**МИНОБРНАУКИ РОССИИ**

**«Юго-Западный государственный университет»**

Кафедра «Информационные системы и технологии»

**КУРСОВАЯ РАБОТА**

по дисциплине «Визуальное программирование»

на тему «Разработка демонстрационно-обучающей программы»

Направление подготовки 09.03.02 Информационные системы и технологии

Автор работы Романов Г.А \_,09.06.2020



фамилия, инициалы подпись, дата

Группа ИС-91б

Руководитель проекта Лапина Т.И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, инициалы подпись, дата

Проект защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лапина Т.И.

подпись, дата фамилия, инициалы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сазонов С.Ю.

подпись, дата фамилия, инициалы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Киселев А.В.

подпись, дата фамилия, инициалы

Курск 2020

**МИНОБРНАУКИ РОССИИ**

**«Юго-Западный государственный университет»**

Кафедра «Информационные системы и технологии»

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студента Романова Георгий Александровича шифр 09.03.02 группа ИС-91б

1.Тема «Разработка демонстрационно-обучающей программы»  
2. Срок представления проекта к защите «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2020 г.

3. Исходные данные для проектирования: Прикладная программа должна выполнять следующие функции: предоставление возможности выбора дисциплины, раздела и темы для изучения; предоставление теоретической информации на выбранную тему; проведения контроля знаний обучающегося на заданную тему по нескольким задачам (не менее трех) и тесту; формирование отчета, отражающего перечень изученных дисциплин, результаты опросов, частный и общий вывод по уровню усвоения материала; сохранение, предварительный просмотр и печать отчета. Разработанная прикладная программа должна иметь удобный пользовательский интерфейс. Все элементы прикладной программы должны быть представлены как единый проект, управление которым осуществляется через меню пользователя.

4. Содержание пояснительной записки курсового проекта:

4.1.Введение  
4.2.Аналитическая часть

4.3.Моделирование базы данных  
4.4.Реализация базы данных

4.5.Заключение

4.6.Список использованных источников

4.7.Приложения

5. Перечень графического материала:в работе должны быть представлены пользовательские формы и формы запросов приложения, отчёты.

Руководитель работы (проекта) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.И. Лапина

подпись, дата инициалы, фамилия

Задание принял к исполнению \_,09.06.2020 Романов Г.А



подпись, дата инициалы, фамилия

РЕФЕРАТ

Курсовая работа: «Разработка демонстрационно-обучающей программы» состоит из четырех глав, введения, заключения и приложения общим объемом 50 страниц. В работе использовано 10 источников литературы.

Объектом исследования является процесс реализации обучения различным предметам.

Предметом исследования в работе являются методы и инструментальные среды разработки, отладки и тестирования программного продукта.

Целью курсовой работы является разработка программного средства для обучения различным дисциплинам. Задачами курсовой работы являются:

* Анализ подходов к реализации процесса обучения;
* Разработка алгоритма обучения различным предметам;
* Выбор инструментальной среды и разработка модели данных приложения;
* Разработка, отладка и тестирование исходных модулей приложения для обучения;
* Анализ результатов разработанного приложения.

В работе дано описание процедуры обучения через приложение. Выявлены основные проблемы в организации деятельности. Разработана модель данных и приложение для лёгкого обучения через приложение.

Приложение позволит людям легко обучится различным предметам.

СОДЕРЖАНИЕ

Введение…………………………………………………………………6

1. Выбор среды программирования и компонентов………………….7

1.1 Объектно-ориентированное программирование………………….7

1.2 Среда разработки borland c++ builder 6.0…………………………10

2. Разработка базы данных приложения……………………………....15

2.1 Понятие базы данных………………………………………………15

2.2 Характеристика баз данных………………………………………..17

2.3 Создание базы данных в microsoft access 2019…………………...17

3. Разработка приложения для обучения и тестирования…………....20

3.1 Структурная схема дерева вызова процедур и программ………..20

3.2 Обзор компонентов, используемых при разработке приложения.22

3.3 Описание программных модулей и форм…………………………23

4. Тестирование приложения…………………………………………..29

Заключение……………………………………………………………...34

Список использованных источников………………………………….34

Листинг программы…………………………………………………….35

ПЕРЕЧЕНЬ УСЛОВНЫХ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

БД – база данных

СУБД – система управления базами данных

ANSI – American National Standards Institute

ВВЕДЕНИЕ

Данная курсовая работа имеет целью разработку прикладной программы в среде Borland C++ Builder 6.0, позволяющей пользователю изучить различные предметы и проверить знания по выбранному предмету.

Во многих образовательных учреждениях сейчас проводятся тестирования по различным предметам. Поэтому создание обучающе - тестирующей программы является достаточно распространенной задачей в программировании. Цель работы - разработка прикладной программы для обучения и тестирования по различным предметам.

Для достижения поставленной цели следует решить следующие задачи:

* изучить и описать среду программирования;
* разработать приложение для обучения и тестирования по русскому языку;
* проанализировать работу разработанного приложения.

Программа будет представлять собой лишь оболочку, в которую при соблюдении некоторых принципов и наличии среды Borland C++ Builder 6.0 возможно загрузить обучающий и тестовый материал по любой дисциплине.

# **ВЫБОР СРЕДЫ ПРОГРАММИРОВАНИЯ И КОМПОНЕНТОВ**

## **Объектно-ориентированное программирование**

Объект - это абстрактная сущность, наделенная характеристиками объектов окружающего нас реального мира. Создание объектов и манипулирование ими - это вовсе не привилегия языка C++, а скорее результат методологии программирования, воплощающей в кодовых конструкциях описания объектов и операции над ними. Каждый объект программы, как и любой реальный объект, отличается собственными атрибутами и характерным поведением. Каждый класс занимает определенное место в иерархии классов, например, все часы принадлежат классу приборов измерения времени (более высокому в иерархии), а класс часов сам включает множество производных вариаций на ту же тему. Таким образом, любой класс определяет некоторую категорию объектов, а всякий объект есть экземпляр некоторого класса.

Объектно-ориентированное программирование (ООП) — это методика, которая концентрирует основное внимание программиста на связях между объектами, а не на деталях их реализации. Borland C++ Builder 6.0 является типичной системой ООП и претендует на кульминационную роль в истории его развития.

Объектно-ориентированное программирование (сокращенно ООП) — это современный подход к построению программ и систем. Любая программа Windows имеет интуитивно понятный пользовательский интерфейс, имеющий вид окна с множеством кнопок, разделов меню, окон редактирования, списков и т.п. Все это объекты. Причем сами по себе они ничего не делают. Они реагируют на различные события — нажатия пользователем клавиш или кнопок мыши, перемещения курсора и т.д. Когда происходит подобное событие, объект получает сообщение об этом и реагирует на него определенным образом: выполняет некоторые вычисления, разворачивает список, заносит символ в окно редактирования. Это и есть объектно-ориентированная программа (Приложение).

Приложение, построенное по принципам объектной ориентации — это не последовательность операторов, не некий жесткий алгоритм. Объектно-ориентрованная программа — это совокупность объектов и способов их взаимодействия. Отдельным (и главным) объектом при таком подходе во многих случаях можно считать пользователя программы. Он же служит и основным, но не единственным, источником событий, управляющих приложением.

Объект – основное понятие ООП – можно определить как некую совокупность данных и способов работы с ними. Данные можно рассматривать как поля записи. Это характеристики объекта. Пользователь и объекты программы должны, конечно, иметь возможность читать эти данные объекта, обрабатывать их и записывать в объект новые значения.

Здесь важнейшее значение имеют принципы инкапсуляции и скрытия данных. Принцип скрытия данных заключается в том, что внешним объектам и пользователю прямой доступ к данным, как правило, запрещен. Делается это из двух соображений.

Во-первых, для надежного функционирования объекта надо поддерживать целостность и непротиворечивость его данных. Если не позаботиться об этом, то внешний объект или пользователь могут занести в объект неверные данные, что приведет к его некорректному функционированию.

Во-вторых, необходимо изолировать внешние объекты от особенностей внутренней реализации данных. Для внешних потребителей данных должен быть доступен только пользовательский интерфейс — описание того, какие имеются данные и функции и как их использовать. А внутренняя реализация должна быть доступна для разработчика объекта. При таком подходе разработчик может в любой момент модернизировать объект, изменить структуру хранения и форму представления данных, но, если при этом не затронут интерфейс, внешний потребитель этого даже не заметит. И, значит, во внешней программе и в поведении пользователя ничего не придется менять.

Чтобы выдержать принцип скрытия данных, в объекте обычно определяются процедуры и функции, обеспечивающие все необходимые операции с данными: их чтение, преобразование, запись. Эти функции и процедуры называются методами и через них происходит общение с данными объекта.

Совокупность данных и методов их чтения и записи называется свойством. Свойства можно устанавливать в процессе проектирования, их можно изменять программно во время выполнения прикладной программы. Причем внешне это все выглядит так, как будто объект имеет какие-то данные, например, целые числа, которые можно прочитать, использовать в каких-то вычислениях, заложить в объект новые значения данных. В процессе проектирования приложения с помощью C++Builder пользователь может видеть значения некоторых из этих данных в окне Инспектора Объектов и изменять эти значения. В действительности общение с данными происходит через методы их чтения и записи. Это происходит и в процессе проектирования, когда среда C++Builder запускает в нужный момент эти методы, и в процессе выполнения приложения, поскольку компилятор C++Builder незримо для разработчика вставляет в нужных местах программы вызовы этих методов. Помимо методов, работающих с отдельными данными, в объекте имеются методы, работающие со всей их совокупностью, меняющие их структуру.

Средой взаимодействия объектов являются сообщения, генерируемые в результате различных событий. События наступают, прежде всего, вследствие действий пользователя — перемещения курсора мыши, нажатия кнопок мыши или клавиш клавиатуры. Но события могут наступать и в результате работы самих объектов. В каждом объекте определено множество событий, на которые он может реагировать. В конкретных экземплярах объекта могут быть заданы обработчики ряда таких событий, которые и определяют реакцию данного экземпляра объекта.

* 1. Среда разработки Borland C++ Builder 6.0

C++ Builder представляет собой SDI-приложение, главное окно которого содержит настраиваемую инструментальную панель (слева) и палитру компонентов (справа). Помимо этого, по умолчанию при запуске C++ Builder появляются окно инспектора объектов (слева) и форма нового приложения (справа). Под окном формы приложения находится окно редактора кода.

Формы являются основой приложений C++ Builder. Создание пользовательского интерфейса приложения заключается в добавлении в окно формы элементов объектов C++ Builder, называемых компонентами. Компоненты C++ Builder располагаются на палитре компонентов, выполненной в виде многостраничного блокнота. Важная особенность C++ Builder состоит в том, что он позволяет создавать собственные компоненты и настраивать палитру компонентов, а также создавать различные версии палитры компонентов для разных проектов.

Компоненты C++ Builder разделяются на видимые (визуальные) и невидимые (невизуальные). Визуальные компоненты появляются во время выполнения точно так же, как и во время проектирования. Примерами являются кнопки и редактируемые поля. Невизуальные компоненты появляются во время проектирования как пиктограммы на форме. Они никогда не видны во время выполнения, но обладают определенной функциональностью (например, обеспечивают доступ к данным, вызывают стандартные диалоги Windows и др.)

Для добавления компонента в форму можно выбрать мышью нужный компонент в палитре и щелкнуть левой клавишей мыши в нужном месте проектируемой формы. Компонент появится на форме, далее его можно перемещать, менять размеры и другие характеристики.

Каждый компонент C++ Builder имеет три разновидности характеристик: свойства, события и методы.

Если выбрать компонент из палитры и добавить его на форму, инспектор объектов автоматически покажет свойства и события, которые могут быть использованы с этим компонентом. В верхней части инспектора объектов имеется выпадающий список, позволяющий выбирать нужный из имеющихся на форме объект.

Свойства являются атрибутами компонента, определяющими его внешний вид и поведение. Многие свойства компонента в колонке свойств имеют значение, устанавливаемое по умолчанию (например, высота кнопок). Свойства компонента отображаются на странице свойств (Properties). Инспектор объектов отображает общие (public) свойства компонентов. Помимо publiс-свойств, компоненты могут и чаще всего имеют опубликованные (published) свойства, которые доступны только во время выполнения приложения. Инспектор объектов используется для установки свойств во время проектирования. Список свойств располагается на странице свойств инспектора объектов. Можно определить свойства во время проектирования или написать код для видоизменения свойств компонента во время выполнения приложения.

При определении свойств компонента во время проектирования нужно выбрать компонент на форме, открыть страницу свойств в инспекторе объектов, выбрать определяемое свойство и изменить его с помощью редактора свойств (это может быть простое поле для ввода текста или числа, выпадающий список, раскрывающийся список, диалоговая панель и т.д.).

Страница событий (Events) инспектора объектов показывает список событий, распознаваемых компонентом (программирование для операционных систем с графическим пользовательским интерфейсом, в частности, для Windows 95 или Windows NT предполагает описание реакции приложения на те или иные события, а сама операционная система занимается постоянным опросом компьютера с целью выявления наступления какого-либо события). Каждый компонент имеет свой собственный набор обработчиков событий. В C++ Builder следует писать функции, называемые обработчиками событий, и связывать события с этими функциями. Создавая обработчик того или иного события, разработчик поручает программе выполнить написанную функцию, если это событие произойдет.

Для того чтобы добавить обработчик событий, нужно выбрать на форме с помощью мыши компонент, которому необходим обработчик событий, затем открыть страницу событий инспектора объектов и дважды щелкнуть левой клавишей мыши на колонке значений рядом с событием, чтобы заставить C++ Builder сгенерировать прототип обработчика событий и показать его в редакторе кода. При этом автоматически генерируется текст пустой функции, и редактор открывается в том месте, где следует вводить код. Курсор позиционируется внутри операторных скобок { ... }. Далее нужно ввести код, который должен выполняться при наступлении события. Обработчик событий может иметь параметры, которые указываются после имени функции в круглых скобках.

Отметим, что при создании формы связанные с ней модуль и заголовочный файл с расширением \*.h генерируются обязательно, тогда как при создании нового модуля он не обязан быть связан с формой (например, если в нем содержатся процедуры расчетов). Имена формы и модуля можно изменить, причем желательно сделать это сразу после создания, пока на них не появилось много ссылок в других формах и модулях.

Файлы, образующие приложение – формы и модули – собраны в проект. Менеджер проектов показывает списки файлов и модулей приложения и позволяет осуществлять навигацию между ними. Можно вызвать менеджер проектов, выбрав пункт меню View/Project Manager. По умолчанию вновь созданный проект получает имя Project1.cpp.

По умолчанию проект первоначально содержит файлы для одной формы и исходного кода одного модуля. Однако большинство проектов содержат несколько форм и модулей. Чтобы добавить модуль или форму к проекту, нужно щелкнуть правой кнопкой мыши и выбрать пункт New Form из контекстного меню. Можно также добавлять существующие формы и модули к проекту, используя кнопку Add контекстного меню менеджера проектов и выбирая модуль или форму, которую нужно добавить. Формы и модули можно удалить в любой момент в течение разработки проекта. Однако, из-за того, что формы всегда связаны с модулем, нельзя удалить одно без удаления другого, за исключением случая, когда модуль не имеет связи с формой. Удалить модуль из проекта можно, используя кнопку Remove менеджера проектов.

Если выбрать кнопку Options в менеджере проектов, откроется диалоговая панель опций проекта, в которой можно выбрать главную форму приложения, определить, какие формы будут создаваться динамически, каковы параметры компиляции модулей (в том числе созданных в Delphi, так как C++ Builder может включать их в проекты) и компоновки.

Важным элементом среды разработки C++ Builder является контекстное меню, появляющееся при нажатии на правую клавишу мыши и предлагающее быстрый доступ к наиболее часто используемым командам.

C++ Builder обладает встроенной системой контекстно-зависимой помощи, доступной для любого элемента интерфейса и являющейся обширным источником справочной информации о C++ Builder.

Первым шагом в разработке приложения C++ Builder является создание проекта. Файлы проекта содержат сгенерированный автоматически исходный текст, который становится частью приложения, когда оно скомпилировано и подготовлено к выполнению. Чтобы создать новый проект, нужно выбрать пункт меню File/New Application.

C++ Builder создает файл проекта с именем по умолчанию Project1.cpp, а также make-файл с именем по умолчанию Project1.mak. При внесении изменений в проект, таких, как добавление новой формы, C++ Builder обновляет файл проекта.

Проект или приложение обычно имеют несколько форм. Добавление формы к проекту создает следующие дополнительные файлы:

* Файл формы с расширением .DFM, содержащий информацию о ресурсах окон для конструирования формы
* Файл модуля с расширением .CPP, содержащий код на C++.
* Заголовочный файл с расширением .H, содержащий описание класса формы.

Когда мы добавляем новую форму, файл проекта автоматически обновляется.

Для того чтобы добавить одну или более форм к проекту, выбирается пункт меню File/New Form. Появится пустая форма, которая будет добавлена к проекту. Можно воспользоваться пунктом меню File/New, выбрать страницу Forms и выбрать.

Для того чтобы просто откомпилировать текущий проект, из меню Compile нужно выбрать пункт меню Compile. Для того чтобы откомпилировать проект и создать исполняемый файл для текущего проекта, из меню Run нужно выбрать пункт меню Run. Компоновка проекта является инкрементной (перекомпилируются только изменившиеся модули).

Если при выполнении приложения возникает ошибка времени выполнения, C++ Builder делает паузу в выполнении программы и показывает редактор кода с курсором, установленным на операторе, являющемся источником ошибки. Прежде чем делать необходимую коррекцию, следует перезапустить приложение, выбирая пункт меню Run из контекстного меню или из меню Run, закрыть приложение и лишь затем вносить изменения в проект. В этом случае уменьшится вероятность потери ресурсов Windows.

**2.Разработка базы данных приложения**

Базы данных всегда были важнейшей темой при изучении информационных систем. В последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Цель базы данных — помочь людям и организациям вести учет определенных вещей. Базы данных используются во многих предприятиях, отраслях науки и т.д. Разработано множество готовых языков программирования, с помощью которых можно создавать базы данных.

Целью данной курсовой работы является создание программы, которая будет вести учет реализации товаров в компьютерном салоне. При использовании этой программы исключается необходимость поиска в ручную большого количества различной бумажной документации. Программа обеспечивает удобство хранения и быстрое нахождение информации, как о различных товарах, так и об их реализации. Также она позволяет производить поиск товаров на складе по базе данных, выдавать информацию о товаре (наименование товара, его производитель, цена, дата продажи и т.д.) или салоне (название, адрес, контактный телефон, количество проданных товаров и т.д.).

Но прежде чем приступить к разработке программы в приложении Borland C++ Builder, необходимо сначала создать базу данных.

2.1 Понятие базы данных

База данных (БД) - именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области. Также существует множество других определений базы данных, однако общепризнанная единая формулировка отсутствует.

2.2 Характеристика баз данных

Наиболее часто при характеристике баз данных используются следующие отличительные признаки: база данных хранится и обрабатывается в вычислительной системе. Таким образом, любые вне компьютерные хранилища информации (архивы, библиотеки, картотеки и т. п.) базами данных не являются. Данные в базе данных логически структурированы (систематизированы) с целью обеспечения возможности их эффективного поиска и обработки в вычислительной системе. Базы данных необходимы для хранения и фильтрации информации, для её обработки и обмена.

Для успешной работы с БД необходимы программные средства, при помощи которых можно было бы создавать и управлять БД. Для этого существуют Системы Управления Базами Данных (СУБД).

СУБД – совокупность языковых и программных средств, обеспечивающих создание, использование и ведение БД.

Существует два вида СУБД: локальные и сетевые. Локальные – это СУБД, работающие на одном компьютере. Наиболее популярной СУБД этого типа является Microsoft Office Access (использовалась версия 2019 года). Сетевые – это СУБД, позволяющие нескольким компьютерам использовать одну и ту же БД с помощью технологии клиент-сервер.

3.3 Основные типы взаимосвязи данных

Существует 4 типа взаимосвязи данных:

1) Один к одному

2) Один ко многим

3) Много к одному

4) Много ко многим

Вид взаимосвязи один к одному означает, что каждая запись одного объекта БД будет указывать на единственную запись другого объекта.

Один ко многим означает, что одной записи объекта БД будет соответствовать несколько записей других объектов. Много к одному означает, что нескольким записям объектов БД будет соответствовать одна запись другого объекта. Много ко многим устанавливается между двумя типами объектов БД.

2.3 Создание базы данных в Microsoft Access 2019

Базы данных существуют для того, чтобы большое количество данных можно было сортировать и без проблем искать нужную информацию. Под базой данных принято понимать объективную форму представления и организации совокупности данных (статей, расчетов и так далее), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью электронной вычислительной машины. В базе данных вся информация собрана в виде таблиц. Каждый столбец имеет имя, не повторяющееся в этой таблице. Строки следуют в произвольном порядке, и не имеют номеров. Поиск строк производится не по номерам, а по идентификаторам (ключам). Строки таблицы обязательно отличаются друг от друга хотя бы единственным значением, что позволяет однозначно идентифицировать любую строку такой таблицы. Столбцам таблицы однозначно присваиваются имена, и в каждом из них размещаются однородные значения данных.

При работе с таблицей ее строки и столбцы можно обрабатывать в любом порядке, т.к. у них есть уникальные имена, а также возможность выделения любой их строки или любого набора строк с указанными признаками. Для поиска и обработки информации служат запросы, а готовый результат выводится в виде отчетов.

Таблицы баз данных, как правило, допускают работу с гораздо большим количеством разных типов данных. Изменение типа данных поля таблицы производится в режиме «Конструктор таблиц». Для создания таблиц этой базы данных применяются следующие типы данных:

Текстовый – тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).

Числовой – тип данных для хранения действительных чисел.

Счетчик – специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование – для порядковой нумерации записей.

Строки таблиц и их типы показаны в Таблице 1:

Таблица 1- поля таблиц базы данных и их характеристики

|  |  |  |
| --- | --- | --- |
| Обозначение объекта | Ограничение | Тип данных |
| Kod\_pred | Совпадения не допускаются | Текстовый |
| Predmet |  | Числовой |
| Tema |  | Текстовый |
| Kod\_tema | Совпадения не допускаются | Текстовый |
| Razdel |  | Текстовый |
| Test |  | Текстовый |
| Kod | Совпадения не допускаются | Текстовый |
| Praktika |  | Текстовый |
| Otvet |  | Текстовый |
| Teoria |  | Текстовый |

В Access первоначально требуется создать таблицы, которые будут составлять базу данных. В режиме конструктора было создано три таблицы: Предмет, Тема, Раздел. Полям таблицы были заданы определённые форматы, что способствует контролю целостности данных.

Таблицы базы данных:

Таблица 2- состав таблицы «Предмет» базы данных предметной области

| Kod\_predmet | Predmet | Tema |
| --- | --- | --- |
| 1.0.0 | Математика | Неравенства и системы неравенств. Системы уравнений |
| 2.0.0 | Физика | Тепловые явления. Изменение состояния вещества |

Таблица 3- состав таблицы «Раздел» базы данных предметной области

| Razdel | | | | |
| --- | --- | --- | --- | --- |
| Kod | Razdel | Praktika | Otvet | Teoria |
| 1.1.1 | Линейные и квадратные неравенства | практика | Ответ к заданиям | теория |
| 1.1.2 | Рациональные неравенства | практика | Ответ к заданиям | теория |
| 1.2.1 | Методы решения систем уравнений | практика | Ответ к заданиям | теория |
| 1.2.2 | Основные понятия | практика | Ответ к заданиям | теория |
| 2.1.1 | количество теплоты | практика | Ответ к заданиям | теория |
| 2.1.2 | удельная теплоёмкость вещества | практика | Ответ к заданиям | теория |
| 2.2.1 | Кипение. Температура кипения | практика | Ответ к заданиям | теория |
| 2.2.2 | Удельная теплота плавления | практика | Ответ к заданиям | теория |

Таблица 4- состав таблицы «Тема» базы данных предметной области

| Tema | | | |
| --- | --- | --- | --- |
| Kod\_tema | Tema | Razdel | test |
| 1.1.0 | Неравенства и системы неравенств | Линейные и квадратные неравенства | Задания для теста |
| 1.2.0 | Системы уравнений | Методы решения систем уравнений | Задания для теста |
| 2.1.0 | Тепловые явления | Кипение. Температура кипения | Задания для теста |
| 2.2.0 | Изменение состояния вещества | Методы решения систем уравнений | Задания для теста |

Далее в схеме данных было проведено связывание этих таблиц (использовался тип соотношения один-ко-многим). Связывание проводилось по полям, помеченным ключом (рис.1):

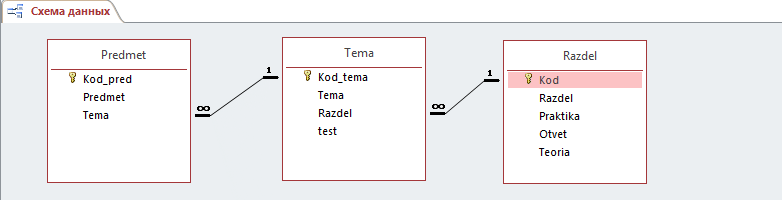


Рис.1 - Схема данных предметной области

После создания связей, работа над базой данных может быть закончена. Для сохранения базы данных используется формат «база данных Access 2007-2016» с расширением .accdb.

3. Разработка приложения для обучения и тестирования

3.1 Структурная схема дерева вызова процедур и программ

Дерево рабочих форм приложения представлено на рисунке 2

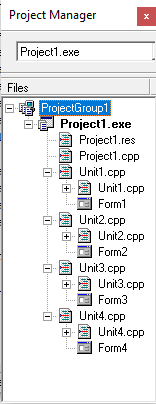


Рисунок 2 - Дерево форм приложения

Проект C++Builder состоит из форм, модулей с их заголовочными файлами и файлами реализации, установок параметров проекта, ресурсов и т.д. Вся эта информация размещается в файлах. Многие из этих файлов автоматически создаются C++Builder, когда мы строем приложение. Ресурсы, такие, как битовые матрицы, пиктограммы и т.д., находятся в файлах, которые мы получаем из других источников или создаете при помощи многочисленных инструментов и редакторов ресурсов. Кроме того, компилятор также создает файлы.

Расширение файла и формат исполняемого файла - exe, применяется в системах DOS, Microsoft Windows. Exe - это исполняемый файл приложения. Представляет из себя специальный самораспаковывающий файл, не требующий никакой программы для открытия. То есть, чтобы запустить файл в таком формате, не надо будет никакой специальной программы для чтения.

Cpp - это препроцессор языка C, который вызывается в качестве первого прохода при любой компиляции C-программ командой cc (cc - компилятор языка C). Формат результатов работы cpp позволяет использовать их в качестве исходных данных для следующего прохода C-компилятора. Поскольку язык C развивается, cpp и остальные команды пакета C-компиляции будут модифицироваться, чтобы следовать изменениям языка. Поэтому использование команды cpp, минуя cc, чревато неприятностями, так как cpp может заработать совсем иначе.

При вызове cpp в качестве аргументов могут быть указаны два имени файла - входной\_файл и выходной\_файл препроцессора. По умолчанию вместо них используются стандартный ввод и стандартный вывод.

Файл ресурсов проекта (.res). Двоичный файл, содержащий ресурсы проекта: пиктограммы, курсоры и т.п. По умолчанию содержит только пиктограмму проекта. Может дополняться с помощью Редактора Изображений (Image Editor).

Заголовочный файл модуля (.h). Каждой созданной форме соответствует не только файл реализации модуля, но и его заголовочный файл с описанием класса формы. Можно самостоятельно создавать необходимые заголовочные файлы.

## **3.2 Обзор компонентов, используемых при разработке приложения**

Рассмотрим каждый из используемых компонентов отдельно.

1. Pagecontrol - многостраничный блокнот.
2. Label – метка, предназначенная для отображения текста.
3. Button – управляющие кнопки.
4. BitBtn – управляющая кнопка. У BitBtn в отличие от кнопки Button становится возможным использование пиктограммы.
5. DBRichEdit **–** окно редактирования многострочного текста. Это RichEdit, только работает с БД.
6. DBGrid - используется для просмотра и редактирования БД в режиме таблицы.
7. ADOTable - представляет собой таблицу БД. Этот компонент невидим при просмотре формы.
8. DataSource - компонент обеспечивает связь между данными, представленными компонентом ADOTable и компонентами отображения данных (DBEdit и DBGrid).
9. DBNavigator - обеспечивает перемещение указателя текущей записи, активацию режима редактирования, добавление и удаление записей.
10. ADOConnection - обеспечивает соединение других компонентов ADO приложения с БД и осуществляет управление транзакциями.
11. ADOQuery - обеспечивает применение запросов SQL при работе с данными через ADO.
12. DBLabel – обычный компонент label, только работает с БД.
13. Edit - окно для ввода данных в одну строку.
14. DBText - Аналог обычной метки TLabel, связанный с данными.
15. RichEdit - много строчное окно редактирования текстов в обогащенном формате .rtf, позволяющее производить выбор цвета, шрифта, поиск текста и т.д.

## **3.3 Описание программных модулей и форм**

При запуске программы открывается главная страница, в которой осуществляется выбор «Начать учится», посмотреть информацию «О

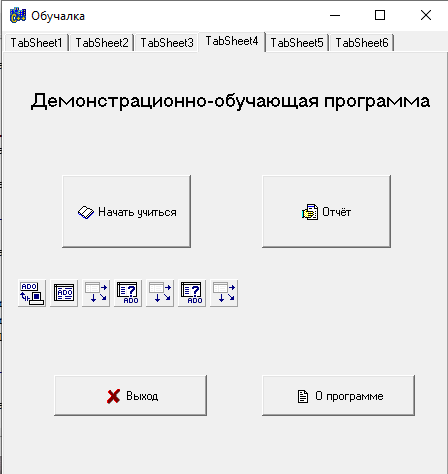
программе», создать «Отчёт» или «Выход».

Рис.3 – главная страница.

Для плавного перехода между страницами TabSheet с компонентами используется компонент Pagecontol.

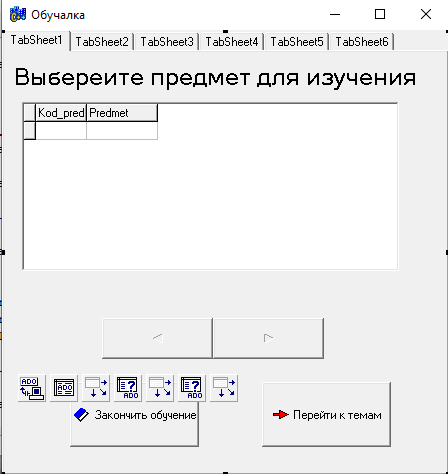


Рис.4 - Страница Tabsheet1.

На этой странице можно выбрать предмет для изучения. Выбор предмета предоставляется с помощью компонента DBNavigator. Для возвращения на начальную страницу и перехода к темам используется компонент BitButton. Для соединения таблицы с БД используется компонент ADOTable.

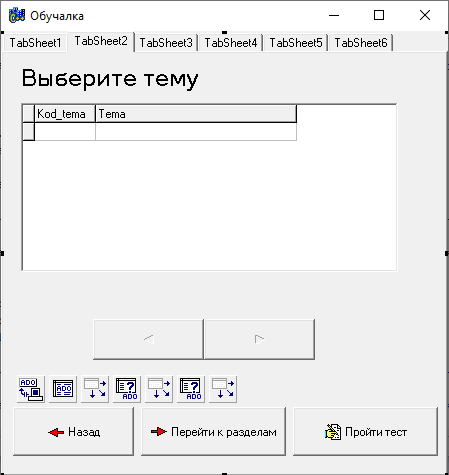


Рис.5 – Страница TabSheet2.

На этой странице можно выбрать тему для выбранного предмета для изучения или пройти тест по выбранной теме. Для соеденения таблицы с БД, используется компонент ADOQuery. Для фильтрации данных используются запросы MySql.

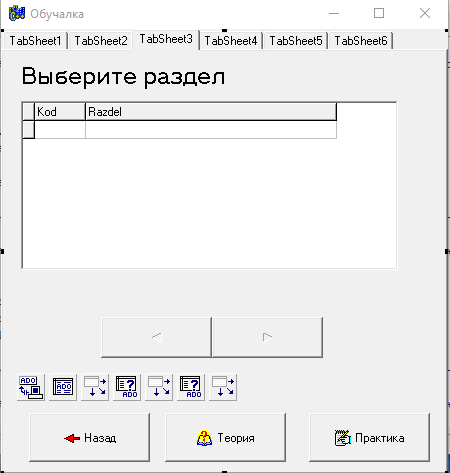


Рис.6 – TabSheet3.

На этой странице можно прочитать теорию или проверить свои знания по выбранному разделу

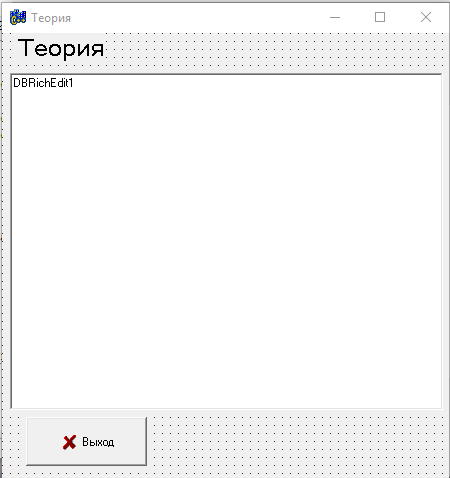
.

Рис.7 – Form2

На этой форме располагается компонент DBRichEdit для отображения теории по выбранному разделу.

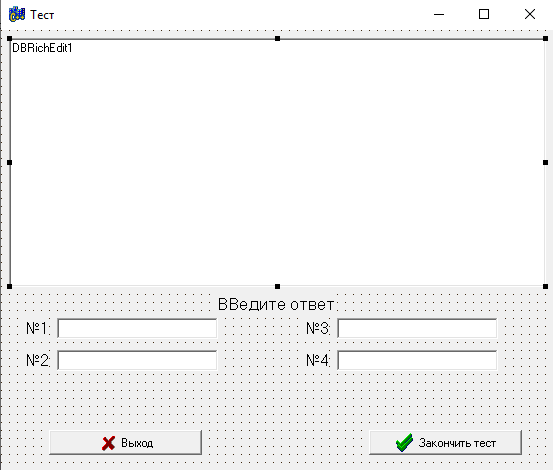


Рис.8 – Form4.

На этой форме производится тестирование знаний по всей выбранной теме.

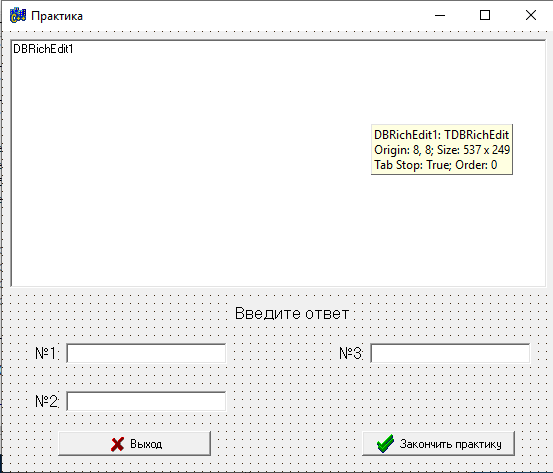


Рис.9 – Form5.

На этой форме производится проверка знаний по выбранной теме.

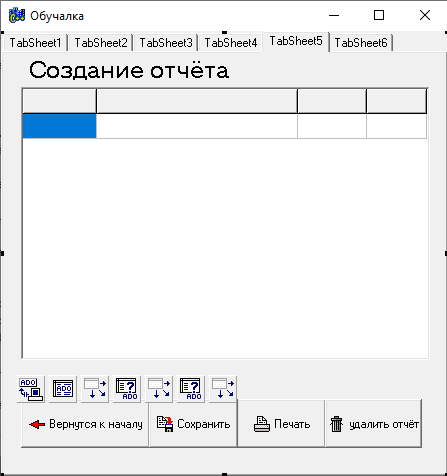


Рис.10 – TabSheet5.

На этой странице производится сохранение отчёта в текстовый файл, печать, и удаление отчёта. Для просмотра отчёта используется компонент StringGrid.

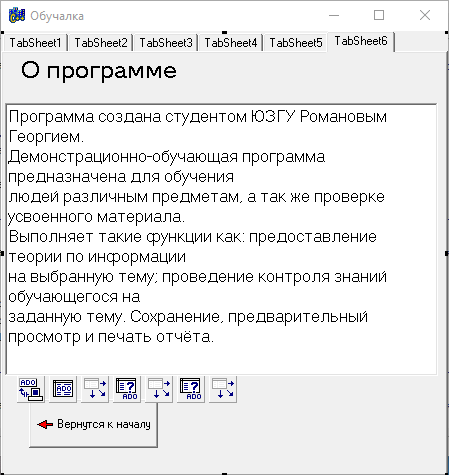


Рис.11 – TabSheet6.

На этой странице описывается информация о программе. Для отображения текста используется компонент RichEdit.

4. Тестирование приложения

Переходим к тестированию приложения. При запуске приложения открывается стартовая форма.

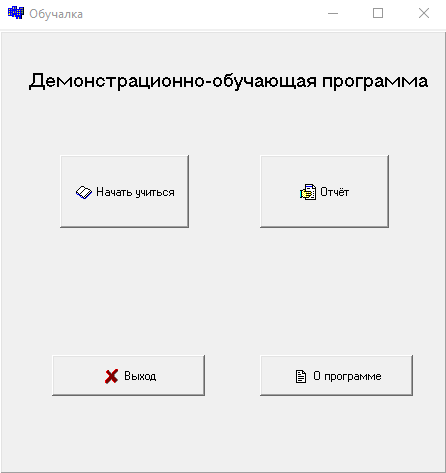


Рис.12 – Главная страница.

При нажатии кнопки «О программе» открывается страница с описанием.

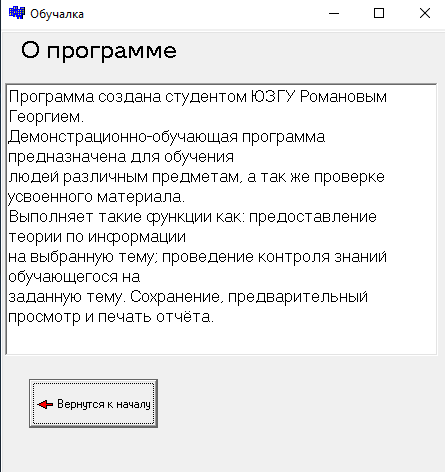


Рис.13 – TabSheet6.

При нажатии кнопки «начать учится» на стартовой форме открывается страница с выбором предмета.

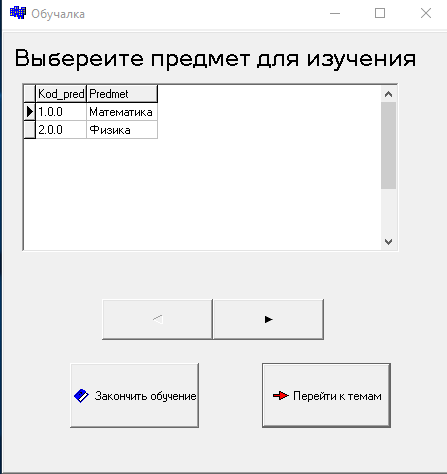


Рис.14 – TabSheet1.

Переходим к темам.

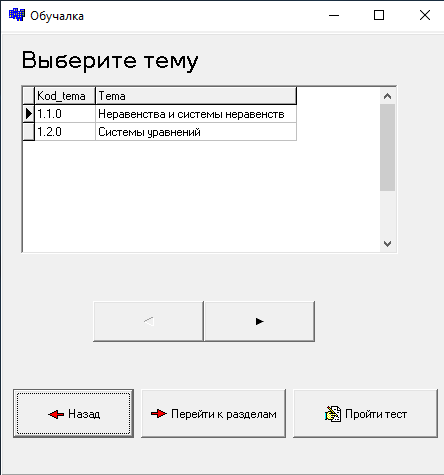


Рис.15 - TabSheet2.

Попробуем пройти тест.

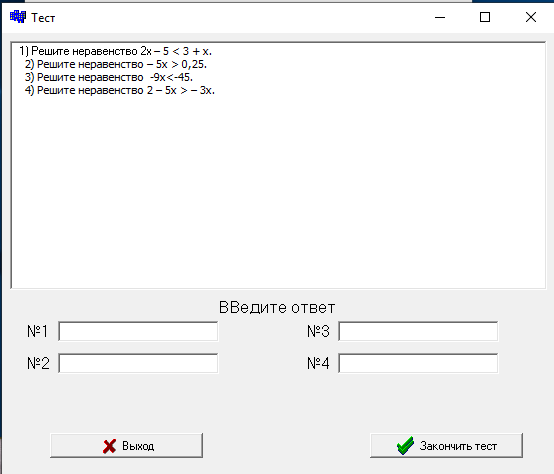


Рис.16 - Form4.

После прохождения теста перейдём к разделам

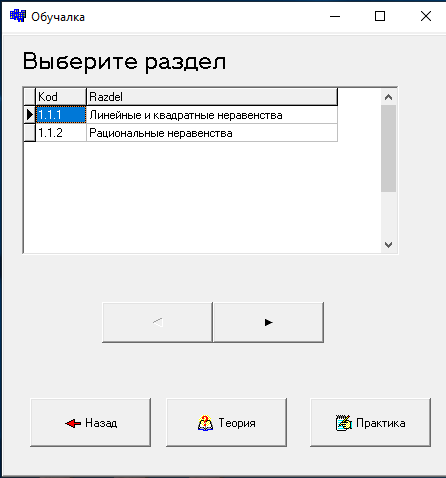


Рис.17-TabSheet3.

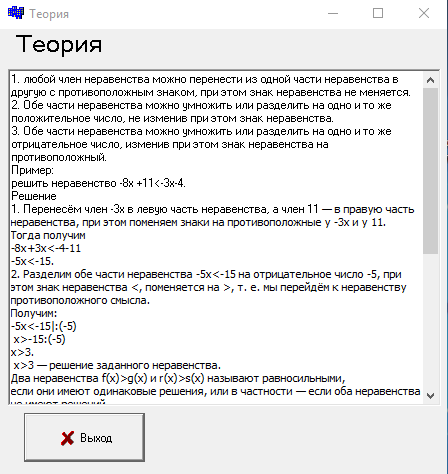
Попробуем прочитать теорию и сделать практику.

Рис.18 – Form2.

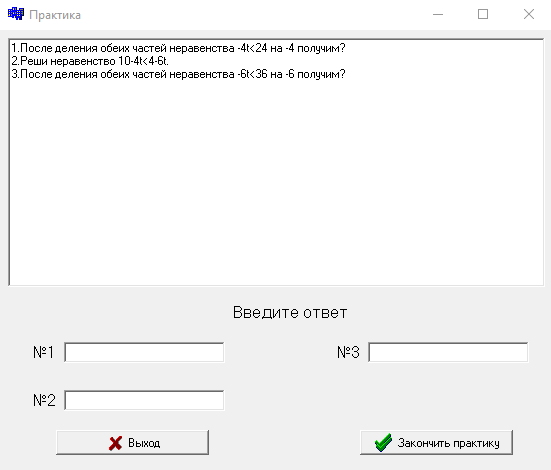


Рис.19 – Form5.

После прохождения практики, создадим отчёт.

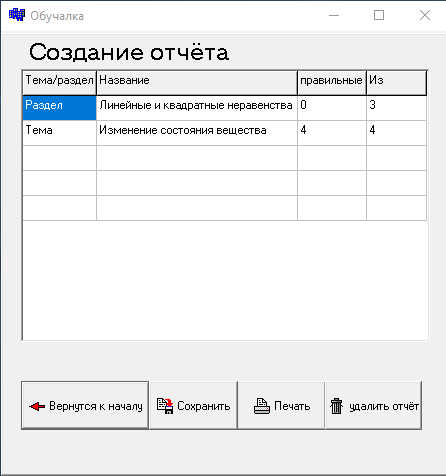


Рис.20 – TabSheet5.

Заключение.

Данная работа выполнена на языке высокого уровня программирования в среде Borland С++Builder. При написании и разработке программного обеспечения использовались различные возможности С++.

Базы данных в наше время имеют особенно важное значение, и иногда их использование просто необходимо. Все их плюсы можно увидеть на примере данной программы. Благодаря этому приложению, можно значительно упростить процесс обучения различным предметам.

В ходе выполнения курсовой работы была разработана база данных различных дисциплин, содержащая три таблицы (Предмет, Тема, Разделы взаимосвязанные между собой). С помощью этой базы данных можно добавить данные о предметах, темах предметов и их разделов.

Разработанная прикладная программа имеет удобный пользовательский интерфейс. Все элементы прикладной программы представлены как единый проект, управление которым осуществляется через окно главной формы проекта. Программа работает без ошибок, выполняет все функции.

**Список использованных источников**

1. Лапина Т.И, Петрик Е.А. Методы и технологии программирования: учебное пособие. Курск.: Изд-во ЮЗГУ, 2011. 211с.
2. Лапина Т.И. Управление данными. Курск.: Изд-во ЮЗГУ, 2011. 360с.
3. Электронный ресурс офисного программного обеспечения / (<http://www.interface.ru/home.asp?artId=8028> - дата обращения: 10.06.2013)
4. Электронный ресурс разработчиков программы Microsoft Office Access 2007/ (<http://office.microsoft.com/ru-ru/access-help/HA010341576.aspx>- дата обращения: 9.06.2013)

**Листинг программы**

**Unit1**

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include "Unit4.h"

#include "Unit5.h"

#include <iostream>

#include <fstream>

using namespace std;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

String b;

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn1Click(TObject \*Sender)//закрыть прогрмму

{

Close();

ofstream fan ("temp.txt");

if (!fan.is\_open()) // если файл не открыт

ShowMessage("не удалось закрыть приложение");

else

fan.close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormShow(TObject \*Sender)//скрытие ярлычков и открытие бд

{

for (int i=0; i < PageControl1->PageCount; ++i)

PageControl1->Pages[i]->TabVisible = false;

PageControl1->ActivePage = TabSheet4;

ADOTable1->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn2Click(TObject \*Sender)// переход к темам

{

PageControl1->ActivePage = TabSheet2;

b = DataSource2->DataSet->FieldByName("Predmet")->Value;

ADOQuery1->Close();

ADOQuery1->SQL->Clear();

if (b=="Математика")

ADOQuery1->SQL->Text ="SELECT \* FROM Tema WHERE Kod\_tema LIKE '1%'";

if (b=="Физика")

ADOQuery1->SQL->Text ="SELECT \* FROM Tema WHERE Kod\_tema LIKE '2%'";

ADOTable1->Close();

ADOQuery1->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn3Click(TObject \*Sender)// назад к предметам

{

PageControl1->ActivePage = TabSheet1;

ADOQuery1->Close();

ADOTable1->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn4Click(TObject \*Sender) //перейти к разделам

{

PageControl1->ActivePage = TabSheet3;

b = DataSource1->DataSet->FieldByName("Kod\_tema")->Value;

ADOQuery2->Close();

ADOQuery2->SQL->Clear();

if (b=="1.1.0")

ADOQuery2->SQL->Text ="SELECT \* FROM Razdel WHERE Kod LIKE '1.1%'";

if (b=="1.2.0")

ADOQuery2->SQL->Text ="SELECT \* FROM Razdel WHERE Kod LIKE '1.2.%'";

if (b=="2.1.0")

ADOQuery2->SQL->Text ="SELECT \* FROM Razdel WHERE Kod LIKE '2.1.%'";

if (b=="2.2.0")

ADOQuery2->SQL->Text ="SELECT \* FROM Razdel WHERE Kod LIKE '2.2.%'";

ADOQuery2->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn8Click(TObject \*Sender)// назад к темам

{

ADOQuery2->Close();

PageControl1->ActivePage = TabSheet2;

ADOQuery1->Close();

ADOQuery1->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn7Click(TObject \*Sender)//практика

{

Form5->Show();

Form5->DBRichEdit1->DataField = "praktika";

b = DataSource4->DataSet->FieldByName("Kod")->Value;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn5Click(TObject \*Sender)// тест

{

Form4->Show();

Form4->DBRichEdit1->DataField = "test";

b = DataSource1->DataSet->FieldByName("Kod\_tema")->Value;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn9Click(TObject \*Sender)// начать обучение

{

PageControl1->ActivePage = TabSheet1;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn11Click(TObject \*Sender)// закончить обучение

{

PageControl1->ActivePage = TabSheet4;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn10Click(TObject \*Sender)// отчёт

{

PageControl1->ActivePage = TabSheet5;

StringGrid1->Cells[0][0]="Тема/раздел";

StringGrid1->Cells[1][0]="Название";

StringGrid1->Cells[2][0]="правильные";

StringGrid1->Cells[3][0]="Из";

ifstream fin("temp.txt");

if (!fin.is\_open()) // если файл не открыт

ShowMessage("файл не открыт");

else

{

int i=1;

string str = "";

while (!fin.eof())

{

StringGrid1->RowCount= StringGrid1->RowCount+1;

for (int f=0;f<4;f++)

{

getline(fin,str);

StringGrid1->Cells[f][i]= str.c\_str();

}

i++;

}

}

fin.close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn13Click(TObject \*Sender)// вернуться к началу

{

PageControl1->ActivePage = TabSheet4;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn14Click(TObject \*Sender)//к началу из о программе

{

PageControl1->ActivePage = TabSheet4;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn12Click(TObject \*Sender)//о программе

{

PageControl1->ActivePage = TabSheet6;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn15Click(TObject \*Sender)//печать отчёта

{

TPrinter \*printer = Printer();

printer->BeginDoc();

printer->Canvas->TextOut(100,100,"Изученные темы и разделы");

printer->Canvas->TextOut(100,200,"Тема/раздел");

printer->Canvas->TextOut(600,200,"Название");

printer->Canvas->TextOut(1900,200,"Правильные");

printer->Canvas->TextOut(2450,200,"Из");

for(int i=1; i<StringGrid1->RowCount-2; i++)

{

printer->Canvas->TextOut(100,200\*i+300,StringGrid1->Cells[0][i]);

printer->Canvas->TextOut(500,200\*i+300,StringGrid1->Cells[1][i]);

printer->Canvas->TextOut(2000,200\*i+300,StringGrid1->Cells[2][i]);

printer->Canvas->TextOut(2500,200\*i+300,StringGrid1->Cells[3][i]);

}

printer->EndDoc();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn17Click(TObject \*Sender)// очищение отчёта

{

ofstream fan ("resultat.txt");

if (!fan.is\_open()) // если файл не открыт

ShowMessage("не удалось очистить");

else

fan.close();

ShowMessage("Данные удалены.");

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn16Click(TObject \*Sender) //сохранение отчёта

{

ofstream AllOrder("resultat.txt", ios::app);

for(int i = 0; i < StringGrid1->RowCount; i++)

{

StringGrid1->Rows[i]->Delimiter = ' ';

String s = StringGrid1->Rows[i]->DelimitedText;

std::string st(AnsiString(s).c\_str());

AllOrder << st << endl;

}

AllOrder.close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BitBtn6Click(TObject \*Sender)//теория

{

Form2->Show();

Form2->DBRichEdit1->DataField = "Teoria";

}

**Unit2**

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

//---------------------------------------------------------------------------

\_\_fastcall TForm2::TForm2(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::BitBtn1Click(TObject \*Sender)

{

Close();

}

//---------------------------------------------------------------------------

**Unit4**

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit4.h"

#include <iostream>

#include <fstream>

using namespace std;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm4 \*Form4;

extern String b;

//---------------------------------------------------------------------------

\_\_fastcall TForm4::TForm4(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::BitBtn1Click(TObject \*Sender)//çàêðûòü òåñò

{

Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::BitBtn2Click(TObject \*Sender)//ïðîâåðêà òåñòà

{

String z;

int t=0;

z = Form1->DataSource1->DataSet->FieldByName("Tema")->Value;

if (b=="1.1.0")

{

if(Edit1->Text == "x<8")

t=t+1;

if(Edit2->Text == "x<-0,05")

t=t+1;

if(Edit3->Text == "x>5")

t=t+1;

if(Edit4->Text == "x<1")

t=t+1;

}

if (b=="1.2.0")

{

if(Edit1->Text == "2")

t=t+1;

if(Edit2->Text == "2")

t=t+1;

if(Edit3->Text == "4")

t=t+1;

if(Edit4->Text == "4")

t=t+1;

}

if (b=="2.1.0")

{

if(Edit1->Text == "1")

t=t+1;

if(Edit2->Text == "2")

t=t+1;

if(Edit3->Text == "3")

t=t+1;

if(Edit4->Text == "4")

t=t+1;

}

if (b=="2.2.0")

{

if(Edit1->Text == "1")

t=t+1;

if(Edit2->Text == "2")

t=t+1;

if(Edit3->Text == "3")

t=t+1;

if(Edit4->Text == "4")

t=t+1;

}

ShowMessage("ïðàâèëüíî " + (IntToStr(t)) + " èç 4");

ofstream fan ("temp.txt",ios::app);

if (!fan.is\_open()) // åñëè ôàéë íå îòêðûò

ShowMessage("Ôàéë íå îòêðûò");

else

{

fan <<"Òåìà" <<endl;

fan <<z.c\_str()<<endl;

fan <<t<< endl;

fan <<"4"<<endl;

}

fan.close();

t=0;

Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::FormShow(TObject \*Sender)

{

for(int i=0; i<ComponentCount; i++)

{

if(Components[i]->ClassNameIs("TEdit"))

((TEdit\*)Components[i])->Text = "";

}

}

**Unit5**

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit5.h"

#include <iostream>

#include <fstream>

using namespace std;

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm5 \*Form5;

extern String b;

//---------------------------------------------------------------------------

\_\_fastcall TForm5::TForm5(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::BitBtn1Click(TObject \*Sender)

{

Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::BitBtn2Click(TObject \*Sender)

{

String z;

int t=0;

z = Form1->DataSource4->DataSet->FieldByName("Razdel")->Value;

if (b=="1.1.1")

{

if(Edit1->Text == "t<-6")

t=t+1;

if(Edit2->Text == "t>-3")

t=t+1;

if(Edit3->Text == "t<-6")

t=t+1;

}

if (b=="1.1.2")

{

if(Edit1->Text == "-2;6")

t=t+1;

if(Edit2->Text == "3")

t=t+1;

if(Edit3->Text == "-1")

t=t+1;

}

if (b=="1.2.1")

{

if(Edit1->Text == "-14")

t=t+1;

if(Edit2->Text == "2")

t=t+1;

if(Edit3->Text == "8")

t=t+1;

}

if (b=="1.2.2")

{

if(Edit1->Text == "5")

t=t+1;

if(Edit2->Text == "-2")

t=t+1;

if(Edit3->Text == "-1")

t=t+1;

}

if (b=="2.1.1")

{

if(Edit1->Text == "126")

t=t+1;

if(Edit2->Text == "325")

t=t+1;

if(Edit3->Text == "196,2")

t=t+1;}

if (b=="2.1.2")

{if(Edit1->Text == "326")

t=t+1;

if(Edit2->Text == "263")

t=t+1;

if(Edit3->Text == "999")

t=t+1;}

if (b=="2.2.1")

{if(Edit1->Text == "476")

t=t+1;

if(Edit2->Text == "911")

t=t+1;

if(Edit3->Text == "25")

t=t+1;}

if (b=="2.2.2")

{if(Edit1->Text == "111")

t=t+1;

if(Edit2->Text == "27")

t=t+1;

if(Edit3->Text == "837")

t=t+1;}

ShowMessage("ïðàâèëüíî " + (IntToStr(t)) + " èç 3");

ofstream fan ("temp.txt",ios::app);

if (!fan.is\_open()) // åñëè ôàéë íå îòêðûò

ShowMessage("Ôàéë íå îòêðûò");

else

{fan <<"Ðàçäåë" <<endl;

fan <<z.c\_str()<<endl;

fan <<t<< endl;

fan <<"3"<<endl;}

fan.close();

t=0;

Close();}