**Министерство по развитию информационных технологий и телекоммуникаций**

**Ташкентский университет информационных технологии**

**Кафедра «Информатика»**

**Проектная работа №1**

*на тему: «Автоматизированная информационная*

*система Аптека»*

Выполнилистуденты группы 225-15:

**Галиуллин Т.**

**Ильхомжонов Д.**

**Уктамов Н.**

Проверили :

**Шихназарова Г.**

**Талипова О**.

Ташкент 2016

**Содержание**

**Введение**

**1. Теоретическая часть**

**1.1. Понятия и определения БД**

**1.2. MicrosoftAccess**

**1.3. СредаC++ Builder**

**2. Создание программы**

**2.1. Руководство программисту**

**2.2. Руководство пользователю**

**3. Заключение**

**4. Литература и Источники**

**5. Приложение**

**5.1. Код программы**

**Введение**

**Понятие информационных систем**

Под системой понимают любой объект, который одновременно рассматривается и как единое целое, и как объединенная в интересах достижения поставленных целей совокупность разнородных элементов. Системы значительно отличаются между собой как по составу, так и по главным целям.

В информатике понятие «система» широко распространено и имеет множество смысловых значений. Чаще всего оно используется применительно к набору технических средств и программ. Системой может называться аппаратная часть компьютера.

Системой может также считаться множество программ для решения конкретных прикладных задач, дополненных процедурами ведения документации и управления расчетами. Добавление к понятию «система» слова «информационная» отражает цель ее создания и функционирования.

Информационные системы обеспечивают сбор, хранение, обработку, поиск, выдачу информации, необходимой в процессе принятия решений задач из любой области. Они помогают анализировать проблемы и создавать новые продукты.

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Информационная система представляет собой хранилище информации, снабженное процедурами ввода, поиска, размещения и выдачи информации. Наличие таких процедур – главная особенность информационных систем, отличающих их от простых скоплений информационных материалов. Например, личная библиотека, в которой может ориентироваться только ее владелец, информационной системой не является. В публичных же библиотеках порядок размещения книг всегда строго определенный. Благодаря ему поиск и выдача книг, а также размещение новых поступлений представляют собой стандартные процедуры, близкие к алгоритмам.

Современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации персонального компьютера. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление.

Необходимо понимать разницу между компьютерами и информационными системами. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система немыслима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

Информационная система определяется следующими свойствами:

* любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем;
* информационная система является динамичной и развивающейся;
* при построении информационной системы необходимо использовать системный подход;
* выходной продукцией информационной системы является информация, на основе которой принимаются решения;
* информационную систему следует воспринимать как человекокомпьютерную систему обработки информации.

**Актуальность работы**.

Аптека - это важное звено лекарственного обеспечения. Аптеки готовых лекарственных форм занимаются продажей лекарств, изготовленных на заводах фирм-производителей. Каждому потребителю или фармацевту необходимо знать информацию об имеющихся лекарствах, эта потребность осуществима благодаря, информационной системе Аптека. Благодаря моей курсовой работе возможно: удобная работа с медикаментами, их учетом; получение максимально достоверной информации о лекарственных средствах. База данных очень удобна в использовании даже для клиента, который плохо знаком с компьютером.

По характеру деятельности аптечные организации подразделяются на:

- изготавливающие лекарственные препараты по рецептам врачей, стандартным прописям с последующей их реализацией в пределах юридического лица по рецептам и без рецептов врача, осуществляющие реализацию готовых лекарственных препаратов и других товаров, разрешенных к отпуску из аптечных организаций, населению по рецептам или без рецептов врача. К ним относятся: аптека с производственным отделом, аптечный пункт с правом изготовления лекарственных препаратов;

- осуществляющие реализацию готовых лекарственных препаратов, разрешенных к отпуску из аптечных организаций, населению по рецептам или без рецептов врача и учреждениям здравоохранения: это аптеки готовых форм, аптечный пункт без права изготовления лекарственных препаратов, аптечный киоск и аптечный магазин.

Большее число лекарственных препаратов мы получаем через аптечные организации. От организации работы аптеки, от ее обеспечения населения лекарственными средствами и другими товарами аптечного ассортимента зависит здоровье людей.

1. **Теоретическая часть.**
   1. **Понятия и определения БД**

База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, причём такое собрание данных, которое поддерживает одну или более областей применения.

База данных — совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Каждая строка таблицы включает данные об одном объекте (например, клиенте, автомобиле, документе), а столбцы таблицы содержат различные характеристики этих объектов — атрибуты (например, наименования и адреса клиентов, марки и цены автомобилей).

Строки таблицы называются записями; все записи имеют одинаковую структуру — они состоят из полей, в которых хранятся атрибуты объекта. Каждое поле записи содержит одну характеристику объекта и имеет строго определенный тип данных (например, текстовая строка, число, дата). Все записи имеют одни и те же поля, только в них содержатся разные значения атрибутов.

  В реляционной БД используются четыре основных типов полей:

* Числовой
* Символьный (слова, тексты, коды и т.д.)
* Дата (календарные даты в форме «день/месяц/год»)
* Логический (принимает два значения: «да» - «нет» или «истина» - «ложь»)

Любая СУБД позволяет выполнять четыре простейшие операции с данными:

* добавлять в таблицу одну или несколько записей;
* удалять из таблицы одну или несколько записей;
* обновлять значения некоторых полей в одной или нескольких записях;
* находить одну или несколько записей, удовлетворяющих заданному условию.

Для выполнения этих операций используется механизм запросов. Результатом выполнения запросов является либо отобранное по определенным критериям множество записей, либо изменения в таблицах. Запросы к базе формируются на специально созданном для этого языке, который так и называется язык структурированных запросов (SQL — StructuredQueryLanguage).

И последняя функция СУБД — это управление данными. Под управлением данными обычно понимают защиту данных от несанкционированного доступа, поддержку многопользовательского режима работы с данными и обеспечение целостности и согласованности данных.

* 1. **Microsoft Access**

Microsoft Access представляет собой мощную систему, обеспечивающую эффективную разработку и сопровождение баз данных и ориентирована на работу с объектами, к которым относятся таблицы базы данных, запросы, а также объекты приложений для работы с базой данных: формы, отчеты, страницы, макросы и модули.

СУБД MS Access состоит из шести основных компонентов:

* таблицы – для хранения структуры БД и самих данных;
* запросы – для создания QBE и SQL запросов к БД (для поиска информации в БД);
* формы – для удобства ввода, просмотра и редактирования данных;
* отчеты – для подготовки и оформления данных в виде документа для распечатки на принтере;
* страницы – для обеспечения доступа в БД с Web – страницы;
* макросы – для автоматизации выполнения повторяющихся операций (макрос состоит из последовательности внутренних команд СУБД);
* модули – для автоматизации выполнения стандартных операций и создания новых процедур средствами языка программирования Visual Basic.

# Таблицы баз данных, как правило, допускают работу с гораздо большим количеством разных типов данных

MicrosoftAccess работают со следующими типами данных:

Текстовый - тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов). Текст или числа, не требующие проведения расчётов.

Числовой - тип данных для хранения действительных чисел. Этот тип данных содержит множество подтипов. От выбора подтипа (размера) зависит точность вычислений.

Поле Мемо - специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он храниться в другом месте базы данных, а в поле храниться указатель на него, но для пользователя такое разделение заметно не всегда. Поле этого типа не может быть ключевым или проиндексированным. Длинный текст или комбинация текста и чисел.

Дата время - тип данных для хранения календарных дат и текущего времени в специальном фиксированном формате.

Денежный - тип данных для хранения денежных сумм. Теоретически, для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа. Включает денежные значения и числовые данные, используемые в математических вычислениях.

Счетчик - специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием, при добавлении новой записи в таблицу. Естественное использование - для порядковой нумерации записей.

Логический - тип для хранения логических данных, а также поля, которые могут содержать одно из двух возможных значений (могут принимать только два значения, например Да или Нет).

Гиперссылка - специальное поле для хранения адресов URL Web-объектов Интернета. При щелчке на ссылке автоматически происходит запуск браузера и воспроизведение объекта в его окне.

Мастер подстановок - это не специальный тип данных. Это объект, настройкой которого можно автоматизировать ввод данных в поле так, чтобы не вводить их вручную, а выбирать их из раскрывающегося списка.

Поле объекта OLE - включает звукозапись, рисунок и прочие типы данных. Поле этого типа не может быть ключевым или проиндексированным.

В таблицах данные распределяются по столбцам (которые называют полями) и строкам (которые называют записями). Все данные, содержащиеся в поле таблицы, должны иметь один и тот же тип. Каждое поле таблицы характеризуется наименованием, типом и шириной поля. При задании типа данных поля можно также указать размер, формат и другие параметры, влияющие на отображение значения поля и точность числовых данных.

* 1. **Среда C++ Builder**

Borland C++ Builder - выпущенное компанией Borland средство быстрой разработки приложений, позволяющее создавать приложения на языке C++, используя при этом среду разработки и библиотеку компонентов Delphi.

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 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 95 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, и 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 к форме, 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 объект из имеющихся на форме.

Свойства являются атрибутами компонента, определяющими его внешний вид и поведение. Многие свойства компонента в колонке свойств имеют значение, устанавиваемое по умолчанию (например, высота кнопок). Свойства компонента отображаются а странице свойств (Properties). Инспектор объектов отображает опубликованные (published) свойства компонентов. Помимо published-свойств, компоненты могут и чаще всего имеют общие (public), опубликованные 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 п остое 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 и ого события, вы поручаете 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 з ачений 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.

Метод является функцией, которая связана с компонентом, и которая объявляется как часть объекта. Создавая обработчики событий, можно вызывать методы, используя следующую нотацию: ->, например:

Edit1->Show();

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 осуществ ять навигацию между ними. Можно вызвать менеджер проектов , выбрав пункт меню View/ProjectManager. 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 NewForm 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 форма связана всегда 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 2.0, 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.

**2. Создание программы**

Создание базы данных в MSAcces

Перейдем непосредственно к процессу разработки. Мы будем использовать MS Access 2016. Запускаем его, сочетанием Ctrl+N создаем новую базу данных, сохраняем и открываем её. Заполняем нужные поля и выбираем их тип данных. Для примера создадим БД со списком лекарств. (Рис.1)

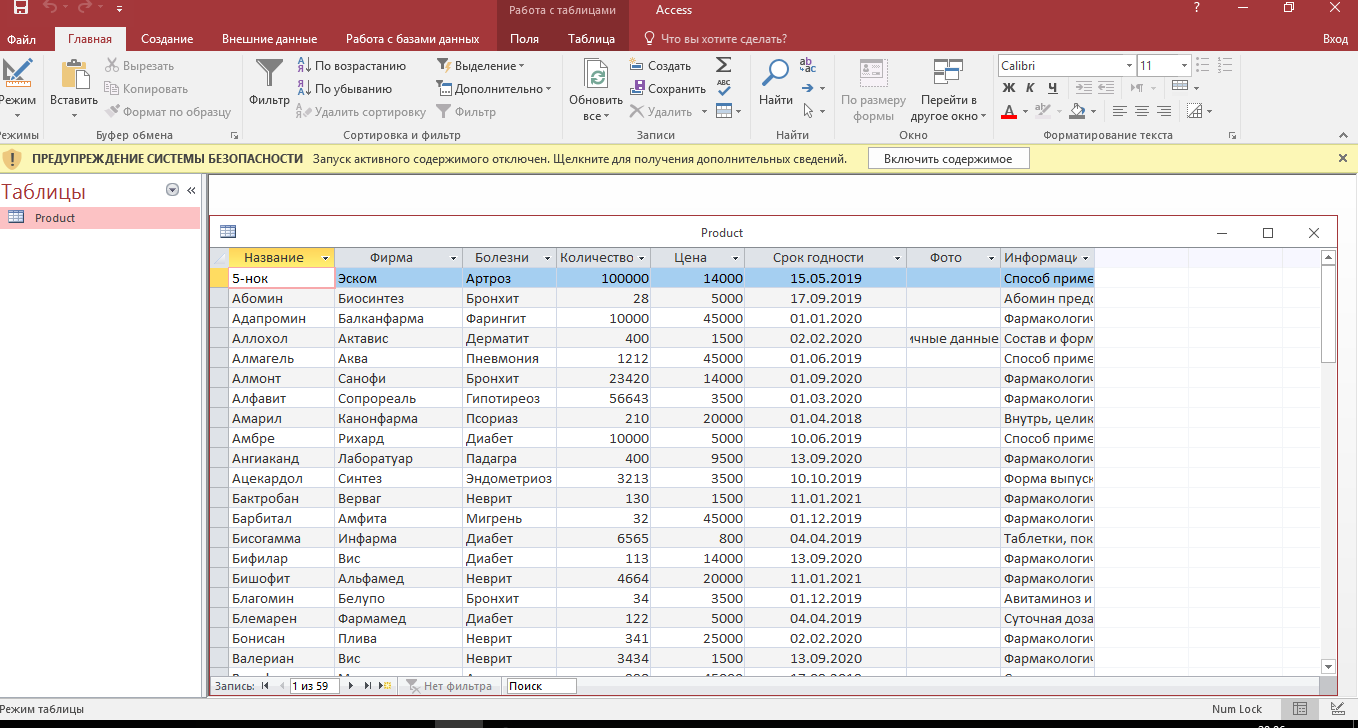


Рис. 1

Следующей создадим таблицу для авторизации администратора и таблицу дохода (Рис.2) (Рис.2.1)

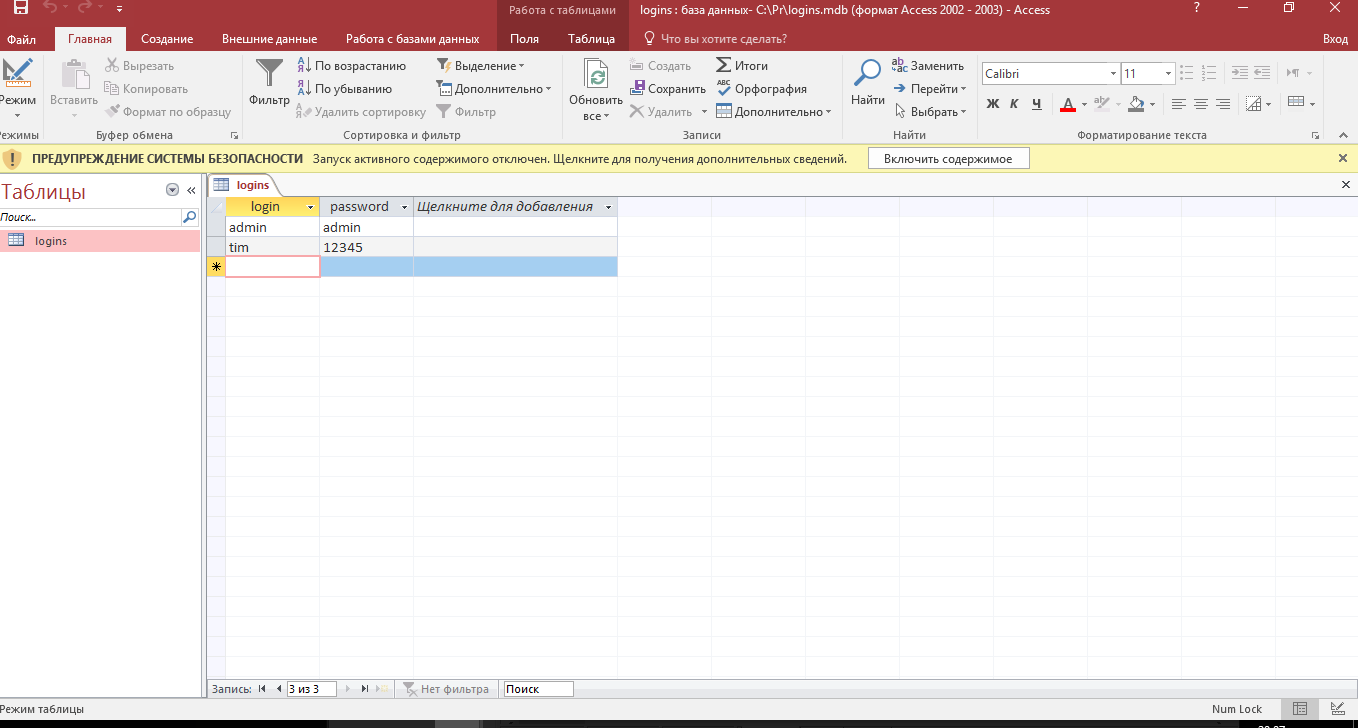


Рис. 2

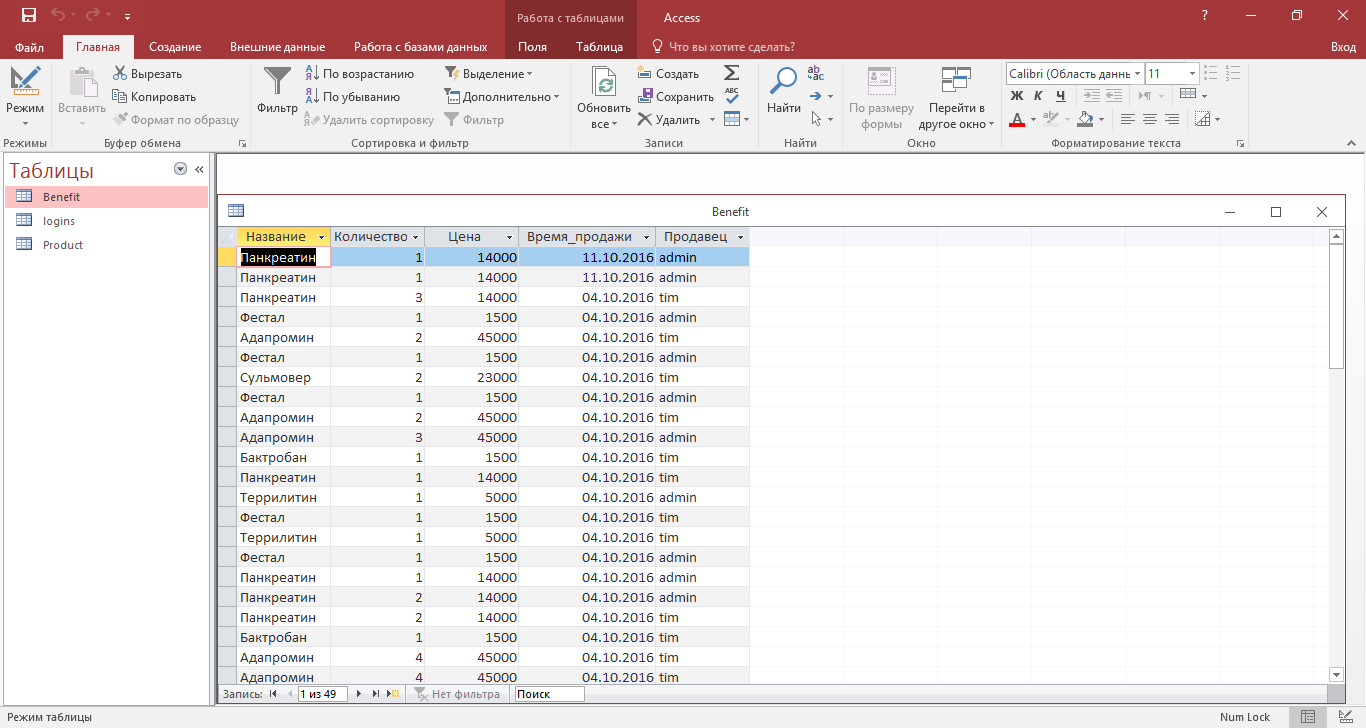


Рис.2.1

Схема данных

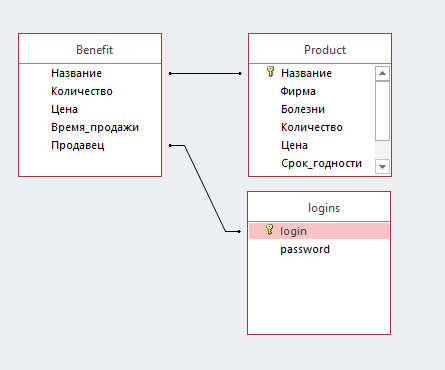


Рис2.2

**2.1 Руководство программисту**

Запускаем C++Builder, сохраняем весть проект (SaveProjectAs...). Путь не должен содержать русских имен. Размещаем на форме компоненты. С вкладки ADO(а в более старших версиях с dbGo) ADOQueryиADOConnection.

Свкладки Data Access компонент DataSource. Свкладки Data Controls компонентыDBGrid.

Щелкаемпо DataSource1 ивDataSetвыбираемADOQuery1. Щелкаемпо ADOQuery1 ивсвойствеConnectionвыбираемADOConnection1.  
Дважды щелкаем по ADOConnection1 и нажимаем кнопкуBuild. Выбираем как на картинке и нажимаем "Далее". И нажимаем два раза "Ок". (Рис. 3)

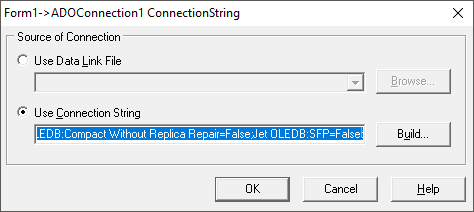


Рис. 3

Теперь, в ObjectInspector ставим Connected в true, в UserName напишем "Admin", поле пароля оставим пустым. Чтобы больше не повторять эту процедуру, установите LoginPrompt в false.

Щелкаем по DBGrid  и в DataSource выбираем DataSource1.

Щелкаем по ADOQuery1 в TableName выбираем нашу таблицу. В свойстве SQLпрописываем “SELECT \* FROMPRODUCT”. Устанавливаем Active в true.

Щелкаем дважды по DBGrid, в появившемся окошечке нажимаем на кнопку "AddAllFields".

Щелкая по полям,можно задать им другие имена для отображения, свойство FieldName и изменить ширину - Width.

Строим форму. (Рис.4)

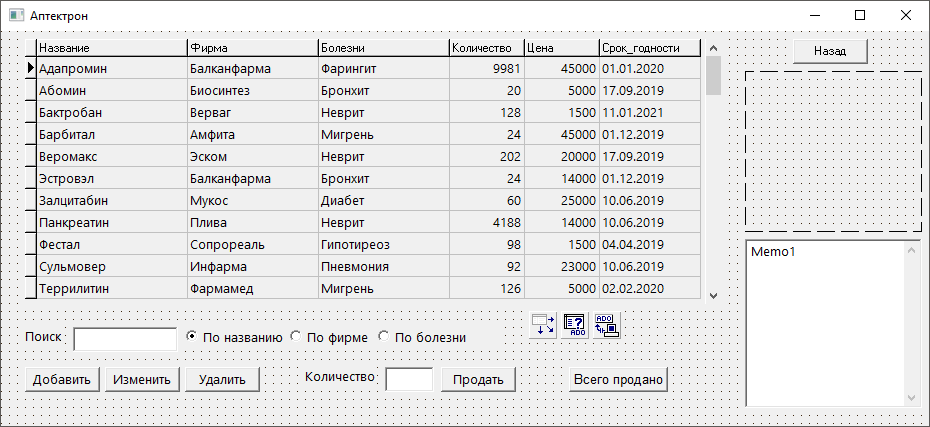


Рис.4

Строим вторую форму для добавления записей. (Рис.5)

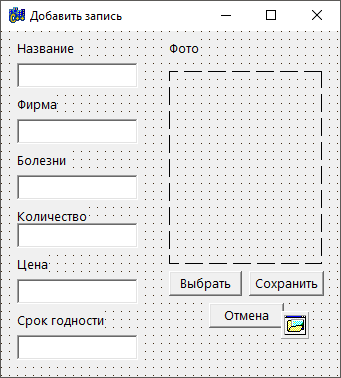


Рис.5

Форма, отображающаяся при запуске программы. (Рис.6)

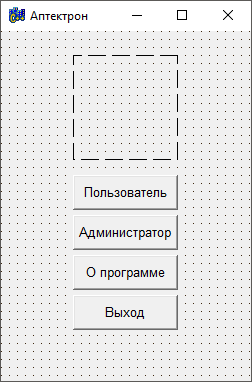


Рис.6

Форма авторизации администратора. (Рис.7)

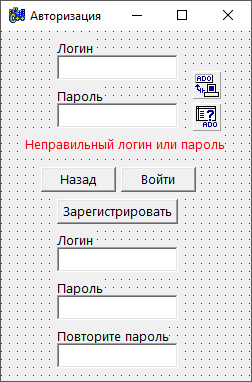


Рис.7

Форма заработанных денег. (Рис.7)

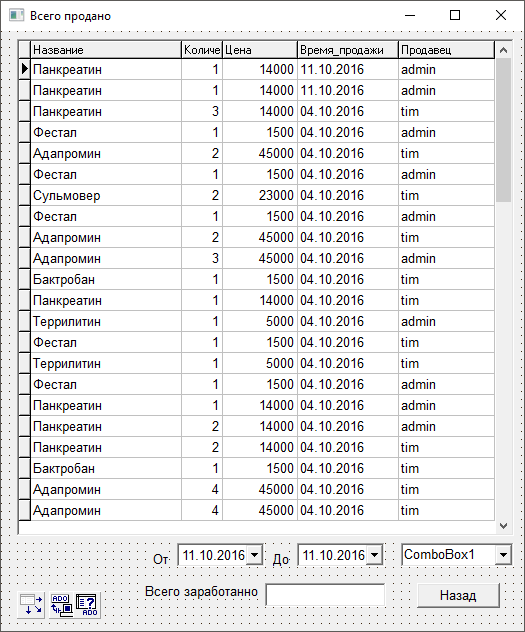


Рис.7.1

2.2 Руководство пользователю.

Для запуска программы необходимо 2 раза левой кнопкой мыши по значку:

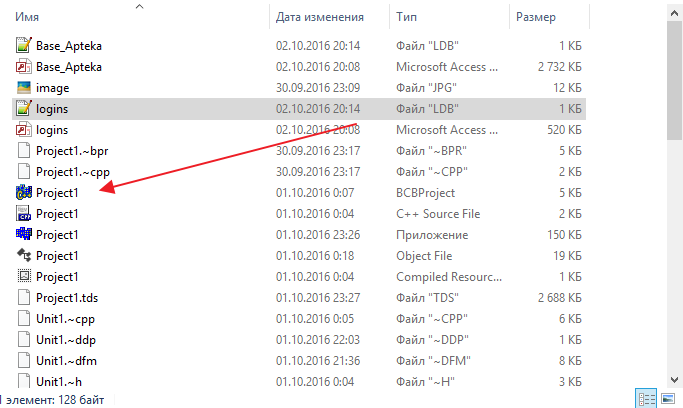


Рис.8

Первое окно – это окно общего вида программы, в котором пользователю нужно выбрать режим работы, и, в случае необходимости ввести пароль и имя администратора (Рис. 9)

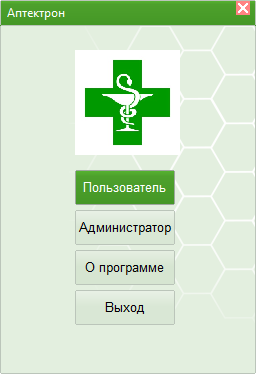


Рис. 10

Для получения полного доступа к данной базе данных, необходимо выбрать режим «Администратор» ввести логин и пароль. В нашем случае логин и пароль «admin»(Рис. 11)

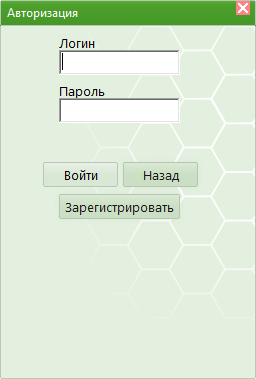


Рис. 11

Если же нет необходимости редактировать записи, то необходимо нажать «Пользователь»( Рис.12)

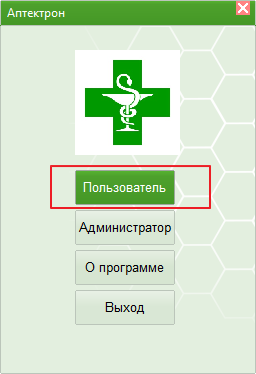


Рис.12

Режим Администратора (Рис.14)

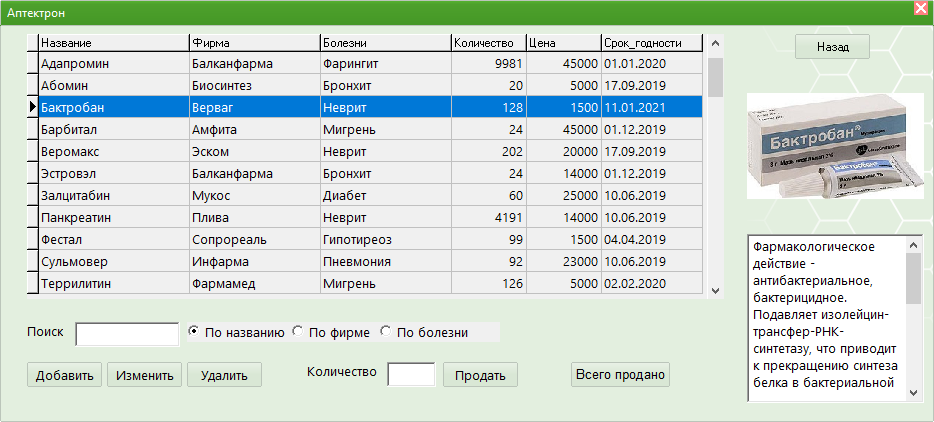


Рис.13

В этом режиме администратор может удалять, добавлять и редактировать записи.

Кликнув по заголовку таблицы, можно отсортировать записи по выбранной категории.

Поиск

Поиск осуществляется по выбранному полю. (Рис.14)

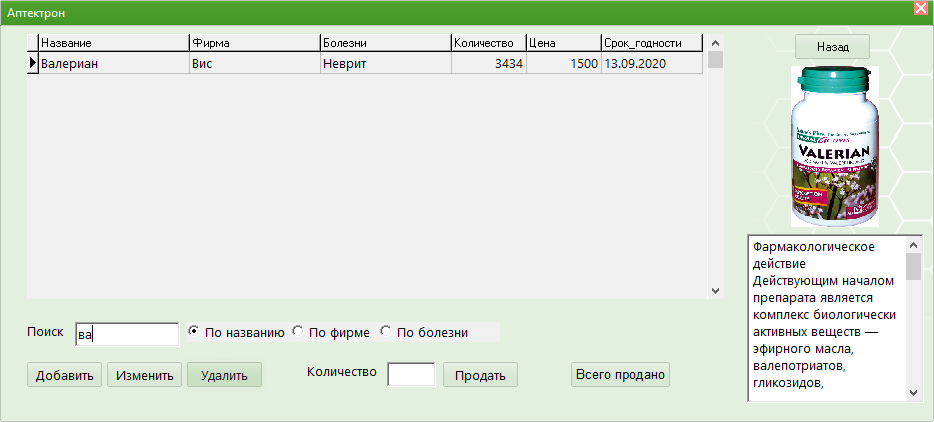


Рис.14

Режим Пользователя (Рис.15)

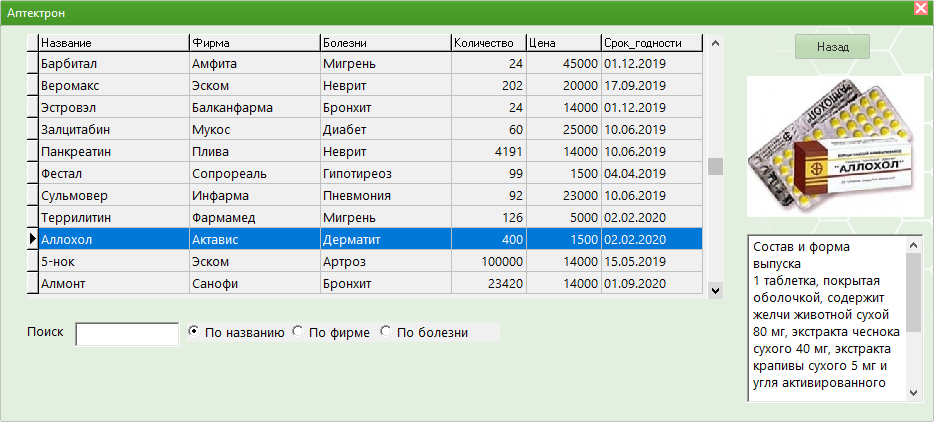


Рис.15

Отличие этого окна от главного заключается в том, что в данном режиме опции редактирования не доступны пользователю.

Нажав на кнопку Всего продано в режиме Администратора, можно узнать количество заработанных денег и просмотреть дату каждой продажи.(Рис.16)

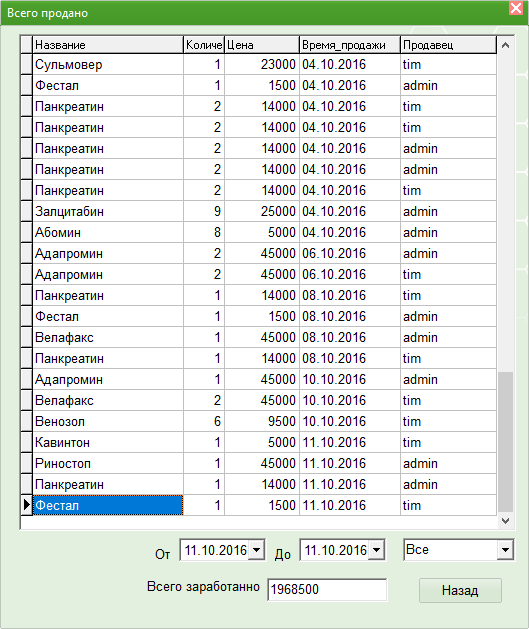


Рис.16

1. **Заключение**

Для тех, кто является новичком в объектно-ориентированном программировании, но хочет стать профессионалом, лучшим способом является написание многих объектно-ориентированных программ.

Программированию лучше всего учиться, программируя. Кроме того, стоит обратить внимание на примеры программ С++, написанные другими людьми. Если возможно, надо изучить С++-код, написанный различными программистами, обращая внимание, каким образом каждая программа спроектирована и реализована.

Необходимо посмотреть на недостатки и на сильные стороны этих программ. Это поспособствует расширению представлений о программировании. И наконец, надо экспериментировать. Только так можно стать экспертом в С++-программировании!

Изобретение языка программирования высшего уровня позволило нам общаться с машиной, понимать её (если конечно Вам знаком используемый язык), как понимает американец немного знакомый с русским языком древнюю азбуку Кириллицы. Проще говоря, мы в нашем развитии науки программирования пока что с ЭВМ на ВЫ.  Поверьте мне это не сарказм вы только посмотрите  как развилась наука программирования с того времени, как появились языки программирования, а ведь язык программирования высшего уровня, судя по всему ещё младенец. Но если мы обратим внимание на темпы роста и развития новейших технологий в области программирования, то можно предположить, что в ближайшем будущем, человеческие познания в этой сфере, помогут произвести на свет языки, умеющие принимать, обрабатывать и передавать информации в виде мысли, слова, звука или жеста. Так и хочется назвать это детище компьютеризированного будущего: «языки программирования "высочайшего" уровня». Возможно, концепция решения этого вопроса проста, а ближайшее будущее этого проекта уже не за горами, и в этот момент, где нибудь в Запорожье, Амстердаме, Токио или Иерусалиме, перед стареньким 133MHz горбится молодой, никем не признанный специалист и разрабатывает новейшую систему  искусственного интеллекта, которая наконец-то позволит человеку, с помощью своих машинных языков, вести диалог с машиной на ТЫ.

Размышляя над этим, хочется верить в прогресс науки и техники, в высоко - компьютеризированное будущее человечества, как единственного существа на планете, пусть и не использующего один, определенный разговорный язык, но способного так быстро прогрессировать и развивать свой интеллект, что и перехода от многоязыковой системы к всеобщему пониманию долго ждать не придется.

Завершить свой труд хорошо бы на такой оптимистичной ноте, но нет, напоследок хочется процитировать человека, фрагменты работы которого, в виде информации о языке Си, вам уже попадались на страницах этого текста:

Единственный способ изучать новый язык программирования – писать на нём программы.

**Аптека**

На основании выше сказанного можно сделать следующие выводы.

Основной задачей аптеки, обслуживающей население является изготовление и реализация лекарственных средств. Аптека должна выполнять функции:

- логистическую (прием, хранение и управление товарными запасами);

- производственную (прием рецептов, изготовление, контроль и отпуск лекарств по рецептам врача и требованиям ЛПУ);

- информационную (обеспечение населения и врачей ЛПУ информацией о лекарственных средствах);

- маркетинговую (формирование и осуществление ассортиментной и ценовой политики);

- медицинскую (оказание при необходимости медицинской помощи).

Все аптеки классифицируются на производственные, занимающиеся как приготовление лекарственных препаратов, так и их реализацией и аптеки, реализующие готовые лекарственные формы. Существуют собственно аптеки, аптечные пункты, аптечные киоски и аптечные магазины, каждая из этих организаций выполняет свои определенные функции. [15]

Аптеке для реализации возложенных на нее функций должны быть созданы следующие отделы: рецептурно-производственный, готовых лекарственных форм, отпуска лекарств без рецептов, парафармацевтической продукции, оптики и т.д.

В каждой аптеке существует определенные помещения: торговый зал, помещение для приготовления лекарств, помещение для получения дистиллированной воды, моечную, кабинет заведующего, комнату персонала, помещение для хранения запасов лекарственных средств, туалет, гардеробная.

**4.1 Литература и источники**

* Методические указания по выполнению курсовой работы по предмету «программированию на С/С++».
* <http://www.cyberforum.ru/cpp-builder-database/>
* <https://ru.wikipedia.org/wiki/Microsoft_Access>

**5.1 Код программы**

Unit1

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

#include <jpeg.hpp>

#include <vcl.h>

#pragma hdrstop

#include "Unit2.h"

#include "Unit1.h"

#include "Unit3.h"

#include "Unit5.h"

#include "Unit4.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

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

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

: TForm(Owner)

{

}

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

void \_\_fastcall TForm1::DBGrid1TitleClick(TColumn \*Column)

{

if(ADOQuery1->Sort==Column->FieldName+" ASC")

ADOQuery1->Sort=Column->FieldName+ " DESC";

else

ADOQuery1->Sort=Column->FieldName+ " ASC";

// if(DBGrid1->Fields[5]->){

ADOQuery1->Close();

ADOQuery1->SQL->Clear();

ADOQuery1->SQL->Add("SELECT \* FROM product ORDER BY Срок\_годности ASC");

ADOQuery1->Open();

} // }

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

void \_\_fastcall TForm1::Edit1Change(TObject \*Sender)

{

if(!Edit1->Text.IsEmpty()&&RadioButton1->Checked)

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "[Название] LIKE '"+Edit1->Text+"%'";

ADOQuery1->Filtered = true;

}

else if(!Edit1->Text.IsEmpty()&&RadioButton2->Checked)

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "[Фирма] LIKE '"+Edit1->Text+"%'";

ADOQuery1->Filtered = true;

}

else if(!Edit1->Text.IsEmpty()&&RadioButton3->Checked)

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "[Болезни] LIKE '"+Edit1->Text+"%'";

ADOQuery1->Filtered = true;

}

else

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "";

ADOQuery1->Filtered = true;

}

}

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

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

Form2->Edit1->Text = "";

Form2->Edit2->Text = "";

Form2->Edit3->Text = "";

Form2->Edit4->Text = "";

Form2->Edit5->Text = "";

Form2->Edit6->Text = ""; ;

Form2->ShowModal();

}

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

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

if (MessageBox(0,"Вы уверены?","Удалениезаписи",MB\_YESNO|MB\_ICONQUESTION) == IDYES)

{

ADOQuery1->RecNo;

ADOQuery1->Delete();

}

}

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

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

if(atoi( (Edit2->Text).c\_str())!=0){

ADOQuery1->Edit();

ADOQuery1->FieldByName("Количество")->AsString = atoi( DBGrid1->Fields[3]->AsString.c\_str() ) - atoi( (Edit2->Text).c\_str() );

ADOQuery1->Post();

Form5->ADOQuery1->Insert();

Form5->ADOQuery1->Edit();

Form5->ADOQuery1->FieldByName("Название")->AsString = ADOQuery1->FieldByName("Название")->AsString;

Form5->ADOQuery1->FieldByName("Количество")->AsString = atoi( (Edit2->Text).c\_str() );

Form5->ADOQuery1->FieldByName("Цена")->AsString = ADOQuery1->FieldByName("Цена")->AsString;

Form5->ADOQuery1->FieldByName("Время\_Продажи")->AsString = Now().DateString();

Form5->ADOQuery1->Post(); }

}

void \_\_fastcall TForm1::DBGrid1CellClick(TColumn \*Column)

{

if(!(TBlobField\*)ADOQuery1->FieldByName("Фото")->IsNull){

if(!Image1->Visible)Image1->Visible=true;

TMemoryStream\* strm = new TMemoryStream;

TJPEGImage \*ptJpg=new TJPEGImage;

((TGraphicField\*)ADOQuery1->FieldByName("Фото"))->SaveToStream(strm);

strm->Seek(0,0);

if (\*(Word\*)strm->Memory == 0xD8FF)Image1->Picture->Graphic=ptJpg;

Image1->Picture->Graphic->LoadFromStream(strm);

delete strm;

delete ptJpg;

}else

Image1->Visible=false;

Memo1->Text = ADOQuery1->FieldByName("Информация")->AsString ;

}

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

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

Memo1->Text="";

}

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

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

Form2->Edit1->Text = Form1->ADOQuery1->FieldByName("Название")->AsString;

Form2->Edit2->Text = Form1->ADOQuery1->FieldByName("Фирма")->AsString;

Form2->Edit3->Text = Form1->ADOQuery1->FieldByName("Болезни")->AsString;

Form2->Edit4->Text = Form1->ADOQuery1->FieldByName("Количество")->AsString;

Form2->Edit5->Text = Form1->ADOQuery1->FieldByName("Цена")->AsString;

Form2->Edit6->Text = Form1->ADOQuery1->FieldByName("Срокгодности")->AsString;

if(!(TBlobField\*)ADOQuery1->FieldByName("Фото")->IsNull){

if(!Form2->Image1->Visible)Form2->Image1->Visible=true;

TMemoryStream\* strm = new TMemoryStream;

TJPEGImage \*ptJpg=new TJPEGImage;

((TGraphicField\*)ADOQuery1->FieldByName("Фото"))->SaveToStream(strm);

strm->Seek(0,0);

if (\*(Word\*)strm->Memory == 0xD8FF)Form2->Image1->Picture->Graphic=ptJpg;

Form2->Image1->Picture->Graphic->LoadFromStream(strm);

delete strm;

delete ptJpg;

}else

Form2->Image1->Visible=false;

Form2->ShowModal();

}

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

void \_\_fastcall TForm1::RadioButton1Click(TObject \*Sender)

{

if(!Edit1->Text.IsEmpty())

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "[Название] LIKE '"+Edit1->Text+"%'";

ADOQuery1->Filtered = true;

}

else

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "";

ADOQuery1->Filtered = true;

}

}

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

void \_\_fastcall TForm1::RadioButton2Click(TObject \*Sender)

{

if(!Edit1->Text.IsEmpty())

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "[Фирма] LIKE '"+Edit1->Text+"%'";

ADOQuery1->Filtered = true;

}

else

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "";

ADOQuery1->Filtered = true;

}

}

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

void \_\_fastcall TForm1::RadioButton3Click(TObject \*Sender)

{

if(!Edit1->Text.IsEmpty())

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "[Болезни] LIKE '"+Edit1->Text+"%'";

ADOQuery1->Filtered = true;

}

else

{

ADOQuery1->Filtered = false;

ADOQuery1->Filter = "";

ADOQuery1->Filtered = true;

}

}

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

void \_\_fastcall TForm1::FormClose(TObject \*Sender, TCloseAction&Action)

{

Form3->Close();

}

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

void \_\_fastcall TForm1::DataSource1DataChange(TObject \*Sender,

TField \*Field)

{

if(!(TBlobField\*)ADOQuery1->FieldByName("Фото")->IsNull){

if(!Image1->Visible)Image1->Visible=true;

TMemoryStream\* strm = new TMemoryStream;

TJPEGImage \*ptJpg=new TJPEGImage;

((TGraphicField\*)ADOQuery1->FieldByName("Фото"))->SaveToStream(strm);

strm->Seek(0,0);

if (\*(Word\*)strm->Memory == 0xD8FF)Image1->Picture->Graphic=ptJpg;

Image1->Picture->Graphic->LoadFromStream(strm);

delete strm;

delete ptJpg;

}else

Image1->Visible=false;

Memo1->Text = ADOQuery1->FieldByName("Информация")->AsString ;

}

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

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

Form5->ADOQuery1->First();

double a=0;

for (int i=0; i<Form5->ADOQuery1->RecordCount; i++)

{

a+= StrToInt(Form5->ADOQuery1->FieldByName("Количество")->AsString)\*StrToInt(Form5->ADOQuery1->FieldByName("Цена")->AsString) ;

Form5->ADOQuery1->Next();

}

Form5->Edit1->Text = a;

Form5->ShowModal();

}

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

void \_\_fastcall TForm1::Button6Click(TObject \*Sender)

{

Form1->Visible=false;

Form4->Edit1->Text="";

Form4->Edit2->Text="";

Form3->Visible=true;

}

Unit 2

#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::Button1Click(TObject \*Sender)

{

if(Edit1->Text!="" && Edit2->Text!="" && Edit3->Text!="" && Edit4->Text!="" && Edit5->Text!="" && Edit6->Text!="")

{

Form1->ADOQuery1->Edit();

Form1->ADOQuery1->FieldByName("Название")->AsString = Edit1->Text;

Form1->ADOQuery1->FieldByName("Фирма")->AsString = Edit2->Text;

Form1->ADOQuery1->FieldByName("Болезни")->AsString = Edit3->Text;

Form1->ADOQuery1->FieldByName("Количество")->AsString = Edit4->Text;

Form1->ADOQuery1->FieldByName("Цена")->AsString = Edit5->Text;

Form1->ADOQuery1->FieldByName("Срокгодности")->AsString = Edit6->Text;

if(Image1->Picture->Graphic != 0)

{

Form1->ADOQuery1->Edit();

TMemoryStream\* strm = new TMemoryStream();

Image1->Picture->Graphic->SaveToStream(strm);

((TGraphicField\*)Form1->ADOQuery1->FieldByName("Фото"))->LoadFromStream(strm);

delete strm;

}

Form1->ADOQuery1->Post();

Form2->Close();

}

}

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

void \_\_fastcall TForm2::Button2Click(TObject \*Sender)

{

if (OpenPictureDialog1->Execute())

{

AnsiStringCurrentFile = OpenPictureDialog1->FileName;

Image1->Picture->LoadFromFile(CurrentFile);

}

}

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

void \_\_fastcall TForm2::Button3Click(TObject \*Sender)

{

Form2->Close();

}

Unit3

#include <vcl.h>

#pragma hdrstop

#include "Unit3.h"

#include "Unit1.h"

#include "Unit4.h"

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

#pragma package(smart\_init)

#pragma link "sSkinManager"

#pragma link "sButton"

#pragma resource "\*.dfm"

TForm3 \*Form3;

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

\_\_fastcall TForm3::TForm3(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm3::FormCreate(TObject \*Sender)

{

Image1->Picture->LoadFromFile("image.jpg");

}

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

void \_\_fastcall TForm3::Button1Click(TObject \*Sender)

{

Form1->Button1->Visible=false;

Form1->Button2->Visible=false;

Form1->Button3->Visible=false;

Form1->Button4->Visible=false;

Form1->Edit2->Visible=false;

Form1->Label2->Visible=false;

Form1->Visible=true;

Form3->Visible=false;

}

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

void \_\_fastcall TForm3::Button2Click(TObject \*Sender)

{

Form4->Visible=true;

Form3->Visible=false;

}

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

void \_\_fastcall TForm3::Button3Click(TObject \*Sender)

{

MessageBox(0,"Аптектрон v1.0.0\nРазработчики:\nГалиуллинТимур\nИлхомжонов Джамшид\nУктамов Нурали\n\nЭлектроннаяпочта:\nandroidsmallmeat@gmail.com","Опрограмме",MB\_OK);

}

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

void \_\_fastcall TForm3::Button4Click(TObject \*Sender)

{

Form3->Close();

}

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

void \_\_fastcall TForm3::sButton1Click(TObject \*Sender)

{

Form1->Button1->Visible=false;

Form1->Button2->Visible=false;

Form1->Button3->Visible=false;

Form1->Button4->Visible=false;

Form1->Edit2->Visible=false;

Form1->Label2->Visible=false;

Form1->Button5->Visible=false;

Form1->Visible=true;

Form3->Visible=false;

}

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

void \_\_fastcall TForm3::sButton2Click(TObject \*Sender)

{

Form4->Visible=true;

Form3->Visible=false;

}

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

void \_\_fastcall TForm3::sButton3Click(TObject \*Sender)

{

MessageBox(0,"Аптектрон v1.2.0\nРазработчики:\nГалиуллинТимур\nИлхомжонов Джамшид\nУктамов Нурали\n\nЭлектроннаяпочта:\nandroidsmallmeat@gmail.com","Опрограмме",MB\_OK);

}

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

void \_\_fastcall TForm3::sButton4Click(TObject \*Sender)

{

Form3->Close();

}

Unit4

#include <vcl.h>

#pragma hdrstop

#include "Unit4.h"

#include "Unit1.h"

#include "Unit3.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm4 \*Form4;

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

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

: TForm(Owner)

{

}

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

void \_\_fastcall TForm4::Button3Click(TObject \*Sender)

{

if(Edit3->Visible==false)

{Edit3->Visible=true;

Edit4->Visible=true;

Edit5->Visible=true;

Label4->Visible=true;

Label5->Visible=true;

Label6->Visible=true;}

AnsiString log = Edit1->Text,

pas = Edit2->Text,

logb,pasb;

bool right = false;

ADOQuery1->First();

for (int i=0;i<ADOQuery1->RecordCount;i++)

{

logb=ADOQuery1->FieldByName("login")->AsString;

pasb=ADOQuery1->FieldByName("password")->AsString;

if(log == logb&& pas == pasb) {right = true; break;}

ADOQuery1->Next();

}

if((Edit4->Text==Edit5->Text )&& right==true&&Edit4->Text!="")

{

ADOQuery1->Insert();

ADOQuery1->Edit();

ADOQuery1->FieldByName("login")->AsString = Edit3->Text;

ADOQuery1->FieldByName("password")->AsString = Edit4->Text;

ADOQuery1->Post();

Label3->Caption = "Добавленадминистратор";

Edit3->Text = "";

Edit4->Text = "";

Edit5->Text = "";

Label3->Visible=true;

}

else if(Edit4->Text!=Edit5->Text)

{ Label3->Caption="Пароли не совпадают";

Label3->Visible=true; }

else if(right!=true)

{ Label3->Caption="Введите данные администратора";

Label3->Visible=true; }

}

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

void \_\_fastcall TForm4::Button2Click(TObject \*Sender)

{

AnsiString log = Edit1->Text,

pas = Edit2->Text,

logb,pasb;

bool right = false;

ADOQuery1->First();

for (int i=0;i<ADOQuery1->RecordCount;i++)

{

logb=ADOQuery1->FieldByName("login")->AsString;

pasb=ADOQuery1->FieldByName("password")->AsString;

if(log == logb&& pas == pasb) {right = true; break;}

ADOQuery1->Next();

}

if(right==true)

{

Form1->Button1->Visible=true;

Form1->Button2->Visible=true;

Form1->Button3->Visible=true;

Form1->Button4->Visible=true;

Form1->Edit2->Visible=true;

Form1->Label2->Visible=true;

Form1->Button5->Visible=true;

Form1->Button6->Visible=true;

Form1->Visible=true;

Form3->Visible=false;

}

else

{

Edit2->Text="";

Label3->Visible=true;

}

}

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

void \_\_fastcall TForm4::FormClose(TObject \*Sender, TCloseAction&Action)

{

Form3->Close();

}

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

void \_\_fastcall TForm4::Button1Click(TObject \*Sender)

{

Edit3->Visible=false;

Edit4->Visible=false;

Edit5->Visible=false;

Label4->Visible=false;

Label5->Visible=false;

Label6->Visible=false;

Edit1->Text = "";

Edit2->Text = "";

Edit3->Text = "";

Edit4->Text = "";

Edit5->Text = "";

Form3->Visible=true;

Form4->Visible=false;

}

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

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

{

Edit1->OnKeyPress;

}

void \_\_fastcall TForm4::Edit2KeyPress(TObject \*Sender, char &Key)

{

if (Key==13)

Button2Click(Sender);

}

Unit 5

#include <vcl.h>

#pragma hdrstop

#include "Unit5.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm5 \*Form5;

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

: TForm(Owner)

{

}

void \_\_fastcall TForm5::Button1Click(TObject \*Sender)

{

Form5->Close();

}