator2 на компонент TPanel5 и присвойте свойство Align=alBottom.



Установить на компонент Panel5 компоненты:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
GroupBox1	Properties	Align	alTop
		Caption	Личные данные
		Font	MS Sans Serif, 8, полужирный
Label2	Properties	Left	24
		Caption	Адрес
		Font	MS Sans Serif, 8, полужирный
Label3	Properties	Left	24
		Caption	Номер телефона
		Font	MS Sans Serif, 8, полужирный
Label5	Properties	Left	24
		Caption	Word
		Font	MS Sans Serif, 8, полужирный
Label6	Properties	Left	24
		Caption	Excel
		Font	MS Sans Serif, 8, полужирный
Label7	Properties	Left	24
		Caption	Access
		Font	MS Sans Serif, 8, полужирный
DBEdit1	Properties	Left	144
DBEdit2	Properties	Left	144
DBEdit3	Properties	Left	144
DBEdit4	Properties	Left	144

Личные данные

Адрес DBEdit1

Номер телефона DBEdit2

Word DBEdit3

Excel DBEdit4

Access DBEdit5

DBGrid

DBNavigator

Компьютерная школа

Компьютерная школа "Вундеркинд"

Код	Фамилия	Имя	Отчество	Год рождения	Школа	Класс	Учебная
1	Чернова	Кристина	Ивановна	1984	1	9	
15	Терещенко	Инна	Алексеевна	1983	3	10	
16	Истратов	Максим	Владимирович	1984	5	9	
17	Бондарь	Ольга	Петровна	1983	1	10	
18	Новоселов	Алексей	Иванович	1984	3	9	

Личные данные

Адрес DBEdit1

Номер телефона DBEdit2

Word DBEdit3

Excel DBEdit4

Access DBEdit5

DBGrid

DBNavigator

Label4

☐ CheckBox1

Лабораторные работы по темам МДК 01.01

Лабораторная работа №1

Принцип программного управления

Цель: углубление и расширения теоретических знаний по теме.

Задание: подготовить доклад по теме «Принцип программного управления»

План доклада:

Что такое компьютер?

Как устроен компьютер?

На каких принципах построены компьютеры?

Что такое команда?

Как выполняется команда?

Что такое архитектура и структура компьютера?

Принцип программного управления.

Программное обеспечение и его основные компоненты.

Этапы развития программного обеспечения.

Тенденции развития программного обеспечения.

Форма контроля:

проверка докладов;

заслушивание и обсуждение вопросов в аудитории;

Лабораторная работа №2

Классическая схема ЭВМ

Цель: углубление и расширения теоретических знаний по теме.

Задание: подготовить доклад по теме «Классическая схема ЭВМ»

План доклада:

О понятии «архитектура ЭВМ»

Классическая архитектура ЭВМ и принципы Фон Неймана

Совершенствование и развитие внутренней структуры ЭВМ

Основной цикл работы ЭВМ

Система команд ЭВМ и способы обращения к данным

Форма контроля:

проверка докладов;

заслушивание и обсуждение вопросов в аудитории;

Лабораторная работа №3

Режимы адресации памяти

Цель: углубление и расширения теоретических знаний по теме.

Задание: подготовить доклад по теме «Режимы адресации памяти»

План доклада:

Режимы адресации.

Сегментация памяти в процессоре x86.

Система команд.

Форма контроля:

проверка докладов;

заслушивание и обсуждение вопросов в аудитории;

Лабораторная работа №4

Основные команды простых программ на Ассемблере. Команды передачи управления

Цель: углубление и расширения теоретических знаний по теме.

Задание: подготовить доклад по теме «Основные команды простых программ на Ассемблере. Команды передачи управления»

План доклада:

Команды языка ассемблера

Операнды команд

Пересылка и арифметические команды

Команды умножения и деления

Форма контроля:

проверка докладов;

заслушивание и обсуждение вопросов в аудитории;

Лабораторная работа №5

Повышение производительности обработки информации

Цель: углубление и расширения теоретических знаний по теме.

Задание: подготовить доклад по теме «Повышение производительности обработки информации»

План доклада:

Основные архитектурные решения в области повышения производительности ЭВМ и вычислительных систем: расслоение памяти, перекрытие, конвейеризация и параллелизм

UNIVAC 1

IBM 7094 II

STRETCH и LARC

IBM 360/91

ILLIAC IV

Многопроцессорные комплексы

Форма контроля:

проверка докладов;

заслушивание и обсуждение вопросов в аудитории;

Лабораторная работа №6

Проектирование многооконных приложений

Цель работы: рассмотреть этапы создания проекта приложения, включающего несколько окон.

Задание 1. Разработка проекта «Электронный учебник»

На основе примера электронного учебника «Основы алгоритмизации» разработайте форму электронного учебника на тему предложенную преподавателем и переложите его на код системы программирования DELPHI.

Пример «Электронный учебник: Основы алгоритмизации»

Для начала нужно создать htm-файлы, которые будут отображаться при нажатии на определённую тему, или практическую работу.

Для этого, откройте в MSWord документ с необходимой темой, и выполните следующие действия:

Щёлкните по кнопке Office.

Выберите пункт «Сохранить как».

В открывшемся контекстном меню выбрать пункт «Другие форматы».

Откроется окно в котором будет предложено выбрать имя для нового файла, и расширение.

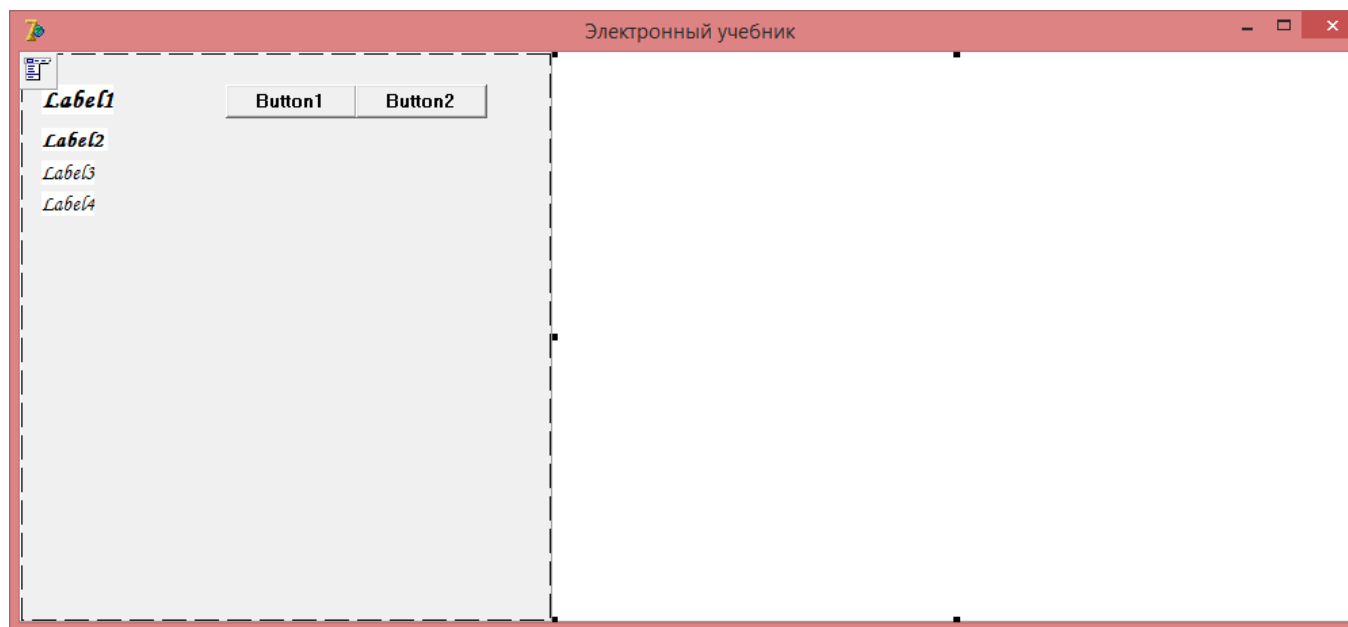
В строчке «Тип файла» выбрать «Веб-страница с фильтром»

Прodelайте эти действия с каждым документом, который будет использован в программе.

Файлы 111.htm и 112.htm будут использованы для отображения тем теоретической части.

Файлы 1.htm, 2.htm, 3.htm будут использованы для отображения практических работ.

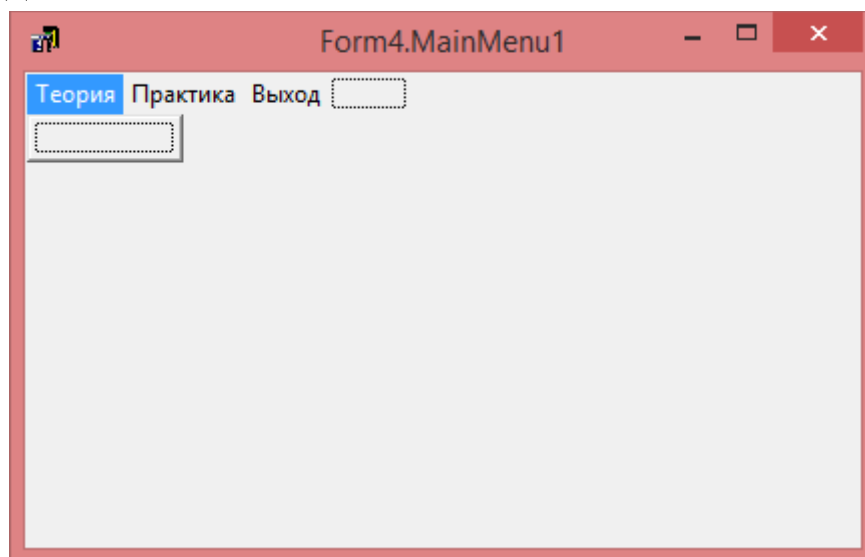
Создайте форму по образцу.



Выделенный объект	Вкладка Object Inspector	Имясвойс тва/ имя события	Действие
Form1	Propertie s	Caption	Электронный учебник
Label1	Propertie s	Caption	Теоретическая часть
	Propertie s	Font.Style. fsBold	True
Label2	Propertie s	Caption	1.1. Основные алгоритмические конструкции
	Propertie s	Visible	False
Label3	Propertie s	Caption	1.1.1. Понятие алгоритма и его свойства
	Events	OnClick	WebBrowser1.Navigate(ExtractFile lePath(Application.ExeName)+'html\111. htm');
	Propertie s	Visible	False
Label4	Propertie s	Caption	1.1.2. Основные алгоритмические конструкции
	Events	OnClick	WebBrowser1.Navigate(ExtractFile lePath(Application.ExeName)+'html\112. htm');
	Propertie s	Visible	False

Button1	Properties	Caption	- (наложение двух компонент Button1, Button2)
	Events	OnClick	Label2.Visible:=False; Label3.Visible:=False; Label4.Visible:=False; Button1.Visible:=False; Button2.Visible:=true;
Button2	Properties	Caption	+ (наложение двух компонент Button1, Button2)
	Events	OnClick	Label2.Visible:=true; Label3.Visible:=true; Label4.Visible:=true; Button2.Visible:=False; Button1.Visible:=true;
Panel1			Выделить компоненты Label1, Label2, Label3, Label4, Button1, Button2 в окне ObjectTreeView и добавить в Panel1
Panel2			Выделить компонент WebBrowser1 в окне ObjectTreeView и добавить в Panel2
WebBrowser1 (вкладка Internet)			
MainMenu1			См. ниже

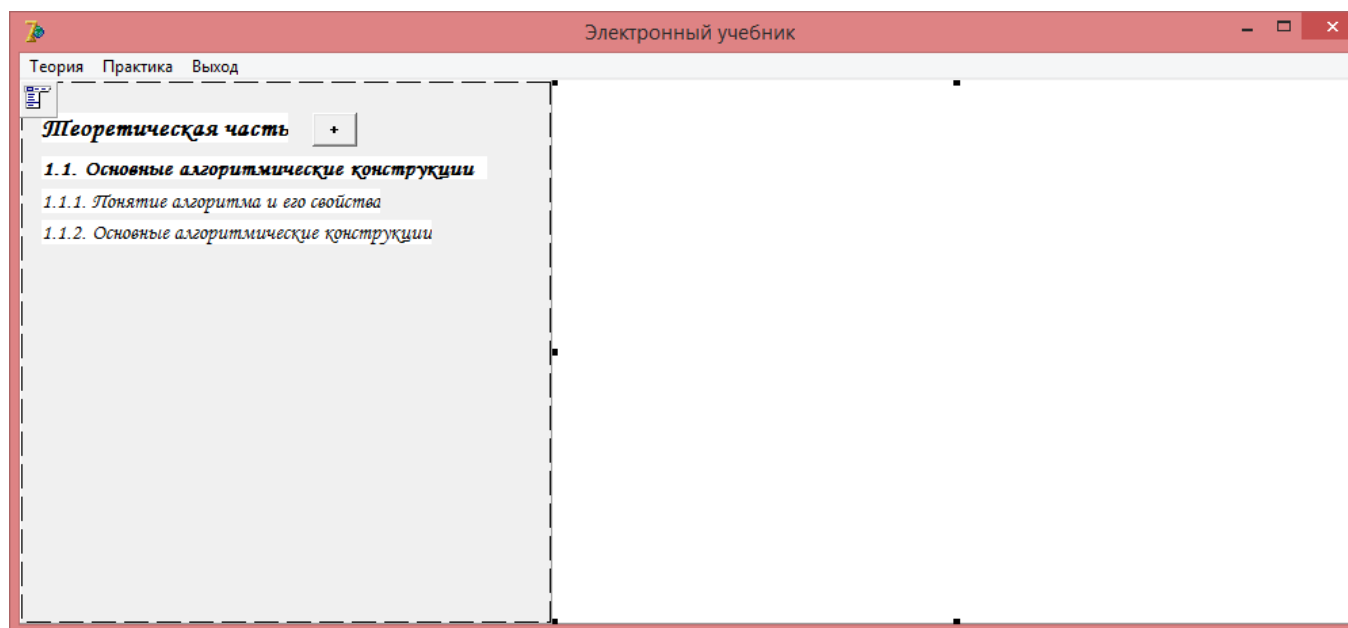
Действия для MainMenu1.



Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
N1	Properties	Caption	Теория

	Events	OnClick	Label2.Visible:=true; Label3.Visible:=true; Label4.Visible:=true; Button2.Visible:=False; Button1.Visible:=true;
N2	Properties	Caption	Практика
	Events	OnClick	Form2.Show; Form1.Hide;
N3	Properties	Caption	Выход
	Events	OnClick	Close;

Расположите компоненты на первой форме согласно образцу. Кнопку “+” поместите на кнопку “-”.

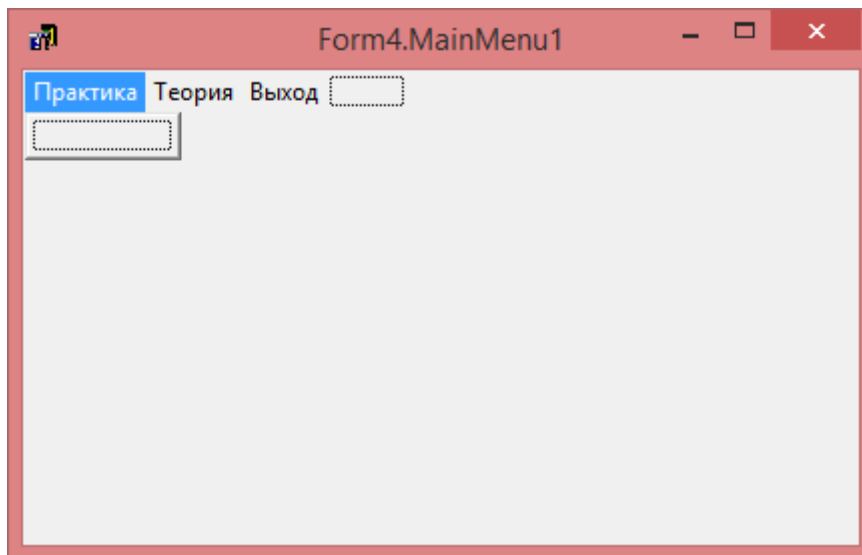


Создайте вторую форму (New/Form), расположите компоненты как и на первой форме.

Можно выделить все компоненты формы 1 (Ctrl+A) и скопировать их на новую форму 2 (Ctrl+C, Ctrl+V)

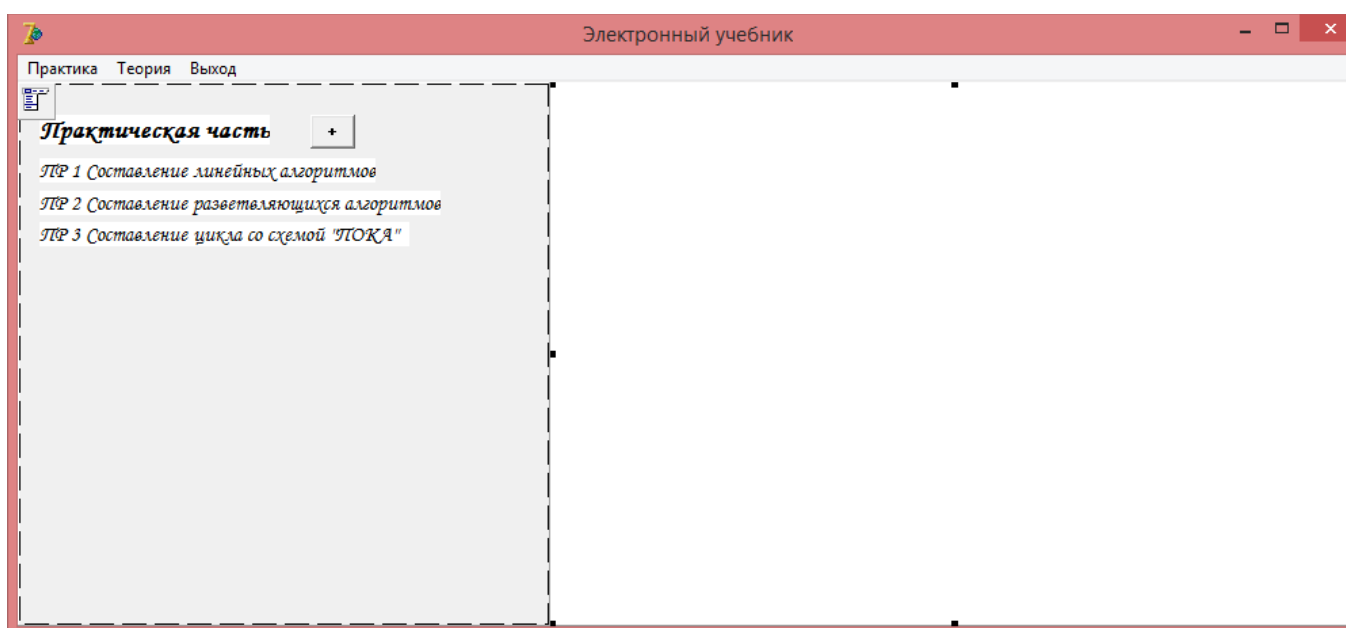
Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Form2	Properties	Caption	Электронный учебник
Label1	Properties	Caption	Практическая часть
	Properties	Font.Style. fsBold	True
Label2	Properties	Caption	ПП 1 Составление линейных алгоритмов

	Events	OnClick	WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\1.htm');
	Properties	Visible	False
Label3	Properties	Caption	ПП 2 Составление разветвляющихся алгоритмов
	Events	OnClick	WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\2.htm');
	Properties	Visible	False
Label4	Properties	Caption	ПП 3 Составление цикла со схемой "ПОКА"
	Events	OnClick	WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\3.htm');
	Properties	Visible	False
Button1	Properties	Caption	-
	Events	OnClick	Label2.Visible:=False; Label3.Visible:=False; Label4.Visible:=False; Button1.Visible:=False; Button2.Visible:=true;
Button2	Properties	Caption	+
	Events	OnClick	Label2.Visible:=true; Label3.Visible:=true; Label4.Visible:=true; Button2.Visible:=False; Button1.Visible:=true;
Panel1			Выделить компоненты Label2, Label3, Label4, Button1, Button2 в окне ObjectTreeView и добавить в Panel1
Panel2			Выделить компонент WebBrowser1 в окне ObjectTreeView и добавить в Panel2
WebBrowser1			
MainMenu1			См. ниже

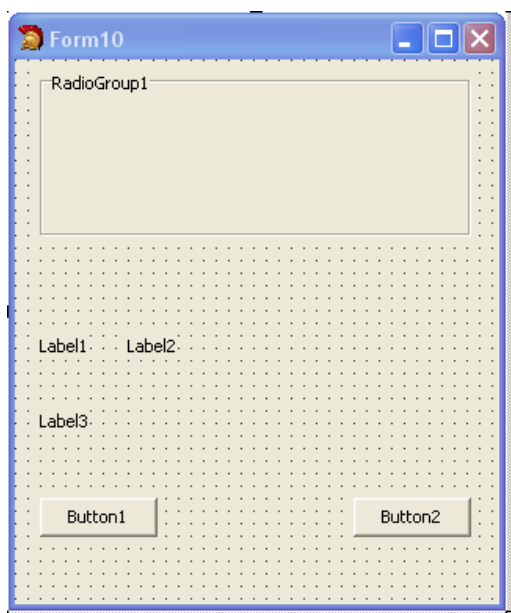


Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
N1	Properties	Caption	Практика
	Events	OnClick	Label2.Visible:=true; Label3.Visible:=true; Label4.Visible:=true; Button2.Visible:=False; Button1.Visible:=true;
N2	Properties	Caption	Теория
	Events	OnClick	Form2.Hide; Form1.Show;
N3	Properties	Caption	Выход
	Events	OnClick	Close;

Расположите компоненты на второй форме согласно образцу.



Следующим шагом будет создание теста. Создайте новую форму и расположите на ней следующие компоненты.



Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Тест 1
RadioGroup1	Properties	Caption	Название вопроса
	Properties	Items	Вписать варианты ответов на вопрос
	Events	OnClick	<pre> a1:=0; if RadioGroup1.ItemIndex=0 then a1:=a1+1 else a1:=a1; Form2.RadioGroup2.Enabled:=True; </pre>
RadioGroup2 (И так далее по количеству заданий)	Properties	Caption	Название вопроса
	Properties	Items	Вписать варианты ответов на вопрос
	Events	OnClick	<pre> a2:=0; if RadioGroup2.ItemIndex=0 then a2:=a2+1 else a2:=a2; Form2.RadioGroup3.Enabled:=True; </pre>
Label1	Properties	Caption	Количество баллов

Label2	s	Propertie	Caption	
Label3	s	Propertie	Caption	
Button1	s	Propertie	Caption	Выход
		Events	OnClick	Form3.Close;
Button2	s	Propertie	Caption	Завершить тестирование
		Events	OnClick	<pre> score:=(a1+a2+a3+a4+a5+a6+a7 +a8+a9+a10+a11+a12+a13+a14+a15+a1 6+a17+a18+a19+a20+a21+a22+a23+a24 +a25); if ((score<=27)and(score>=25)) then begin Label2.Caption:=IntToStr(score); Label3.Caption:='Отлично!'; end else if ((score<25)and(score>=22)) then begin Label2.Caption:=IntToStr(score); Label3.Caption:= ('Хорошо! Но Вам необходимо повторить теоретический материал. '); end else if ((score<22)and(score>=19)) then begin Label2.Caption:=IntToStr(score); Label3.Caption:='Плохо! Учитель!'; end else if ((score<19)and(score>=0)) then begin Label2.Caption:=IntToStr(score); Label3.Caption:='Очень плохо! Задумайтесь над этим!'; </pre>

			end; end;
--	--	--	--------------

В событии кнопки Button2 указываются переменные от a2 до a25. Они означают количество вопросов в тесте. Критерии оценки выбрать самостоятельно.

Листинг программы (Form1)

```
unitUnit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, OleCtrls, SHDocVw, Menus, ExtCtrls;
type
TForm1 = class(TForm)
  MainMenu1: TMainMenu;
  N1: TMenuItem;
  WebBrowser1: TWebBrowser;
  N3: TMenuItem;
  N2: TMenuItem;
  Panel2: TPanel;
  Panel1: TPanel;
  Label4: TLabel;
  Label3: TLabel;
  Label2: TLabel;
  Label1: TLabel;
  Button1: TButton;
  Button2: TButton;
  procedure Label3Click(Sender: TObject);
  procedure Label4Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure N3Click(Sender: TObject);
  procedure N1Click(Sender: TObject);
  procedure N2Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;

implementation
uses Unit2;
{$R *.dfm}
procedure TForm1.Label3Click(Sender: TObject);
```

```

begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\111.htm');
end;
procedure TForm1.Label4Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\112.htm');
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
Label2.Visible:=False;
Label3.Visible:=False;
Label4.Visible:=False;
Button1.Visible:=False;
Button2.Visible:=true;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
Label2.Visible:=true;
Label3.Visible:=true;
Label4.Visible:=true;
Button2.Visible:=False;
Button1.Visible:=true;
end;
procedure TForm1.N3Click(Sender: TObject);
begin
Close;
end;
procedure TForm1.N1Click(Sender: TObject);
begin
Label2.Visible:=true;
Label3.Visible:=true;
Label4.Visible:=true;
Button2.Visible:=False;
Button1.Visible:=true;
end;
procedure TForm1.N2Click(Sender: TObject);
begin
Form2.Show;
Form1.Hide;
end;
end.

```

Листинг программы (Form2)

```

unit Unit2;
interface

```

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, OleCtrls, SHDocVw, Menus, ExtCtrls;

type

```
TForm1 = class(TForm)
  MainMenu1: TMainMenu;
  N1: TMenuItem;
  WebBrowser1: TWebBrowser;
  N3: TMenuItem;
  N2: TMenuItem;
  Panel2: TPanel;
  Panel1: TPanel;
  Label4: TLabel;
  Label3: TLabel;
  Label2: TLabel;
  Label1: TLabel;
  Button1: TButton;
  Button2: TButton;
  procedure Label3Click(Sender: TObject);
  procedure Label3Click(Sender: TObject);
  procedure Label4Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure N3Click(Sender: TObject);
  procedure N1Click(Sender: TObject);
  procedure N2Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2 TForm2;

implementation

uses Unit1;

{ \$R *.dfm }

procedure TForm1.Label2Click(Sender: TObject);

begin

WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\1.htm');

end;

procedure TForm1.Label3Click(Sender: TObject);

begin

```

WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\2.htm');
end;
procedure TForm1.Label4Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'html\3.htm');
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
Label2.Visible:=False;
Label3.Visible:=False;
Label4.Visible:=False;
Button1.Visible:=False;
Button2.Visible:=true;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
Label2.Visible:=true;
Label3.Visible:=true;
Label4.Visible:=true;
Button2.Visible:=False;
Button1.Visible:=true;
end;
procedure TForm1.N3Click(Sender: TObject);
begin
Close;
end;
procedure TForm1.N1Click(Sender: TObject);
begin
Label2.Visible:=true;
Label3.Visible:=true;
Label4.Visible:=true;
Button2.Visible:=False;
Button1.Visible:=true;
end;
procedure TForm1.N2Click(Sender: TObject);
begin
Form1.Show;
Form2.Hide;
end;
end.

```

Листинг программы (Form3)

```

unit Unit3;
interface
uses

```

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, jpeg, ExtCtrls;

type

TForm2 = class(TForm)

Button1: TButton;

RadioGroup1: TRadioGroup;

Button2: TButton;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

procedure Button1Click(Sender: TObject);

procedure RadioGroup1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,a19,a20,a21,a22,a23,a24,a25,s

core:integer;

implementation

uses Unit1, Unit2;

{ \$R *.dfm }

procedure TForm3.Button1Click(Sender: TObject);

begin

Close;

end;

procedure TForm3.RadioGroup1Click(Sender: TObject);

begin

a1:=0;

if RadioGroup1.ItemIndex=0 then a1:=a1+1 else a1:=a1;

Form2.RadioGroup2.Enabled:=True;

end;

procedure TForm3.Button2Click(Sender: TObject);

begin

score:=(a1+a2+a3+a4+a5+a6+a7+a8+a9+a10+a11+a12+a13+a14+a15+a16+a17+a18+a19+a20
+a21+a22+a23+a24+a25);

if ((score<=27)and(score>=25)) then

begin

Label2.Caption:=IntToStr(score);

Label3.Caption:='Отлично!';

end

```

else if ((score<25)and(score>=22)) then
begin
Label2.Caption:=IntToStr(score);
Label3.Caption:= ('Хорошо! Но Вам необходимо повторить теоретический материал. ');
end
else if ((score<22)and(score>=19)) then
begin
Label2.Caption:=IntToStr(score);
Label3.Caption:='Плохо! Учите!';
end
else if ((score<19)and(score>=0)) then
begin
Label2.Caption:=IntToStr(score);
Label3.Caption:='Очень плохо! Задумайтесь над этим!';
end;
end;
end.

```

Лабораторная работа №7

Работа с графикой и средствами мультимедиа

Цель работы

рассмотреть практическое применение классов для реализации графических операций;

на примере рассмотреть методы рисования.

Задание 1. Разработка проекта «Движущиеся объекты»

На основе примера кроссворда «Синее море, белый пароход» разработать свою модель движущегося объекта предложенную преподавателем и переложите его на код системы программирования DELPHI.

Пример «Движущиеся объекты: Синее море, белый пароход»

```

unit ship_;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

```

```

end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
var
  x,y: integer; // координаты корабля (базовой точки)
  // вычерчивает на поверхности формы кораблик
  // или стирает его
procedure Parohod(x,y: integer; mode: boolean);
  // x,y - координаты базовой точки кораблика
  // mode: True - нарисовать, False - стереть
const
  { Базовые точки кораблика находятся в узлах сетки,
    шаг которой определяет размер кораблика. }
  dx = 5; // шаг сетки по X
  dy = 5; // шаг сетки по Y
var
  // корпус и надстройку будем рисовать
  // при помощи метода Polygon
  p1: array[1..7] of TPoint; // координаты точек корпуса
  p2: array[1..8] of TPoint; // координаты точек надстройки
  pc, bc: TColor; // текущий цвет карандаша и кисти
begin
  if not mode then
  begin
    // стереть
    Form1.Canvas.Brush.Color := clNavy;
    Form1.Canvas.Pen.Color := clNavy;
    Form1.Canvas.Rectangle(x,y+1,x+17*dx,y-10*dy);
    Form1.Canvas.Brush.Color := clNavy;
    // небо
    if y-10*dy < 80 then
    begin
      // конецмачтынафоненеба
      Form1.Canvas.Pen.Color := clSkyBlue;
      Form1.Canvas.Brush.Color := clSkyBlue;
      Form1.Canvas.Rectangle(x,y-10*dy,x+17*dx,80);
    end;
    exit;
  end;
  // рисуем
  with Form1.Canvas do
  begin

```

```

pc := Pen.Color; // сохраним текущий цвет карандаша
bc := Brush.Color; // и кисти
Pen.Color:=clBlack; // установим нужный цвет
Brush.Color := clWhite;
// рисуем ...
// корпус
p1[1].X := x; p1[1].Y := y;
p1[2].X := x; p1[2].Y := y-2*dy;
p1[3].X :=x+10*dx; p1[3].Y := y-2*dy;
p1[4].X :=x+11*dx; p1[4].Y := y-3*dy;
p1[5].X :=x+17*dx; p1[5].Y :=y-3*dy;
p1[6].X :=x+14*dx; p1[6].Y :=y;
p1[7].X :=x; p1[7].Y :=y;
Polygon(p1);
// надстройка
p2[1].X := x+3*dx; p2[1].Y := y-2*dy;
p2[2].X := x+4*dx; p2[2].Y := y-3*dy;
p2[3].X := x+4*dx; p2[3].Y := y-4*dy;
p2[4].X := x+13*dx; p2[4].Y := y-4*dy;
p2[5].X := x+13*dx; p2[5].Y := y-3*dy;
p2[6].X := x+11*dx; p2[6].Y := y-3*dy;
p2[7].X := x+10*dx; p2[7].Y := y-2*dy;
p2[8].X := x+3*dx; p2[8].Y := y-2*dy;
Polygon(p2);
MoveTo(x+5*dx,y-3*dy);
LineTo(x+9*dx,y-3*dy);
// капитанский мостик
Rectangle(x+8*dx,y-4*dy,x+11*dx,y-5*dy);
// труба
Rectangle(x+7*dx,y-4*dy,x+8*dx,y-7*dy);
// иллюминаторы
Ellipse(x+11*dx,y-2*dy,x+12*dx,y-1*dy);
Ellipse(x+13*dx,y-2*dy,x+14*dx,y-1*dy);
// мачта
MoveTo(x+10*dx,y-5*dy);
LineTo(x+10*dx,y-10*dy);
// оснастка
Pen.Color := clWhite;
MoveTo(x+17*dx,y-3*dy);
LineTo(x+10*dx,y-10*dy);
LineTo(x,y-2*dy);
Pen.Color:=pc; // восстановим старый цвет карандаша
end;
end;

```

```

// обработка сигнала таймера
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Parohod(x,y, False); // стереть рисунок
  if x < Form1.ClientWidth
  then x := x+2
  else begin // новый рейс
    x := 0;
    y := Random(50) + 100;
  end;
  Parohod(x,y,True); // нарисовать в новой точке
end;
procedure TForm1.FormPaint(Sender: TObject);
begin
  // небо
  Canvas.Brush.Color := clSkyBlue;
  Canvas.Pen.Color := clSkyBlue;
  Canvas.Rectangle(0,0,ClientWidth,80);
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  x:=0; y:=80; // исходное положение парохода
  Form1.Color:=clNavy; // цвет моря
  Timer1.Interval := 50; // сигнал таймера каждые 50 мСек
end;
end.

```

Лабораторная работа №8

Разработка и проектирование баз данных

Цель работы: освоить технологию создания базы данных и наполнения ее информацией в системе DELPHI.

Задание 1. Разработка проекта «База данных»

Напишите программу работы с локальной базой данных.

Вариант 1. Учет авиапассажиров.

Таблица авиапассажиров: код авиапредприятия, код города назначения, номер рейса, дата вылета, фамилия, адрес пассажира.

Справочники: авиапредприятия (код, наименование, адрес, телефон), города (код и наименование).

Отчет по авиапредприятиям и городам с числом пассажиров по авиапредприятиям и городам и в целом по отчету.

Вариант 2. Учет выпуска изделий.

Таблица выпущенных изделий: код изделия, код предприятия, количество выпущенных изделий (шт.), их себестоимость, отпускная цена, затраты на электроэнергию, затраты на зарплату, дата выпуска.

Справочники: изделия (код, наименование), предприятия (код, наименование, адрес, телефон).

Отчет по изделиям и предприятиям со средними затратами по каждому изделию.

Вариант 3. Учет поставок товаров.

Таблица поставленных товаров: код поставщика, код товара, количество, цена, единица измерения, дата поставки.

Справочники: поставщики (код, наименования, адрес, телефон), товары (код, наименование).

Отчет по товарам и поставщикам с итоговой стоимостью по товарам, поставщикам и по отчету.

Вариант 4. Учет уволившихся с предприятия.

Таблица уволенных: код подразделения, код причины увольнения, дата увольнения, фамилия, номер приказа.

Справочники: подразделения (код, наименование), причины увольнения (код и наименование).

Отчет по причинам увольнения и подразделениям с итоговым числом уволившихся по причинам увольнения, подразделениям и по отчету.

Вариант 5. Учет рождаемости.

Таблица новорожденных: код населенного пункта, код национальности; вес, рост новорожденного, дата рождения, фамилия, имя, отчество.

Справочники: населенные пункты (код, наименование), национальности (код, наименование).

Отчет по населенным пунктам и национальностям с итоговым числом новорожденных по населенным пунктам, национальностям и по отчету.

Вариант 6. Учет участников Олимпиады.

Таблица олимпийцев: код страны, код вида спорта, фамилия участника, дата рождения, рост, вес, пол.

Справочники: страны (код, наименование, начальник команды, телефон), виды спорта (код, наименование).

Отчет по странам и видам спорта с итогами по странам и видам спорта: число участников, средний вес, средний рост.

Вариант 7. Учет проданных товаров.

Таблица проданного товара: код торгового предприятия, код товара, стоимость проданного товара за последние 3 года (три поля).

Справочники: торгующие предприятия (код, наименование, директор, телефон), товары (код, наименование).

Отчет по товарам и торгующим предприятиям с итогами по товарам и предприятиям.

Вариант 8. Учет малых предприятий.

Таблица малых предприятий: код населенного пункта, код основного вида деятельности МП, численность работающих, адрес, объем выполняемых работ за год (всего) и по основному виду деятельности, директор, дата создания.

Справочники: населенные пункты (код, наименование), виды деятельности (код, наименование).

Отчет по населенным пунктам и видам деятельности с итогами по видам деятельности, населенным пунктам и по отчету в целом.

Вариант 9. Учет платежных поручений в банке.

Таблица платежных поручений: код клиента, код вида платежа, дата, сумма, номер поручения.

Справочники: клиенты банка (код, наименование, адрес, директор, телефон), виды платежей (код, наименование).

Отчет по клиентам и видам платежа с итогами по клиентам, видам платежа и в целом по отчету.

Вариант 10. Учет договоров займа.

Таблица договоров займа: код организации, номер договора, дата заключения, дата окончания, сумма, процент, сумма возврата, возвращенная сумма, особые условия, процент штрафа за каждый просроченный день, код денежной единицы.

Справочники: организации (код, наименование, адрес, директор, телефон); денежные единицы (код, обозначение, наименование).

Отчет по видам валют и дням возврата с итогами по видам валют и дням возврата.

Вариант 11. Учет проданных ценных бумаг.

Таблица проданных ценных бумаг: код эмитента, код типа ценной бумаги, число, номинальная стоимость, курсовая стоимость, дата выпуска, номер регистрации выпуска ценных бумаг.

Справочники: эмитенты (код, наименование, адрес, директор, телефон), типы ценных бумаг (код, наименование).

Отчет по типам ценных бумаг и эмитентам с итогами по типам ценных бумаг и эмитентам: суммарное число бумаг, средняя стоимость.

14. Учет кадров.

Таблица сотрудников предприятия: табельный номер работника, фамилия, дата рождения, код специальности, код подразделения, дата приема, дата начала трудовой деятельности, оклад, образование.

Справочники: специальности (код, наименование); подразделения (код, наименование, фамилия начальника, телефон).

Отчет по подразделениям и специальностям с итогами (число принятых) по подразделениям, специальностям и по предприятию.

Вариант 12. Учет покупки и продажи валюты.

Таблица продажи и покупки валюты: дата, фамилия, паспортные данные, код валюты, количество валюты проданной и купленной, количество рублей, код обменного пункта.

Справочники: валюта (код, наименование, текущий курс) и обменные пункты (код, наименование).

Отчет по видам валют и обменным пунктам с итогами по видам валют и обменным пунктам (сумма проданной и купленной валюты).

Вариант 13. Учет успеваемости студентов.

Таблица успеваемости студентов: код предмета, дата, оценка, номер зачетной книжки.

Справочники: предметы (код и наименование), студенты (группа, номер зачетной книжки, фамилия).

Отчет по группам и студентам со средней оценкой по группам, студентам и в целом по ВУЗу.

Вариант 14. Каталог почтовых марок.

Концептуальные требования:

Номер	Год издания	Тематика	Ф.И.О. художника	Сюжет
-------	-------------	----------	------------------	-------

Номинал	Стоимость	Тираж	Способ печати	Размеры
---------	-----------	-------	---------------	---------

Фото

Режимы работы:

Сортировка по полям – Номер, Год издания, Тематика, Способ печати.

Поиск по полям - ФИО художника, Номер, Номинал.

Фильтрация по полям на диапазон значений- ФИО художника, Год издания, Способ печати.

Вариант 15. Каталог монет России.

Концептуальные требования:

Номер	Номинал	Материал	Размер	Вес	Год выпуска
-------	---------	----------	--------	-----	-------------

Ф.И.О. гравера	Монетный двор	Тираж	Редкость	Фото	
----------------	---------------	-------	----------	------	--

Режимы работы:

Сортировка по полям – Номер, Год выпуска, Материал, Монетный двор.

Поиск по полям - ФИО гравёра, Номер, Номинал, Редкость.

Фильтрация по полям на диапазон значений - ФИО гравёра, Год выпуска, Материал,

Номинал

Вариант 16. Исторические здания г. Перми.

Концептуальные требования:

Наименование	Адрес	Дата постройки	Ф.И.О.	архитект.
--------------	-------	----------------	--------	-----------

Владелец

Размеры	Этажность	Фото	Дата ремонта
---------	-----------	------	--------------

Режимы работы:

Сортировка по полям – Наименование, Дата постройки, Адрес.

Поиск по полям - ФИО архитектора, Дата постройки, Наименование.

Фильтрация по полям на диапазон значений- ФИО архитектора, Дата постройки,

Владелец.

Вариант 17. Каталог периодических изданий

Концептуальные требования:

Наименование	Ф.И.О. гл. редакт.	Вид	Цена 1 экз.	Периодичность
--------------	--------------------	-----	-------------	---------------

Объем	Тираж	Подп. Индекс	Объем подписки
-------	-------	--------------	----------------

Режимы работы:

Сортировка по полям – Наименование, Объём, Тираж, Подписной индекс.

Поиск по полям - Наименование, ФИО главного редактора, Цена.

Фильтрация по полям на диапазон значений - Вид, Периодичность, Тираж, Объём,

Цена.

3.2. Типовые задания для оценки освоения МДК 01.02 Разработка программных модулей

ЛАБОРАТОРНАЯ РАБОТА № 1. Тестирование программ методами «белого ящика»

Цель работы: изучить методы тестирования логики программы, формализованные описания результатов тестирования и стандарты по составлению схем программ.

Подготовка к лабораторной работе

1. Ознакомиться с лекционным материалом по теме «Отладка и тестирование программного обеспечения» МДК 01.02 Поддержка и тестирование программных модулей».

2. Ознакомиться теоретической частью лабораторной работы.

3. Выполнить задания.

Теоретическая часть. Виды тестирования

Тестирование программного обеспечения включает в себя целый комплекс действий, аналогичных последовательности процессов разработки программного обеспечения. В него входят :

- постановка задачи для теста;
- проектирование теста;
- написание тестов;
- тестирование тестов;
- выполнение тестов;
- изучение результатов тестирования.

Наиболее важным является проектирование тестов. Существуют разные подходы к проектированию тестов.

Первый состоит в том, что тесты проектируются на основе внешних спецификаций программ и модулей либо спецификаций сопряжения модуля с другими модулями, программа при

этом рассматривается как «черный ящик». Смысл теста заключается в том, чтобы проверить, соответствует ли программа внешним спецификациям. При этом содержание модуля не имеет значения. Такой подход получил название — стратегия «черного ящика».

Второй подход — стратегия «белого ящика», основан на анализе логики программы. При таком подходе тестирование заключается в проверке каждого пути, каждой ветви алгоритма. При этом внешняя спецификация во внимание не принимается.

Ни один из этих подходов не является оптимальным. Реализация тестирования методом «черного ящика» сводится к проверке всех возможных комбинаций входных данных. Невозможно протестировать программу, подавая на вход бесконечное множество значений, поэтому ограничиваются определенным набором данных. При этом исходят из максимальной отдачи теста по сравнению с затратами на его создание. Она измеряется вероятностью того, что тест выявит ошибки, если они имеются в программе. Затраты измеряются временем и стоимостью подготовки, выполнения и проверки результатов теста.

Тестирование методом «белого ящика» также не дает 100%-ной гарантии того, что модуль не содержит ошибок. Даже если предположить, что выполнены тесты для всех ветвей алгоритма, нельзя с полной уверенностью утверждать, что программа соответствует ее спецификациям. Например, если требовалось написать программу для вычисления кубического корня, а программа фактически вычисляет корень квадратный, то реализация будет совершенно неправильной, даже если проверить все пути. Вторая проблема — отсутствующие пути. Если программа реализует спецификации не полностью (например, отсутствует такая специализированная функция, как проверка на отрицательное значение входных данных

программы вычисления квадратного корня), никакое тестирование существующих путей не выявит такой ошибки. И наконец, проблема зависимости результатов тестирования от входных данных. Одни данные будут давать правильные результаты, а другие нет. Например, если для определения равенства трех чисел программируется выражение вида:

$$?(A + B + C)/B = A,$$

то оно будет верным не для всех значений A , B и C (ошибка возникает в том случае, когда из двух значений B или C одно больше, а другое на столько же меньше A). Если концентрировать внимание только на тестировании путей, нет гарантии, что эта ошибка будет выявлена.

Таким образом, полное тестирование программы невозможно, т. е. никакое тестирование не гарантирует полное отсутствие ошибок в программе. Поэтому необходимо проектировать тесты таким образом, чтобы увеличить вероятность обнаружения ошибки в программе.

Стратегия «белого ящика»

Существуют следующие методы тестирования по принципу «белого ящика»:

- покрытие операторов;
- покрытие решений;
- покрытие условий;
- покрытие решений/условий;
- комбинаторное покрытие условий.

Метод покрытия операторов

Целью этого метода тестирования является выполнение каждого оператора программы хотя бы один раз.

Пример.

Если для тестирования задать значения переменных $A = 2$, $B = 0$, $X = 3$, будет реализован путь *ace*, т. е. каждый оператор программы выполнится один раз (рис. Л1.1, *a*). Но если внести в алгоритм ошибки — заменить в первом условии *and* на *or*, а во втором $X > 1$ на $X < 1$ (рис. Л1.1, *б*), ни одна ошибка не будет обнаружена (табл. Л1.1). Кроме того, путь *abd* вообще не будет охвачен тестом, и если в нем есть ошибка, она также не будет обнаружена. В табл. Л1.1 ожидаемый результат определяется по блок-схеме на рис. Л1.1, *a*, а фактический — по рис. Л1.1, *б*.

Как видно из этой таблицы, ни одна из внесенных в алгоритм ошибок не будет обнаружена.

Таблица Л1.1. Результат тестирования методом покрытия операторов

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A = 2, B = 0, X = 3$	$X = 2,5$	$X = 2,5$	Неуспешно

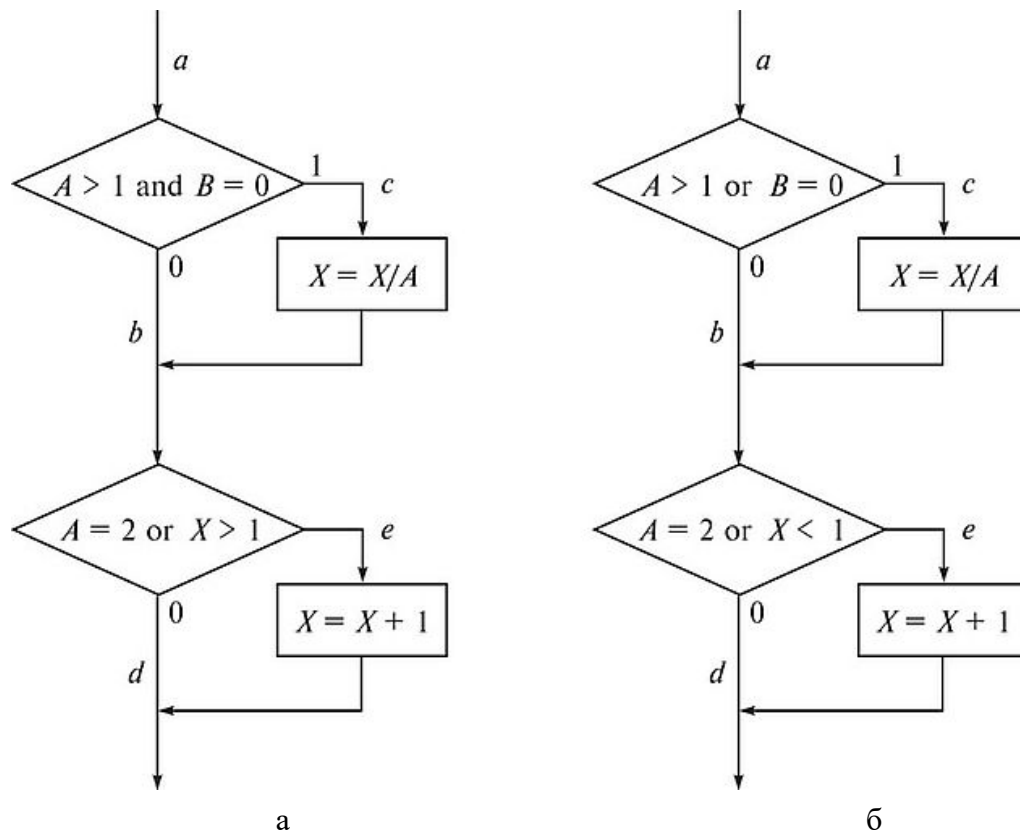


Рис. Л1.1. Пример алгоритма программы: *a* — правильный; *б* — с ошибкой

Метод покрытия решений (покрытия переходов)

Согласно методу покрытия решений каждое направление перехода должно быть реализовано, по крайней мере, один раз. Этот метод включает в себя критерий покрытия операторов, так как при выполнении всех направлений переходов выполнятся все операторы, находящиеся на этих направлениях.

Для программы, приведенной на рис. Л1.1, покрытие решений может быть выполнено двумя тестами, покрывающими пути $\{ace, abc!\}$, либо $\{acc1, abe\}$. Для этого выберем следующие исходные данные; $\{A = 3, B = 0, X = 3\}$ — в первом случае и $\{A = 2, B = 1, X = 1\}$ — во втором. Однако путь, где X не меняется, будет проверен с вероятностью 50 %: если во втором условии вместо условия $X > 1$ записано $X < 1$, то ошибка не будет обнаружена двумя тестами.

Результаты тестирования приведены в табл. Л1.2.

Таблица Л1.2. Результат тестирования методом покрытия решений

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A = 3, B = 0, X = 2$	$X = 1$	$X = 1$	Неуспешно
$A = 2, B = 1, X = 1$	$X = 2$	$X = 1$	Успешно

Метод покрытия условий

Этот метод может дать лучшие результаты по сравнению с предыдущими. В соответствии с методом покрытия условий записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз.

В рассматриваемом примере имеем четыре условия: $\{A > 1, 5=0\}$, $\{A = 2, X > 1\}$. Следовательно, требуется достаточное число тестов, такое, чтобы реализовать ситуации, где $A > 1$, $A < 1$, $5 = 0$ и $5 \neq 0$ в точке a и $5 = 2$, $A \neq 2$, $X > 1$ и $X < 1$ в точке b . Тесты, удовлетворяющие критерию покрытия условий (табл. Л1.3), и соответствующие им пути:

- а) $A = 2, 5 = 0, X=4$ $a \rightarrow c \rightarrow e$;
- б) $A = 1, 5 = 1, X=0$ $a \rightarrow b \rightarrow c \rightarrow f$.

Таблица Л1.3. Результаты тестирования методом покрытия условий

Тест	Ожидаемый	Фактический	Результат
	результат	результат	тестирования
$5 = 2, B = 0, X=4$	$5 \neq 3$	$X = 3$	Неуспешно
$A = 1, B = 1, X=0$	$5 \neq 0$	$X = 1$	Успешно

Метод покрытия решений/условий

Критерий покрытия решений/условий требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия выполнялись по крайней мере один раз, все результаты каждого решения выполнялись по крайней мере один раз и, кроме того, каждой точке входа передавалось управление по крайней мере один раз.

Недостатки метода:

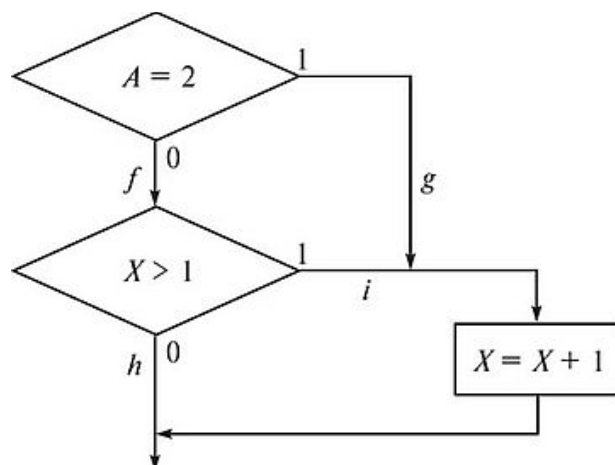
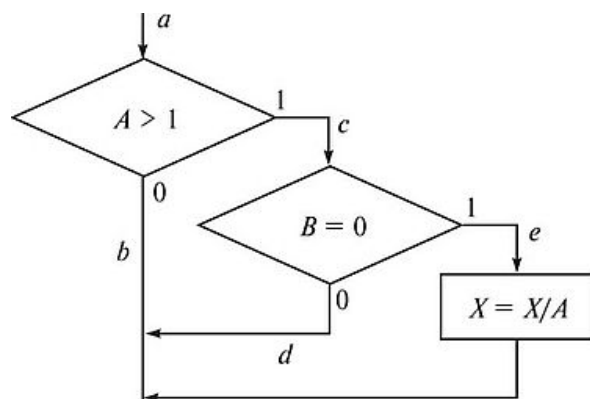
- не всегда можно проверить все условия;
- невозможно проверить условия, которые скрыты другими условиями;
- недостаточная чувствительность к ошибкам в логических выражениях.

Так, в рассматриваемом примере два теста метода покрытия условий

- а) $A = 2, B = 0, X = 4$ $a \rightarrow c \rightarrow e$;
- б) $A = 1, B = 1, X = 0$ $a \rightarrow b \rightarrow c \rightarrow f$!

отвечают и критерию покрытия решений/условий. Это является следствием того, что одни условия приведенных решений скрывают другие условия в этих решениях. Так, если условие $A > 1$ будет ложным, транслятор может не проверять условия $B = 0$, поскольку при любом результате условия $B = 0$ результат решения $((A > 1) \& (B = 0))$ примет значение *ложь*. То есть в варианте на рис. Л1.1 не все результаты всех условий выполняются в процессе тестирования.

Рассмотрим реализацию того же примера на рис. Л1.2. Наиболее полное покрытие тестами в этом случае осуществляется



так, чтобы выполнялись все возможные результаты каждого простого решения. Для этого нужно покрыть пути *acseg* (тест $A = 2, B = 0, X = 4$), *acē/k* (тест $A = 3, B = 1, X = 0$), *aB/k* (тест $A = 0, B = 0, X = 0$), *aBp* (тест $A = 0, B = 0, X = 2$).

Протестировав алгоритм на рис. Л1.2, нетрудно убедиться в том, что критерии покрытия условий и критерии покрытия решений/условий недостаточно чувствительны к ошибкам в логических выражениях.

Метод комбинаторного покрытия условий

Критерий комбинаторного покрытия условий удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

Этот метод требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении выполнялись по крайней мере один раз. По этому критерию в рассматриваемом примере должны быть покрыты тестами следующие восемь комбинаций:

1. $A > 1, B = 0$. 5. $A = 2, X > 1$.
2. $A > 1, B \neq 0$.
3. $A < 1, B = 0$.
4. $A < 1, B \neq 0$.
6. $A = 2, 1$.
7. $A \neq 2, X > 1$.
8. $A \neq 2, X < 1$.

Для того чтобы протестировать эти комбинации, необязательно использовать все 8 тестов. Фактически они могут быть покрыты четырьмя тестами (табл. Л1.4):

- $A = 2, B = 0, X = 4$ (покрывает 1,5);

- $A = 2, B = 1, X = 1$ (покрывает 2,6);
- $A = 0,5, B = 0, X = 2$ (покрывает 3, 7);
- $A = 1, B = 0, X =$ (покрывает 4, 8).

Таблица Л1.4. Результаты тестирования методом комбинаторного покрытия условий

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
$A=2, B=0, X=4$	$*=3$	$*=3$	Неуспешно
$A = 2, B = 1, X = 1$	$X = 2$	1 Г) СЧ П	Успешно
$A = 0,5, B = 0, X = 2$	$X = 3$	$X = 4$	Успешно
$A = 1, B = 0, X = 1$	$X = 1$	$X = 1$	Неуспешно

Порядок выполнения работы

1. Спроектировать тесты по принципу «белого ящика» для программы, разработанной в лабораторной работе № 4. Использовать схемы алгоритмов, разработанные и уточненные в лабораторных работах № 2, 3.

2. Выбрать несколько алгоритмов для тестирования и обозначить буквами или цифрами ветви этих алгоритмов.

3. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования.

4. Записать тесты, которые позволят пройти по путям алгоритма.

5. Протестировать разработанную вами программу. Результаты оформить в виде таблиц (см. табл. Л1.1— Л1.4).

6. Проверить все виды тестов и сделать выводы об их эффективности.

7. Оформить отчет по лабораторной работе.

8. Сдать и защитить работу.

Защита отчета по лабораторной работе

Отчет по лабораторной работе должен состоять из:

1. Постановки задачи.
2. Блок-схемы программ.
3. Тестов.
4. Таблиц тестирования программы.
5. Выводов по результатам тестирования (не забывайте, что целью тестирования является обнаружение ошибок в программе).

Контрольные вопросы

1. Охарактеризуйте этап реализации и тестирования программного продукта.

2. Какие существуют виды тестирования?
3. Назовите критерии выбора тестов.
4. Перечислите свойства тестов.
5. Приведите критерии надежности программ.
6. В чем заключается оценка надежности программ?

ЛАБОРАТОРНАЯ РАБОТА № 2. Тестирование программ методами «черного ящика»

Цель работы: провести функциональное тестирование разработанного программного средства в соответствии с заданным вариантом

Подготовка к лабораторной работе

Ознакомиться с лекционным материалом по теме «Отладка и тестирование программного обеспечения» МДК 01.02 Поддержка и тестирование программных модулей».

2. Ознакомиться с теоретической частью лабораторной работы.

3. Выполнить задания.

Теоретическая часть. Тестирование программ

Процесс тестирования состоит из трёх этапов:

1. Проектирование тестов.
2. Исполнение тестов.
3. Анализ полученных результатов.

На первом этапе решается вопрос о выборе некоторого подмножества множества тестов, которое сможет найти наибольшее количество ошибок за наименьший промежуток времени. На этапе исполнения тестов проводят, запуск тестов и отлавливают ошибки в тестируемом программном продукте.

Виды тестов

Функциональные тесты составляются на уровне спецификации, до решения задачи. Будущий алгоритм рассматривается как «черный ящик» - функция с неизвестной (или не рассматриваемой) структурой, преобразующая входы в выходы. Суть функциональных тестов: каким бы способом ни решалась задача, при заданных входных значениях должны получиться соответствующие выходные значения.

Структурные тесты составляются для проверки логики решения, или логики работы уже готового алгоритма. Логика определяется последовательностью операций, их условным выполнением или повторением (т.е. композицией базовых конструкций). Совокупность структурных тестов должна обеспечить проверку каждой из таких конструкций.

Чаще всего совокупность тщательно составленных функциональных тестов покрывает множество структурных тестов.

Приведенные понятия различаются тем, что первое рассматривает программу только с точки зрения входов и выходов, тогда как второе относится к ее структуре; но оба понятия не касаются процесса организации тестирования.

Общая последовательность разработки тестов

Наиболее рациональная процедура заключается в том, что сначала разрабатываются функциональные тесты, а затем – структурные.

Функциональное тестирование (тестирование «черного ящика»)

При функциональном тестировании выявляются следующие категории ошибок: некорректность или отсутствие функций;

ошибки интерфейса;
ошибки в структурах данных;
ошибки машинных характеристик (нехватка памяти и др.);
ошибки инициализации и завершения.

Техника тестирования ориентирована:

на сокращение необходимого количества тестовых вариантов;
на выявление классов ошибок, а не отдельных ошибок.

Способы функционального тестирования

Разбиение на классы эквивалентности

Это самый популярный способ. Его суть заключается в разделении области входных данных программы на классы эквивалентности и разработке для каждого класса одного тестового варианта.

Класс эквивалентности – набор данных с общими свойствами, в силу чего при обработке любого набора данных этого класса задействуется один и тот же набор операторов¹.

Классы эквивалентности определяются по спецификации программы. Тесты строятся в соответствии с классами эквивалентности, а именно: выбирается вариант исходных данных некоторого класса и определяются соответствующие выходные данные.

Самыми общими классами эквивалентности являются классы допустимых и недопустимых (аномальных) исходных данных. Описание класса строится как комбинация условий, описывающих каждое входное данное.

Условия допустимости или недопустимости данных задают возможные значения данных и могут описывать:

некоторое конкретное значение; определяется один допустимый и два недопустимых класса эквивалентности: заданное значение, множество значений меньше заданного, множество значений больше заданного;

диапазон значений; определяется один допустимый и два недопустимых класса эквивалентности: множество значений в границах диапазона; множество значений, выходящих за левую границу диапазона; множество значений, выходящих за правую границу диапазона;

множество конкретных значений; определяется один допустимый и один недопустимый класс эквивалентности: заданное множество и множество значений, в него не входящих.

Такие классы можно описать языком логики, например, языком исчисления предикатов. Описания более сложных условий и соответствующих классов (например, элементы массива должны находиться в некотором диапазоне и при этом массив не должен содержать нулевых элементов) могут быть построены на основании приведенных выше условий.

В примере, приводимом в вопросе 2 темы 3, при построении тестов неформально использовался описанный метод. В методических целях были выделены только основные классы тестов. Кроме того, исходя из условия задачи, были выделены классы эквивалентности внутри класса правильных данных.

Анализ граничных значений

¹ Здесь видна связь структурного и функционального тестирования, поскольку набор операторов – это элемент структуры программы, которая, по определению функционального тестирования, игнорируется.

Этот способ построения тестов дополняет предыдущий и предполагает анализ значений, лежащих на границе допустимых и недопустимых данных. Построение таких тестов часто диктуется интуицией.

Основные правила построения тестов:

если условие правильности данных задает диапазон, то строятся тесты для левой и правой границы диапазона; для значений чуть левее левой и чуть правее правой границы;

если условие правильности данных задает дискретное множество значений, то строятся тесты для минимального и максимального значений; для значений чуть меньше минимума и чуть больше максимума;

если используются структуры данных с переменными границами (массивы), то строятся тесты для минимального и максимального значения границ.

Диаграммы причин-следствий

Взаимосвязь классов эквивалентности и соответствующих им действий описывается формально в виде графа на основе автоматного подхода. Граф преобразуется в таблицу решений, столбцы которой в свою очередь преобразуются в тестовые варианты.

Порядок выполнения работы

По результатам лабораторных работ 1-4 разработать тестовые наборы для функционального тестирования (Приложение 1).

Провести тестирование программы и представить результаты в виде таблицы. (Приложение 1).

Выработать рекомендации для корректировки тестируемой программы.

Представить отчет по лабораторной работе для защиты.

Защита отчета по лабораторной работе

Отчет по лабораторной работе должен быть оформлен согласно требований СТО ВГУЭС и состоять из следующих структурных элементов:

титульный лист;

текстовая часть;

Текстовая часть отчета должна включать пункты:

условие задачи;

порядок выполнения

полученные результаты.

Защита отчета по лабораторной работе заключается в предъявлении преподавателю полученных результатов, демонстрации полученных навыков в ответах на вопросы преподавателя.

Контрольные вопросы

Что такое тестирование ПС?

Чем тестирование отличается от отладки ПС?

Для чего проводится функциональное тестирование?

Что такое комплексное тестирование?

Каковы правила тестирования программы «как черного ящика»?

Как проводится тестирования программы по принципу «белого ящика»?

Что такое модульное тестирование?

Как осуществляется сборка программы при модульно тестировании?

Приложение 1

Шаблон таблицы для представления результатов

Тест (значения для входных данных)	Ожидаемый результат (значения для выходных данных)	Фактический результат (полученные значения выходных данных)	Результат тестирования (успешно/неуспешно)
---	---	--	--

ЛАБОРАТОРНАЯ

РАБОТА

№

3.

Модульное

тестирование

Цель работы:

1. Изучение назначения и задач модульного тестирования.
2. Программная реализация тестов, производящих модульное тестирование алгоритма пирамидальной сортировки из курса лабораторных работ.

Подготовка к лабораторной работе

Ознакомиться с лекционным материалом по теме «Отладка и тестирование программного обеспечения» МДК 01.02 Поддержка и тестирование программных модулей».

2. Ознакомиться теоретической частью лабораторной работы.

3. Выполнить задания

Юнит-тестирование

Модульное тестирование, или юнит-тестирование (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок. В данной работе мы будем использовать юнит-тесты для проверки функциональных требований программы.

Для использования юнит-тестов будем использовать JUnit. JUnit — библиотека для модульного тестирования программного обеспечения на языке Java. Скачать свежую версию данной библиотеки можно по адресу <http://www.junit.org>.

Пример юнит-теста, проверяющего равенство $2+2=4$, приведен в листинге 3.1.

Листинг 3.1 — пример юнит-теста

```
import org.junit.Test;
import junit.framework.Assert;

public class MathTest {

    @Test
    public void testEquals() {
        Assert.assertEquals(4, 2 + 2);
        Assert.assertTrue(4 == 2 + 2);
    }

    @Test
    public void testNotEquals() {
        Assert.assertFalse(5 == 2 + 2);
    }
}
```

```
}  
}
```

Для модульного тестирования необходимо использовать драйверы и заглушки. Unit (Элемент) — наименьший компонент, который можно скомпилировать. Драйверы — модули тестов, которые запускают тестируемый элемент. Заглушки — заменяют недостающие компоненты, которые вызываются элементом и выполняют следующие действия:

- возвращаются к элементу, не выполняя никаких других действий;
- отображают трассировочное сообщение и иногда предлагают тестеру продолжить тестирование;
- возвращают постоянное значение или предлагают тестеру самому ввести возвращаемое значение;
- осуществляют упрощенную реализацию недостающей компоненты;
- имитируют исключительные или аварийные условия.

Предварительная подготовка к работе

Так как алгоритм реализован на языке Java, то юнит-тесты следует писать с использованием библиотеки JUnit (<http://www.junit.org>). Допускается использовать любую удобную среду разработки (IDEA / Eclipse / др.).

Для упрощения задачи, предлагается выполнить интеграционное тестирование по восходящему подходу. При использовании этого метода подразумевается, что сначала тестируются все программные модули, входящие в состав системы, и только затем они объединяются для интеграционного тестирования. При таком подходе значительно упрощается локализация ошибок: если модули протестированы по отдельности, то ошибка при их совместной работе есть проблема их интерфейса.

Несмотря на то, что интеграционные тесты не являются юнит-тестами в классическом понимании, при выполнении задач лабораторной работы необходимо использовать библиотеку JUnit и написать модуль юнит-тестирования вручную (используя Java-аннотации @Test и документацию библиотеки JUnit).

Реализовать юнит-тесты следует так, чтобы они выполнял функциональное тестирование программы сортировки в соответствии с целями работы.

JUnit

Таблица 1. Аннотации

Аннотация	Описание
@Test public void method()	Помечает метод как тестовый
@Before public void method()	Метод будет выполняться перед каждым тестом, может быть использован для подготовки тестового окружения (инициализации, чтения данных)
@After public void method()	Метод будет выполняться после каждого теста. Может использоваться для отчистки окружения, удаления временных данных.

Таблица 2. Тестовые методы

Сигнатура	Описание
fail(String)	Вызывает сбой. Может быть использован, чтобы удостовериться, что определенная часть кода не достигнута или пока тестовый метод не реализован.
assertTrue(true) / assertTrue(false)	Постоянно будет true / false. Может быть использован, чтобы предопределить результат теста, пока он не реализован.
assertEquals([String message], expected, actual)	Проверяет, равны ли две величины. Важно: для массивов проверяется ссылка а не содержимое.
assertEquals([String message], expected, actual, tolerance)	Проверяет, совпадают ли величины float и double
assertNull([message], object)	Проверяет, что объект null.
assertNotNull([message], object)	Проверяет, что объект не null.
assertSame([String], actual)	expected, Проверяет, что обе переменные ссылаются на один объект.
assertNotSame([String], actual)	expected, Проверяет, что переменные ссылаются на разные объекты.

Сигнатура

Описание

`assertTrue([message], boolean condition)` Проверяет условие на истинность.

Порядок выполнения работы

Откройте выбранную IDE и создайте проект на основе существующих программных кодов, реализующих алгоритм пирамидальной сортировки.

Подключите к проекту библиотеку JUnit.

Создайте каркас для юнит-тестов (Например, в IDE Eclipse можно выбрать нужный класс, открыть контекстное меню, и выбрать New->JUnit test case, в появившемся диалоговом окне выбрать методы, для которых понадобятся юнит-тесты).

Создайте юнит-тест согласно описанным требованиям.

Отладьте и запустите юнит-тест.

Оцените результаты выполнения юнит-тестирования и сделайте соответствующие выводы.

Содержание отчёта

Необходимо представить исходный код, описание и результаты работы юнит-теста на проверку правильности интеграции модулей системы.

Изложить вводы по результатам тестирования и проделанной лабораторной работе.

ТЕСТ по МДК 01.02 Поддержка и тестирование программных модулей

В. Что такое динамическое тестирование?

О. Это тестирование за счет выполнения кода или программы с различными входными значениями и подтверждением результатов.

В. Что такое GUI-тестирование (GUI Testing)?

О. Тестирование GUI (графического интерфейса пользователя): интерфейс программного обеспечения проверяется на предмет соответствия требованиям.

В. Что такое формальное тестирование?

О. Верификация программного обеспечения, согласно тест-плану, тестовым процедурам и соответствующей документации, с учетом пожеланий клиента.

В. Что такое тестирование на основе рисков?

О. Определяются наиболее важные части системы, затем устанавливается порядок их тестирования, затем следует, собственно, тестирование.

В. Что такое раннее тестирование?

О. Тестирование по возможности проводится как можно раньше, чтобы выявить дефекты на ранних этапах SDLC. Это позволяет быстрее обнаружить и устранить дефекты, экономит расходы.

В. Что такое исчерпывающее тестирование?

О. Тестирование функциональности, с использованием неверных и верных данных ввода и входных условий.

В. Что такое скопление дефектов?

О. Даже небольшой модуль или функциональность могут содержать в себе ряд дефектов, поэтому необходимо больше уделять внимания тестированию функциональности.

В. Что такое «парадокс пестицида»?

О. Если с помощью имеющихся тестовых сценариев не получается обнаружить дефекты, возможно, стоит дополнить/пересмотреть тест-кейсы, чтобы можно было находить больше дефектов.

В. Что такое статическое тестирование?

О. Верификация кода вручную без программы. В этом процессе проблемы находятся в коде, во время его проверки и сравнения с требованиями.

В. Что такое позитивное тестирование?

О. Тестирование, которое проводится в приложении с целью определить, насколько система функциональна. Такой подход больше известен как «тест на прохождение».

В. Что такое негативное тестирование?

О. Тестирование негативных сценариев в ПО: высвечивает ли система ошибку, когда она должна это делать, или не должна.

В. Что такое сквозное тестирование (end-to-end)?

О. Тестирование общей функциональности системы, включая интеграцию данных в модулях.

В. Что такое исследовательское тестирование?

О. Это исследование приложения, чтобы составить представление о его функциональности, добавление (или) изменение существующих тест-кейсов для более качественного тестирования.

В. Что такое «обезьянье тестирование» (Monkey Testing)?

О. Тестирование приложения без какого-либо плана, тестирование выборочных мест, чтобы обнаружить какие-то сложные системные сбои, а затем и дефекты, которые к этому привели.

В. Что такое нефункциональное тестирование?

О. Валидация различных нефункциональных аспектов системы, таких как пользовательские интерфейсы, совместимость, производительность и прочее.

В. Что такое [юзабилити-тестирование](#)?

О. Проверка на предмет того, насколько легко конечные пользователи способны понять и управлять приложением.

В. [Тестирование безопасности](#).

О. Проверяется, насколько хорошо реализованы в приложении все условия безопасности.

В. Что такое [тестирование производительности](#)?

О. Анализ эффективности различных характеристик системы — времени ответа, общей производительности с целью установить, как быстро система работает под нагрузкой.

В. Что такое нагрузочное тестирование?

О. Анализ функциональности и производительности приложения в разных условиях.

В. Что такое стресс-тестирование?

О. Проверка устойчивости системы в условиях превышения пределов обычного функционирования. Или снижение ресурсов системы и сохранение нагрузки на определенном уровне, чтобы проверить, как приложения при этом себя ведет.

В. Что такое процесс?

О. Процесс — это набор практик для достижения определенной цели; может включать инструменты, методы, материалы и людей.

В. Что такое конфигурационное управление?

О. Процесс поиска, организации и контроля изменений в разработке ПО. Или методология контроля и управления проектом разработки ПО.

В. Что такое [процесс тестирования/жизненный цикл](#)?

О. Составление:

Тест-плана

Тест-сценариев

Тест-кейсов

Выполнение тест-кейсов

Проверка результатов

Составление отчетов о дефектах

Дефект-трекинг

Заккрытие дефектов

Тестовый релиз

В. Как расшифровывается CMMI?

О. Capability Maturity Model Integration (Модель зрелости процессов разработки).

В. Что такое разбор программы?

О. Неформальный анализ исходного кода программы с целью выявить дефекты и верифицировать техники программирования.

В. Что такое [модульное тестирование](#)?

О. Тестирование отдельных программ, модулей или элементов кода.

В. Что такое тестирование уровня интеграции?

О. Тестирование соответствующих программ, модулей (или) единиц кода.

В. Что такое тестирование на уровне системы?

О. Тестирование всей компьютерной системы по всем модулям. Такая разновидность тестирования может включать функциональное и структурное тестирование.

В. Что такое альфа-тестирование?

О. Тестирование всей компьютерной системы перед этапом пользовательского тестирования (UAT).

В. Что такое UAT?

О. Тестирование компьютерной системы клиентом, чтобы проверить, соответствует ли система требованиям.

В. Что такое тестовый план?

О. Документ, описывающий масштаб, подход, ресурсы и график тестирования, в котором определены тестовые элементы, отдельные части функционала, тестовые задания, специалисты, которые будут проводить конкретные тесты, и любые риски, требующие дополнительного планирования.

В. Что такое сценарий тестирования?

О. Идентификация всех возможных зон тестирования.

В. Что такое ECP (Equivalence Class Partition)?

О. Метод генерации тест-кейсов.

В. Что такое дефект?

О. Любое несовершенство в работе софта. Или когда ожидаемый результат не соответствует фактической работе приложения.

В. Что такое критичность?

О. Определяет уровень дефекта с функциональной точки зрения, т.е. насколько критичен дефект для приложения.

В. Что такое приоритет?

О. Указывает на срочность устранения дефекта.

В. Что такое повторное тестирование?

О. Повторное тестирование приложения с целью узнать, устранены ли дефекты.

В. Что такое [регрессионное тестирование](#)?

О. Верификация существующих функциональных и нефункциональных зон после того, как были изменены отдельные части приложения или добавлены новые функциональные возможности.

В. Что такое тестирование восстановления?

О. Проверяется возможность системы справиться с некоторыми неожиданными ситуациями.

В. Что такое тестирование глобализации (Globalization Testing)?

О. Тестируется возможность запуска приложения независимо от его географической и культурной среды. Проверяется возможность смены языка, даты, формата и валюты, если приложение разработано для пользователей из нескольких стран.

В. Что такое тестирование локализации?

О. Проверка на предмет того, подходит ли приложение для отдельной локальной группы пользователей, культурных и географических условий.

В. Что такое тестирование установки?

О. Проверяется возможность успешной установки ПО, в соответствии с документацией по установке.

В. Что такое тестирование удаления?

О. Проверка возможности удаления ПО.

В. Что такое тестирование на совместимость?

О. Проверяется совместимость приложения с другим программным и аппаратным обеспечением.

В. Что такое стратегия тестирования?

О. Это часть тест-плана, описывающая, как проводится тестирование и какие разновидности тестирования необходимо сделать.

В. Что такое тест-кейс?

О. Тест-кейс — набор определенных шагов, по которым проверяется функциональность системы.

В. Что такое тест-кейс для валидации бизнес-процессов?

О. Этот тест-кейс составляется для того, что проверить определенное условие или требование.

В. Как определяется хороший тест?

О. Тест-кейс, у которого высокий приоритет обнаружения дефектов.

В. Что такое тестирование по сценарию использования?

О. Такое тестирование определяет, было ли ПО разработано согласно случаю использования.

В. Что такое возраст дефекта?

О. Время между датой обнаружения и датой закрытия дефекта.

В. Что такое дефект Showstopper?

О. Дефект, который вынуждает остановить ход тестирования.

В. Что такое [завершение тестирования](#)?

О. Это последний этап STLC. Руководство составляет отчеты по тестам, разъясняет статистику проекта, исходя из имеющихся данных.

В. Что такое Bucket Testing?

О. Bucket Testing, или A/B-тестирование. Чаще всего исследуется эффект разного дизайна, используется метрика для веб-сайтов. Две версии сайта запускаются на одной или нескольких веб-страницах, чтобы определить разницу в кликах.

В. Что такое критерии запуска и завершения тестирования?

О. Критерии запуска — процесс, который должен быть представлен в начале системы. Это может быть:

SRS – ПО

FRS

Случай использования

Тест-кейс

План тестирования

Критерий завершенности определяет готовность приложения к релизу. Это может быть:

Отчет по тестированию

Метрики

Отчет по анализу теста

В. Что такое тестирование валюты?

О. Это комплексное пользовательское тестирование одновременного доступа к приложению, для верификации влияния на код, модуль или базу данных. Главным образом обнаруживает тупиковые ситуации в коде.

В. Что такое [тестирование веб-приложения](#)?

О. Тестирование веб-приложения проводится на веб-сайте для проверки загрузки, производительности, безопасности, функциональности, интерфейса, совместимости и других вопросов, относящихся к юзабилити.

В. Что такое [функциональное тестирование](#)?

О. Тестирование элементов (или побочное тестирование) позволяет проверить отдельные работу модулей исходного кода.

В. Что такое тестирование интерфейса?

О. Тестирование интерфейса проверяет взаимодействие отдельных модулей. Чаще всего используется для тестирования пользовательского интерфейса приложений с GUI.

В. Что такое гамма-тестирование?

О. Гамма-тестирование проводится , когда ПО уже готово к релизу, проверяется соответствие требованиям.

Лабораторная работа № 3

Тема: Методы и средства разработки технической документации программного продукта.

Цель: Разработать техническую документацию для программного продукта Qt.

Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата
Умения Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования программных средств
Умения Использовать инструментальные средства для автоматизации оформления документации.	Использование текстового редактора.
Знания Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
Знания Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

Задание.

Для готового программного модуля, разработанного в Qt, создать документ, содержащий техническую документацию.

Методические указания:

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.

Программная документация, включает:

техническое задание (назначение, область применения программы, требования, предъявляемые к программе);

текст программы (запись программы с необходимыми комментариями);

описание программы (сведения о логической структуре и функционировании программы);

пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);

эксплуатационные документы.

Программный документ "Пояснительная записка" составляется на стадии эскизного или технического проектов программы. Как правило, на стадии рабочего проекта не используется.

К эксплуатационным документам относят:

описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);

руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);

руководство программиста (сведения для эксплуатации программы);

руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);

описание языка (описание синтаксиса и семантики языка);

руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов.

Эксплуатационный документ "Описание языка" включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.

Эксплуатационный документ "Руководство по техническому обслуживанию" включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

В техническое задание включают:

введение (наименование, краткая характеристика области применения программы);

основания для разработки (документы, на основании которых ведётся разработка, организация, утвердившая документы, дата утверждения, наименование и обозначение темы разработки);

назначение разработки (функциональное и эксплуатационное назначение программы);

требования к программе и программной документации;

техничко-экономические показатели;

стадии и этапы разработки;

порядок контроля и приёмки.

Наиболее существенной частью технического задания является раздел "требования..." В этом разделе приводятся:

требования к функциональным характеристикам (состав выполняемых функций, организация входных и выходных данных, временные характеристики);

требования к надёжности (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа);

требования к информационной и программной совместимости (требования к информационным структурам на входе и выходе, методам решения, исходным кодам, языкам программирования и программным средствам; требования к защите информации);

требования к составу и параметрам технических средств;

требования к программной документации.

Данный раздел может содержать требования к маркировке, упаковке, транспортировке и хранению, а также условия эксплуатации.

Кроме явно описанных в техническом задании требований, следует придерживаться общепринятых правил разработки программ с учётом выбранной парадигмы программирования:

Программа не должна содержать избыточные элементы (все элементы программы адекватны поставленной задаче: нет циклов, массивов и т. п. элементов, без которых можно обойтись).

Алгоритм должен быть структурирован: для функционального стиля программирования - адекватное разбиение на функции (процедуры), для объектно-ориентированного - адекватная иерархия классов. Каждая функция (метод класса) должна реализовывать ровно одно действие.

У функций (методов классов) должны быть параметры. Следует избегать использования в функциях глобальных переменных.

Программа должна аккуратно использовать память: работать с динамическими массивами, в ней не должно быть неиспользуемых блоков памяти, лишних переменных.

Должны проверяться диапазоны вводимых пользователем значений и параметров, передаваемых между модулями программы.

При использовании в программе каких-либо готовых компонент (библиотечных функций, классов) если функция или метод класса может завершиться неудачей, необходимо обязательно проверять это, не полагаясь на незначительность вероятности такого события.

Программа должна быть конфигурируема (важные параметры программы следует выделить в единый блок).

Текст программы

Текст программы представляет собой символическую запись на исходном или промежуточном языке или символическое представление машинных кодов. Текст программы оформляется моноширинным шрифтом (Courier, Lucida Console и т. п.):

Количество операторов на строке должно быть равно 1.

Все операторы, входящие в составной оператор, должны быть сдвинуты вправо на одинаковое количество позиций, при этом операторные скобки (т. е. то, что ограничивает составной оператор), относящиеся к одному блоку, должны располагаться следующим образом: открывающая скобка должна находиться на той же строке, что и оператор, открывающий блок, а закрывающая должна находиться в той же колонке, с которой начинается оператор, открывающий блок. Допускается располагать открывающую скобку на строке, следующей за оператором, открывающим блок, в той же колонке, с которой начинается этот оператор.

Строка исходного текста программы должна целиком располагаться в одной типографской строке (до 80 символов в зависимости от шрифта). Несоблюдение этого правила говорит о слишком большой вложенности блоков, что означает неудачный алгоритм или структуру программы. В таком случае рекомендуется переосмыслить структуру программы, ввести дополнительные функции, заменив какие-то большие части кода их вызовами, переделать алгоритм и т.п.

Если синтаксис языка позволяет, желательно отделять знаки операций пробелами от операндов. Как и в обычном тексте, после запятой должен следовать пробел.

Определения функций или логические части программы следует отделять друг от друга пустыми строками.

Идентификаторы (названия переменных, типов, подпрограмм) должны быть значимыми настолько, чтобы читающий текст программы мог понимать их смысл без присутствия рядом автора. При необходимости объявление переменной или типа может сопровождаться комментарием.

Текст программы должен содержать комментарии, отражающие функциональное назначение того или иного блока программы, структуру программы.

Описание программы

Документ "Описание программы" содержит:

общие сведения (обозначение наименование программы, программное обеспечение, необходимое для функционирования программы, языки программирования, на которых написана программа);

функциональное назначение (классы решаемых задач, сведения о функциональных ограничениях на применение);

описание логической структуры (алгоритм программы, используемые методы, структура программы с описанием составных частей и связи между ними);

используемые технические средства (типы ЭВМ и устройств, которые используются при работе программы);

вызов и загрузка (способ вызова программы с соответствующего носителя данных);

входные данные (характер, организация и предварительная подготовка входных данных, а также их формат, описание и способ кодирования);

выходные данные (характер и организация выходных данных, а также их формат, описание и способ кодирования).

Описание логической структуры программы следует сопровождать блок-схемой программы.

Документ "Описание программы" может содержать также схемы данных, схемы взаимодействия программ, схемы ресурсов системы и проч., оформленные в соответствии с ГОСТ 19.701-90.

Описание применения

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

назначение программы (возможности, основные характеристики, ограничения области применения);

условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);

описание задачи (указываются определения задачи и методы её решения);

входные и выходные данные.

Руководство системного программиста

Документ "Руководство системного программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания системным программистом. Документ состоит из следующих разделов:

общие сведения о программе (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);

структура программы (сведения о структуре, взаимосвязи между модулями программы и с другими программами);

настройка программы (настройка на состав технических средств, выбор функций и т. п.);

проверка программы (способы и методики проверки, контрольные примеры, методы прогона, результаты);

дополнительные возможности;

сообщения системному программисту (тексты сообщений, выдаваемых в ходе выполнения настройки, проверки программы, в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство программиста

Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:

назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);

характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);

обращение к программе (способы передачи управления и параметров данных);

входные и выходные данные (формат и кодирование);

сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство оператора

Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:

назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);

условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);

выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);

сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

3.3 Типовые задания для оценки освоения МДК 01.03 Разработка мобильных приложений

Лабораторная работа №5. Первая программа для Android

Цель работы: Ознакомиться с процессом написания программы в Eclipse и ADT (Android Development Tools) для операционной системы Android. Ознакомиться с принципом добавления в программу элементов интерфейса (кнопки, текстовые поля ввода и т.п.) и обработкой событий этих элементов в Android.

Методические указания.

Считаем, что после предыдущих лабораторных работ (№1..№4) у нас на компьютере уже установлены Java, Eclipse и ADT.

1. Запустите Eclipse. Выберите File/New/Android Application Project (рис. 1).

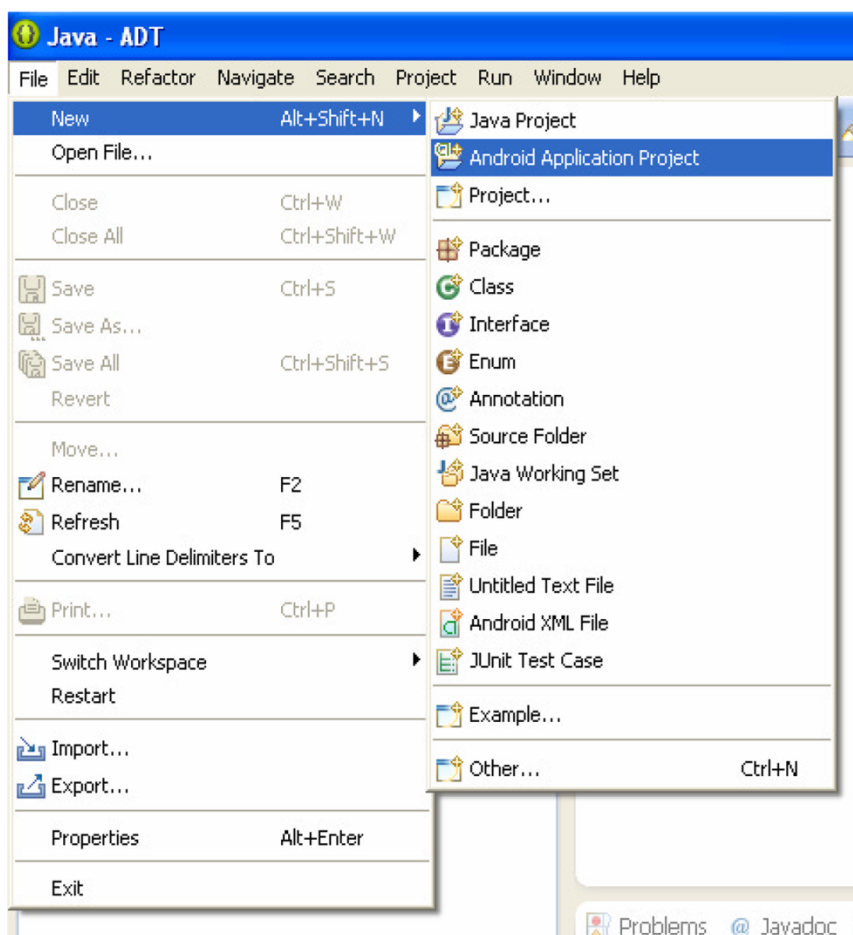


Рис. 1. Создание нового проекта для приложения Android

2. На экране появится окно выбора параметров проекта, приведенное на рис. 2.

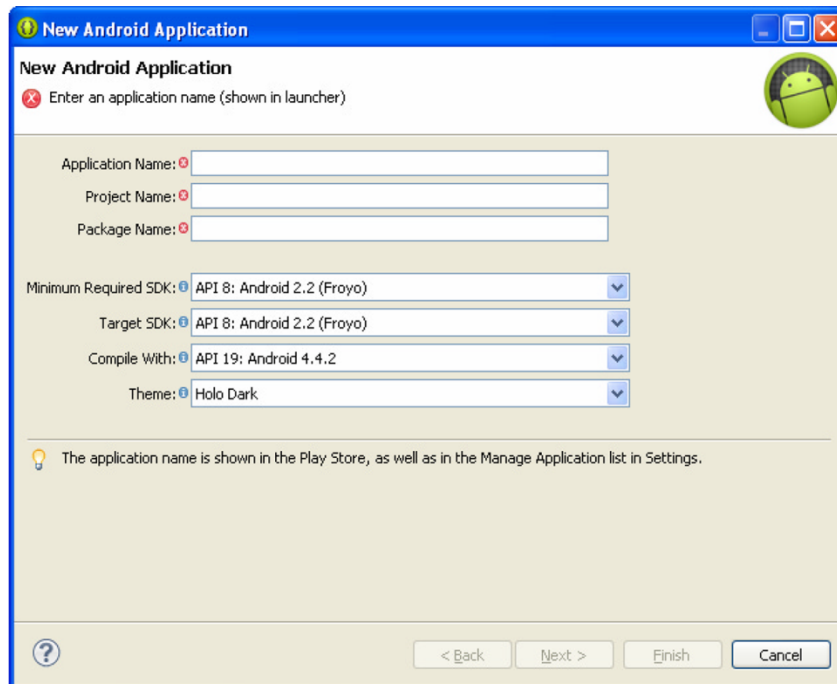


Рис. 2. Окно выбора параметров проекта

В качестве “Application Name” задайте свою фамилию латинскими буквами.

В полях “Minimum Required SDK:” и “Target SDK:” выберите “API 8: Android 2.2 (Froyo)”. В поле “Compile With:” выберите “API 19: Android 4.4.2”. В поле “Theme:” выберите “Holo Dark” (рис. 3).

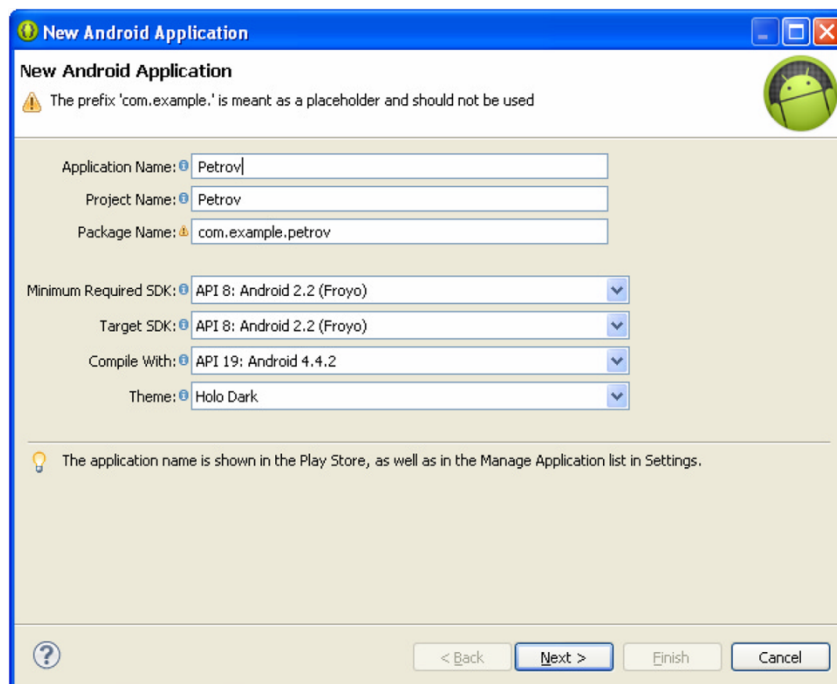


Рис. 3. Окно с установленными параметрами

Появится окно (рис. 4), в котором оставьте все без изменений и нажмите Next.

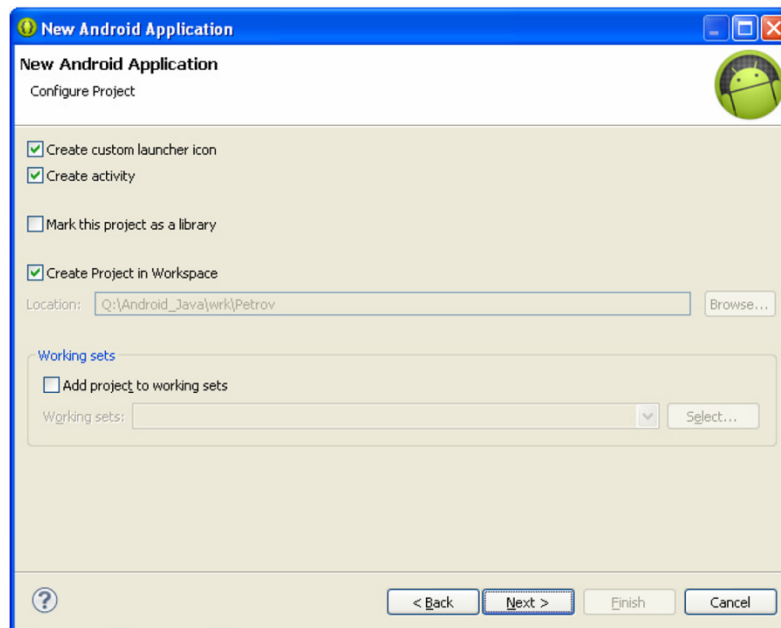


Рис. 4. Окно конфигурации проекта

Появится окно (рис. 5) настройки иконки, в котором тоже оставьте все без изменений и нажмите Next.

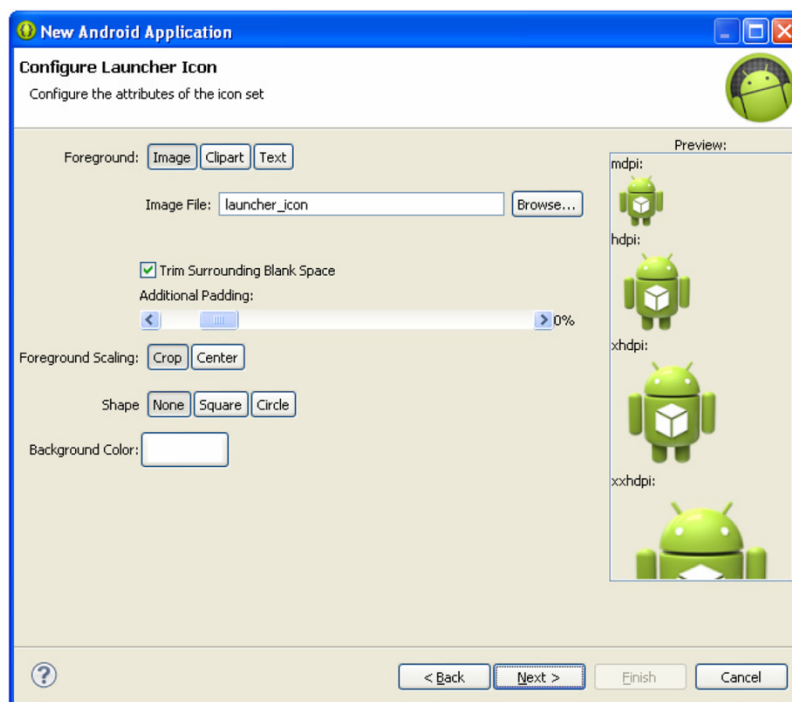


Рис. 5. Окно конфигурации иконки

В появившемся окне (рис. 6) выберите “Blank Activity” и нажмите Next.

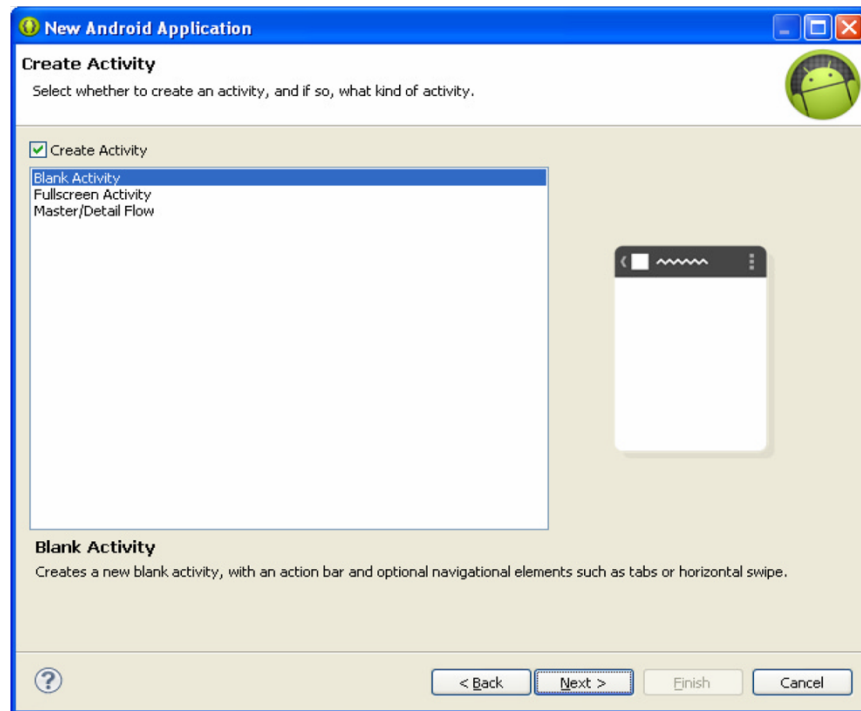


Рис. 6. Окно выбора вида экрана программы

В появившемся окне (рис. 7) оставьте все без изменений и нажмите Finish.

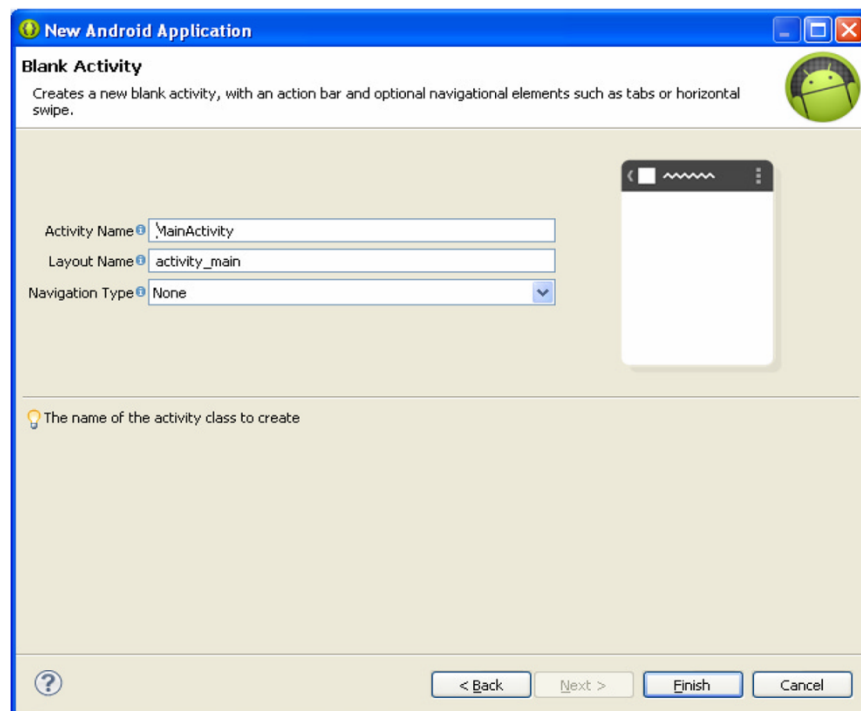


Рис. 7. Завершающее окно настройки проекта

3. После всего сделанного проект (программу) уже можно откомпилировать, установить на телефон или планшет, и запустить.

Чтобы запустить программу на компьютере, нужно вначале создать и запустить виртуальный телефон.

Для того, чтобы создать виртуальный телефон, запустите Android Virtual Device Manager (пункт меню “Window\Android Virtual Device Manager) и в нем нажмите кнопку New. Появится окно, показанное на рис. 8.

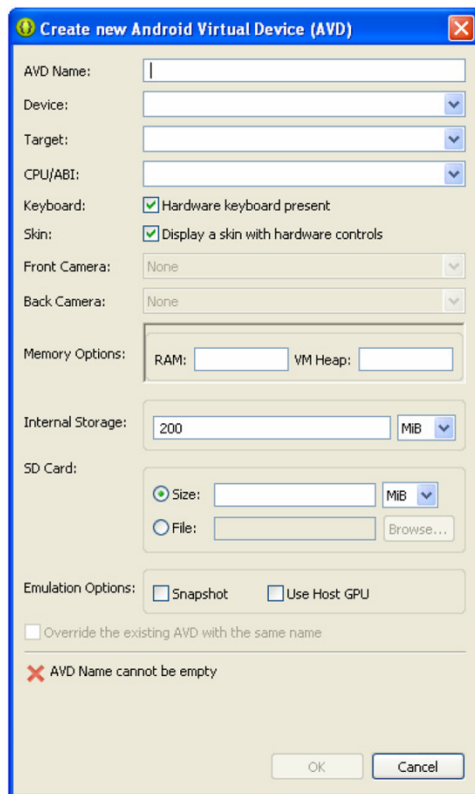


Рис. 8. Окно создания виртуального устройства (телефона, планшета)

В поле “AVD Name” введите имя телефона (любое имя латинскими буквами).

В поле “Device” выбирается тип телефона. Чем больше будет экран и память этого телефона, тем медленнее будет работать виртуальная модель этого телефона и больше потребуется памяти нашего компьютера. Поэтому, для этой лабораторной работы выберите телефон с небольшим экраном - 2.7” QVGA slider (240 x 320: ldpi).

В поле Target укажите версию операционной системы Android, которая будет установлена на этом виртуальном телефоне. Чем более новой будет эта версия, тем медленнее будет загружаться телефон и работать эмулятор. Для данной лабораторной работы укажите “Android 2.2 - API Level 8”.

Поле CPU/ABI заполнится автоматически.

Для поля Back Camera оставьте “None”.

В поле “Memory Options” поставьте “RAM” 512 и “VM Heap” 16. Это оперативная память телефона и, если она будет слишком большой, то эмулятор

телефона будет работать слишком медленно. Если же она будет слишком маленькой, то операционная система и программы могут не запускаться на этом телефоне.

В поле “Internal Storage” ставьте 100 MiB. Это объем встроенной карты памяти телефона.

Поле “SD Card” оставьте пустым.

В итоге заполненное окно должно выглядеть, как это показано на рис. 9.

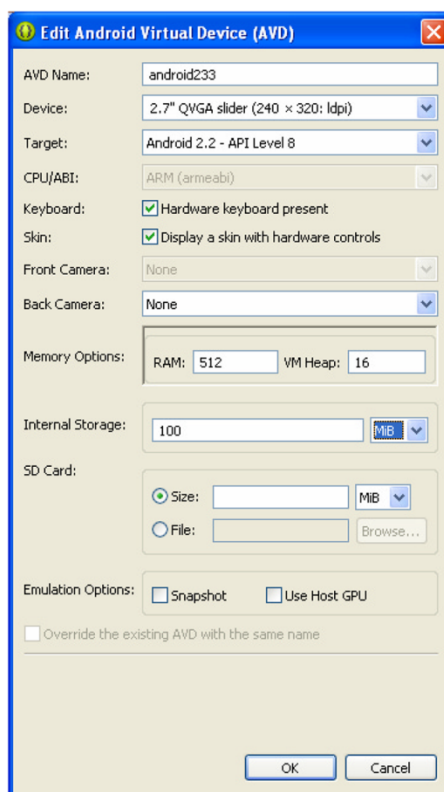


Рис. 9. Выбранные параметры для виртуального телефона

В окне “Android Virtual Device Manager” (рис. 10) нажмите кнопку “Start”.

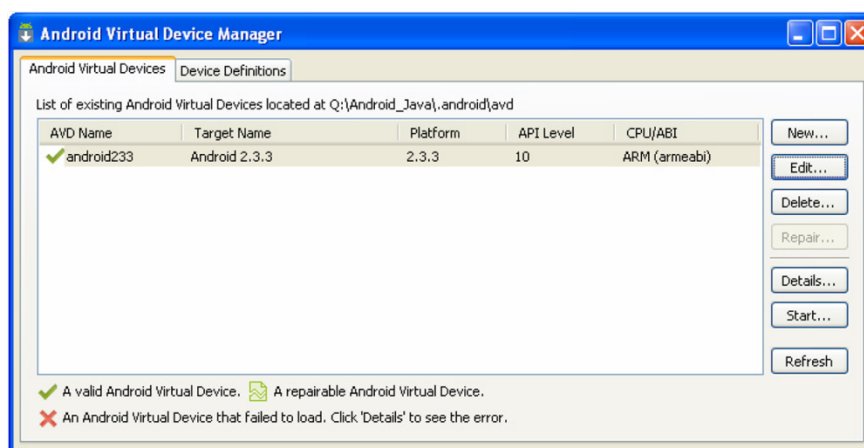


Рис. 10. Окно “Android Virtual Device Manager” с созданным устройством

Появится окно запуска (рис. 11).

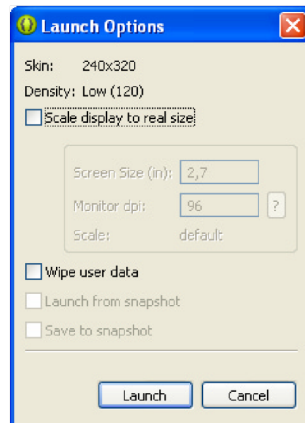


Рис. 11. Окно запуска эмулятора телефона

В поле “Scale display to real size” уберите галочку, если она там есть, и нажмите кнопку Launch. Начнет запускаться эмулятор телефона. При этом для связи Eclipse и этого телефона создается специальный локальный сервер и, если ваш файрвол будет в этот момент спрашивать, разрешать ли соединение с ним, то разрешайте.

Появится окно виртуального телефона и начнется загрузка операционной системы на телефоне. Этот процесс займет некоторое время (несколько минут) и завершится примерно таким экраном, как на рис. 12.

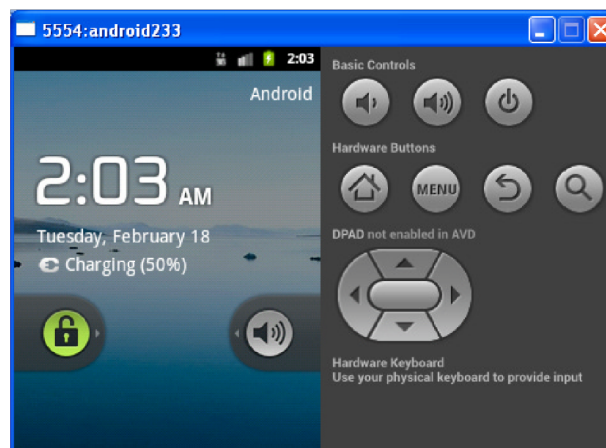


Рис. 12. Экран виртуального телефона после загрузки

Это практически полноценный телефон, на котором установлены самые настоящие программы и, который даже показывает зарядку батареи. После этого окно Android Virtual Device Manager можно закрыть, а окно с виртуальным телефоном должно оставаться открытым.

4. Теперь попробуйте запустить созданный вами проект (программу) на этом виртуальном телефоне. Для этого в меню Project Explorer нажмите правую кнопку

мыши на названии проекта и в выпавшем меню выберите “Run As” и “Android Application” (рис.

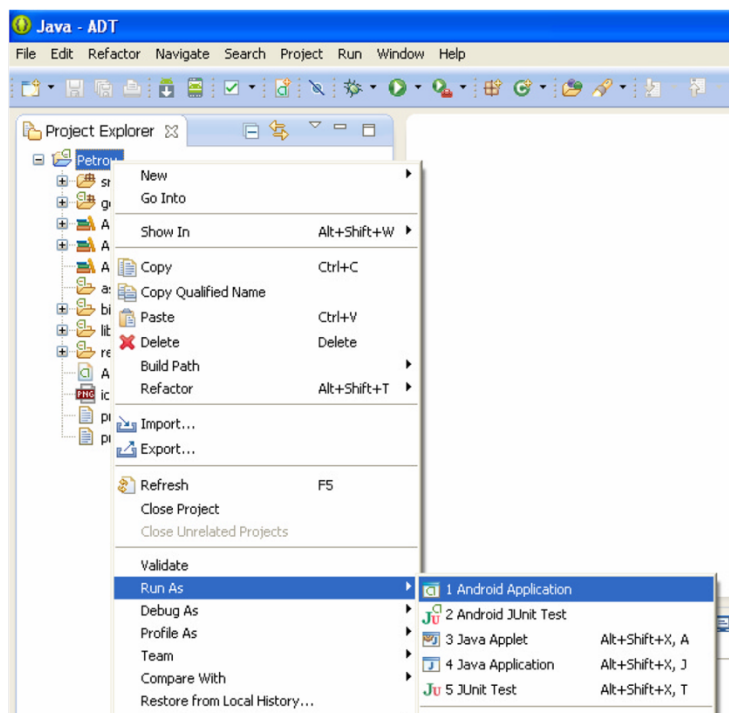


Рис. 13. Запуск приложения

Если проект не содержит ошибок, то пройдет некоторое время (возможно, несколько минут), пока проект откомпилируется и установится на виртуальном телефоне и, разблокировав экран виртуального телефона, Вы увидите запущенное приложение с надписью “Hello, world!” (рис. 14). Это созданная нами программа. В этот момент файрвол так же может беспокоиться и запрашивать разрешение для выполнения запросов в серверу. Разрешайте их.

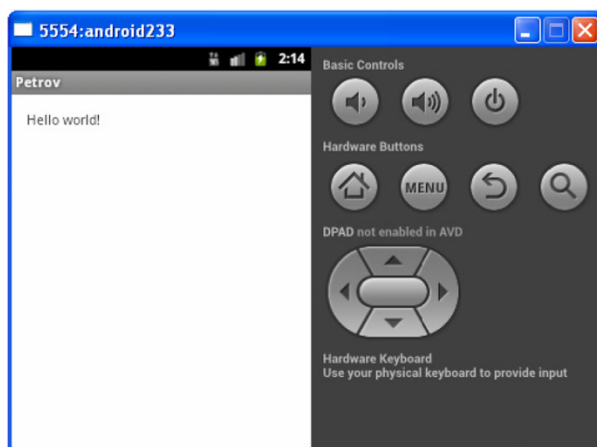


Рис. 14. Откомпилированный и запущенный проект

5. Откомпилированный и запущенный только что созданный проект, как мы убедились, уже содержит надпись “Hello, world!” и больше ничего. Как убрать эту надпись и как вместо нее добавить что-то иное?

В Project Explorer выберите ваш проект и нажмите `res\layout\activity_main.xml` (рис. 15).

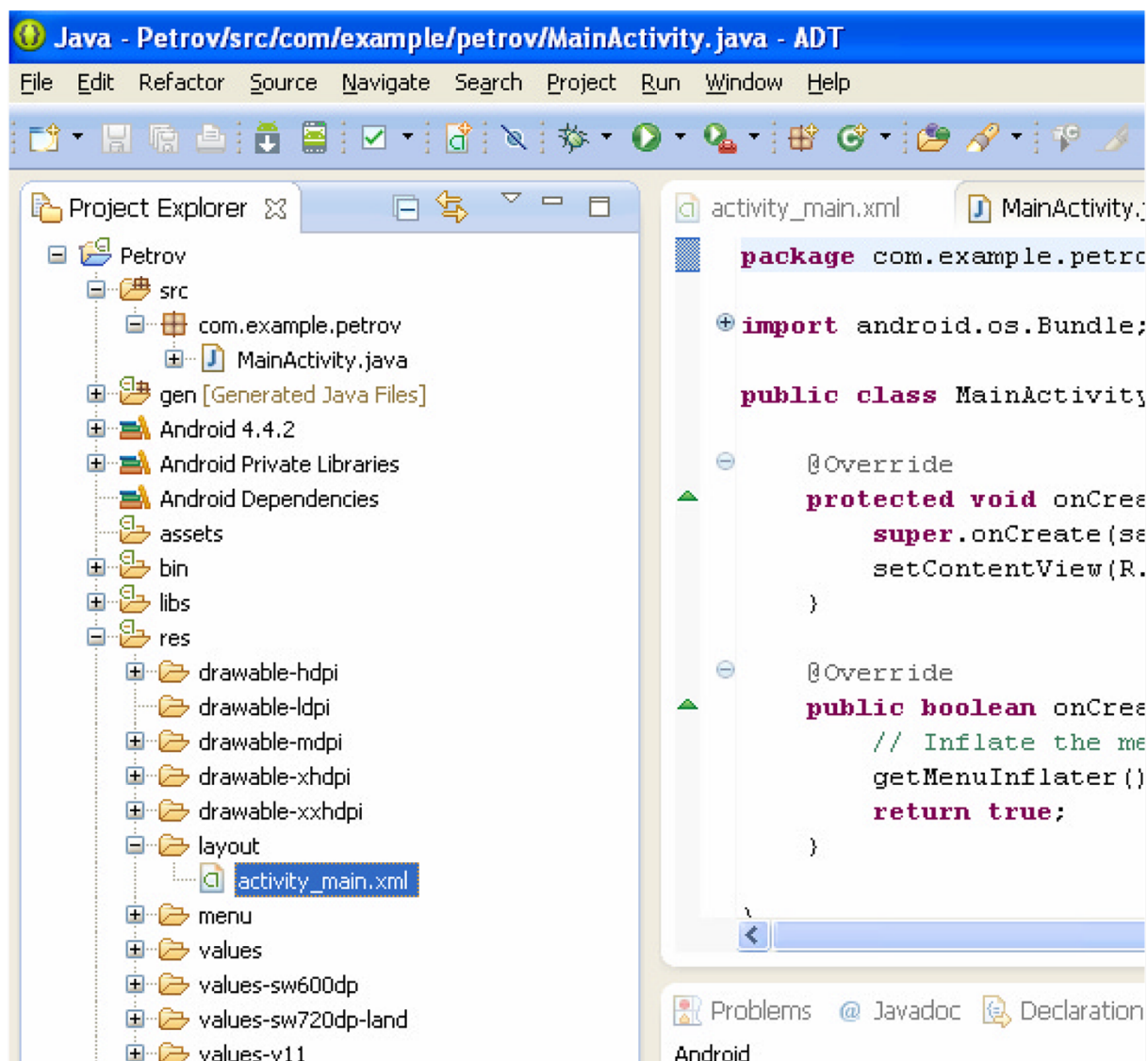


Рис. 15. Окно Project Explorer

Откроется окно визуального редактирования содержимого экрана нашей программы (приложения) (рис. 16). С расположенного в левой части окна Palette перетащите на окно программы кнопку “Button”. Нажмите на эту кнопку и в окне Properties в правой части экрана в поле текст введите новую надпись для нее “Нажми меня!” (рис. 17).

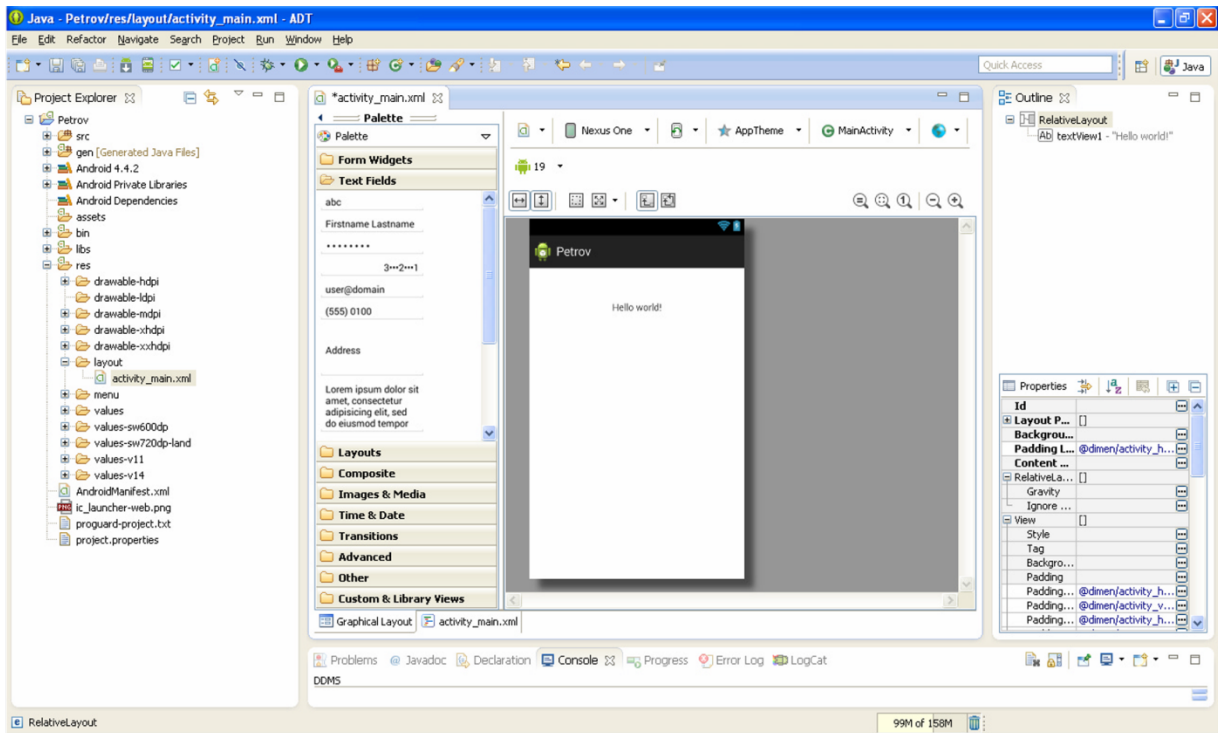


Рис. 16. Окно визуального редактирования содержимого экрана программы

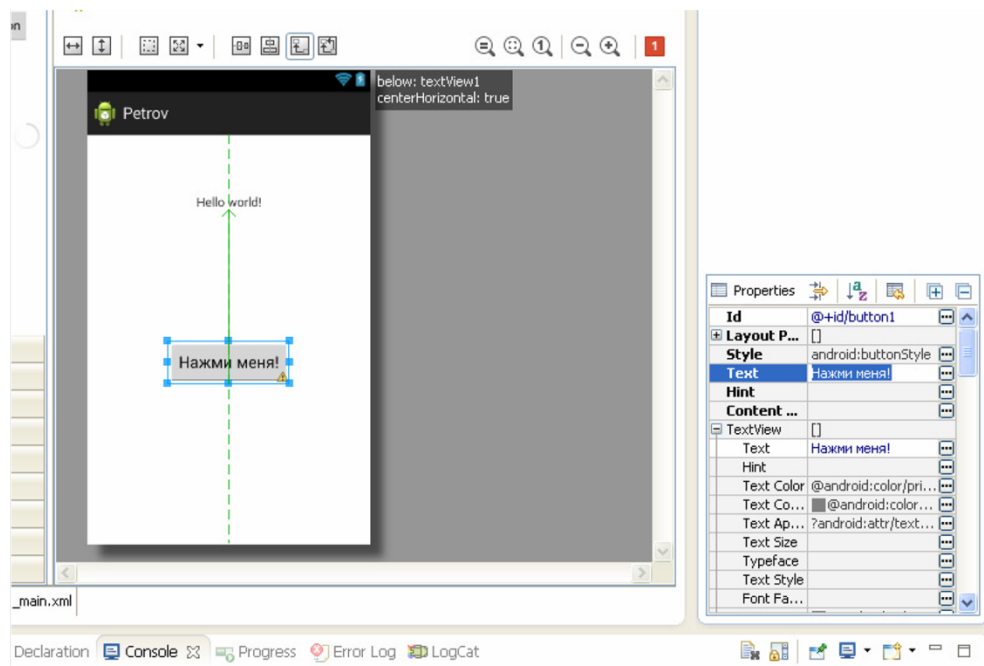


Рис. 17. Добавление кнопки и изменение надписи на ней

Таким же образом измените и текст надписи “Hello, world!” на свои имя и фамилию, например, на Владимир Петров. Снова запустите проект на выполнение и в окне виртуального телефона появится новый вариант нашей программы (рис. 18).

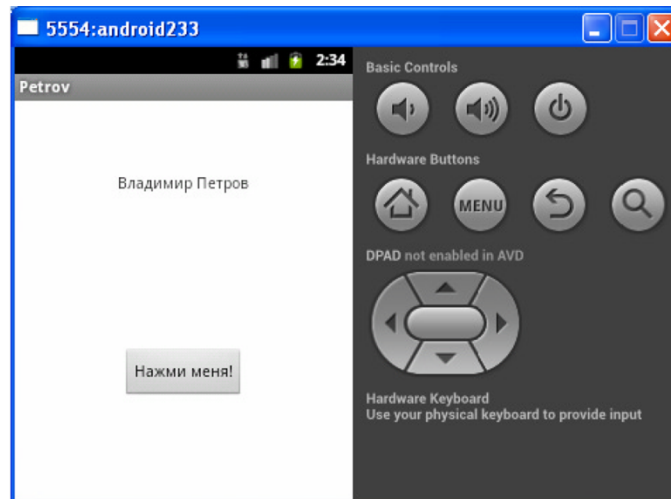


Рис. 18. Измененная программа

Можно попробовать понажимать на эту кнопку, но ничего не произойдет. Почему? Мы пока не написали никакого текста программы, который бы реагировал на это нажатие.

6. В редакторе нажмите на кнопку и посмотрите в окне Properties значение поля Id. Допустим, там написано “@+id/button1”. Это значит, что Id кнопки - “button1”. По этому Id к ней можно обращаться из нашей программы.

Аналогично, нажав на надпись, можно узнать и ее Id. Допустим, в поле Id надписи написано «@+id/textView1». Значит Id надписи - “textView1”.

В Project Explorer для своего проекта выберите src\MainActivity.java (рис. 19).

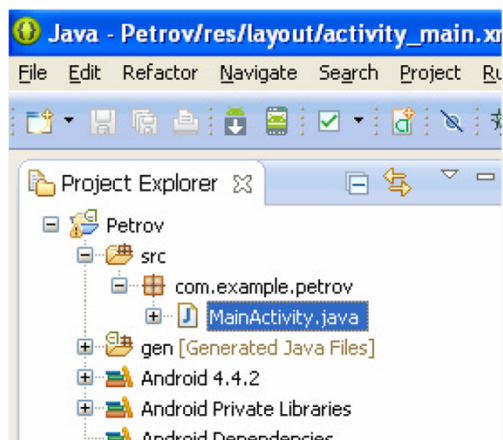


Рис. 19. Выбор для редактирования MainActivity.java

MainActivity.java откроется в окне редактора.

Чтобы работать с текстовой надписью и кнопкой, для начала нужно создать поля (переменные) этих типов. В Java для Android тип данных для кнопок называется Button, а тип данных для надписей - TextView.

Добавьте в класс MainActivity поле кнопки и поле текстовой строки, придумав для них любые названия, например katya и olesya:

```
TextView katya;  
Button olesya;
```

Eclipse подчеркнет TextView и Button как ошибки. Чтобы устранить ошибку, добавьте в программу import обоих типов данных.

Теперь в методе onCreate (он будет запускаться при инициализации окна программы) добавьте следующие две строки:

```
katya = (TextView) findViewById(R.id.textView1);  
olesya = (Button) findViewById(R.id.button1);
```

Смысл этих двух строк в том, что по известным нам ранее Id текстовой строки и кнопки (textView1 и button1) мы находим их в окне программы и запоминаем в специально для этого созданных нами полях katya и olesya.

Теперь остается только назначить обработчик для события нажатия на кнопку. Для этого в этом же методе onCreate добавьте строку:

```
olesya.setOnClickListener(обрabotka);
```

В класс добавьте новый экземпляр класса OnClickListener следующим образом:

```
OnClickListener obrabotka = new OnClickListener() {  
    public void onClick(View v) {  
        katya.setText("Кнопка была нажата!");  
    }  
};
```

Здесь действие katya.setText("Кнопка была нажата!"); будет выполняться при нажатии кнопки. Вместо этого действия можно по аналогии прописать в качестве реакции на нажатие кнопки любое другое действие.

Eclipse подчеркнет OnClickListener как ошибку и нужно его импортировать в программу. Eclipse предложит два варианта импорта (рис. 20).

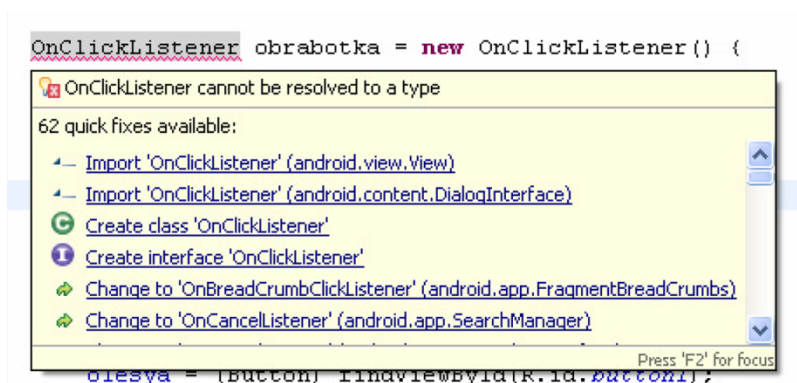


Рис. 20. Для импорта OnClickListener нужно выбрать android.view.View

Так же придется импортировать тип данных View, подведя к нему мышку в Eclipse. После этого ошибок не останется и можно снова откомпилировать и запустить программу. В этот раз после нажатия на кнопку текст с “Владимир Петров” изменится на “Кнопка была нажата!” (рис. 21).

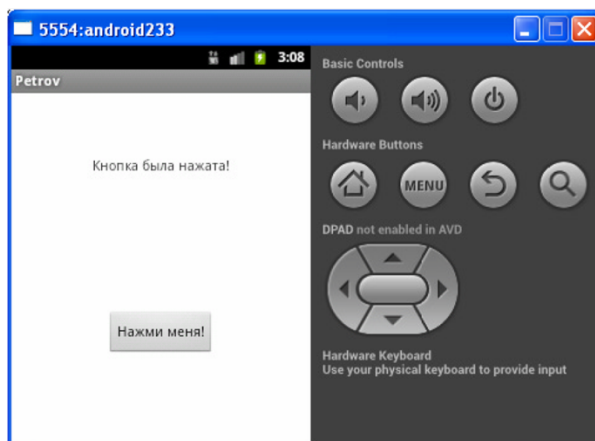


Рис. 21. Результат нажатия на кнопку в запущенной программе

7. Покажите откомпилированную и запущенную программу преподавателю.

8. Если у вас есть реальный планшет или телефон, работающий под управлением операционной системы Android, то попробуйте установить туда написанную вами программу и запустить ее. Для этого из папки bin вашего проекта возьмите файл с именем проекта и расширением “.apk”. Это стандартное расширение для установочных программ в операционной системе Android. Самостоятельно разберитесь или почитайте в Google, как устанавливать программу из apk-файла.

9. Дома самостоятельно попробуйте модифицировать написанную Вами программу, добавляя другие элементы интерфейса, реакции на нажатия кнопки и т.д.

3.4. Типовые задания для оценки освоения МДК 01.04 Системное программирование

Лабораторная работа №1 Основы программирования на Ассемблере

Цель работы

- 1.Использование регистров, доступных программисту.
- 2.Знакомство со структурой программы.
- 3.Написание элементов программы.

Общие сведения

Язык Ассемблер разработан для максимального использования конкретной специфики компьютера. Следовательно, для того, чтобы написать программу на Ассемблере, важно знать архитектуру компьютера.

Внутренние регистры

Микропроцессор IBM PC содержит три группы регистров данных и адресов, 16- битовый указатель команд IP (Instruction Pointer) и 16-битовый регистр флагов.

Регистры работы с данными

К первой группе относятся 4 регистра общего назначения: AX, BX, CX, DX. Их можно рассматривать как четыре 16-битовых или восемь 8-битовых регистров. Эти регистры образованы из 8-битовых регистров: регистр AX состоит из двух байтовых регистров AL и AH, регистр BX из BL, BH, CX из CL, CH, и DX из DL и DH. Здесь L и H означают младшие (Low-order) и старшие (High-order) байты 16- битовых регистров. Например, регистры AL и AH образуют соответственно младший и старший байты регистра AX. Всеми этими регистрами можно пользоваться при программировании, но следует учитывать, что ряд команд использует их неявным образом, в частности:

регистр AX, аккумулятор (accumulator), используется при умножении и делении слов, в операциях ввода-вывода и в некоторых операциях над строками; регистр AL используется при выполнении аналогичных операций над байтами.

регистр BX, базовый регистр (base register), часто используется при адресации данных в памяти;

регистр CX, счетчик (count register), используется как счетчик числа повторений цикла и в качестве номера позиции элемента данных при операциях над строками. Регистр CL используется как счетчик при операциях сдвига и циклического сдвига на несколько битов;

регистр DX, регистр данных (data register), используется при умножении и делении слов. Кроме того, в операциях ввода-вывода он используется как номер порта и для хранения номера строки и столбца для выводимого символа.

Регистры сегментов

Оперативная память разделена на сегменты. Процессор может иметь дело одновременно с четырьмя сегментами. Начальные адреса этих сегментов содержатся в его четырех регистрах сегментов. Эти регистры выполняют следующие функции: регистр сегмента команд CS (code segment) указывает на сегмент, содержащий текущую исполняемую программу;

регистр сегмента стека SS (stack segment) указывает на текущий сегмент стека; регистр сегмента данных DS (data segment) указывает на текущий сегмент данных;

регистр дополнительного сегмента ES (extra segment) указывает на текущий дополнительный сегмент.

Регистры указателей и индексов

Для вычисления адреса команды в сегменте команд процессор извлекает номер сегмента памяти из регистра CS, а смещение - из регистра IP. Подобным образом за счет выбора номера сегмента из соответствующего регистра сегмента, а смещение - из другого регистра осуществляется доступ к данным других сегментов. Для доступа к сегменту данных микропроцессор извлекает номер сегмента из регистра DS, а смещение - из регистра BX или индексного регистра (SI или DI). Для доступа к сегменту стека процессор извлекает номер сегмента из регистра SS, а смещение - из регистра указателя (SP или BP). Выбирая номер сегмента из регистра ES, процессор может также получить доступ к дополнительному сегменту.

Флаги

Очень важным является регистр, в котором содержатся признаки последней выполненной операции, именуемый флагами. В 16-битовом регистре флагов размещены девять флагов.

Это:

1. флаг переноса CF (carry flag) (Бит 0) изменяется во многих операциях;
2. флаг четности PF (parity flag) (Бит 2); PF=1, если в результате операции количество битов в состоянии 1 четно.
3. флаг вспомогательного переноса AF (auxiliary carry flag). (Бит 4);
4. флаг нуля ZF (zero flag), (Бит 6) равен 1, если результат операции равен 0;
5. флаг знака SF (sign flag) (Бит 7) SF = 1, если результат отрицательный;
6. флаг трассировки TF (trap flag), (Бит 8);
7. флаг прерывания IF (interrupt enable flag), (Бит 9) разрешает прерывания от внешних устройств. Если IF=0 прерывания запрещены;
8. флаг направления DF (direction flag). (Бит 10);
9. флаг переполнения OF (overflow flag). (Бит 11).

Для проверки состояния флагов используются команды условного перехода.

Простейшая программа на Ассемблере ;

Первая программа на Ассемблере

```

; Данные программы
DATA SEGMENT                                ;Начало сегмента данных
    HELLO    DB 'Здравствуйте!$'           ;Данные программы
DATA ENDS                                    ;Конец сегмента данных
;
;Код программы
CODE SEGMENT                                ;Начало сегмента кодов
    ASSUME CS: CODE, DS: DATA
    START:  MOV AX, DATA                    ;Занесение адреса сегмента
            MOV DS, AX                      ;данных в регистр сегмента данных
            MOV AH, 9                       ;Функция вывода строки сообщения
            MOV DX, OFFSET HELLO            ;Смещение HELLO в регистр DX
            INT 21H                         ;Вызов программы ввода/вывода
            MOV AH, 4CH                     ;Функция выхода из программы
            INT 21H                         ;Вызов программы ввода/вывода
CODE ENDS                                    ;Конец сегмента кодов
;
END START                                    ;последний оператор текста программы с меткой
                                           ;начала выполнения программы.

```

Текст программы на Ассемблере можно набрать в редакторе текстов, например, в **NotePad** (в папке **Accessories (Стандартные)**).

Подготовка программы состоит из 3 шагов (Рис. 1):

1. Подготовка исходного текста программы (Например, **HELLO.ASM**)
2. Создание файла **.OBJ**,
3. Создание файла **.EXE**,
4. Выполнение **EXE** файла.

Создание программы



Рисунок 1. Этапы создания программы на Ассемблере

После набора текста программы HELLO.ASM в NotePad откройте окно MS- DOS Prompt и туда, с помощью мышки из окна Проводника и папки где находится исходный текст, втащите название файла TASM или MASM и, после пробела, также втащите туда же название файла с программой. Нажмите ввод. Если ассемблирование пройдет без ошибок, будет создан объектный файл с расширением .OBJ. В окне MS-DOS Prompt таким же образом запустите редактор связей TLINK или LINK с названием объектного файла. Будет создан выполняемый файл .EXE, который можно запустить на выполнение.

Запустить Ассемблер MASM или TASM можно также из Norton Commander или FAR.

ПРАВИЛА ЗАПИСИ ПРОГРАММЫ НА АССЕМБЛЕРЕ

Программа, как правило, состоит из 3 сегментов: сегмента стека, сегмента данных и сегмента кодов (программы).

Каждый сегмент начинается с названия сегмента и ключевого слова segment. Вторым оператором в сегменте кодов должен быть записан оператор assume, который устанавливает соответствие между регистрами сегментов и названиями сегментов. Сегмент заканчивается оператором с названием сегмента и слова ends.

Сегмент кодов обычно содержит основную процедуру и несколько процедур, и для каждой из них записывается название (например, myproc1); после названия в начале подпрограммы пишется слово proc, а конце - endp. Каждая процедура заканчивается словом ret.

myproc1 proc

ret

myproc1 endp.

Данные обычно записываются в специальном сегменте данных, пример которого дан ниже.

mydata segment x db ?

y dw ?

mydata ends

Программа, состоящая из указанных сегментов и включающая в себя основные директивы, а также некоторые дополнительные конструкции, необходимые для создания работающей программы, представлена на рисунке:

;MODEL.ASM

```

SSEG SEGMENT STACK ;Начало сегмента стека
DB 256 DUP (?)      ;Указана длина стека
SSEG ENDS           ;Конец сегмента стека
;
DSEG SEGMENT        ;Начало сегмента данных
    CATS DB 'A'     ;Ваши данные
    RATS DB 'B'     ;Ваши данные
    BATS DB 'C'     ;Ваши данные
DSEG ENDS           ;Конец сегмента данных
;
CSEG SEGMENT        ;Начало сегмента кодов
ASSUME CS:CSEG,DS:DSEG,SS:SSEG ; указание названий сегментов
регистров сегментов
START PROC ;Начало главной процедуры START
PUSH DS ;Работа программы начинается
PUSH AX ;с установки AX=0
MOV BX, DSEG ;Задание базового адреса
MOV DS, BX ; сегмента данных
CALL MAIN ;Вызов процедуры MAIN
RET
START ENDP ;конец процедуры START

```

```

MAIN PROC NEAR      ;Начало процедуры MAIN
    MOV AH, CATS    ;Ваша программа
    MOV BH, RATS    ;Ваша программа
    MOV CH, BATS    ;Ваша программа
    MOV DL, CATS    ;Занесение буквы А в DL для вывода
    MOV AH, 2       ;Задание функции вывода символа
    INT 21H         ;Обращение к системной программе
    RET             ;Организация возврата
MAIN ENDP           ;Конец процедуры MAIN
;
CSEG ENDS           ;конец сегмента кодов
END START           ;указываем имя первой выполняемой процедуры

```

Текст программы заканчивается оператором **END**, в котором указывается имя процедуры или оператора, с которого начинается выполнение программы. Оператор **END** указывает Ассемблеру, где завершить трансляцию программы.

Описание данных

Все данные и переменные, используемые в программе, должны быть описаны явно в сегменте данных.

Синтаксис оператора описания данных:

[имя переменной] Dx список значений

[имя переменной] является необязательным параметром и указывает на имя данных, по которому можно к ним обращаться.

Тип данных определяется параметром Dx. Возможны следующие значения **DB, DW, DD, DQ, DT**.

DB определяет данные длиной байт. Например:

MY_BYTE DB 233 ;переменная MY_BYTE занимает 1 байт и значение 233

MY_LIST DB 15, 19, 23, 22 ;список байтовых данных под именем MY_LIST

X DB ? ;переменная X длиной 1 байт, значение ее не определено

Y DB 10 DUP (?) ;переменная Y длиной 10 байт, значение ее не определено

STRING DB 'ASSEMBLER' ;строка из символов ASCII

DW определяет данные длиной слово. Например:

MY_WORD DW 3453 ;переменная MY_WORD занимает слово, значение 34533

Формат команды Ассемблера

[метка :] КОП [Операнд 1] [,Операнд 2];Комментарий

Элементы команды, укатанные в квадратных скобках, в различных командах могут отсутствовать.

Пересылка данных

К наиболее простым и часто используемым командам Ассемблера относится команда **MOV** (переслать). Она используется для того, чтобы

- 1) данные длиной 1 байт или 2 байта переслать:
 - а) из регистра в регистр;
 - б) из регистра в память;
 - в) из памяти в регистр;

2) данные, непосредственно записанные в команде, переслать в регистр или по заданному адресу в память.

передастся системной программе ДОС. Одновременно с выводом символа на экране курсор перемещается на одну позицию вправо. Затем управление возвращается прикладной программе, и начинается выполнение команды, следующей за командой **INT21H**.

Ввод символа с клавиатуры по прерыванию типа 21H

Для ввода символа с клавиатуры используется непосредственное обращение к функции ввода/вывода системной программы:

MOV AH, 1 указываем системе, что хотим ввести символ

INT 21H вызываем системную программу

Программа ждет нажатия клавиши для ввода символа. После нажатия символ в коде ASCII попадает в регистр **AL**.

Организация циклов

В программе часто необходимо выполнять многократно некоторые фрагменты программы. Для этого в Ассемблере имеется удобное решение организации циклов. Количество повторений цикла заносится в регистр **CX** (только в **CX!**), а команда **LOOP** обеспечивает выполнение заданного количества повторений. Как правило, команда **LOOP** ставится в конце повторяющейся последовательности команд программы. Формат команды:

Loop метка

Команда вычитает единицу из содержимого регистра **CX** и, если содержимое неравно нулю, передает управление на метку. Если регистр **CX** становится равен нулю, выполняется следующая за **LOOP** команда. В регистр **CX** заносится количество повторений цикла.

Ассемблеры MASM и TASM

MASM является двухпроходным Ассемблером. Это означает, что исходный текст программы он просматривает два раза. За первый проход он определяет все переменные в программе и строит таблицу их адресов. Таблица содержит имена переменных, адреса, в которых они находятся и тип данных. Во время второго прохода Ассемблер заменяет каждую команду на машинный код и определяет длину каждой команды.

TASM является однопроходным ассемблером. Он создает машинный код и таблицу адресов за один проход.

Листинг программы

```
Microsoft (R) Macro Assembler Version 5.10          9/3/96 17:54:04
0000                                     sseg segment stack
0000 0100[                                     db 256 dup (?)
    ??
    ]
0100                                     sseg ends
0000                                     dseg segment
```

```

0000 61          cats db 'a'
0001 62          rats db 'b'
0002 63          bats db 'c'
0003           dseg ends
;
0000           cseg segment
           assume cs:cseg, ds:dseg, ss:sseg

0000           start proc far
0000 1E          push ds          ;
0001 50          push ax          ;
0002 BB ---- R   mov bx,dseg
0005 8E DB       mov ds,bx
0007 E8 000B R   call main
000A CB         ret
000B           start endp
000B           main proc near
000B 8A 26 0000 R mov ah,cats
000F 8A 3E 0001 R mov bh,rats
0013 8A 2E 0002 R mov ch,bats
0017 8A 16 0000 R mov dl,cats
001B B4 02       mov ah,2
001D CD 21       int 21h
001F CB         retf
0020           main endp
0020           cseg ends
           end start

```

Рассмотрим листинг подробнее. Слева в каждой строке листинга стоит четырехзначное число. Это значение смещения для команды, текст которой напечатан в той же строке справа между значением смещения и исходной командой записано машинное представление этой команды. В сегменте данных (ему присвоено имя DSEG) описаны три переменные с именами CATS, RATS, BATS. Их машинные коды 61, 62, 63 и для них выделены три байта памяти (сегмент заканчивается значением 3). В сегменте кодов (его имя CSEG) в левой части указаны смещения адресов соответствующих команд. По их значениям можно определить длины машинных команд.

Работа с отладчиком DEBUG

Отлаживать программу удобнее с отладчиком. Отладчик **DEBUG** позволяет отслеживать выполнение программы по шагам, просматривать содержимое регистров, запускать программу с заданных точек.

Запустите программу **DEBUG** с выполняемым файлом **.EXE**. После появления знака – (тире) наберите команду.

Команды DEBUG:

-R высветить содержимое регистров

-Т выполнить одну команду

-С[адрес] выполнить программу до указанного адреса -Q выйти из DEBUG

-Р выполнить процедуру (фрагмент программы, начинающийся с CALL или INT).

Выполнение лабораторной работы

1. Текст программы на Ассемблере наберите в редакторе текстов, например, NotePad. Файл должен иметь расширение .ASM

2. Для трансляции программы на ассемблере выполните программу MASM или TASM с указанием имени файла с вашей программой, которая имеет расширение .ASM. На экране появится сообщение:

Source filename [.ASM]: (наберите имя-файла программы)

Object filename [имя-файла.OBJ]: (нажмите ввод)

Source listing {NUL.LST}: (укажите имя-файла листинга или нажмите ввод)

Cross-reference [NUL.CRF]: (нажмите ввод)

Результатом работы является объектный модуль, который записывается в файл с расширением .OBJ.

3. Если нет ошибок (сообщение Severe errors 0) выполните программу LINK, функция которой сформировать выполняемый модуль с расширением .EXE, подключив к нему необходимые стандартные программы. Появится сообщение:

Object modules [.OBJ]: (наберите имя-файла с расширением .OBJ)

Runfile [имя-файла.EXE]: (нажмите ввод)

Libraries [.LIB]: (нажмите ввод)

4. Запустите на выполнение полученный .EXE файл

5. Подготовьте отчет по лабораторной работе. Отчет должен содержать вариант задания, исходный текст программы с комментариями каждой команды, листинг программы, пояснения к листингу.

Варианты заданий

1. Напишите программу вывода инициалов вашего имени и фамилии, после каждой буквы поставьте точки.

2. Выведите на экран двоичное представление символа. Программа вводит символ в коде ASCII с клавиатуры, выводит на экран его двоичное представление (например, двоичное представление буквы V - это 01010110).

3. Напишите программу сложения пяти чисел 1, 2, 3, 4, 5. Ведите двоичное число, представляющее сумму этих чисел.

4. Напишите программу ввода с клавиатуры двух символов, вычисления их суммы и вывода двоичного представления суммы на экран.

Контрольные вопросы

1. Что такое оперативная память?

2. Какие регистры имеются в ШИМ РС?

3. Какова длина и назначение регистров?

4. Что находится в регистре сегмента?
5. Что делает команда пересылки?
6. Что такое сегмент?
7. Из каких частей состоит программа на Ассемблере?
8. Что такое процедура?
9. Какая информация содержится в листинге программы?
10. Как вычислить смещение адреса по листингу программы?
11. Куда вводится символ при вводе по прерыванию INT 21H?
12. Откуда выводится символ при выводе по прерыванию INT 21H?
13. Как организуется цикл в программе?

IV ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ

4. 1 Промежуточная аттестация по ПМ01

Задачи для промежуточной аттестации по ПМ01

Разработать приложение, позволяющее производить конвертацию валюты. Список валют и их курс хранятся в текстовом файле. Данные из текстового файла считываются в массив, состоящий из записей, содержащих название валюты и курс. Пользователь выбирает валюту из списка → название валюты и курс появляются в поля Label. В поле Edit пользователь вводит сумму в рублях. Далее после щелчка по кнопке «Рассчитать» программа выводит результат

1. Разработать приложение «Тест», состоящий из 10 вопросов и выставяющий оценку. Вопросы и варианты ответов, считываются из текстового файла
2. Разработать приложение, позволяющее придумывать сложный пароль по фамилии, имени, отчеству и году рождения. Алгоритм кодирования придумать самостоятельно. Данные для формирования паролей считываются с текстового файла. Пароли сохраняются в другой текстовый файл.
3. Разработать программу «Пароль». При загрузке программы запрашивает логин и пароль. Если пользователь не зарегистрирован, предлагает зарегистрироваться. Организовать хранение логинов и паролей пользователей в текстовом файле, который загружает данные в массив при запуске приложения.
4. Разработать программу, которая формирует ведомость успеваемости студента. Фамилия и имя студента считываются из текстового файла. Для каждого студента вводятся 3 оценки, рассчитывается средний балл и вся информация (Фамилия, имя, оценка1, оценка2, оценка3, средний_балл) сохраняется в другой текстовый файл по выбору пользователя.
5. Разработать приложение, которое переводит числа из десятичной системы счисления в 2сс, 8сс, 16 сс
6. Разработать приложение, которое переводит числа из 2сс, 8сс, 16 сс в десятичную
7. Разработать приложение «Двоичный калькулятор», позволяющее производить операции сложения, вычитания двоичных чисел
8. Разработать приложение «Восьмеричный калькулятор», позволяющее производить операции сложения, вычитания восьмеричных чисел
9. Разработать приложение «Шестнадцатеричный калькулятор», позволяющее производить операции сложения, вычитания шестнадцатеричных чисел

10. Разработать приложение, позволяющее придумывать сложный пароль по фамилии, имени, отчеству и году рождения. Алгоритм кодирования придумать самостоятельно
11. Разработать приложение «Тест », состоящий из 10 вопросов и выставяющий оценку
12. Разработать приложение «Бегающий текст», которое передаёт текстовое сообщение, введённое в поле EDIT между компонентами формы по нажатию на клавишу Enter, пока не вернётся в поле EDIT.
13. Разработать приложение «Поймай кнопку...». На форме разместить кнопку (можно кнопку заменить рисунком) и при подведении указателя мыши к кнопке, она «убегала» или «пряталась»
14. Составить проект для нахождения корней квадратного уравнения по трем его коэффициентам, вводимым с клавиатуры. Предусмотреть реакции на некорректный ввод символов в полях редактирования (например, поставить защиту от ввода букв или второй десятичной точки). При нажатии Enter переносить курсор в следующее поле редактирования (а на завершении ввода выполнять вычисления). Предусмотреть кнопки "расчет", "новые данные" и "выход". Всем введенным компонентам задать ярлычки с оперативной подсказкой (Hints). При оформлении компонент использовать по возможности различные цвета и шрифты.
15. Составить проект для визуализации выбираемого стиля, размера и цвета шрифта. Сам шрифт как набор всех латинских и русских букв (как прописных, так и строчных) отображать на метке. Каждую характеристику шрифта выбирать из набора минимум четырех радиокнопок.
16. Составить проект для анализа введенной в строке Edit информации: текстовая, числовая, прочая. В качестве индикаторов использовать набор из трех компонент CheckBox. Ввести четвертый индикатор для анализа очередного набираемого символа.
17. Предусмотреть кнопки "новые данные" и "выход". Всем введенным компонентам задать ярлычки с оперативной подсказкой (Hints). При оформлении компонент использовать по возможности различные цвета и шрифты.

4.2 Промежуточная аттестация МДК 01.01

Тест 1.

1. Операционная система:
 - a) **система программ, которая обеспечивает совместную работу всех устройств компьютера по обработке информации**
 - b) система математических операций для решения отдельных задач
 - c) система планового ремонта и технического обслуживания компьютерной техники
2. Программное обеспечение (ПО) – это:
 - a) **совокупность программ, позволяющих организовать решение задач на компьютере**
 - b) возможность обновления программ за счет бюджетных средств
 - c) список имеющихся в кабинете программ, заверен администрацией школы
3. Загрузка операционной системы – это:
 - a) запуск специальной программы, содержащей математические операции над числами
 - b) **загрузка комплекса программ, которые управляют работой компьютера и организуют диалог пользователя с компьютером**
 - c) вложение дискеты в дисковод
4. Система программирования – это:
 - a) комплекс любимых программ программиста
 - b) **комплекс программ, облегчающий работу программиста**

- с) комплекс программ, обучающих начальным шагам программиста
- 5. Прикладное программное обеспечение – это:
 - а) справочное приложение к программам
 - б) текстовый и графический редакторы, обучающие и тестирующие программы, игры**
 - с) набор игровых программ
- 6. Прикладное программное обеспечение:
 - а) программы для обеспечения работы других программ
 - б) программы для решения конкретных задач обработки информации**
 - с) программы, обеспечивающие качество работы печатающих устройств
- 7. Операционные системы:
 - а) DOS, Windows, Unix**
 - б) Word, Excel, PowerPoint
 - с) (состав отделения больницы): зав. отделением, 2 хирурга, 4 мед. Сестры
- 8. Системное программное обеспечение:
 - а) программы для организации совместной работы устройств компьютера как единой системы**
 - б) программы для организации удобной системы размещения программ на диске
 - с) набор программ для работы устройства системного блока компьютера
- 9. Сервисные (обслуживающие) программы:
 - а) программы сервисных организаций по бухгалтерскому учету
 - б) программы обслуживающих организаций по ведению делопроизводства
 - с) системные оболочки, утилиты, драйвера устройств, антивирусные и сетевые программы**
- 10. Системные оболочки – это:
 - а) специальная кассета для удобного размещения дискет с операционной системой
 - б) специальная программа, упрощающая диалог пользователь – компьютер, выполняет команды операционной системы**
 - с) система приемов и способов работы конкретной программы при загрузке программ и завершении работы
- 11. Использование одного имени для задания общих для класса действий, что означает способность объектов выбирать внутренний метод, исходя из типа данных, определяет свойство ООП
 - а) Полиморфизм
 - б) Управление событиями
 - с) Инкапсуляция**
 - д) Наследование
- 12. Понятие «инкапсуляция» относится к
 - а) Технологии модульного программирования
 - б) Технологии объектно – ориентированного программирования**
 - с) Технологии императивного программирования
 - д) Технологии модульного программирования
- 13. Свойство ООП, которое может быть смоделировано с помощью таксономической классификационной схемы (иерархии) называется
 - а) Инкапсуляция
 - б) Управление событиями
 - с) Полиморфизм
 - д) Наследование**
- 14. Понятие класса в ООП включает в себя
 - а) Поля и методы класса**
 - б) Процедуры и функции обработки
 - с) Поля и функции обработки
 - д) Поля и процедуры обработки
- 15. Назначение конструктора объекта

- a) **Только выделяет память под объект**
 - b) Выделяет память и задает начальное значение полям
 - c) Задает начальное значение полям
 - d) Выделяет память, задает начальное значение полям, выполняет любые проверки, заданные программистом
16. Как описывается конструктор объекта
- a) procedurecreate;
 - b) constructorcreate;
 - c) functioncreate;
 - d) **functionconstructor;**
17. Как описывается деструктор объекта
- a) procedurefree;
 - b) **destructorfree;**
 - c) free;
 - d) functionfree;
18. Понятия объекта в ООП - это
- a) представитель класса
 - b) **конкретные данные, заданные в классе.**
 - c) компонент панели инструментов
 - d) встроенный объект Delphi
19. Моделями типа «черный ящик» являются
- a) Модели мышления
 - b) Модели, описывающие зависимость параметров состояния объекта от входных параметров
 - c) **Модели, описывающие входные и выходные параметры объекта без учета внутренней структуры объекта**
 - d) Модели «аварийного» ящика на самолетах
20. Моделями типа «белый ящик» являются
- a) Модели мышления
 - b) **Модели, описывающие зависимость параметров состояния объекта от входных параметров**
 - c) Модели, описывающие входные и выходные параметры объекта с учетом внутренней структуры объекта
 - d) Модели, описывающие выходные данные в программе
21. Программа «драйвер» служит для
- a) запуска программы на выполнение
 - b) имитации запуска программы на выполнение
 - c) **проверки правильности работы программы**
 - d) передачи параметров в процедуры и функции
22. Программа «заглушка» служит для
- a) запуска программы на выполнение
 - b) **имитации запуска другой программы на выполнение**
 - c) проверки правильности работы программы
 - d) имитации передачи параметров в другой модуль
23. Какие методы сборки программы существуют
- a) монолитная
 - b) **пошаговая**
 - c) **одновременная**
 - d) постепенная
24. Какой метод тестирования программы учитывает закон распределения входных данных
- a) детерминированное тестирование
 - b) функциональное тестирование
 - c) стохастическое тестирование

d) логическое тестирование

25. Программирование сверху вниз – это

- a) Процесс, при котором от начального предположения осуществляется движение по направлению к лучшим решениям
- b) Процесс пошагового разбиения алгоритма на все более мелкие части с целью получения таких элементов, для которых можно написать конкретные команды**
- c) Метод сведения трудной задачи к последовательности более простых
- d) Исследование древовидной модели пространства решений и ориентация на поиск оптимального решения

26. Загрузочный модуль программы – результат работы

- a) Грамматики
- b) Транслятора**
- c) Интерпретатора
- d) Редактора связей (компоновщика)

27. Интегрированная система программирования включает компонент для перевода исходного текста программы в машинный код, который называется

- a) строителем кода
- b) компилятор**
- c) переводчиком
- d) преобразователем

28. Результатом компиляции программы на языке высокого уровня является

- a) Командный файл
- b) Объектный файл**
- c) Исходный текст программы на языке высокого уровня
- d) Дисплейный файл

Тест. Объектно-ориентированное программирование

1. Delphi – это...

- 1) Язык программирования низкого уровня
- 2) Система объектно-ориентированного визуального программирования**
- 3) Программа обработки видео
- 4) Программа для перевода кода программы на язык машинных кодов.

2. RAD – это...

- 1) Программа записи видео
- 2) Средство модульного программирования
- 3) Среда быстрой разработки приложений**

3. Язык программирования Delphi

- 1) Object Pascal**
- 2) Java
- 3) C#

4. IDE – это...

- 1) История развития Delphi
- 2) Приложение, разрабатываемое программистом
- 3) Интегрированная среда разработки**

5. IDE не включает в себя: (несколько вариантов ответа)

- 1) Редактор кода
- 2) Высокопроизводительный компилятор в машинный код
- 3) Объектно-ориентированную модель компонент
- 4) Эффекты и переходы**

- 5) Визуальное построение приложений
- 6) Сопровождение ПП**
- 7) Средство для построения баз данных
- 6. Объектно-ориентированная программа - ...
 - 1) Программирование, основанное на объектах
 - 2) Совокупность объектов и способов их взаимодействия**
 - 3) Структура среды программирования

- 7. Установить соответствие

1) Объект	а) Атрибуты (основные характеристики), которые описывают особенности объекта (цвет, ширина, положение и т.д.)
2) Событие	б) Совокупность данных (компонентов) и методов работы с ними
3) Свойство	с) Отклик на внешнее воздействие
1 – б, 2 – с, 3 – а	

- 8. Дополнить предложение. Основным окном разрабатываемого приложения является ...
 - 1) Код
 - 2) Форма**
 - 3) Библиотека
 - 4) Объект
- 9. Дополнить предложение. Коды для стандартных окон диалога и кнопок в системе Delphi получены от ...
 - 1) Компилятора
 - 2) C++
 - 3) Windows**
- 10. Названия процедур на определенное событие:
 - 1) Компоненты
 - 2) Обработчики событий**
 - 3) Редактор кода
- 11. Назначение главного меню:
 - 1) Осуществление функций управления при разработке программ**
 - 2) Сопровождение программных продуктов
 - 3) Автоматизирует процесс тестирования программ
- 12. Назначение панели инструментов:
 - 1) Обработка событий
 - 2) Меню команд быстрого доступа к командам, содержащее набор кнопок, функции которых эквивалентны наиболее часто употребляемым командам Главного меню**
 - 3) Построение баз данных
- 13. Назначение палитры компонентов:
 - 1) содержит пиктограммы, которые представляют компоненты VCL , которые можно включить в приложение**
 - 2) Подделка подписей
 - 3) Управление Paint

14. Дополнить предложение. Главное окно управляет окнами ... *(несколько вариантов ответа)*
- 1) **Инспектор объектов**
 - 2) Язык ассемблера
 - 3) **Редактора кода**
 - 4) **Проектировщика форм**
 - 5) Трансляции программы
15. Заготовка разрабатываемого приложения - ...
- 1) Компилятор
 - 2) Свойство
 - 3) **Окно проектировщика форм**
 - 4) Главное окно
16. Действия которые нельзя выполнить с помощью проектировщика форм
- 1) Добавить компоненты в форму
 - 2) **Сменить язык программирования**
 - 3) Модифицировать форму и её компоненты
 - 4) Связать обработчики событий компонента с программой на Object Pascal, содержащейся в редакторе кода
 - 5) **Документирование программы**
17. Дополнить предложение. Инспектор объектов позволяет ... *(несколько вариантов ответа)*
- 1) **Устанавливать свойства объектов**
 - 2) Изменять структуру программного кода
 - 3) Изменять наследование классов объектов
 - 4) **Назначать методы обработки событий**
18. Дополнить предложение. Инспектор объектов состоит из ... *(несколько вариантов ответа)*
- 1) Unit – программного модуля
 - 2) **Properties – списка свойств,**
 - 3) **Events – списка событий.**
 - 4) Begin...end.
19. Окно содержащее текст программы на языке Object Pascal, связанный с каждой формой в приложении:
- 1) Окно компилятора
 - 2) **Окно редактора кода**
 - 3) Окно проектировщика
 - 4) Главное окно
20. Задание метки текста из редактора ввода:
- 1) Memo1.Caption := Edit1.Text;
 - 2) **Label1. Caption := Edit1.Text;**
 - 3) Form1. Caption := Edit1.Text;
21. Обнуление строки ввода:
- 1) **Edit1.Text :='';**
 - 2) Edit1.Text := TMemo;
 - 3) Edit1.Text :='Закругляемся'
22. Передача фокуса ввода на редактор ввода

- 1) Label1.SetFocus;
 - 2) Edit1.Add;
 - 3) **Edit1.SetFocus;**
23. Открыть файл проекта Project1 можно, нажав:
- 1) CTRL+F4
 - 2) CTRL+ALT+F10
 - 3) **CTRL+F12**
 - 4) ALT+F12
24. Для обозначения комментария не используются:
- 1) (*комментарий*)
 - 2) **\комментарий/**
 - 3) {комментарий}
 - 4) //комментарий
25. Зарезервированное слово, объявляющее блок подключаемых к проекту модулей:
- 1) Program
 - 2) Begin..end
 - 3) **Uses**
 - 4) Forms
26. Дополнить предложение. Объект Application определяет ...
- 1) **Свойства и методы программы-приложения для Windows**
 - 2) Отображение размещения формы
 - 3) Перемещение по программному коду
27. Метод Application.Run:
- 1) Создает окно формы, регистрирует входящие в него компоненты
 - 2) **Подключает автоматический цикл обработки сообщений Windows к приложению**
 - 3) Выполняет подпрограммы раздела инициализации модулей приложения в случае их наличия
28. Контейнер, содержащий все элементы программы:
- 1) Unit1
 - 2) **Form1**
 - 3) Edit1
 - 4) Memo1
29. Дополнить предложение. Свойство **Position** определяет ...
- 1) **Размещение и размер формы**
 - 2) Подключение модулей
 - 3) Цвет шрифта
30. **poDesigned:**
- 1) Форма выводится в центр экрана
 - 2) Windows автоматически определяет начальную позицию и размеры формы
 - 3) **Форма отображается в той позиции и с теми размерами, которые были установлены при её конструировании**
31. Определяет способ выравнивания компонента внутри контейнера:
- 1) Свойство Position
 - 2) **Свойство Align**
 - 3) Свойство Memo

4) Свойство ScrollBars

32. **alClient:**

- 1) **Компонент занимает всю поверхность контейнера**
- 2) Выравнивание не используется
- 3) Компонент помещается в нижнюю часть контейнера

33. **alCustom:**

- 1) Компонент помещается в верхнюю часть контейнера
- 2) Компонент помещается в нижнюю часть контейнера
- 3) **Размеры и положение компонента устанавливаются разработчиком**

34. Объект **Memo1** - ...

- 1) Однострочный редактор
- 2) **Многострочный редактор, содержащий несколько строк текста**
- 3) Текстовая надпись

35. Свойство **ScrollBars:**

- 1) **Задаёт в поле редактирования полосы прокрутки**
- 2) Определяет способ выравнивания компонента внутри контейнера
- 3) Задаёт имя объекта

36. Функциональная кнопка с изображением устанавливается из:

- 1) Палитра компонентов -> Standart -> Button
- 2) Палитра компонентов -> Standart -> Label
- 3) **Палитра компонентов -> Additional -> BitBtn**

37. Дополнить предложение. За стиль оформления внешней и внутренней рамок отвечают свойства ... (несколько вариантов ответа)

- 1) Align
- 2) **BevelOuter**
- 3) **BevelInner**
- 4) ssBoth
- 5) Caption

38. Установить соответствие

1) bvLowered	a) Компонент помещается в нижнюю часть контейнера
2) poScreenCenter	b) Есть обе полосы прокрутки
3) bvNone	c) Рамка вдавлена
4) alBottom	d) Форма выводится в центре экрана, её высота и ширина не изменяются
5) ssBoth	e) Рамка отсутствует
1 – c, 2- d, 3 – e, 4 – a, 5 -b	

39. VCL – это...

- 1) Приложение Delphi
- 2) Алгоритмизация решения задачи
- 3) **Библиотека визуальных компонентов**
- 4) Характеристики объектов

40. Дополнить предложение. Компоненты, которые видны на форме, как во время создания приложения, так и во время работы приложения называются ...

- 1) **Визуальными**
- 2) Реальными

3) Невизуальными

4) Основными

41. К невидуальным компонентам относят:

1) Кнопки, метки, списки блоков

2) Таймеры, компоненты для работы с базами данных, списки изображений

3) Библиотеки, звуки, коды

42. Установить соответствие:

1) Standart	a) Обеспечивает доступ к 32-битным элементам Windows
2) Additional	b) Компоненты, реализующие интерфейс с пользователем и процесс управления данными для БД
3) Win32	c) Включает стандартные компоненты, обеспечивающие некоторые функции интерфейса пользователя
4) System	d) Специализированные компоненты, организующие доступ к БД
5) DataAccess	e) Набор компонентов для доступа к системным ресурсам (OLE, DDE)
6) DataControl	f) Дополнительные интерфейсные компоненты для красочного оформления приложения
1) – c, 2) – f, 3) – a, 4) – e, 5) – d, 6) – b;	

43. Дополнить предложение. Имя компонента состоит из... (несколько вариантов ответа)

1) Названия компонента

2) Заготовки компонента

3) Порядкового номера компонента

4) Произвольного значения ItemIndex

44. Дополнить предложение. Свойства Height и Width показывают...

1) Какое действие должен выполнить программист

2) Размеры компонента

3) Исполняемый блок программы

45. Дополнить предложение. Проект Delphi представляет собой...

1) Набор программных единиц – модулей

2) Основной интерфейсный элемент в Delphi

3) Атрибуты (основные характеристики), которые описывают особенности объекта

46. Модуль – это...

1) Средство для создания, изменения исходных файлов, которые содержат написанную на языке программирования программу

2) Автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно некоторые исполняемые операторы иницирующей части

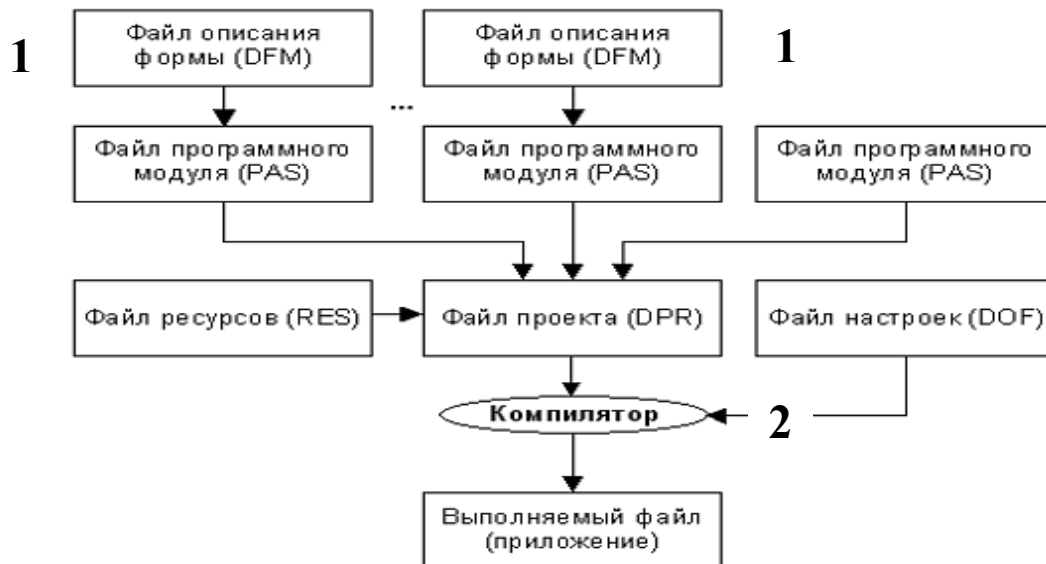
3) Набор инструментов, которые используются для преобразования символов

47. Установить соответствие

1) Файл проекта	a) (.pas) соответствующий файл модуль для хранения кода
2) Файл модуля	b) (.res) содержит пиктограмму и прочие ресурсы
3) Файл формы	c) (.dpr) текстовый файл используется для хранения информации о формах и модулях, содержит операторы инициализации и запуска программы на выполнение

4) Файл опций проекта	d) (.dfm) двоичный файл, который создается для хранения информации о ваших формах и фреймах
5) Файл ресурсов	e) (.cfg) хранит установки проекта
6) Файл конфигурации проекта	f) (.dof) хранит установки опций проекта
1) – c, 2) – a, 3) – d, 4) – f, 5) – b, 6) – e	

48. Заполнить схему:



1) 1 – файл формы (DFM); 2 – файл проекта (DPR)

2) 1 – файл проекта (DPR); 2 – файл формы (DFM)

3) 1 – файл формы (DPR); 2 – файл проекта (DFM)

4) 1 – файл проекта (DFM); 2 – файл формы (DPR)

49. Дополнить предложение. Набор всех файлов, необходимых для создания приложения, называется...

- 1) Модулем
- 2) **Проектом**
- 3) Задачей
- 4) Формой

50. Файлы описания форм – это...

- 1) Отклики на внешнее воздействие
- 2) Файлы подключающие все используемые программные модули
- 3) **Текстовые файлы с расширением DFM, в котором сохраняются значения свойств формы и ее компонентов**

51. Содержимое dfm-файла вызывается:

- 1) **Контекстное меню формы -> View as Text**
- 2) Контекстное редактора кода -> View as Form
- 3) File -> Save as...

52. Дополнить предложение. При сборке приложения описание из DFM-файла помещается в...

- 1) Файл модуля
- 2) **Область ресурсов**
- 3) Область установок проекта

53. Программный модуль (Unit) содержит:

- 1) **Все относящиеся к форме объявления и методы обработки событий, написанные на языке Object Pascal**
- 2) Заготовку, макет одного из окон разрабатываемого приложения
- 3) Пиктограммы, которые представляют компоненты VCL , которые можно включить в приложение

54. Дополнить предложение. Имя модуля должно совпадать с...

- 1) Приложением Windows
- 2) Спецификацией задачи
- 3) Объектом в Delphi
- 4) **Именем файла при сохранении File->Save as...**

55. Интерфейсная секция модуля (interface) содержит: (несколько вариантов ответа)

- 1) **Список стандартных модулей библиотеки VCL**
- 2) **Раздел описания типов**
- 3) Компилятор
- 4) **Раздел описания переменных (объявление собственно объекта формы)**
- 5) Инспектор объектов и проектировщик форм

56. Дополнить предложение. Класс формы объявляется в разделе...

- 1) **Type**
- 2) Uses
- 3) Project
- 4) Var

57. Дополнить предложение. В описании класса форм помещенные на форму компоненты представлены...

- 1) Границами формы
- 2) **Полями формы**
- 3) Индексами формы

58. Published – это...

- 1) **Атрибут видимости, позволяющий работать с компонентами и методами обработки событий на визуальном уровне**
- 2) Директива препроцессора
- 3) Текст программы на языке Object Pascal

59. Пустые секции, в которых можно размещать любые вспомогательные поля, методы и свойства: (несколько вариантов ответа)

- 1) **Private**
- 2) Uses
- 3) Var
- 4) **Public**
- 5) Program

60. Объявление объекта формы:

- 1) unit Unit1;
- 2) type TForm1 = class(TForm)
- 3) **var Form1: TForm1;**

61. implementation – это...

- 1) Тип данных Delphi
- 2) Племя в Южной Африке

3) Раздел реализации объявлений интерфейса

62. Подключение файла описания формы:

- 1) **{SR *.dfm}**
- 2) {SR *.pas}
- 3) {R *.dpr}

63. Дополнить предложение. После подключения файла описания формы следует...

- 1) Раздел описания типов
- 2) Атрибут видимости
- 3) Описание размеров используемых компонентов
- 4) **Реализация методов обработки событий**

64. Установить метод обработки события Button1Click: *(несколько вариантов ответа)*

- 1) **Двойной щелчок по компоненту Button1 на форме**
- 2) **Выделить Button1 ->Events -> OnClick**
- 3) Выделить BitBtn1 ->Events -> OnClick

65. Программный файл проекта – это...

- 1) Программист
- 2) **Главная часть приложения, с которой начинается выполнение программы и которая обеспечивает инициализацию других модулей**
- 3) Пустая секция

66. К комментариям не относятся: *(несколько вариантов ответа)*

- 1) **Program**
- 2) {Form1}
- 3) **{SR *.res}**
- 4) //выполнение команды

67. Дополнить предложение. Сочетание end со следующей за ней точкой называется...

- 1) Терминалом программной единицы
- 2) **Терминатором программной единицы**
- 3) Андроидом программной единицы

68. Содержимое DPR-файла вызывается: *(несколько вариантов ответа)*

- 1) Ctrl+alt
- 2) File -> New->Application
- 3) **Project -> View Source**
- 4) **Ctrl+F12**

69. Дополнить предложение. Зарезервированное слово **uses** сообщает компилятору о...

- 1) **Модулях, которые должны рассматриваться как необъемлемые части программы и которые содержатся в других файлах**
- 2) Начале работы приложения
- 3) Ошибках, которые могут возникнуть при компилировании

70. Модуль Forms – это...

- 1) Дополнительный модуль, содержащий неиспользуемые компоненты
- 2) Модуль для построения более красочного приложения
- 3) **Обязательный для всех программ модуль, который содержит определение объекта Application**

71. Запись {SR *.res} означает:

- 1) Комментарий, указывающий программисту где искать ресурсы

2) Директиву, подключающую к результирующему выполняемому файлу ресурсы

3) Директиву, подключающую к результирующему выполняемому файлу форму

72. Установить соответствие:

1) Application.Initialize	a) загружает и инициализирует форму
2) Application.CreateForm	b) активизирует форму и начинает выполнение приложения
3) Application.Run	c) подготавливает приложение к работе
1 – c; 2 – a; 3 – b	

73. Включение множественного выбора элементов компонента ListBox1

1) ListBox2.MultiSelect:=false;

2) **ListBox1.MultiSelect:=true;**

3) ListBox1.MultiSelect:=false;

4) ListBox1.Checked:=true;

74. Запись ListBox1.Selected[3]:=true означает что:

1) Необходимо установить три дополнительных элемента

2) Необходимо удалить три лишних элемента

3) **Выделен четвертый элемент в компоненте**

4) Выделен третий элемент в компоненте

75. Простейшая функция вывода на экран текстовых сообщений:

1) ListBox1.Items.Add();

2) **ShowMessage();**

3) Case...of;

4) CheckBox1.Checked:=true;

76. Полная очистка списка:

1) **ListBox1.Clear;**

2) ListBox1.Items.Add();

3) RadioGroup1.ItemIndex:=0;

77. Снятие «флажка» компонента CheckBox1

1) ListBox1.Checked:=true;

2) **CheckBox1.Checked:=false;**

3) CheckBox1.Checked:=true;

4) CheckBox1.Caption := 'Снять флажок';

78. Свойство Align=AllBottom:

1) Компонент перемещается в центр формы

2) Компонент перемещается в верхнюю часть контейнера

3) **Компонент перемещается в нижнюю часть контейнера и его ширина становится равной ширине контейнера**

79. Объект ComboBox - ...

1) Многострочный редактор

2) **Комбинированный список**

3) Функциональная кнопка

80. Установка в ComboBox1 первой строки массива Item:

1) **ItemIndex=0**

2) ItemIndex=-1

3) ItemIndex=1

- 4) ItemIndex=356
81. Объект RadioGroup - ...
- 1) Команда «Авторадио»
 - 2) Панель группы радиокнопок**
 - 3) Метка
82. Дополнить предложение. Компонент-флажок, который используется для обозначения включения или отключения какой либо опции, называется...
- 1) RadioGroup
 - 2) Функциональная кнопка
 - 3) CheckBox**
 - 4) ListBox

1.3 Промежуточная аттестация по МДК 01.02

Тест

1. Какие программы можно отнести к системному программному обеспечению:
Варианты ответа:
1. **операционные системы;**
 2. прикладные программы;
 3. игровые программы.
2. Какие программы можно отнести к системному ПО:
Варианты ответа:
- 1) **драйверы;**
 - 2) текстовые редакторы;
 - 3) электронные таблицы;
 - 4) графические редакторы.
3. Специфические особенности ПО как продукта:
- 1) *продажа по ценам ниже себестоимости (лицензирование);*
 - 2) низкие материальные затраты при создании программ;**
 - 3) возможность создание программ небольшие коллективом или даже одним человеком;**
 - 4) разнообразие решаемых задач с помощью программных средств.**
4. Какие программы можно отнести к системному ПО:
Варианты ответа:
- 1) программа расчета заработной платы;
 - 2) электронные таблицы;
 - 3) СУБД (системы управления базами данных).**
5. Какие программы нельзя отнести к системному ПО:
Варианты ответа:
- 1) **игровые программы;**
 - 2) компиляторы языков программирования;
 - 3) операционные системы;
 - 4) системы управления базами данных.
6. Какие программы можно отнести к прикладному программному обеспечению:
Варианты ответа:
- 1) электронные таблицы;**
 - 2) таблицы решений;
 - 3) СУБД (системы управления базами данных).
7. Какие программы можно отнести к прикладному ПО:
Варианты ответа:
- 1) программа расчета заработной платы;**
 - 2) диспетчер программ;
 - 3) программа «Проводник» (Explorer).

8. Какие программы нельзя отнести к прикладному ПО:

Варианты ответа:

- 1) **компиляторы и (или) интерпретаторы;**
- 2) текстовые и (или) графические редакторы;
- 3) электронные таблицы.

9. Можно ли отнести операционную систему к программному обеспечению:

Варианты ответа:

- 1) **да;**
- 2) нет.

10. Можно ли отнести операционную систему к прикладному программному обеспечению:

Варианты ответа:

- 1) **да;**
- 2) *нет.*

11. Специфические особенности ПО как продукта:

Варианты ответа:

- 1) **низкие затраты при дублировании;**
- 2) универсальность;
- 3) простота эксплуатации;
- 4) наличие поддержки (сопровождения) со стороны разработчика.

12. Какие программы можно отнести к системному ПО:

Варианты ответа:

- 1) **утилиты;**
- 2) экономические программы;
- 3) статистические программы;
- 4) мультимедийные программы.

13. Этап, занимающий наибольшее время, в жизненном цикле программы:

Варианты ответа:

- 1) **сопровождение;**
- 2) проектирование;
- 3) тестирование;
- 4) программирование;
- 5) формулировка требований.

14. Этап, занимающий наибольшее время, при разработке программы:

Варианты ответа:

- 1) **тестирование;**
- 2) сопровождение;
- 3) проектирование;
- 4) программирование;
- 5) формулировка требований.

15. Первый этап в жизненном цикле программы:

Варианты ответа:

- 1) **формулирование требований;**
- 2) анализ требований;
- 3) проектирование;
- 4) автономное тестирование;
- 5) комплексное тестирование.

16. Один из необязательных этапов жизненного цикла программы:

Варианты ответа:

- 1) **оптимизация;**
- 2) проектирование;

- 3) тестирование;
- 4) программирование;
- 5) анализ требований.

17. Самый большой этап в жизненном цикле программы:

Варианты ответа:

- 1) **эксплуатация;**
- 2) изучение предметной области;
- 3) программирование;
- 4) тестирование;
- 5) корректировка ошибок.

18. Какой этап выполняется раньше:

Варианты ответа:

- 1) **отладка;**
- 2) тестирование.

19. Какой этап выполняется раньше:

Варианты ответа:

- 1) отладка;
- 2) оптимизация;
- 3) **программирование;**
- 4) тестирование.

20. Что выполняется раньше:

Варианты ответа:

- 1) **компиляция;**
- 2) отладка;
- 3) компоновка;
- 4) тестирование.

21. Что выполняется раньше:

Варианты ответа:

- 1) **проектирование;**
- 2) программирование;
- 3) отладка;
- 4) тестирование.

22. В стадии разработки программы не входит:

Варианты ответа:

- 1) **автоматизация программирования;**
- 2) постановка задачи;
- 3) составление спецификаций;
- 4) эскизный проект;
- 5) тестирование.

23. Самый важный критерий качества программы:

Варианты ответа:

- 1) **работоспособность;**
- 2) надежность;
- 3) эффективность;
- 4) быстродействие;
- 5) простота эксплуатации.

24. Способы оценки качества:

Варианты ответа:

- 1) **сравнение с аналогами;**
- 2) наличие документации;
- 3) оптимизация программы;
- 4) структурирование алгоритма.

25. Существует ли связь между эффективностью и оптимизацией программы:

Варианты ответа:

- 1) **да;**
- 2) **нет.**

26. Наиболее важный критерий качества:

Варианты ответа:

- 1) **надежность;**
- 2) **быстродействие;**
- 3) **удобство в эксплуатации;**
- 4) **удобный интерфейс;**
- 5) **эффективность.**

27. Способы оценки надежности:

Варианты ответа:

- 1) **тестирование;**
- 2) **сравнение с аналогами;**
- 3) **трассировка;**
- 4) **оптимизация.**

28. Повышает ли качество программ оптимизация:

Варианты ответа:

- 1) **да;**
- 2) **нет.**

29. Существует ли связь между надежностью и быстродействием:

Варианты ответа:

- 1) **нет;**
- 2) **да.**

30. В каких единицах можно измерить надежность:

Варианты ответа:

- 1) **отказов/час;**
- 2) **км/час;**
- 3) **Кбайт/сек;**
- 4) **операций/сек.**

31. В каких единицах можно измерить быстродействие:

Варианты ответа:

- 1) **отказов/час;**
- 2) **км/час;**
- 3) **Кбайт/сек;**
- 4) **операций/сек.**

32. Что относится к этапу программирования:

Варианты ответа:

- 1) **написание кода программы;**
- 2) **В) разработка интерфейса;**
- 3) **С) работоспособность;**
- 4) **анализ требований.**

33. Последовательность этапов программирования:

Варианты ответа:

- 1) **компилирование, компоновка, отладка;**
- 2) **В) компоновка, отладка, компилирование;**
- 3) **отладка, компилирование, компоновка;**
- 4) **компилирование, отладка, компоновка.**

34) Инструментальные средства программирования:

Варианты ответа:

- 1) **компиляторы, интерпретаторы;**
- 2) **СУБД (системы управления базами данных);**
- 3) **BIOS (базовая система ввода-вывода);**

4) ОС (операционные системы).

35. На языке программирования составляется:

Варианты ответа:

- 1) **исходный код;**
- 2) исполняемый код;
- 3) объектный код;
- 4) алгоритм.

36. Правила, которым должна следовать программа это:

Варианты ответа:

- 1) **алгоритм;**
- 2) структура;
- 3) спецификация;
- 4) состав информации.

37. Можно ли внутри цикла поместить еще один цикл:

Варианты ответа:

- 1) **да;**
- 2) нет.

38. Можно ли внутри условного оператора поместить еще одно условие:

Варианты ответа:

- 1) **да;**
- 2) нет.

39. Можно ли одно большое (длинное) выражение разбить на несколько выражений:

Варианты ответа:

- 1) **да;**
- 2) нет.

40. Если имеется стандартная функция, нужно ли писать собственную:

Варианты ответа:

- 1) **нет;**
- 2) да.

41. Доступ, при котором записи файла читаются в физической последовательности, называется:

Варианты ответа:

- 1) **последовательным;**
- 2) прямым;
- 3) простым;
- 4) основным.

42. Доступ, при котором записи файла обрабатываются в произвольной последовательности, называется:

Варианты ответа:

- 1) **прямым;**
- 2) последовательным;
- 3) простым;
- 4) основным.

43. Методы программирования (укажите НЕ верный ответ):

Варианты ответа:

- 1) **логическое;**
- 2) структурное;
- 3) модульное.

44. Что выполняется раньше:

Варианты ответа:

- 1) **разработка алгоритма;**

- 2) выбор языка программирования;
- 3) написание исходного кода;
- 4) компиляция.

45. Можно ли переменным присваивать произвольные идентификаторы:

Варианты ответа:

- 1) **да;**
- 2) нет.

46. Найдите НЕ правильное условие для создания имен:

Варианты ответа:

- 1) **имена могут содержать пробелы;**
- 2) длинное имя можно сократить;
- 3) из имени лучше выбрасывать гласные;
- 4) можно использовать большие буквы.

47. Какие символы не допускаются в именах переменных:

Варианты ответа:

- 1) **пробелы;**
- 2) цифры;
- 3) подчеркивание

48. Можно ли использовать имена, которые уже были использованы в другой программе (модуле):

Варианты ответа:

- 1) **да;**
- 2) нет.

49. Можно ли ставить знак подчеркивания в начале имени:

Варианты ответа:

- 1) **да, но не рекомендуется;**
- 2) да, без ограничений;
- 3) нет.

50. Как называется способ составления имен переменных, когда в начале имени сообщается тип переменной:

Варианты ответа:

- 1) прямым указанием;
- 2) **венгерской нотацией;**
- 3) структурным программированием;
- 4) поляризацией.

51. Можно ли писать комментарии в отдельной строке:

Варианты ответа:

- 1) **да;**
- 2) нет.

52. Транслируются ли комментарии:

Варианты ответа:

- 1) да;
- 2) **нет.**

53. Наличие комментариев позволяет:

Варианты ответа:

- 1) **быстрее найти ошибки в программе;**
- 2) быстрее писать программы;
- 3) быстрее выполнять программы.

54. Наличие комментариев позволяет:

Варианты ответа:

- 1) **легче разобраться в программе;**
- 2) применять сложные структуры;
- 3) увеличить быстродействие.

55. Наличие комментариев позволяет:

Варианты ответа:

- 1) **улучшить читабельность программы;**
- 2) улучшить эксплуатацию программы;
- 3) повысить надежность программы.

56. Что определяет выбор языка программирования:

Варианты ответа:

- 1) **область приложения;**
- 2) знание языка;
- 3) наличие дополнительных библиотек.

57. Возможно ли комбинирование языков программирования в рамках одной задачи:

Варианты ответа:

- 1) **да;**
- 2) нет.

58. Допустимо ли комбинирование языков программирования в рамках одной задачи :

Варианты ответа:

- 1) **да;**
- 2) нет.

59. Для каких задач характерно использование большого количества исходных данных, выполнение операций поиска, группировки:

Варианты ответа:

- 1) **для экономических задач;**
- 2) для системных задач;
- 3) для инженерных задач.

60. Для каких задач характерен большой объем вычислений, использование сложного математического аппарата:

Варианты ответа:

- 1) **для инженерных задач;**
- 2) для системных задач;
- 3) для экономических задач.

61. На каком этапе производится выбор языка программирования:

Варианты ответа:

- 1) **проектирование;**
- 2) программирование;
- 3) отладка;
- 4) тестирование.

62. Можно ли использовать комбинацию языков программирования в рамках одного проекта:

Варианты ответа:

- 1) **да;**
- 2) нет.

63. Для решения экономических задач характерно применение:

Варианты ответа:

- 1) **СУБД (систем управления базами данных);**
- 2) языков высокого уровня;
- 3) языков низкого уровня;
- 4) применение сложных математических расчетов.

64. Для решения инженерных задач характерно применение:

Варианты ответа:

- 1) **САПР (систем автоматизированного проектирования);**
- 2) СУБД (систем управления базами данных);
- 3) ОС (операционных систем).

65. Причины синтаксических ошибок:

Варианты ответа:

- 1) **плохое знание языка программирования;**
- 2) ошибки в исходных данных;
- 3) ошибки, допущенные на более ранних этапах;
- 4) неправильное применение процедуры тестирования.

66. Когда можно обнаружить синтаксические ошибки:

Варианты ответа:

- 1) **при компиляции;**
- 2) при отладке;
- 3) при тестировании;
- 4) на этапе проектирования;
- 5) при эксплуатации.

67. Ошибки компоновки заключаются в том, что:

Варианты ответа:

- 1) **указано внешнее имя, но не объявлено;**
- 2) неправильно использовано зарезервированное слово;
- 3) составлено неверное выражение;
- 4) указан неверный тип переменной.

68. Могут ли проявиться ошибки при изменении условий эксплуатации:

Варианты ответа:

- 1) **да;**
- 2) нет.

69. Могут ли проявиться ошибки при изменении в предметной области:

Варианты ответа:

- 1) **да;**
- 2) нет.

70. Возможно ли программирование с защитой от ошибок:

Варианты ответа:

- 1) **да;**
- 2) нет.

71. Есть ли недостатки программирования с защитой от ошибок:

Варианты ответа:

- 1) **да;**
- 2) нет.

72. Защитное программирование это:

Варианты ответа:

- 1) **встраивание в программу отладочных средств;**
- 2) создание задач защищенных от копирования;
- 3) разделение доступа в программе;
- 4) использование паролей;
- 5) оформление авторских прав на программу.

73. Вид ошибки с неправильным написанием служебных слов (операторов):

Варианты ответа:

- 1) **синтаксическая;**
- 2) семантическая;
- 3) логическая;
- 4) символьная.

74. Вид ошибки с неправильным использованием служебных слов (операторов):

Варианты ответа:

- 1) **семантическая;**
- 2) синтаксическая;
- 3) логическая;
- 4) символная.

75. Ошибки при написании программы бывают:

Варианты ответа:

- 1) **синтаксические;**
- 2) орфографические;
- 3) лексические;
- 4) фонетические;
- 5) морфологические.

76. Процедура поиска ошибки, когда известно, что она есть это:

Варианты ответа:

- 1) **отладка;**
- 2) тестирование;
- 3) компоновка;
- 4) транзакция;
- 5) трансляция.

77. Программа для просмотра значений переменных при выполнении программы:

Варианты ответа:

- 1) **отладчик;**
- 2) компилятор;
- 3) интерпретатор;
- 4) трассировка;
- 5) тестирование.

78. Отладка – это:

Варианты ответа:

- 1) **процедура поиска ошибок, когда известно, что ошибка есть;**
- 2) определение списка параметров;
- 3) правило вызова процедур (функций);
- 4) составление блок-схемы алгоритма.

79. Когда программист может проследить последовательность выполнения команд программы:

Варианты ответа:

- 1) **при трассировке;**
- 2) при тестировании;
- 3) при компиляции;
- 4) при выполнении программы;
- 5) при компоновке.

80. На каком этапе создания программы могут появиться синтаксические ошибки:

Варианты ответа:

- 1) **программирование;**
- 2) проектирование;
- 3) анализ требований;
- 4) тестирование.

81. Когда приступают к тестированию программы:

Варианты ответа:

- 1) **когда программа уже закончена;**
- 2) после постановки задачи;
- 3) на этапе программирования;
- 4) на этапе проектирования;
- 5) после составления спецификаций,

82. Тестирование бывает:

Варианты ответа:

- 1) **автономное;**
- 2) инструментальное;
- 3) визуальное;
- 4) алгоритмическое.

83. Тестирование бывает:

Варианты ответа:

- 1) **комплексное;**
- 2) инструментальное;
- 3) визуальное;
- 4) алгоритмическое.

84. Существует ли различие между отладкой и тестированием:

Варианты ответа:

- 1) **да;**
- 2) нет.

85. При комплексном тестировании проверяются:

Варианты ответа:

- 1) **согласованность работы отдельных частей программы;**
- 2) правильность работы отдельных частей программы;
- 3) быстродействие программы;
- 4) эффективность программы.

86. Чему нужно уделять больше времени, чтобы получить хорошую программу:

Варианты ответа:

- 1) **тестированию;**
- 2) программированию;
- 3) отладке;
- 4) проектированию.

87. Процесс исполнения программы с целью обнаружения ошибок:

Варианты ответа:

- 1) **тестирование;**
- 2) кодирование;
- 3) сопровождение;
- 4) проектирование.

88. Автономное тестирование это:

Варианты ответа:

- 1) **тестирование отдельных частей программы;**
- 2) инструментальное средство отладки;
- 3) составление блок-схем;
- 4) пошаговая проверка выполнения программы.

89. Трассировка это:

Варианты ответа:

- 1) **проверка пошагового выполнения программы;**
- 2) тестирование исходного кода;
- 3) отладка модуля;
- 4) составление блок-схемы алгоритма.

90. Локализация ошибки:

Варианты ответа:

- 1) **определение места возникновения ошибки;**
- 2) определение причин ошибки;
- 3) обнаружение причин ошибки;
- 4) исправление ошибки.

91. Назначение тестирования:

Варианты ответа:

- 1) **повышение надежности программы;**
- 2) обнаружение ошибок;
- 3) повышение эффективности программы;
- 4) улучшение эксплуатационных характеристик;
- 5) приведение программы к структурированному виду.

92. Назначение отладки:

Варианты ответа:

- 1) **поиск причин существующих ошибок;**
- 2) поиск возможных ошибок;
- 3) составление спецификаций;
- 4) разработка алгоритма.

93. Инструментальные средства отладки (НЕ правильный ответ):

Варианты ответа:

- 1) **компиляторы;**
- 2) отладчики;
- 3) трассировка.

94. Отладка программ это:

Варианты ответа:

- 1) **локализация и исправление ошибок;**
- 2) алгоритмизация программирования;
- 3) компиляция и компоновка.

95. Что выполняется раньше, автономная или комплексная отладка:

Варианты ответа:

- 1) **автономная;**
- 2) комплексная.

96. Что выполняется раньше, отладка или тестирование:

Варианты ответа:

- 1) **отладка;**
- 2) тестирование.

97. Что такое автоматизация программирования:

Варианты ответа:

- 1) **создание исходного кода программными средствами;**
- 2) создание исходного кода при помощи компилятора;
- 3) создание исходного кода без разработки алгоритма.

98. В чем сущность автоматизации программирования:

Варианты ответа:

- 1) **создание программы без написания ее текста;**
- 2) получение готовой программы без выполнения компоновки;
- 3) в отсутствии компиляции.

99. Возможна ли автоматизация программирования:

Варианты ответа:

- 1) **да;**
- 2) нет.

100. Создание исполняемого кода программы без написания исходного кода называется:

Варианты ответа:

- 1) составлением спецификаций;
- 2) отладкой;
- 3) проектированием.
- 4) **автоматизацией программирования;**

101. Одно из преимуществ автоматизации программирования:

Варианты ответа:

- 1) **наглядное программирование с визуальным контролем;**
- 2) получение стандартной программы;
- 3) создание программы с оптимальным кодом.

102. Один из методов автоматизации программирования:

Варианты ответа:

- 1) структурное программирование;
- 2) модульное программирование;
- 3) **визуальное программирование;**
- 4) объектно-ориентированное программирование.

103. Влияет ли автоматизация программирования на эффективность программы:

Варианты ответа:

- 1) **нет;**
- 2) да

104. Автоматизация программирования позволяет:

Варианты ответа:

- 1) повысить надежность программы;
- 2) **сократить время разработки программы;**
- 3) повысить быстродействие программы.

105. Позволяет ли автоматизация программирования всегда создавать эффективные программы:

Варианты ответа:

- 1) да.
- 2) **нет;**

106. Позволяет ли автоматизация программирования всегда создавать надежные программы:

Варианты ответа:

- a) **нет;**
- b) да.

107. Недостаток автоматизации программирования;

Варианты ответа:

- a) низкое быстродействие;
- b) **большой размер программы;**
- c) сложность программы.

108. Возможны ли ошибки при автоматизации программирования:

Варианты ответа:

- a) **да;**
- b) нет.

109. Нужно ли выполнять тестирование при автоматизации программирования:

Варианты ответа:

- a) **да;**
- b) нет.

110. Выполняется ли процедура компиляции при автоматизации программирования:

Варианты ответа:

- A) **да;**
- B) нет.

111. Что легко поддается автоматизации:

Варианты ответа:

- A) **интерфейс;**

- В) работа с файлами;
- С) сложные логические задачи;
- Д) алгоритмизация.

112. Относится ли визуальное программирование к средствам автоматизации:

Варианты ответа:

- А) да;**
- В) нет.

113. Нахождение наилучшего варианта из множества возможных:

Варианты ответа:

- А) оптимизация;**
- В) тестирование;
- С) автоматизация;
- Д) отладка;
- Е) сопровождение.

114. Что такое оптимизация программ:

Варианты ответа:

- А) улучшение работы существующей программы;**
- В) создание удобного интерфейса пользователя;
- С) разработка модульной конструкции программы;
- Д) применение методов объектно-ориентированного программирования.

115. Критерии оптимизации:

Варианты ответа:

- А) время выполнения или размер требуемой памяти;**
- В) размер программы и ее эффективность;
- С) независимость модулей;
- Д) качество программы, ее надежность.

116. Критерии оптимизации:

Варианты ответа:

- А) эффективность использования ресурсов;**
- В) структурирование алгоритма;
- С) структурирование программы.

117. Возможна ли оптимизация программ без участия программиста:

Варианты ответа:

- А) да;**
- В) нет.

118. Возможна ли оптимизация циклов:

Варианты ответа:

- А) да;**
- В) нет.

119. В чем заключается оптимизация условных выражений:

Варианты ответа:

- А) в изменении порядка следования элементов выражения;**
- В) в использовании простых логических выражений;
- С) в использовании сложных логических выражений;
- Д) в использовании операций AND, OR и NOT.

120. Оптимизация циклов заключается в:

Варианты ответа:

- А) уменьшении количества повторений тела цикла;**
- В) просмотре задачи с другой стороны;
- С) упрощение задачи за счет включения логических операций.

121. Оптимизация программы это:

Варианты ответа:

A) модификация;

B) отладка;

C) повышение сложности программы;

D) уменьшение сложности программы.

122. Критерии оптимизации программы:

Варианты ответа:

A) быстродействие или размер программы;

B) быстродействие и размер программы;

C) надежность или эффективность;

D) надежность и эффективность.

123. Результат оптимизации программы:

Варианты ответа:

A) эффективность;

B) надежность;

C) машино-независимость;

D) мобильность.

124. Сущность оптимизации циклов:

Варианты ответа:

A) сокращение количества повторений выполнения тела цикла;

B) сокращение тела цикла;

C) представление циклов в виде блок-схем;

D) трассировка циклов;

E) поиск ошибок в циклах.

125. В чем сущность модульного программирования:

Варианты ответа:

A) в разбиении программы на отдельные функционально независимые части;

B) в разбиении программы на отдельные равные части;

C) в разбиение программы на процедуры и функции;

126. Можно ли сочетать модульное и структурное программирование:

Варианты ответа:

A) да;

B) нет.

127. Может ли модуль включать несколько процедур или функций:

Варианты ответа:

A) да;

B) нет.

128. Рекомендуемые размеры модулей:

Варианты ответа:

A) небольшие;

B) большие;

C) равные;

D) фиксированной длины.

129. В чем заключается независимость модуля:

Варианты ответа:

A) в написании, отладке и тестировании независимо от остальных модулей;

B) в разработке и написании независимо от других модулей;

C) в независимости от работы основной программы.

130. При модульном программировании желательно, чтобы модуль имел:

Варианты ответа:

A) большой размер;

B) небольшой размер;

C) фиксированный размер;

D) любой размер.

131. Модульное программирование это:

Варианты ответа:

A) разбиение программы на отдельные части;

B) структурирование;

C) использование стандартных процедур и функций.

132. Можно ли использовать оператор GO TO в модульных программах:

Варианты ответа:

A) можно;

B) нельзя.

133. Разрешается ли использование циклов при модульном программировании:

Варианты ответа:

A) да;

B) нет.

134. Разрешается ли использование условных операторов при модульном программировании:

Варианты ответа:

A) да;

B) нет.

135. Сократится ли размер программы, если ее написать в виде набора модулей:

Варианты ответа:

A) нет;

B) да.

136. Достоинство модульного программирования:

Варианты ответа:

A) создание программы по частям в произвольном порядке;

B) не требует компоновки;

C) всегда дает эффективные программы;

D) снижает количество ошибок.

137. Недостаток модульного программирования:

Варианты ответа:

A) увеличивает трудоемкость программирования;

B) усложняет процедуру комплексного тестирования;

C) снижает быстродействие программы;

D) не позволяет выполнять оптимизацию программы.

138. Достоинство модульного программирования:

Варианты ответа:

A) возможность приступить к тестированию до завершения написания всей программы;

B) не требует комплексного тестирования;

C) уменьшает размер программы;

D) повышает надежность программы.

139. Допустимо ли использование оператора GO TO при структурном программировании:

1. Варианты ответа:

A) нет;

B) да.

140. Можно ли сочетать структурное программирование с модульным:

Варианты ответа:

A) можно;

B) нельзя;

C) только в особых случаях.

141. Любую ли программу можно привести к структурированному виду:

Варианты ответа:

A) любую;

B) не все;

C) нельзя.

142. Можно ли использовать оператор GO TO в структурированных программах:

Варианты ответа:

A) можно;

B) нельзя;

C) только в особых случаях.

143. Возможно, ли преобразовать неструктурированную программу к структурному виду:

Варианты ответа:

A) да;

B) нет.

144. Возможно ли программирование без оператора GO TO:

Варианты ответа:

A) да;

B) нет.

145. При структурном программировании задача выполняется:

Варианты ответа:

A) поэтапным разбиением на более легкие задачи;

B) без участия программиста;

C) объединением отдельных модулей программы.

146. Разрешается ли использование оператора GO TO при структурном программировании:

Варианты ответа:

A) нет;

B) да;

C) иногда.

147. Разрешается ли использование циклов при структурном программировании:

Варианты ответа:

A) да;

B) нет.

148. Разрешается ли использование оператора IF при структурном программировании:

Варианты ответа:

A) да;

B) нет.

149. Программирование без GO TO применяется. при:

Варианты ответа:

A) структурном программировании;

B) модульном программировании;

C) объектно-ориентированном программировании;

D) все ответы верные.

150. Достоинство структурного программирования:

Варианты ответа:

A) можно приступить к комплексному тестированию на раннем этапе разработки;

B) можно приступить к автономному тестированию на раннем этапе разработки;

C) нет необходимости выполнять тестирование;

D) можно пренебречь отладкой.

151. Достоинство структурного программирования:

Варианты ответа:

A) облегчает работу над большими и сложными проектами;

B) повышает быстродействие программы;

С) снижает затраты на программирование.

152. Недостаток структурного программирования:

Варианты ответа:

- А) **увеличивает размер программы;**
- В) снижает эффективность;
- С) уменьшает количество ошибок;
- Д) не требует отладки.

153. Повышает ли читабельность программ структурное кодирование:

Варианты ответа:

- А) **да;**
- В) нет.

154. Разрешается ли использование циклов при объектно-ориентированном программировании:

Варианты ответа:

- А) **да;**
- В) нет.

2. 155. Разрешается ли использование оператора IF при объектно-ориентированном программировании:

3. Варианты ответа:

- А) **да;**
- В) нет.

156. Предусматривает ли объектно-ориентированное программирование использование стандартных процедур и функций:

- А) **да;**
- В) нет.

157. Можно ли сочетать объектно-ориентированное и структурное программирование

Варианты ответа:

- А) **можно;**
- В) нельзя.

158) Можно ли сочетать объектно-ориентированное и модульное программирование:

Варианты ответа:

- А) **можно;**
- В) нельзя.

159. Что такое объект, в объектно-ориентированном программировании:

Варианты ответа:

- А) **тип данных;**
- В) структура данных;
- С) событие;
- Д) обработка событий;
- Е) использование стандартных процедур.

160. Инкапсуляция это:

Варианты ответа:

- А) определение новых типов данных;
- В) определение новых структур данных;
- С) **объединение переменных, процедур и функций в одно целое;**
- Д) разделение переменных, процедур и функций;
- Е) применение стандартных процедур и функций.

161. Наследование это:

Варианты ответа:

- А) передача свойств экземплярам;

- В) передача свойств предкам;
- С) передача свойств потомкам;**
- Д) передача событий потомкам.

162. Полиморфизм это:

Варианты ответа:

- А) изменение поведения потомков, имеющих общих предков;**
- В) передача свойств по наследству;
- С) изменение поведения потомков на разные события;
- Д) изменение поведения экземпляров, имеющих общих предков;

163. Три "кита" объектно-ориентированного метода программирования:

Варианты ответа:

- А) предки, родители, потомки;
- В) полиморфизм, инкапсуляция, наследование;**
- С) свойства, события, методы;
- Д) визуальные, не визуальные компоненты и запросы.

164. Какое утверждение верно:

Варианты ответа:

- А) предки наследуют свойства родителей;
- В) родители наследуют свойства потомков;
- С) потомки не могут иметь общих предков;
- Д) потомки наследуют свойства родителей.**

165. Может ли дочерний элемент иметь двух родителей:

Варианты ответа:

- А) да;
- В) нет;**
- С) только для визуальных элементов;
- Д) если их свойства совпадают.

165. Могут ли два визуальных компонента иметь общего предка:

Варианты ответа:

- А) да;**
- В) нет;
- С) если их свойства совпадают;
- Д) если их методы совпадают.

167. Есть ли различие между объектом и экземпляром:

Варианты ответа:

- А) да;**
- В) нет;
- С) если у них общий предок.

168. Есть ли различие в поведении объекта и экземпляра того же типа:

Варианты ответа:

- А) да;
- В) если у них есть общий предок;
- С) нет;**
- Д) если у них нет общего предков.

169. Изменение свойств, приводит к изменению поведения экземпляра:

Варианты ответа:

- A) нет;
- B) только для визуальных;
- C) только НЕ для визуальных ;
- D) да .**

170. Можно ли свойствам присваивать значения:

Варианты ответа:

- A) да (всегда);
- B) не всегда;**
- C) нет.

171. Можно ли переопределять методы:

Варианты ответа:

- A) да;**
- B) нет.

172. Можно ли переопределять свойства:

Варианты ответа:

- A) да;
- B) нет.**

173. Могут ли два различных объекта реагировать на событие по-разному:

Варианты ответа:

- A) да;**
- B) нет.

174. Могут ли два экземпляра одного объекта реагировать на событие по-разному:

Варианты ответа:

- A) да;**
- B) нет.

175. Какой методикой проектирования пользуются при структурном программировании:

Варианты ответа:

- A) сверху вниз;**
- A) снизу-вверх.

176. Какой этап проектирования может быть исключен:

Варианты ответа:

- A) эскизный проект;**
- B) технический проект;
- C) рабочий проект.

177. Какие этапы проектирования можно объединять:

Варианты ответа:

- A) технический и рабочий;**
- B) эскизный и рабочий;
- C) технический и эскизный.

178. Модульное программирование применимо при:

Варианты ответа:

- A) проектировании сверху вниз;
- B) проектирование снизу-вверх;**

179. Процесс преобразования постановки задачи в план алгоритмического или вычислительного решения это:

Варианты ответа:

- A) проектирование;**

- В) анализ требований;
- С) программирование;
- Д) тестирование.

180. Составление спецификаций это:

Варианты ответа:

- А) формализация задачи;**
- В) эскизный проект;
- С) поиск алгоритма;
- Д) отладка.

181. Этап разработки программы, на котором дается характеристика области применения программы:

Варианты ответа:

- А) техническое задание;**
- В) эскизный проект;
- С) технический проект;
- Д) внедрение;
- Е) рабочий проект.

182. Укажите правильную последовательность создания программы:

Варианты ответа:

- А) формулирование задачи, анализ требований, проектирование, программирование;**
- В) анализ требований, проектирование, программирование, тестирование, отладка;
- С) анализ требований, программирование, проектирование, тестирование;
- Д) анализ требований, проектирование, программирование, модификация, трассировка;
- Е) формулирование задачи, анализ требований, программирование, проектирование, отладка.

183. Уточнение структуры входных и выходных данных, разработка алгоритмов, определение элементов интерфейса входят в:

Варианты ответа:

- А) технический проект;**
- В) рабочий проект;
- С) эскизный проект.

184. Несуществующий метод проектирования:

Варианты ответа:

- А) алгоритмическое;**
- В) нисходящее;
- С) восходящее.

185. Метод проектирования:

Варианты ответа:

- А) нисходящее;**
- В) алгоритмическое;
- С) логическое;
- Д) использование языков программирования;
- Е) составление блок-схем.

186. Нисходящее проектирование это:

Варианты ответа:

- А) последовательное уточнение (детализация);**
- В) составление блок-схем;
- С) разделение программы на отдельные участки (блоки);
- Д) трассировка.

187. Признаки нисходящего программирования:

Варианты ответа:

- А) последовательная детализация;**
- В) наличие оптимизации;
- С) наличие тестирования;
- Д) автоматизация программирования.

188. Какой методикой пользуются при структурном программировании:

Варианты ответа:

A) сверху вниз;

B) снизу-вверх.

189. Проектирование сверху вниз это:

Варианты ответа:

A) последовательное разбиение общих задач на более мелкие;

B) составление из отдельных модулей большой программы.

190. Проектирование снизу-вверх это:

Варианты ответа:

A) составление из отдельных модулей большой программы;

B) последовательное разбиение общих задач на более мелкие.

191. Модульное программирование применимо при:

Варианты ответа:

A) проектировании сверху вниз;

B) проектирование снизу-вверх;

C) и в том, и другом случае;

D) ни в коем случае.

192. Какой методикой проектирования пользуются при структурном программировании:

Варианты ответа:

A) сверху вниз;

B) снизу-вверх.

193. В чем заключается иерархический подход в решении задачи:

Варианты ответа:

A) в последовательном разбиении задачи на более мелкие составные части;

B) в выделении основных и второстепенных элементов;

C) в возможности параллельного выполнения отдельных частей задачи.

4. 194. Какой метод проектирования соответствует иерархическому подходу в решении задачи:

5. Варианты ответа:

A) нисходящее (сверху вниз);

B) восходящее (снизу-вверх).

195. В каких единицах измеряются затраты на проектирование:

Варианты ответа:

A) в человеко-днях;

B) в долларах;

C) в тенге;

D) в килобайтах.

196. Зависит ли трудоемкость разработки от сложности алгоритма:

Варианты ответа:

A) да;

B) нет.

197. Зависит ли трудоемкость разработки от количества программистов:

Варианты ответа:

A) да;

B) нет.

198. Зависит ли трудоемкость разработки от языка или системы программирования:

Варианты ответа:

A) да;

B) нет.

199. Зависит ли трудоемкость разработки от количества обрабатываемой информации:

Варианты ответа:

- A) да;
- B) нет.**

200. Зависит ли трудоемкость разработки от вида информации:

Варианты ответа:

- A) да;**
- B) нет.

201. Если вы приобрели программу законным путем, являетесь ли вы собственником программы:

Варианты ответа:

- A) нет;**
- B) да.

202. Если вы приобрели программы законным путем, имеете ли вы право вносить в нее изменения:

Варианты ответа:

- A) нет;**
- B) да

203. Если вы приобрели программы законным путем, имеете ли вы право продать ее:

Варианты ответа:

- A) да;**
- B) нет.

204. Кому принадлежит право собственности на ПО:

Варианты ответа:

- A) разработчику;**
- A) продавцу;
- B) покупателю.

205. Кому принадлежит авторское право на ПО:

Варианты ответа:

- 1) разработчику;**
- 2) продавцу;
- 3) покупателю.

206. Что охраняется законом:

Варианты ответа:

- 1) структура базы данных;
- 2) содержание базы данны**

4.4 Промежуточная аттестация по МДК 01.03

Тест на знание основ разработки приложений для ОС Android

1. Ядро какой операционной системы использовалось в качестве базы для ОС Android?
(Отметьте один правильный вариант ответа.)

Вариант 1 Linux

Вариант 2 OS/2

Вариант 3 Mac OS

Вариант 4 Windows

2. Какой движок баз данных используется в ОС Android?

(Отметьте один правильный вариант ответа.)

Вариант 1 DBM

Вариант 2 SQLite

Вариант 3 InnoDB

Вариант 4 MyISAM

3. Какой компонент архитектуры Android позволяет любому приложению использовать уже реализованные возможности других приложений, к которым разрешен доступ?

(Отметьте один правильный вариант ответа.)

Вариант 1 Application Framework

Вариант 2 Linux Kernel

Вариант 3 Applications

Вариант 4 Libraries & Android Runtime

4. Инструмент разработки, позволяющий адаптировать код C/C++ для работы на Android, это -

(Отметьте один правильный вариант ответа.)

Вариант 1 Google Android SDK (ADT Bundle)

Вариант 2 Marmalade SDK

Вариант 3 Intel* Software Manager

Вариант 4 Android NDK

5. Чем являются Eclipse и IDEA?

(Отметьте один правильный вариант ответа.)

Вариант 1 API (интерфейс прикладного программирования)

Вариант 2 IDE (интегрированная среда разработки)

Вариант 3 SDK (набор средств разработки)

Вариант 4 ADT (инструменты разработки под Android)

6. К проблемам разработки под ОС Android можно отнести:

(Отметьте один правильный вариант ответа.)

Вариант 1 отсутствие эффективных инструментов разработки

Вариант 2 ненадежную изоляцию ядра системы от выполняемых приложений

Вариант 3 все варианты ответа верны

Вариант 4 большое разнообразие устройств, невозможность проверки приложения на всех

7. К преимуществам среды разработки Intel XDK можно отнести:

(Отметьте один правильный вариант ответа.)

Вариант 1 легкость разработки кроссплатформенных приложений

Вариант 2 все варианты ответа верны

Вариант 3 наличие облачного хранилища для разработанных приложений

Вариант 4 наличие собственного эффективного эмулятора

8. **Среда** разработки Intel XDK поддерживает следующие языки программирования:

(Отметьте один правильный вариант ответа.)

Вариант 1 C++

Вариант 2 C#

Вариант 3 Java

Вариант 4 JavaScript

9. Для запуска приложений, разработанных в Android IDE, необходимо:

(Ответ считается верным, если отмечены все правильные варианты ответов.)

Вариант 1 настроить среду и запустить проект на устройстве

Вариант 2 запустить виртуальную машину с установленной ОС Android

Вариант 3 настроить компьютер (для Windows необходимо установить нужный драйвер вручную, нужны права администратора)

Вариант 4 настроить устройство (включить режим отладки по USB)

10. Не является средством разработки под Android:

(Отметьте один правильный вариант ответа.)

Вариант 1 ProGuard

Вариант 2 Device Monitor

Вариант 3 AVD Manager

Вариант 4 SDK Manager

11. С какой целью был создан Open Handset Alliance?

(Отметьте один правильный вариант ответа.)

Вариант 1 писать историю развития ОС Android

Вариант 2 продавать смартфоны под управлением Android

Вариант 3 разрабатывать открытые стандарты для мобильных устройств

Вариант 4 рекламировать смартфоны под управлением Android

12. Какое название получила версия Android 4.4?

(Отметьте один правильный вариант ответа.)

Вариант 1 Krispy Kreme

Вариант 2 kream-karamel

Вариант 3 kitekat

Вариант 4 Kit Kat

13. С какой целью инструмент Intel* Hardware Accelerated Execution Manager (Intel* HAXM) используется в среде разработки Intel* Beacon Mountain?

(Отметьте один правильный вариант ответа.)

Вариант 1 для ускорения работы эмулятора в среде разработки

Вариант 2 для эффективного распараллеливания C++ мобильных приложений

Вариант 3 для оптимизированной обработки данных и изображений

Вариант 4 для оптимизации загрузки системы при использовании процедур OpenGL

14. С какой целью инструмент Intel* Graphics Performance Analyzers (Intel* GPA) System Analyzer используется в среде разработки Intel* Beacon Mountain?

(Отметьте один правильный вариант ответа.)

Вариант 1 позволить разработчикам эффективно распараллелить C++ мобильные приложения

Вариант 2 позволить разработчикам оптимизировать загрузенность системы при использовании процедур OpenGL

Вариант 3 для ускорения работы эмулятора в среде разработки

Вариант 4 для оптимизированной обработки данных и изображений

15. Тестирование на реальных устройствах приложений, разработанных в Intel XDK

(Отметьте один правильный вариант ответа.)

Вариант 1 невозможно

Вариант 2 можно проводить, установив на устройство специальную программу или собрав проект и установив результат сборки на устройство

Вариант 3 можно проводить, установив на устройство специальную программу

Вариант 4 можно проводить, собрав проект и установив результат сборки на устройство

16. Эмулятор среды разработки Intel XDK:

(Отметьте один правильный вариант ответа.)

Вариант 1 не требует больших системных ресурсов

Вариант 2 позволяет проверить работу приложений, использующих акселерометр

Вариант 3 позволяет моделировать исполнение приложения на ряде устройств

Вариант 4 все варианты ответа верны

17. Intel XDK поддерживает разработку под:

(Отметьте один правильный вариант ответа.)

Вариант 1 JavaFX Mobile

Вариант 2 Apple iOS, BlackBerry OS

Вариант 3 Android, Apple iOS, Microsoft Windows 8, Tizen

Вариант 4 MtkOS, Symbian OS, Microsoft Windows 8

18. Какая графическая библиотека входит в набор библиотек ОС Android?

(Отметьте один правильный вариант ответа.)

Вариант 1 OpenCV

Вариант 2 DirectX

Вариант 3 Open GL

Вариант 4 OpenCL

19. Приложения, не имеющие GUI и выполняющиеся в фоновом режиме - это

(Отметьте один правильный вариант ответа.)

Вариант 1 Intents

Вариант 2 Content Providers

Вариант 3 Activities

Вариант 4 Services

20. Удобное средство обмена между двумя NFC-устройствами:

(Отметьте один правильный вариант ответа.)

Вариант 1 Wi-Fi Direct

Вариант 2 Dalvik

Вариант 3 AndroidBeam

Вариант 4 Bluetooth

21. Преимуществом эмуляторов является:

(Отметьте один правильный вариант ответа.)

Вариант 1 невозможность полноценной отладки

Вариант 2 необходимость дополнительных системных ресурсов

Вариант 3 низкая стоимость

Вариант 4 медленный запуск

22. Фоновые приложения ...

(Отметьте один правильный вариант ответа.)

Вариант 1 после настройки не предполагают взаимодействия с пользователем, большую часть времени находятся и работают в скрытом состоянии

Вариант 2 небольшие приложения, отображаемые в виде графического объекта на рабочем столе

Вариант 3 выполняют свои функции и когда видимы на экране, и когда скрыты другими приложениями

Вариант 4 большую часть времени работают в фоновом режиме, однако допускают взаимодействие с пользователем и после настройки

23. Какая папка в структуре Android-приложения содержит файлы с исходным кодом на языке Java?

(Отметьте один правильный вариант ответа.)

Вариант 1 gen

Вариант 2 src

Вариант 3 bin

Вариант 4 res

24. Приложение какого вида имеет смысл использовать для отображения динамической информации, такой как заряд батареи, прогноз погоды, дата и время?

(Отметьте один правильный вариант ответа.)

Вариант 1 фоновое приложение

Вариант 2 приложение переднего плана

Вариант 3 виджет

Вариант 4 смешанное приложение

25. Что находится в папке gen?

(Отметьте один правильный вариант ответа.)

Вариант 1 ресурсы приложения

Вариант 2 аудио файлы

Вариант 3 изображения

Вариант 4 автоматически сгенерированные java-файлы

26. Каждый приемник широковещательных сообщений является наследником класса ...

(Отметьте один правильный вариант ответа.)

Вариант 1 ContentProvider

Вариант 2 BroadcastReceiver

Вариант 3 ViewReceiver

Вариант 4 IntentReceiver

27. Выберите верные утверждения относительно объекта-намерения (Intent).

(Ответ считается верным, если отмечены все правильные варианты ответов.)

Вариант 1 представляет собой структуру данных, содержащую описание операции, которая должна быть выполнена, и обычно используется для запуска активности или сервиса

Вариант 2 используются для передачи сообщений между основными компонентами приложений

Вариант 3 используется для получения инструкций от пользователя

Вариант 4 используется для передачи сообщений пользователю

28. Какие методы может использовать активность для запуска сервиса?

(Ответ считается верным, если отмечены все правильные варианты ответов.)

Вариант 1 bindService()

Вариант 2 openService()

Вариант 3 startService()

Вариант 4 createService()

29. Выберите верную последовательность действий, необходимых для создания в приложении контент-провайдера.

(Отметьте один правильный вариант ответа.)

Вариант 1 Проектирование способа хранения данных; Создание класса-наследника от класса ContentProvider; Определение строки авторизации провайдера, URI для его строк и имен столбцов

Вариант 2 Создание класса наследника от класса ContentProvider; Заполнение контент-провайдера данными; Определение способа работы с данными

Вариант 3 Проектирование способа хранения данных; Определение способа организации данных; Определение способа работы с данными

Вариант 4 Создание класса наследника от класса ContentProvider; Определение способа организации данных; Заполнение контент-провайдера данными

30. Какой метод вызывается системой в случае, когда активность теряет фокус?

(Отметьте один правильный вариант ответа.)

Вариант 1 onPause()

Вариант 2 onRestart()

Вариант 3 onDestroy()

Вариант 4 onStop()

31. Какой вид компоновки определяет табличный способ расположения компонентов графического интерфейса пользователя в приложениях под Android?

(Отметьте один правильный вариант ответа.)

Вариант 1 TableLayout

Вариант 2 RowLayout

Вариант 3 GridBagLayout

Вариант 4 GridLayout

32. Дизайн или проектирование интерфейса для графических дизайнеров: (Отметьте один правильный вариант ответа.)

Вариант 1 передача информации о поведении посредством ожидаемого назначения

Вариант 2 прозрачность и понятность информации

Вариант 3 тон, стиль, композиция, которые являются атрибутами бренда

Вариант 4 все варианты ответа верны

32. Следующие утверждения верны:

(Отметьте один правильный вариант ответа.)

Вариант 1 текстура бесполезна для передачи различий или привлечения внимания

Вариант 2 люди легко воспринимают контрастность

Вариант 3 восприятие направления затруднено при больших размерах объектов

Вариант 4 все варианты ответа верны

33. Элементы управления "Радиокнопки":

(Отметьте один правильный вариант ответа.)

Вариант 1 должны иметь квадратную форму

Вариант 2 должны иметь круглую форму

Вариант 3 не должны иметь круглую форму

Вариант 4 могут иметь любую форму

35. Какое средство организации интерфейса помогает визуально уравновесить элементы?

(Отметьте один правильный вариант ответа.)

Вариант 1 сетка

Вариант 2 выравнивание

Вариант 3 логический маршрут

Вариант 4 симметрия

36. При создании справочной системы необходимо:

(Отметьте один правильный вариант ответа.)

Вариант 1 использовать всплывающие подсказки

Вариант 2 создать руководство по "быстрому старту"

Вариант 3 все вышеперечисленное

Вариант 4 учитывать, что она не должна становиться костылем для продукта

37. К элементам ввода относят:

(Отметьте один правильный вариант ответа.)

Вариант 1 ограничивающие элементы ввода

Вариант 2 ползунки

Вариант 3 счетчики

Вариант 4 все вышеперечисленное

38. Нижняя панель (Navigation Bar) предназначена

(Отметьте один правильный вариант ответа.)

Вариант 1 для вывода времени

Вариант 2 для вывода уровня заряда батареи

Вариант 3 для вывода уровня сигнала сотовой сети

Вариант 4 для навигации на тех устройствах, которые не имеют аппаратных навигационных клавиш

39. Один DP равен:

(Отметьте один правильный вариант ответа.)

Вариант 1 одному пикселю на экране типа MDPI

Вариант 2 0,5 дюйма

Вариант 3 300 dpi

Вариант 4 0,3 мм

40. Возможны следующие виды списков - элементов управления

(Ответ считается верным, если отмечены все правильные варианты ответов.)

Вариант 1 комбо-списки

Вариант 2 а и b

Вариант 3 раскрывающие списки

Вариант 4 нумерованные списки

41. Возможность прокрутки сеток стоит осуществлять

(Отметьте один правильный вариант ответа.)

Вариант 1 только по горизонтали

Вариант 2 по горизонтали и по вертикали

Вариант 3 по горизонтали или по вертикали

Вариант 4 только по вертикали

42. Текстовые гиперссылки при программировании для мобильных устройств:

(Отметьте один правильный вариант ответа.)

Вариант 1 стоит использовать в исключительных случаях

Вариант 2 стоит использовать для перехода между окнами приложения

Вариант 3 все варианты ответа неверны

Вариант 4 стоит использовать повсеместно

43. Какое визуальное свойство является самым главным в определении сущности объекта?

(Отметьте один правильный вариант ответа.)

Вариант 1 размер

Вариант 2 яркость

Вариант 3 форма

Вариант 4 цвет

44. При проектировании окон приложения необходимо:

(Отметьте один правильный вариант ответа.)

Вариант 1 интегрировать в рабочее пространство инструменты, используемые наиболее часто

Вариант 2 для каждой функции делать отдельное диалоговое окно

Вариант 3 все вышеперечисленное

Вариант 4 позволять диалоговому окну оперировать только с одной функцией

45. Примерами комбо-элементов не являются:

(Отметьте один правильный вариант ответа.)

Вариант 1 комбо-список

Вариант 2 все вышеперечисленное

Вариант 3 комбо-поле

Вариант 4 комбо-кнопка

46. К традиционным типографическим инструментам не относят
(Отметьте один правильный вариант ответа.)

Вариант 1 цвет

Вариант 2 разреженность

Вариант 3 выравнивание по сетке

Вариант 4 масштаб

47. Минимальный размер элемента управления
(Отметьте один правильный вариант ответа.)

Вариант 1 64 dp

Вариант 2 48dp

Вариант 3 32px

Вариант 4 30pt

48. Выделяют следующие категории элементов управления:
(Отметьте один правильный вариант ответа.)

Вариант 1 системные элементы управления

Вариант 2 все варианты ответа верны

Вариант 3 элементы управления интерфейсами

Вариант 4 командные элементы управления

49. В чем заключается суть структурных шаблонов проектирования?
(Отметьте один правильный вариант ответа.)

Вариант 1 решают проблемы, относящиеся к взаимодействию с функциональными элементами

Вариант 2 решают проблемы, связанные с управлением отображением информации

Вариант 3 определяют тип продукта в отношении к пользователю

Вариант 4 все перечисленные

50. Какие элементы управления применяются для действий по настройке?
(Отметьте один правильный вариант ответа.)

Вариант 1 элементы ввода

Вариант 2 элементы отображения

Вариант 3 командные элементы управления

Вариант 4 элементы выбора

51. Следующие утверждения не верны:
(Отметьте один правильный вариант ответа.)

Вариант 1 на любом шаге должна быть возможность вернуться назад

Вариант 2 если объекты похожи, они должны выполнять сходные действия

Вариант 3 картинки работают быстрее, чем слова

Вариант 4 не используйте интерфейсные элементы

52. Расстояние между элементами управления рекомендуется делать кратным:
(Отметьте один правильный вариант ответа.)

Вариант 1 24dp

Вариант 2 8dp

Вариант 3 4dp

Вариант 4 16dp

53. Основные вкладки (FixedTabs) удобны при отображении
(Отметьте один правильный вариант ответа.)

Вариант 1 двух вкладок

Вариант 2 от четырех вкладок

Вариант 3 трех и менее вкладок

Вариант 4 трех и более вкладок

54. Объектами информационного дизайна являются

(Отметьте один правильный вариант ответа.)

Вариант 1 диаграммы

Вариант 2 все варианты ответа верны

Вариант 3 инфографика

Вариант 4 всевозможные графики

55. Расположение элементов мобильного приложения:

(Отметьте один правильный вариант ответа.)

Вариант 1 влияет на удобство использования

Вариант 2 полезно для передачи иерархии

Вариант 3 все варианты ответа верны

Вариант 4 полезно для создания пространственных отношений между объектами на экране и объектами реального мира

56. Использование ярлычков для всех панелей сетки и полупрозрачных "занавесок" для неактивных панелей

(Отметьте один правильный вариант ответа.)

Вариант 1 мешает пользователю понять, что скрывается под той или иной панелью

Вариант 2 скрывает от пользователя нужные панели

Вариант 3 нет правильного варианта ответа

Вариант 4 помогает пользователю быстрее понять, что скрывается под той или иной панелью

57. Строительные блоки визуального дизайна интерфейсов:

(Отметьте один правильный вариант ответа.)

Вариант 1 цвет

Вариант 2 все варианты ответа верны

Вариант 3 размер

Вариант 4 форма

58. Более крупные элементы:

(Отметьте один правильный вариант ответа.)

Вариант 1 привлекают больше внимания

Вариант 2 привлекают меньше внимания

Вариант 3 размер не влияет на уровень внимания

Вариант 4 все варианты ответа верны

59. К элементам управления отображением не относятся:

(Отметьте один правильный вариант ответа.)

Вариант 1 полосы прокрутки

Вариант 2 сетки и рамки

Вариант 3 разделители

Вариант 4 рукоятки

60. Выделяют следующие категории плотности экрана для Android-устройств:

(Отметьте один правильный вариант ответа.)

Вариант 1 правильный вариант ответа отсутствует

Вариант 2 HDPI, XHDPI, XXHDPI, и XXXHDPI

Вариант 3 LDPI, MDPI, HDPI

Вариант 4 LDPI, MDPI, HDPI, XHDPI, XXHDPI, и XXXHDPI

61. Всплывающие подсказки стоит использовать, когда (Отметьте один правильный вариант ответа.)

Вариант 1 сообщение является важным и требует немедленного прочтения и ответа

Вариант 2 сообщение является важным, однако требует немедленного прочтения, но не ответа

Вариант 3 сообщение является важным, однако не требует немедленного прочтения и ответа

Вариант 4 сообщение не требует ответа пользователя, но важно для продолжения его работы

61. ProgressDialog это:

(Отметьте один правильный вариант ответа.)

Вариант 1 диалоговое окно, которое может содержать заголовок, до трех кнопок, список выбираемых значений или настраиваемое содержимое

Вариант 2 контейнер для создания собственных диалоговых окон

Вариант 3 диалоговое окно с предопределенным интерфейсом, позволяющее выбрать дату или время

Вариант 4 диалоговое окно, содержащее линейку процесса выполнения какого-то действия

62. DialogFragment это:

(Отметьте один правильный вариант ответа.)

Вариант 1 диалоговое окно, которое может содержать заголовок, до трех кнопок, список выбираемых значений или настраиваемое содержимое

Вариант 2 контейнер для создания собственных диалоговых окон

Вариант 3 диалоговое окно с предопределенным интерфейсом, позволяющее выбрать дату или время

Вариант 4 диалоговое окно, содержащее линейку процесса выполнения какого-то действия

63. При загрузке приложения первой появляется активность,

(Отметьте один правильный вариант ответа.)

Вариант 1 чье описание первым в лексикографическом порядке

Вариант 2 чье описание находится первым в манифесте

Вариант 3 чье описание находится последним в манифесте

Вариант 4 правильный вариант ответа отсутствует

64. Для чего предназначен класс ListActivity?

(Отметьте один правильный вариант ответа.)

Вариант 1 для создания нового листа

Вариант 2 для создания активности, основным элементом которой является список

Вариант 3 для создания обычной активности

Вариант 4 для создания активности, основным элементом которой является таблица

65. Интерфейс Adapter позволяет:

(Отметьте один правильный вариант ответа.)

Вариант 1 связывать список и названия его элементов

Вариант 2 оптимизировать процесс зарядки телефона

Вариант 3 обрабатывать нажатия

Вариант 4 адаптировать приложение к любой версии Android SDK

66. Для создания собственных диалоговых окон используют контейнер:

(Отметьте один правильный вариант ответа.)

Вариант 1 AlertDialog

Вариант 2 ProgressDialog

Вариант 3 DialogFragment

Вариант 4 DatePickerDialog

67. Можно ли создать диалоговое окно без управляющих кнопок?

(Отметьте один правильный вариант ответа.)

Вариант 1 нет, ни в коем случае

Вариант 2 да, если у него отсутствует содержательная часть

Вариант 3 нет, диалоговое окно должно содержать от одной до трех кнопок

Вариант 4 да, если элементы содержательной части являются кликабельными

68. Заголовок диалогового окна:

(Отметьте один правильный вариант ответа.)

Вариант 1 генерируется автоматически

Вариант 2 является обязательным элементом

Вариант 3 является необязательным элементом

Вариант 4 является необязательным элементом, если отсутствуют содержательная часть и управляющие кнопки

69. Шаблон Fixed Tabs + Swipe позволяет переключаться между вкладками:

(Отметьте один правильный вариант ответа.)

Вариант 1 только с помощью кнопок

Вариант 2 только с помощью перелистывания

Вариант 3 как с помощью кнопок, так и с помощью перелистывания

Вариант 4 не позволяет переключаться между вкладками

4.5 Промежуточная аттестация по МДК 01.04

Тест

1. Установите порядок, в котором осуществляется перевод произвольного двоичного числа в системе счисления с основанием $q=2^n$

Укажите порядок следования всех 3 вариантов ответа: 1. рассмотреть каждую группу как n -разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием $q = 2^n$; 2. двоичное число разбить слева и справа (целую и дробную части) на группы по цифр в каждой; 3. если в последних правой и левой группах окажется меньше разрядов, то их надо дополнить справа и слева нулями до нужного числа разрядов

a) **3 1 2**

b) 2 1 3

c) 1 2 3

d) 3 2 1

2. По сколько цифр в группе нужно разбивать двоичное число при переводе его в четверичную систему счисления?

a) 6

b) 2

c) 3

d) 4

3. Переведите двоичное число 100011111011 в шестандцатеричную систему счисления:

a) 5fa

b) 8fb

c) 8fc

d) 8fa

4. Чему будет равен результат сложение двоичных чисел 1001 и 1000

a) 10001

b) 11001

c) 10011

d) 10101

5. Выполните деление в двоичной системе счисления $1110:10=$
- 101
 - 011
 - 110
 - 111**
6. Переведите двоичное число 1010101 в восьмеричную систему счисления
- 125**
 - 124
 - 126
 - 123
7. Установите порядок, в котором осуществляется перевод целого числа в системе счисления с основанием $q=2^n$. Укажите порядок следования всех 3 вариантов ответа: 1. данное двоичное число разбить справа налево на группы по цифр в каждой; 2. рассмотреть каждую группу как n -разрядное двоичное число и записать ее соответствующей цифрой в системе счисления с основанием $q = 2^n$; 3. если в последней левой группе окажется меньше разрядов, то ее надо дополнить слева нулями до нужного числа разрядов
- 1 3 2**
 - 1 2 3
 - 3 2 1
 - 2 1 3
8. Переведите шестнадцатеричное число FACC в двоичную систему счисления
- 1111001011011001
 - 1111101011001100**
 - 1011010111111100
 - 1101100111011010
9. Выполните вычисление в двоичной системе счисления $110101-101=$
- 110100
 - 110010
 - 110000**
 - 110001
10. В саду 100 фруктовых деревьев - 14 яблонь и 42 груши. В какой системе счисления посчитаны деревья?
- В шестеричной**
 - В шестнадцатеричной
 - В двоичной
 - В восьмеричной
11. Умножьте в двоичной системе счисления $1111*11=$
- 1111111
 - 101111
 - 1010101
 - 101101**
12. Установите соответствие. Укажите соответствие для всех 4 вариантов ответа: 1) 4 2) 5
3) 6 4) 7
- 4=101 5=110 6=100 7=111
 - 5=101 6=110 4=100 7=111**
 - 6=101 5=110 4=100 7=111
 - 4=101 7=110 6=100 5=111
13. Выполните умножение в системе счисления $10101*101=$
- 110101011
 - 1101001**
 - 111001001
 - 1010101
14. Установите соответствие. Укажите соответствие для всех 4 вариантов ответа:
- A=1010 B=1111 F=1110 E=1011**

- b) **A=1010 F=1111 E=1110 B=1011**
c) A=1010 C=1111 F=1110 E=1011
d) A=1010 B=1111 F=1110 C=1011
15. В какой системе счисления будет верным равенство $7+8=16$?
a) 12
b) 6
c) **9**
d) 16
16. $FFFF+1=$
a) 1FFF
b) **10000**
c) FFFE
d) FFF1
17. Выполните вычитание в двоичной системе счисления $1111-101$
a) **1010**
b) 1110
c) 1100
d) 1001
18. Результат деления в двоичной системе счисления числа 110 на 11 равен
a) 11
b) **10**
c) 01
d) 101
19. Перевести смешанное двоичное $111100000,101$ число в восьмеричную систему счисления
a) 610,4
b) **740,5**
c) 170,5
d) 1E0,A
20. Переведите восьмеричное число 0,25 в двоичную систему счисления
a) **0,010101**
b) 0,101010
c) 0,110111
d) 0,101011
21. Алгоритм-это:
a) Указание на выполнение действий,
b) **Система правил, описывающая последовательность действий, которые необходимо выполнить для решения задачи,**
c) Процесс выполнения вычислений, приводящих к решению задачи
22. Свойство алгоритма – дискретность, выражает, что:
a) Команды должны следовать последовательно друг за другом,
b) Каждая команда должна быть описана в расчете на конкретного исполнителя,
c) **Разбиение алгоритма на конечное число команд**
23. Формальное исполнение алгоритма – это:
a) Исполнение алгоритма конкретным исполнителем с полной записью его рассуждений,
b) Разбиение алгоритма на конкретное число команд и пошаговое их исполнение,
c) **Исполнение алгоритма не требует рассуждений, а осуществляется исполнителем автоматически**
d) Исполнение алгоритма осуществляется исполнителем на уровне его знаний
24. Самое важное свойство алгоритма:
a) Визуальность,
b) **Массовость,**
c) Дискретность,
d) Аудиальность,
25. Какой алгоритм называется линейным:

- a) Выполнение операций зависит от условия,
 - b) Операции выполняются друг за другом,**
 - c) Одни и те же операции выполняются многократно
 - d) Присутствие всех возможных операций в одном алгоритме
26. Графическое задание алгоритма – это:
- a) Способ представления алгоритма с помощью геометрических фигур,**
 - b) Представление алгоритма в форме таблиц и расчетных формул,
 - c) Система обозначений и правил для единообразной и точной записи алгоритмов и их исполнения
27. В расчете на кого должен строиться алгоритм:
- a) В расчете на ЭВМ,
 - b) В расчете на умственные способности товарища,
 - c) В расчете на конкретного исполнителя**
28. Какое из перечисленных свойств относится к свойствам алгоритма:
- a) Визуальность,
 - b) Совокупность,
 - c) Аудиальность,
 - d) Понятность**
29. Псевдокоманда DB означает
- a) определить байт;**
 - b) определить слово (2 байта);
 - c) определить двойное слово (4 байта);
 - d) определить 6 байт
 - e) определить учетверенное слово (8 байт);
30. Псевдокоманда DW означает
- a) определить байт;
 - b) определить слово (2 байта);**
 - c) определить двойное слово (4 байта);
 - d) определить 6 байт
 - e) определить учетверенное слово (8 байт);
31. Псевдокоманда DD означает
- a) определить байт;
 - b) определить слово (2 байта);
 - c) определить двойное слово (4 байта);**
 - d) определить 6 байт
 - e) определить учетверенное слово (8 байт);
32. К сегментным регистрам относят
- a) CS, DS, SS и ES.**
 - b) AX, BX, CX и DX
 - c) SP и BP.
 - d) SI и DI
33. К индексным регистрам относят
- a) CS, DS, SS и ES.
 - b) AX, BX, CX и DX
 - c) SP и BP.
 - d) SI и DI**
34. Устанавливает бит результата в 1, если оба бита, бит источника и бит приемника установлены в 1.
- a) AND
 - b) OR**
 - c) XOR (НЕ ИЛИ)
 - d) NOT
35. устанавливает бит результата в 1, если бит источника отличается от бита приемника.
- a) AND**

- b) OR
 - c) XOR
 - d) NOT
36. Запись регистра в память осуществляется при помощи команды
- a) **MOV**
 - b) XCHG
 - c) XLAT
37. Порядковый номер элемента в массиве, который всегда начинается с нуля
- a) **Индекс**
 - b) Команда
 - c) Псевдокоманда
 - d) Макрокоманда
38. Позволяет сохранять длину исходного текста программы
- a) Индекс
 - b) Команда
 - c) Псевдокоманда
 - d) **Макрокоманда**

Задачи

Для решения следующих задач разработать программы на Ассемблере.

1. Поменять местами содержимое двух переменных ($X \leftrightarrow Y$).
2. Выполнить циклический обмен содержимого трех переменных (X, Y, Z).
3. Сохранить в переменных X, Y, Z, Произведения числа, расположенного в заданной ячейке памяти, на числа 11, 12, 13, 14.
4. Пусть в четырех последовательных ячейках памяти расположены числа (соответственно) x, y, 10, 14. Вычислить выражение: $10x - 14y$.
5. Вычислить значения выражений:

1) $2x^2 + 3y^2 + 35xy - 18$;	2) $7x^3 + 4x + 5$;	3) $x^2 - 5x + 7$;
4) $ax^2 + bx + c$;	5) $x^4 - 15x + 20$.	
6. Для заданного X вычислите значение X^{32} , используя наименьшее количество команд и переменных.
7. Вычислите значения выражений:

1) $ x^2 - 7x - 1 $;	2) $x^2 - 5x + 7 $.
-----------------------	-----------------------
8. Вычислите значения выражений:

1) $y = \begin{cases} x - 5 , & \text{если } x \leq 0, \\ x - 5, & \text{если } x > 0; \end{cases}$	2) $y = \begin{cases} x + 5, & \text{если } x < 0, \\ x - 5, & \text{если } x \geq 0; \end{cases}$
1) $y = \begin{cases} x - 5, & \text{если } x \leq 0, \\ x - 5 , & \text{если } x > 0. \end{cases}$	
9. Пусть задано некоторое целое положительное число X. С использованием конструкции цикла просуммировать все числа от 1 до X.
10. Определить делится ли целое число X на целое число Y ($X > Y$), используя вычитание Y из X в цикле.
11. Определить наибольший общий делитель (НОД(x, y)) двух натуральных чисел с помощью алгоритма Евклида, используя очевидное свойство: $\text{НОД}(x, y) = \text{НОД}(x - y, y)$, при $x > y$.
12. Дано целое число X и натуральное N. Вычислить значение выражения X^N .
13. Числа Фибоначчи определяются по формуле $U_{n+1} = U_n + U_{n-1}$, $U_1 = U_2 = 1$, т.е. каждый последующий член равен сумме двух предыдущих.
 - 1) Определить значение числа Фибоначчи с заданным номером.
 - 2) Найти сумму чисел Фибоначчи, не превышающих заданного K.
14. Вычислить сумму числовой последовательности $1 + 3 + 5 + \dots + (2n - 1)$ и убедиться, что она равна n^2 .
15. Вычислить сумму числовой последовательности $1 + 8 + 16 + \dots + 8(n - 1)$ и убедиться, что она равна $(2n - 1)^2$.

16. Отсортировать заданную последовательность чисел в возрастающем (убывающем) порядке.
17. Даны два отсортированных массива, объединить их в один, тоже отсортированный.
18. Для заданной последовательности чисел $x_1, x_2, \dots, x_n$ определить их сумму: $S = x_1 + x_2 + \dots + x_n$ и сумму квадратов: $D = x_1^2 + x_2^2 + \dots + x_n^2$.

4.6 Контрольные вопросы к экзамену по МДК.01.01.

1. Программы и программное обеспечение
2. Система счисления. Двоичные числа
3. Система счисления. Шестнадцатеричные числа
4. Алгоритм Банкира
5. Алгоритм Медника
6. Алгоритм замещения страниц
7. Исследование файловой системы ввода/вывода
8. Директивы определения данных
9. Директивы распределения памяти
10. Регистры
11. Архитектура микропроцессоров
12. Предложения языка Ассемблер
13. Операнды команд
14. Алгоритмы работы Ассемблеров
15. Описание данных в Ассемблер
16. Команды пересылки общего назначения
17. Команды загрузки адресных значений и обращения к стеку
18. Команды ввода/вывода
19. Арифметические операции над двоичными числами. Сложение (вычитание) двоичных чисел без знака
20. Арифметические операции над двоичными числами. Сложение (вычитание) двоичных чисел со знаком
21. Арифметические операции над двоичными числами. Умножение двоичных чисел
22. Арифметические операции над двоичными числами. Деление двоичных чисел
23. Арифметические операции над десятичными числами. Сложение десятичных чисел
24. Арифметические операции над десятичными числами. Умножение десятичных чисел
25. Арифметические операции над десятичными числами. Деление десятичных чисел
26. Логические команды. Команды логических операций.
27. Логические команды. Команды сравнения.
28. Логические команды. Команды сдвигов.
29. Организация подпрограмм. Передача параметров в подпрограммы.
30. Организация макрокоманд. Передача параметров в макрокоманды

4.7. Контрольные вопросы к экзамену по МДК 01.02

1. Тестирование программных продуктов. Цель, стадии тестирования, виды тестирования, подходы к формированию тестов.
2. Тестирование программных продуктов. Стоимость исправления ошибок, этапы тестирования, место стадии тестирования в жизненном цикле программного продукта.
3. Тестирование программных продуктов. Подходы к формированию тестов. Ручной контроль. Методы ручного контроля.
4. Тестирование программных продуктов. Подходы к формированию тестов. Структурное тестирование. Критерии формирования тестовых наборов при структурном тестировании.
5. Тестирование программных продуктов. Подходы к формированию тестов. Функциональное тестирование. Методы формирования тестовых наборов при функциональном тестировании.

6. Нисходящее и восходящее тестирование, критерии завершения тестирования, оценочное тестирование и его виды.
7. Отладка программного продукта, причины сложности отладки программного продукта. Классификация ошибок при отладке программного продукта. Общая методика отладки программного продукта.
8. Отладка программного продукта. Классификация ошибок при отладке программного продукта. Методы и средства получения дополнительной информации при отладке программного продукта. Общая методика отладки программного продукта.
9. Программная документация. Виды программных документов. Основные правила оформления текстовых документов.

4.8 Контрольные вопросы к экзамену по МДК 01.03

1. Мобильное программирование, платформы для разработки.
2. Система Windows Phone 7.
3. Microsoft Visual Studio Express for Windows Phone.
4. Аппаратные средства устройств, поддерживающих Windows Phone 7.
5. Инструменты разработки Windows Phone7-приложений. Пример простейших программ Windows Phone 7-приложения. Запуск приложения на эмуляторе.
6. История появления, аспекты применения Microsoft Silverlight.
7. Основы работы с сенсорным вводом. Обработка нескольких касаний. Использование изображений.
8. Акселерометр и служба определения местоположения, вторичные потоки выполнения, обработка асинхронных операций и доступ к Веб-сервисам.
9. Особенности использования pivot и panorama.
10. Принципы интерфейса системы и приложений Metro
11. Типографика.
12. Краткая история ОС Android.
13. Intel для Android: партнерство и инструментарий разработчика.
14. Архитектура приложений для Android. Ресурсы приложения. Пользовательский интерфейс. Инструментарий разработки приложений для Android.
15. Обзор шагов разработки типичного приложения под Android. Особенности разработки с использованием эмулятора. Отладка кода в эмуляторе и на реальных устройствах. Пример простейших программ Android-приложения. Запуск приложения на эмуляторе.
16. Тестирование приложения с помощью Dalvik Debug Monitor Server (DDMS).
17. Планирование кадровой анимации, анимирование, анимация шаблонов, видов, использование класса Camera.
18. Проверка безопасности, работа со службами, основанными на местоположении, использование HTTP-служб, службы AIDL.
19. Подготовка AndroidManifest.xml для загрузки, локализация приложения, подготовка ярлыка приложения, подготовка APK-файла для загрузки, работа пользователя с Android Market.
20. Intel Power Monitoring Tool. Intel Graphics Performance Analyzer. Intel Energy Checker SDK. Intel Hardware Accelerated Execution Manager.
21. Работа с инструментами Intel для оптимизации отладки Android-приложений.
22. Текстовые элементы управления, кнопки, списки, таблицы, управление датой и временем, MapView.
23. Галерея, счетчик, диспетчеры шаблонов, адаптеры.
24. Создание меню, расширенные меню, загрузка меню при помощи XML-файлов, создание диалоговых окон, диалоговые окна с подсказками и предупреждениями.

4.9 Контрольные вопросы к экзамену по МДК 01.04.

1. Программы и программное обеспечение

2. Система счисления. Двоичные числа
3. Система счисления. Шестнадцатеричные числа
4. Алгоритм Банкира
5. Алгоритм Медника
6. Алгоритм замещения страниц
7. Исследование файловой системы ввода/вывода
8. Директивы определения данных
9. Директивы распределения памяти
10. Регистры
11. Архитектура микропроцессоров
12. Предложения языка Ассемблер
13. Операнды команд
14. Алгоритмы работы Ассемблеров
15. Описание данных в Ассемблер
16. Команды пересылки общего назначения
17. Команды загрузки адресных значений и обращения к стеку
18. Команды ввода/вывода
19. Арифметические операции над двоичными числами. Сложение (вычитание) двоичных чисел без знака
20. Арифметические операции над двоичными числами. Сложение (вычитание) двоичных чисел со знаком
21. Арифметические операции над двоичными числами. Умножение двоичных чисел
22. Арифметические операции над двоичными числами. Деление двоичных чисел
23. Арифметические операции над десятичными числами. Сложение десятичных чисел
24. Арифметические операции над десятичными числами. Умножение десятичных чисел
25. Арифметические операции над десятичными числами. Деление десятичных чисел
26. Логические команды. Команды логических операций.
27. Логические команды. Команды сравнения.
28. Логические команды. Команды сдвигов.
29. Организация подпрограмм. Передача параметров в подпрограммы.
30. Организация макрокоманд. Передача параметров в макрокоманды

Критерии оценки МДК 01.01, МДК 01.02, МДК 01.03, МДК 01.04

- «Отлично» - теоретическое содержание МДК освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.
- «Хорошо» - теоретическое содержание МДК освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.
- «Удовлетворительно» - теоретическое содержание МДК освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.
- «Неудовлетворительно» - теоретическое содержание МДК не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.

V ПЕРЕЧЕНЬ ВИДОВ РАБОТ ДЛЯ ПРОВЕРКИ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОГРАММЫ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ НА ПРАКТИКЕ

5.1. Требования к дифференцированному зачёту учебной практике

Целью оценки по учебной практике является оценка профессиональных и общих компетенций; практического опыта и умений.

Оценка по учебной практике выставляется на основании данных аттестационного листа (характеристики профессиональной деятельности студента на практике) с указанием видов работ, выполненных обучающимся во время практики, их объема, качества выполнения в соответствии с технологией и требованиями организации, в которой проходила практика.

Виды работ учебной практики и проверяемые результаты обучения по профессиональному модулю

Виды работ	Проверяемые результаты (ПК, ОК)	Оценка
Инструктаж о прохождении практики. Знакомство с программой практики и порядок её проведения, изучение правил внутреннего распорядка, знакомство с графиком работы студентов, ведения дневника практики, составление отчета. Инструктаж по технике безопасности, пожаробезопасности, производственной санитарии под роспись в журнале. Правила безопасности при работе с компьютером.	ОК 1 ПК 1.6	
Анализ поставленной задачи. Выбор методов и разработка основных алгоритмов решения задачи. Анализ входных и выходных данных. Определение границ входных и выходных данных. Разработка технического задания. Примерные темы заданий для выполнения практической работы: • Обработка сообщений • Рисование геометрических фигур в окне • Вывод текста • Диалог с пользователем • Чтение и запись файлов в библиотеке Win32 API • Диалоговые окна • Растровая графика • Анимация • Библиотеки динамической компоновки DLL Варианты заданий: • Создание диалоговых окон. • Создание главных окон. • Графика 2D и 3D. • Технология “drag-and-drop”. • Управление компоновкой виджетов на форме. • Обработка событий во время продолжительных процессов. • Работа с каталогами.	ОК 1-9 ПК 1.1 – 1.2	
Разработка структуры и конкретных компонент разрабатываемого программного обеспечения, в том числе блок-схемы поставленной задачи	ОК 1-9 ПК 1.1 – 1.2	

алгоритмов ,их общее описание, обоснование принятых технических решений. Математическая формализация. Построение информационной модели для решения поставленной задачи. Выделение объектов и процессов.		
Описание соотношений между характеристиками объектов моделирования. Системный анализ объектов моделирования поставленной задачи.	ОК 1-9 ПК 1.1 – 1.2	
Реализация метода и основного алгоритма решения задачи методом последовательной детализации. Определение свойств входных и выходных данных поставленной задачи.	ОК 1-9 ПК 1.1 – 1.5	
Анализ процесса обработки информации и выбор структур данных для её хранения	ОК 1-9 ПК 1.1 – 1.5	
Построение алгоритма решения поставленной задачи средствами автоматизированного проектирования..	ОК 1-9 ПК 1.1 – 1.2	
Выбор технологии и среды программирования. Разработка структурной схемы программного продукта.	ОК 1-9 ПК 1.1 – 1.5	
Анализ и уточнение требований к программному продукту. Применение технологии разработки многомодульных программ. Построение каркаса приложения.	ОК 1-9 ПК 1.1 – 1.2	
Проектирование интерфейса пользователя. Проектирование классов предметной области. Организация обработки сообщений.	ОК 1-9 ПК 1.1 – 1.2	
Построение графа диалога. Разработка форм ввода-вывода информации.	ОК 1-9 ПК 1.1 – 1.2	
Использование директив препроцессора для создания гибких и мобильных программ. Организация диалога с пользователем. Тестирование элементов управления.	ОК 1-9 ПК 1.1 – 1.2	
Работа с панелями инструментов. Чтение и запись файлов в библиотеке. Организация работы с файлами.	ОК 1-9 ПК 1.1 – 1.2	
Разработка кода программного продукта для выбранного задания на соответствующем языке программирования	ОК 1-9 ПК 1.1 – 1.2	
Выбор стратегии тестирования и разработка тестов. Отладка кода программного продукта, используя возможности отладчика.	ОК 1-9 ПК 1.1 – 1.2	
Использование средств отладки, предоставляемых интерфейсом пользователя. Определение мест программы, в которых необходимо установить точки останова.	ОК 1-9 ПК 1.3 – 1.5	
Ручной контроль программного модуля. Проверка структуры программного модуля.	ОК 1-9 ПК 1.3 – 1.5	
Структурное тестирование. Устранение утечки памяти. Исследование возможных причин утечки памяти.	ОК 1-9 ПК 1.3 – 1.5	

Функциональное тестирование. Особенности отладки приложений, использующих шаблоны функций и классов.	ОК 1-9 ПК 1.3 – 1.5	
Оценочное тестирование программного продукта. Анализ соответствия разработанного программного продукта постановке задачи.	ОК 1-9 ПК 1.3 – 1.5	
Составление программной документации. Определение сведений, необходимых для сопровождения и эксплуатации программного продукта. Разработка пояснительной записки, содержащей информацию о структуре и конкретных компонентах программного обеспечения, в том числе схемы алгоритмов, их общее описание, обоснование принятых технических решений.	ОК 1-9 ПК 1.6	
Разработка спецификаций всех файлов программного продукта. Описание сведений о логической структуре и функционировании программы.	ОК 1-9 ПК 1.6	
Разработка описания применения, содержащего сведения о назначении программного продукта, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств.	ОК 1-9 ПК 1.6	
Разработка руководства системного программиста, содержащего сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения.	ОК 1-9 ПК 1.6	
Разработка руководства по техническому обслуживанию, содержащего сведения для применения тестовых и диагностических программ при обслуживании технических средств.	ОК 1-9 ПК 1.6	

5.2. Требования к дифференцированному зачёту по производственной практике

Дифференцированный зачёт по производственной практике выставляется на основании данных аттестационного листа (характеристика профессиональной деятельности студента на практике) с указанием вида работ, выполненных студентами во время практики, их объёма, качества выполнения в соответствии с требованиями организации, в которой проходила практика. Уровень подготовки при проведении практики оценивается: отлично, хорошо, удовлетворительно, неудовлетворительно. Предметом оценки по производственной практике является приобретение практический опыта, а также освоение общих и профессиональных компетенций.

Контроль и оценка по производственной практике проводится на основе утвержденного документационного обеспечения практики: приказ, договор с предприятием, содержание производственной практик, сопроводительная и отчетная документация по практике, аттестационный лист по практике, в котором отражаются виды работ, выполненные обучающимся во время практики, их объем, качество выполнения в соответствии с технологией и требованиями организации, на базе которой проходила практика.

VI СТРУКТУРА КОНТРОЛЬНО-ОЦЕНОЧНЫХ МАТЕРИАЛОВ ДЛЯ ЭКЗАМЕНА (КВАЛИФИКАЦИОННОГО)

6.1 Задания для экзаменуемого

Разработать приложение, содержащее базу данных:

- автоматизация учёта результатов физического развития студентов колледжа;
- автоматизация учёта результатов медицинских осмотров студентов;
- автоматизация учёта участия студентов в различных конкурсах, олимпиадах, соревнованиях, конференциях и т.д.
- автоматизация учёта прохождения студентами производственных практик
- учёт книгообеспеченности библиотеки колледжа;
- организация удалённого доступа к электронным ресурсам колледжа
- автоматизация разработки индивидуальных планов работы преподавателей колледжа;
- автоматизация учёта сведений по слушателям курсов дополнительного образования колледжа;
- автоматизация учёта результатов бильярдных турниров;
- учёт выдачи книг в библиотеке.

6.2 Условия выполнения заданий

1. Место (время) выполнения задания: Лаборатория «Программного обеспечения и сопровождения компьютерных систем»
2. Максимальное время выполнения задания: 270 мин.
3. Вы можете воспользоваться:

Основные источники:

1. Основы программирования. Учебник с практикумом (для СПО). Учебник : учебник / Н.В. Макарова. — Москва : КноРус, 2018. — 452 с. — ISBN 978-5-406-06505-1. Режим доступа: <https://www.book.ru/book/930074>
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учеб. пособие / Г.Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - Режим доступа: <http://znanium.com/>
3. Соколова, В. В. Разработка мобильных приложений : учеб. пособие для СПО / В. В. Соколова. — М. : Издательство Юрайт, 2019. — 175 с. — (Серия : Профессиональное образование). — ISBN 978-5-534-10680-0. — Режим доступа : www.biblio-online.ru/book/57114D71-FBD8-49ED-B562-26A75AEAAF9F.
4. Технология программирования : учебник / Г.С. Иванова. — Москва : КноРус, 2018. — 333 с. — Для бакалавров. — ISBN 978-5-406-06109-1. - Режим доступа: <https://www.book.ru/book/926372>

6.3 Структура контрольно-оценочных материалов для экзамена (квалификационного)

В процессе итогового оценивания освоения содержания программы профессионального модуля ПМ.01 Разработка модулей программного обеспечения для компьютерных систем используются:

- положительная аттестация по МДК.01.01;
- положительная аттестация по МДК.01.02;
- положительная аттестация по МДК.01.03;
- положительная аттестация по МДК.01.04;
- оценка по итогам учебной практики
- оценка по итогам производственной практики.

6.4 Общая оценка результата освоения вида профессиональной деятельности

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем: неудовлетворительно, удовлетворительно, хорошо, отлично.

Условием положительной аттестации на экзамене (квалификационном) является положительная оценка (80 % выполнения задания) освоения всех профессиональных компетенций по всем контролируемым показателям.

При отрицательном заключении хотя бы по одной из профессиональных компетенций принимается решение: оценка «неудовлетворительно».

6.5. Пакет экзаменатора

ПАКЕТ ЭКЗАМЕНАТОРА		
Объекты оценки	Критерии оценки результата (в соответствии с разделом 1 «Паспорт комплекта контрольно-оценочных средств»)	Отметка о выполнении
ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием. Формализация и алгоритмизация поставленных задач	Использование методов и приемов формализации задач Использование методов и приемов алгоритмизации поставленных задач Использование программных продуктов для графического отображения алгоритмов Применение стандартных алгоритмов в соответствующих областях	
ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.	Применение выбранных языков программирования для написания программного кода Использование выбранной среды программирования и средства системы управления базами данных Использование возможности имеющейся технической и/или программной архитектуры	
ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств.	Выявление ошибок в программном коде Применение методов и приемов отладки программного кода Интерпретирование сообщения об ошибках, предупреждения, записи технологических журналов Применение современных компиляторов, отладчиков и оптимизаторов программного кода	
ПК 1.4 Выполнять тестирование программных модулей. ПК 1.5. Осуществлять оптимизацию программного кода	Разработка и оформление контрольных примеров для проверки работоспособности программного обеспечения Разработка процедур генерации тестовых наборов данных с заданными характеристиками Подготовка наборов данных, используемых в процессе проверки работоспособности программного обеспечения	

6.6 Перечень материалов, оборудования и информационных источников, используемых при аттестации

Технические средства обучения:

- компьютеры по количеству обучающихся;
- локальная компьютерная сеть и глобальная сеть Интернет;
- лицензионное системное и прикладное программное обеспечение;
- лицензионное антивирусное программное обеспечение;
- лицензионное специализированное программное обеспечение;
- мультимедиапроектор;
- система дистанционного обучения Модус.

Информационное обеспечение обучения

Основные источники

Электронные источники

1. Основы программирования. Учебник с практикумом (для СПО). Учебник : учебник / Н.В. Макарова. — Москва : КноРус, 2018. — 452 с. — ISBN 978-5-406-06505-1. Режим доступа: <https://www.book.ru/book/930074>
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учеб. пособие / Г.Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - Режим доступа: <http://znanium.com/>
3. Соколова, В. В. Разработка мобильных приложений : учеб. пособие для СПО / В. В. Соколова. — М. : Издательство Юрайт, 2019. — 175 с. — (Серия : Профессиональное образование). — ISBN 978-5-534-10680-0. — Режим доступа : www.biblio-online.ru/book/57114D71-FBD8-49ED-B562-26A75AEAAF9F.
4. Технология программирования : учебник / Г.С. Иванова. — Москва : КноРус, 2018. — 333 с. — Для бакалавров. — ISBN 978-5-406-06109-1. - Режим доступа: <https://www.book.ru/book/926372>

Дополнительные источники

5. Программирование на языке C++: учеб. пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРА-М, 2019. — 512 с. — (Среднее профессиональное образование). - Режим доступа: <http://znanium.com/catalog/product/1000008>
6. Основы алгоритмизации и программирования (среда PascalABC.Net) : учеб. пособие / И.Г. Фризен. — М. : ФОРУМ : ИНФРА-М, 2019. — 392 с. — (Среднее профессиональное образование). - Режим доступа: <http://znanium.com/catalog/product/1007486>
7. Программирование. (Бакалавриат). Учебник : учебник / Г.С. Иванова. — Москва : КноРус, 2019. — 426 с. — ISBN 978-5-406-06774-1. Режим доступа: <https://www.book.ru/book/931234>
8. Технология Microsoft ADO .NET : курс лекций / Ч.А. Кариев. — Москва : Интуит НОУ, 2016. — 666 с. — ISBN 978-5-9556-0097-0.
9. Объектно-ориентированное программирование на Visual Basic в среде Visual Studio .NET : учеб. пособие / В.Н. Шакин, А.В. Загвоздкина, Г.К. Сосновиков. — М. : ФОРУМ : ИНФРА-М, 2018. — 398 с. — (Среднее профессиональное образование). - Режим доступа: <http://znanium.com/catalog/product/961516>
10. Программирование на языке Object Pascal : учеб. пособие / Т.И. Немцова, С.Ю. Голова, И.В. Абрамова ; под ред. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРА-М, 2018. — 496 с. +

Доп. материалы [Электронный ресурс; Режим доступа <http://www.znanium.com>]. —
(Профессиональное образование). - Режим доступа:
<http://znanium.com/catalog/product/944326>

ПРИЛОЖЕНИЕ 1 ФОРМА ОЦЕНОЧНОЙ ВЕДОМОСТИ

ОЦЕНОЧНАЯ ВЕДОМОСТЬ ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ ПМ. 01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Ф.И.О. _____

Обучающийся на _____ курсе по специальности 09.02.07 Информационные системы и программирование освоил программу профессионального модуля
ПМ. 01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ
КОМПЬЮТЕРНЫХ СИСТЕМ

в объеме _____ часов с «_____» _____ 20__ года по «_____» _____ 20__ года.

Результаты промежуточной аттестации по элементам профессионального модуля.

Результат оценки: вид профессиональной деятельности: _____

«_____» _____ 201__ год

Подписи членов экзаменационной комиссии

_____ (_____)

_____ (_____)

_____ (_____)

Пермский финансово-экономический колледж – филиал ФГОБУ ВО «Финансовый университет при Правительстве РФ»