ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОМУНИКАЦИОННЫХ ТЕХНОЛОГИИ РЕСПУБЛИКИ УЗБЕКИСТАН
НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИИ

КУРСОВАЯ РАБОТА

ПО ПРЕДМЕТУ “ПРОГРАММИРОВАНИЕ НА С++”

НА ТЕМУ: “ОБЬЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. C++ BUILDER”

ВЫПОЛНИЛ: ***СТУДЕНТ 2^{В} КУРСА ПО НАПРАВЛЕНИЮ КОМПЬЮТЕРНОГО ИНЖИНИРИНГА САЙТОВ АЗИЗБЕК***

ПРИНЯЛ: ***ЯДГАРОВ Ш.***

НУКУС 2015

План:

- ***Введение***

1. С и С++.

2. С++ Builder 5.0.

3. С++ визуальная среда разработки.

4. Основные компоненты С++ Builder.

5. Прикладная работа..

- ***Заключение.***
- ***Список литературы***

Введение.

В начале компьютерной эры программисты были рабами вычислительных машин. Разработчики программного обеспечения должны были писать свои команды на единственном языке, который понимали компьютеры, — в двоичном коде, и программы выглядели как последовательность нулей и единиц. По мере того как время шло, и алгоритмы усложнялись, программирование требовало все больше времени, а внесение изменений в программы и их модернизация становились практически невозможными. Так появились языки программирования высокого уровня: Фортран, Бейсик, Паскаль.

Требования к программам росли, времени для их написания отводилось все меньше, программистам надо было сосредоточиться на сложных алгоритмах, их эффективной реализации, не отвлекаясь на внутреннюю структуру компьютера. А тут еще проблемы переносимости программ на новые компьютеры с новыми возможностями... Был необходим новый подход — и он появился в виде объектно-ориентированного программирования.

Язык Simula, использовавшийся в 70-80-х годах в норвежских вооруженных силах, является одним из первых языков, основанных на понятии класс (класс — подмножество, состоящее из данных и связанных с ними функций). Примерно в то же время был утвержден стандарт нового языка программирования, получивший название C (Си) и обладавший большой мощью, гибкостью и эффективностью. Достаточно сказать, что это был один из первых языков высокого уровня, позволявший работать с оборудованием, например, организовывать обмен данными между компьютерами.

В 1978 году были сделаны первые попытки объединить достоинства этих двух языков: так появился язык «Си с классами». Лишь спустя несколько лет он стал тем C++, который так широко применяется сегодня.

Проходит время, меняются требования и подходы к программированию, и в соответствии с этими требованиями меняется язык . Сейчас существует множество различных платформ и версий C++. Среди них можно выделить наиболее часто используемые Microsoft C++ (Dos,Windows) и Borland C++ (Dos,Windows), Visual C++ (Windows), GNU C++ (Linux). В связи с этим можно говорить только о единстве ядра C++, в то время как интерфейс среды программирования и некоторые инструкции различаются для разных платформ и версий. Среда Borland C++ 5, зарекомендовала себя как одна из самых надежных и может быть особенно привлекательна для начинающих программировать на C++.

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. Объекты можно классифицировать по разным категориям: например, мои цифровые наручные часы "Cassio" принадлежат к классу часов. Программная реализация часов входит, как стандартное приложение, в состав операционной системы вашего компьютера.

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. В этой главе основные принципы ООП (инкапсуляция, наследование, полиморфизм, создание классов и объектов) интерпретируются и дополняются новыми понятиями и терминологией, принятыми интегрированной средой визуальной обработки C++Builder. Приводится описание расширений языка новыми возможностями (компоненты, свойства, обработчики событий) и последних дополнений стандарта ANSI C++ (шаблоны, пространства имен, явные и непостоянные объявления, идентификация типов при выполнении программы, исключения).

C и C++

Язык C++ не требует обязательного применения объектов в программах Это позволяет модернизировать ранее на писанные и создавать новые программы, пользуясь практически синтаксисом C и выполняя лишь более строгие требования C++ к типам: наличие в начале программы прототипов всех функций, определенных пользователей явного приведения типов для указателей на разные типы и некоторых других. Архитектура современных операционных систем становится все более и более объектно-ориентированной. При работе в таких системах не обойтись без понимания использования таких основополагающих понятий объектно-ориентированного программирования, как объекты, инкапсуляция и полиморфизм.

Класс *- это тип, описывающий устройство объектов. Понятие «класс» подразумевает некоторое поведение и способ представления. Понятие «объект» подразумевает нечто, что обладает определённым поведением и способом представления. Говорят, что объект - это экземпляр класса.*

Объект *- сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса.*

Прототип *- это объект-образец, по образу и подобию которого создаются другие объекты.*

Объектно-ориентированный язык программирования характеризуется тремя основными свойствами:

Инкапсуляция*. Комбинирование записей с процедурами и функциями, манипулирующими полями этих записей, формирует новый тип данных - объект.*

Наследование*. Определение объекта и его дальнейшее использование для построения иерархии порожденных объектов с возможностью для каждого порожденного объекта, относящегося к иерархии, доступа к коду и данным всех порождающих объектов.*

Полиморфизм*. Присваивание действию одного имени, которое затем совместно используется вниз и вверх по иерархии объектов, причем каждый объект иерархии выполняет это действие способом, именно ему подходящим*

C++ Builder 5.0.

Пятая версия продукта Borland C++ Builder , вышедшая в начале 2000 года, сегодня является наиболее совершенной визуальной средой быстрой разработки на Си++ для Windows. В ее состав входит около 200 самых разных компонентов, а создание законченной программы требует минимума усилий. Ближайший конкурент Borland C++Builder — это не

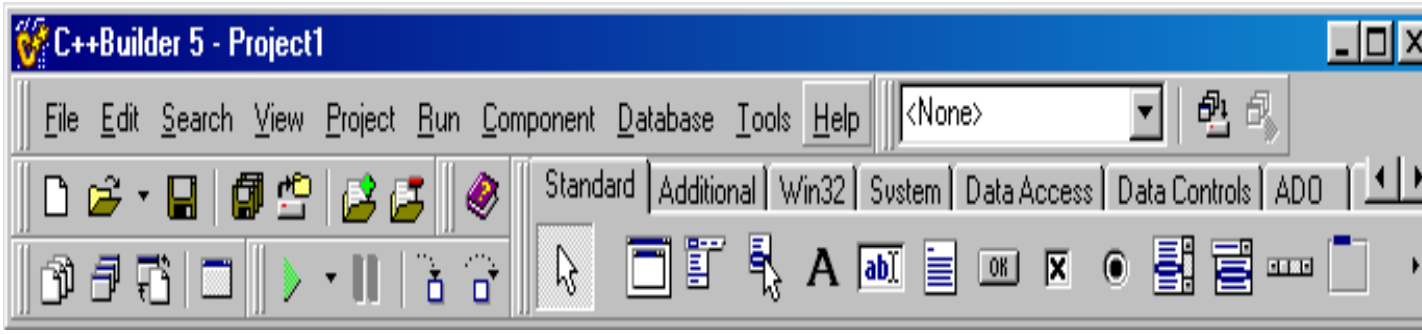
система Microsoft Visual C++, которая построена по другой схеме и не является RAD-системой, а Microsoft Visual Basic, типичная среда разработки. Однако эффективность программ, создаваемых с помощью C++Builder, в десятки раз превосходит быстродействие программ, написанных на MS Visual Basic. Да и по числу свободных доступных компонентов равных среде C++Builder сегодня не найти. У этой системы есть родной брат — RAD-среда Borland Delphi, технология работы с которой полностью совпадает с технологией, принятой в C++Builder. Только в Delphi программный код пишется не на языке C++, а на языке программирования Паскаль, точнее на его объектно-ориентированной версии ObjectPascal. Но самое интересное, что Borland C++Builder позволяет писать программу при желании одновременно и на Си++, и на Паскале!

C++ визуальная среда разработки.

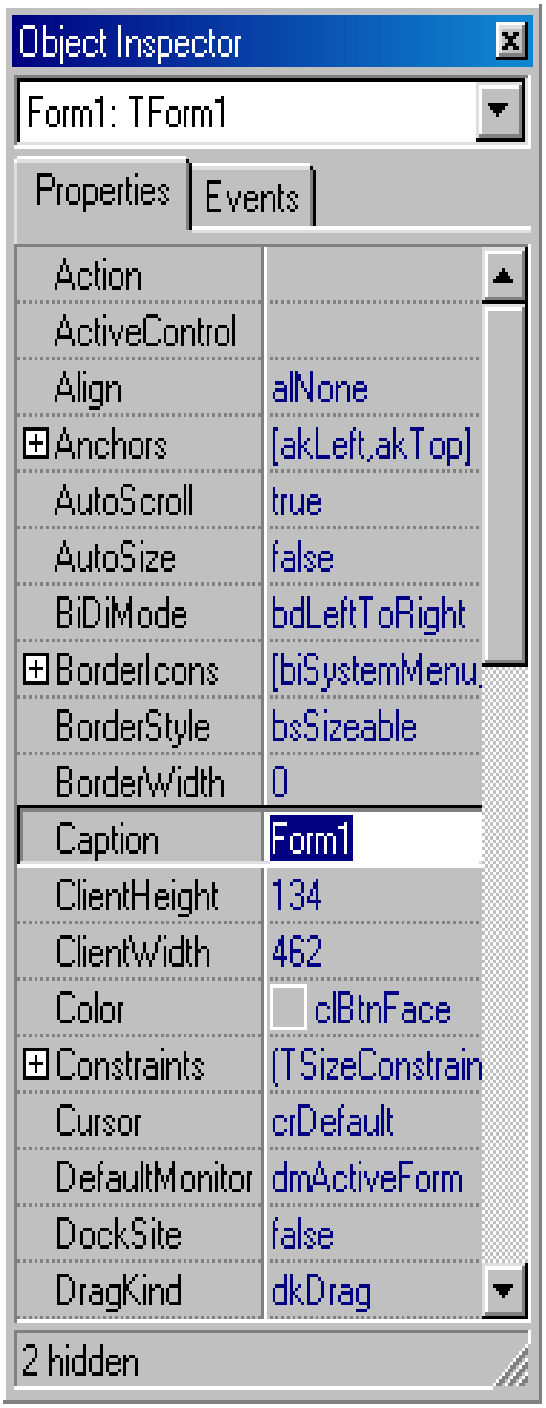
Различные элементы управления, такие, как кнопки, переключатели, значки и другие объекты Windows (которые в терминологии RAD-систем называются компонентами), можно перетаскивать в проектируемом окне с помощью мыши. Процесс создания интерфейса будущей программы напоминает забаву с игровым компьютерным конструктором. Поэтому RAD-среды еще называют визуальными средами разработки: какими мы видим рабочие и диалоговые окна программы при проектировании, такими они и будут, когда программа заработает.

При запуске программы открывается визуальной среды разработки (IDE) в начальном состоянии, которая состоит из четырех компонентов:

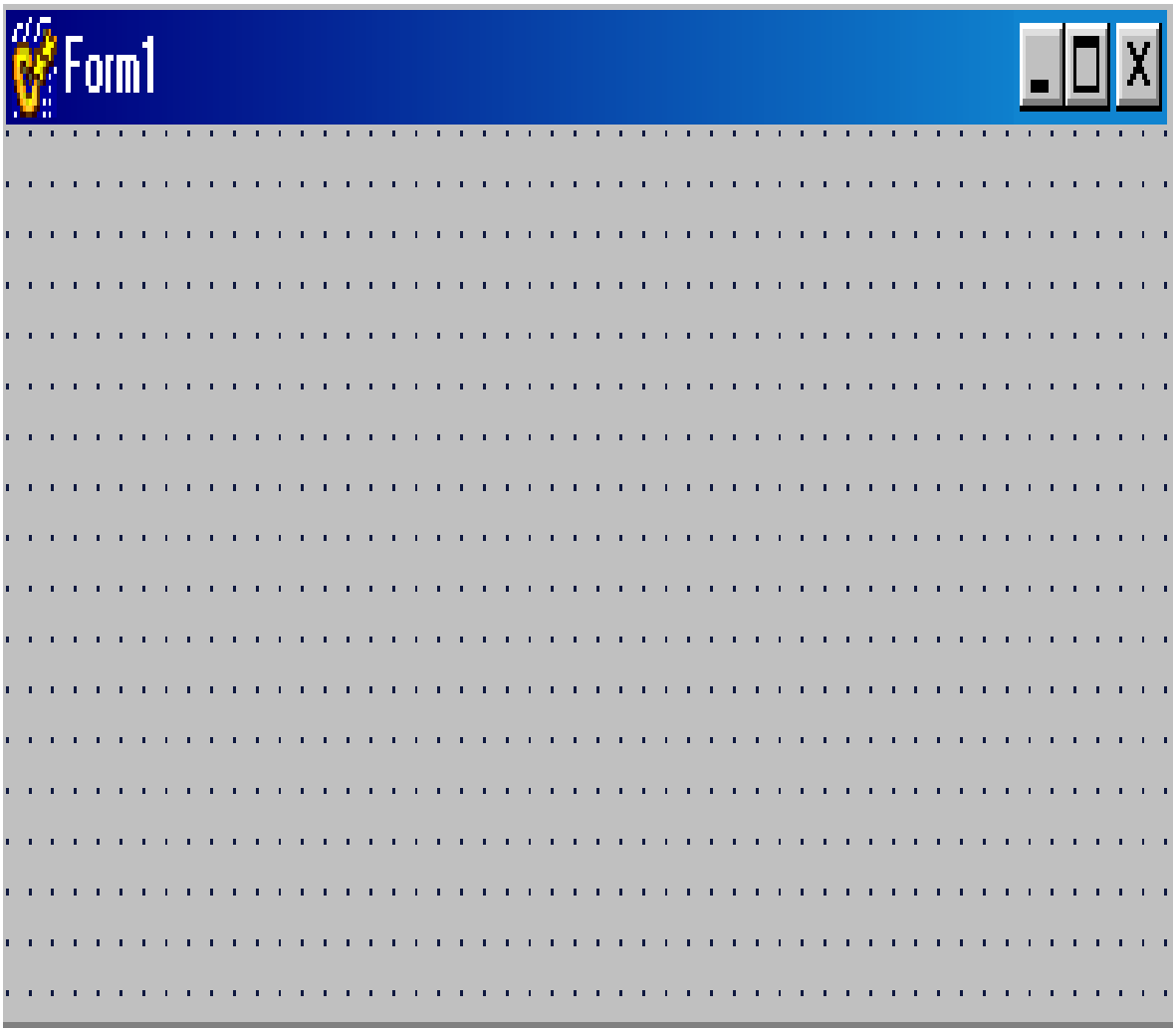
1. Панели управления.



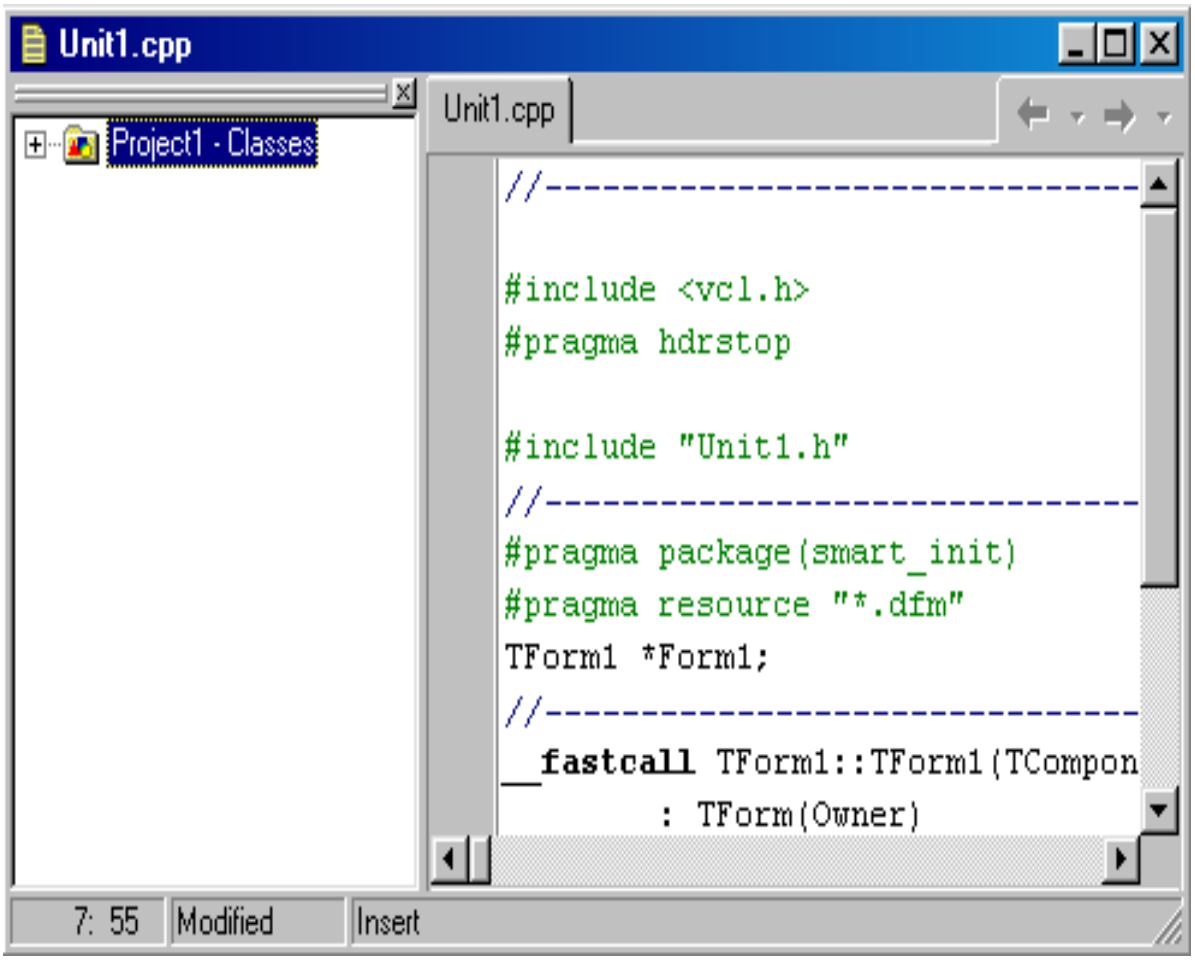
2. Панели Инспектора объектов (Object Inspektor).



3. Визуальный проектировщик рабочих окон (Форма).



4. Окна редактора программы.



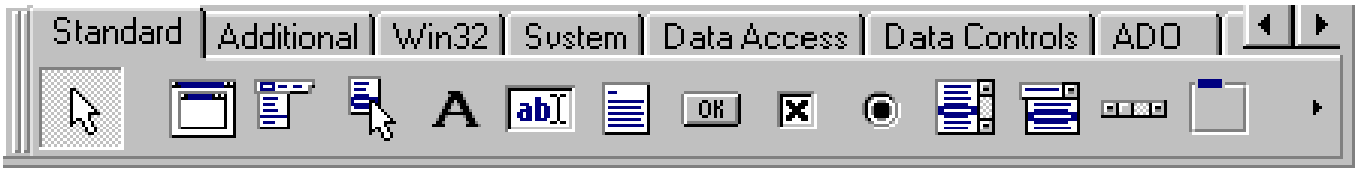
Просмотрщик классов. ***Редактор текста.***

Главное окно C++ Bilder (панель управления).

Главное окно не разворачивается на весь экран, но его можно перетащить в любое удобное место. Традиционно его располагают в верхней части экрана. В состав главного окна входят:

- *строка заголовка;*
- *строка меню;*
- *панель инструментов, на которой располагаются командные кнопки для выполнения наиболее часто требуемых действий;*

• палитра компонентов, содержащая набор заготовок для элементов управления, из которых будет собираться интерфейс вашей программы. Каждый компонент представлен на палитре своим значком.



Палитра компонентов состоит из четырнадцати панелей, на которых компоненты сгруппированы по конкретным областям применения.

Кратко опишем области, охватываемые компонентами каждой панели:

- ***Standard*** *— стандартные элементы управления Windows;*
- ***Additional*** *— дополнительные, нестандартные элементы управления Windows;*
- ***Win32*** *— элементы управления Windows 9x;*
- ***System*** *— системные объекты (таймер, музыкальный проигрыватель и другие);*
- ***Internet*** *— все для приложений, работающих с Интернетом;*
- ***Data Access*** *— компоненты для организации связи с базами данных;*
- ***Data Controls*** *— управляющие элементы для работ с базами данных;*
- ***ADO*** *— компоненты для доступа к данным на основе одной из самых передовых на сегодняшний день Windows-технологии Microsoft ActiveX Data Objects (ADO);*
- ***InterBase*** *— компоненты для работы с СУБД InterBase производства корпорации Inprise;*
- ***Midas*** *— компоненты для создания приложений, способных работать на нескольких компьютерах;*
- ***InternetExpress и Internet*** *— средства быстрого создания приложений для Интернета;*
- ***FastNet*** *— компоненты, поддерживающие основные сетевые протоколы, ориентированные на Интернет;*
- ***Decision Cube*** *— компоненты системы анализа данных;*
- ***QReport*** *— компоненты создания различных отчетов;*
- ***Dialogs*** *— стандартные диалоговые окна Windows;*
- ***Win 3.1*** *— элементы управления Windows 3.1;*
- ***Samples*** *— примеры компонентов, входящие в поставку системы;*
- ***ActiveX*** *— ActiveX-компоненты (ActiveX — формат активных компонентов, разработанный фирмой Microsoft. Borland C++Builder 4 также поддерживает этот формат);*
- ***Servers*** *— набор компонентов, с помощью которых можно управлять работой офисных программ Word, Excel, PowerPoint, Outlook и др.*

Визуальный проектировщик рабочих форм

Форма — это окно Windows, в котором размещаются различные элементы управления (кнопки, меню, переключатели, списки, элементы ввода и т. д.). Когда создаваемая программа будет откомпилирована и запущена, форма превратится в обычное окно Windows и станет выполнять те действия, которые для нее определены. Таких окон в программе может быть сколько угодно.

Инспектор объектов (Object Inspector)

Инспектор объектов — очень важная часть среды разработки. Он предназначен для задания свойств объектов и определения их реакции на различные события.

Свойства объектов

Свойство объекта — это одна из его характеристик, такая, как ширина для кнопки, название для окна, наличие полос прокрутки для списка, цвет и стиль для шрифта, имя файла для рисунка и т. д. Каждый объект имеет большое число свойств, свойства многих элементов управления схожи; свойства других объектов могут сильно различаться

Инспектор объектов позволяет быстро и удобно менять любые свойства текущего (выделенного на форме) объекта. При этом вносимые изменения немедленно сказываются на внешнем виде этого объекта. Например, если мы с помощью Инспектора изменим, текст надписи на кнопке, это изменение мгновенно отобразится на самой кнопке в проектируемой форме.

Окно редактора программы

За главной формой скрыто окно редактора программы Между текущей формой и редактором можно переключаться с помощью клавиши F12. Окно редактора, состоит из двух панелей: панели Просмотрщика классов и панели редактора исходного текста программы на Си++. Просмотрщик классов визуально отображает структуру связей между различными объектами нашей программы и позволяет быстро перемещаться по ее тексту.

Прикладная работа..

Пример создания простейшей программы на C++ Bilder 5.0

<u>Валютный калькулятор</u>

Эту простейшую программу мы будем складывать из готовых «кирпичиков», эти «кирпичики» называются компонентами. Компоненты могут быть визуальными и невизуальными.

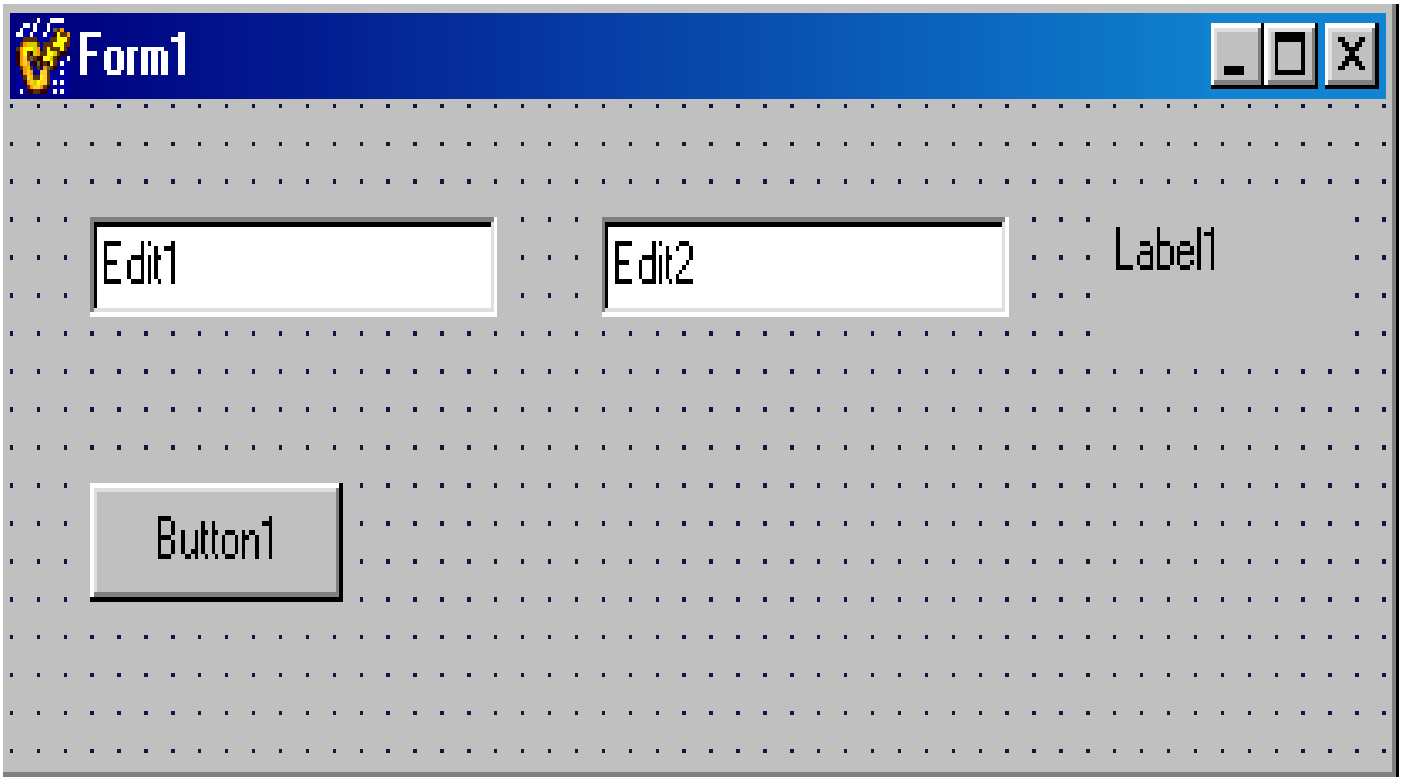
Визуальный компонент*-это элемент управления (например кнопка или список), с которым пользователь будет взаимодействовать во время ее работы.*

Невизуальный компонент*-это элемент управления, с которым пользователь взаимодействовать не может.*

Программа будет называться «Валютный калькулятор», в него вводят сумму в долларах США и текущий курс ММВБ, а результат получают в рублях. Устроен он будет так: в рабочем окне программы присутствуют три поля и одна командная кнопка. В первое поле

вводят денежную сумму, во второе-курс пересчета, а в третьем поле программа выдает результат вычисления после нажатия на командную кнопку.

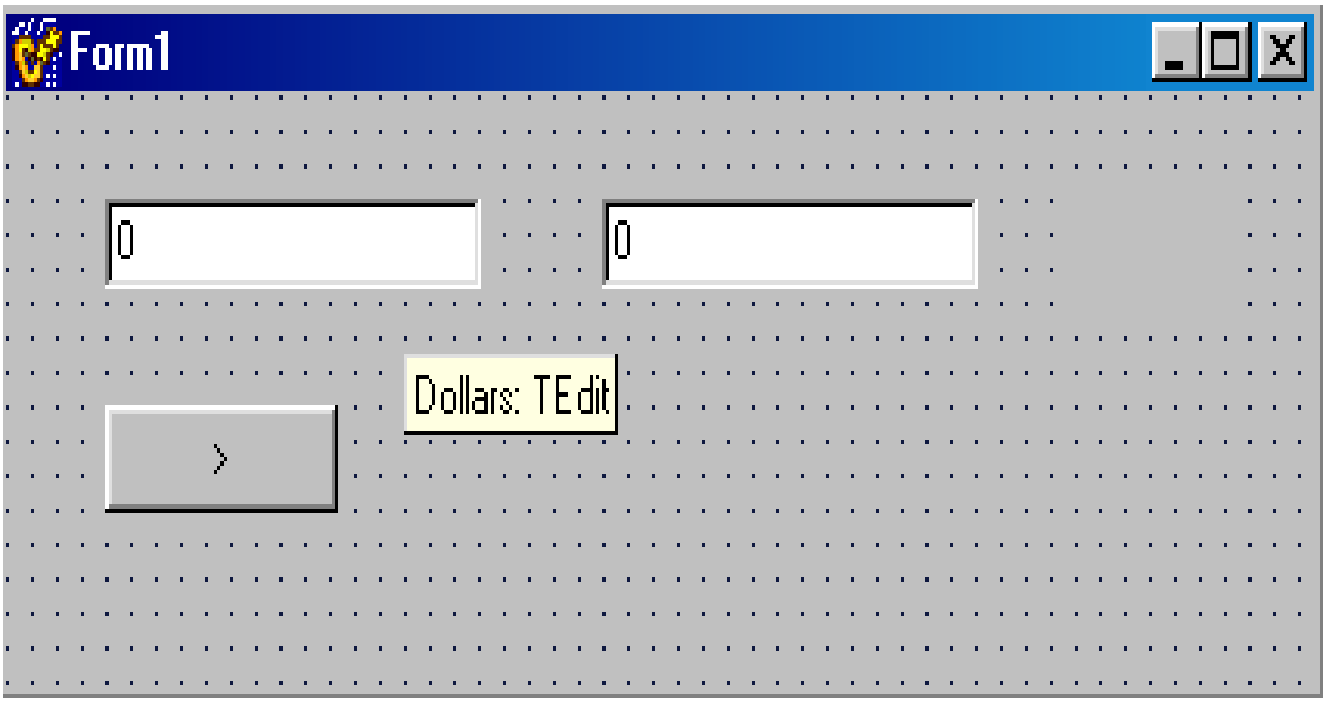
И так начинаем дизайн нашей будущей программы, на «визуальный проектировщик рабочих окон» (Форму) поместим четыре визуальных компонента, это два поля Edit1, Edit2, одно Label и компонент Button.



Форма, на которой мы размещаем компоненты, покрыта мелкой сеткой.

При установке компонентов происходит автоматическое притягивание к ее узлам такой подход к проектированию внешнего вида очень удобен, а расстояние между линиями сетки нетрудно изменить.

И вот мы видим окно нашей будущей программы и ее компоненты, эти компоненты подписаны своими начальными именами (Label, Edit и т.д.) мы заменим их на новые с помощью инспектора объектов и получим:



1.Edit1-Dollarss.

2.Edit2-Rate (Rate-курс обмена).

3.Label1-Убираем начальный текст Label1.

4.Button-TotalButton >.

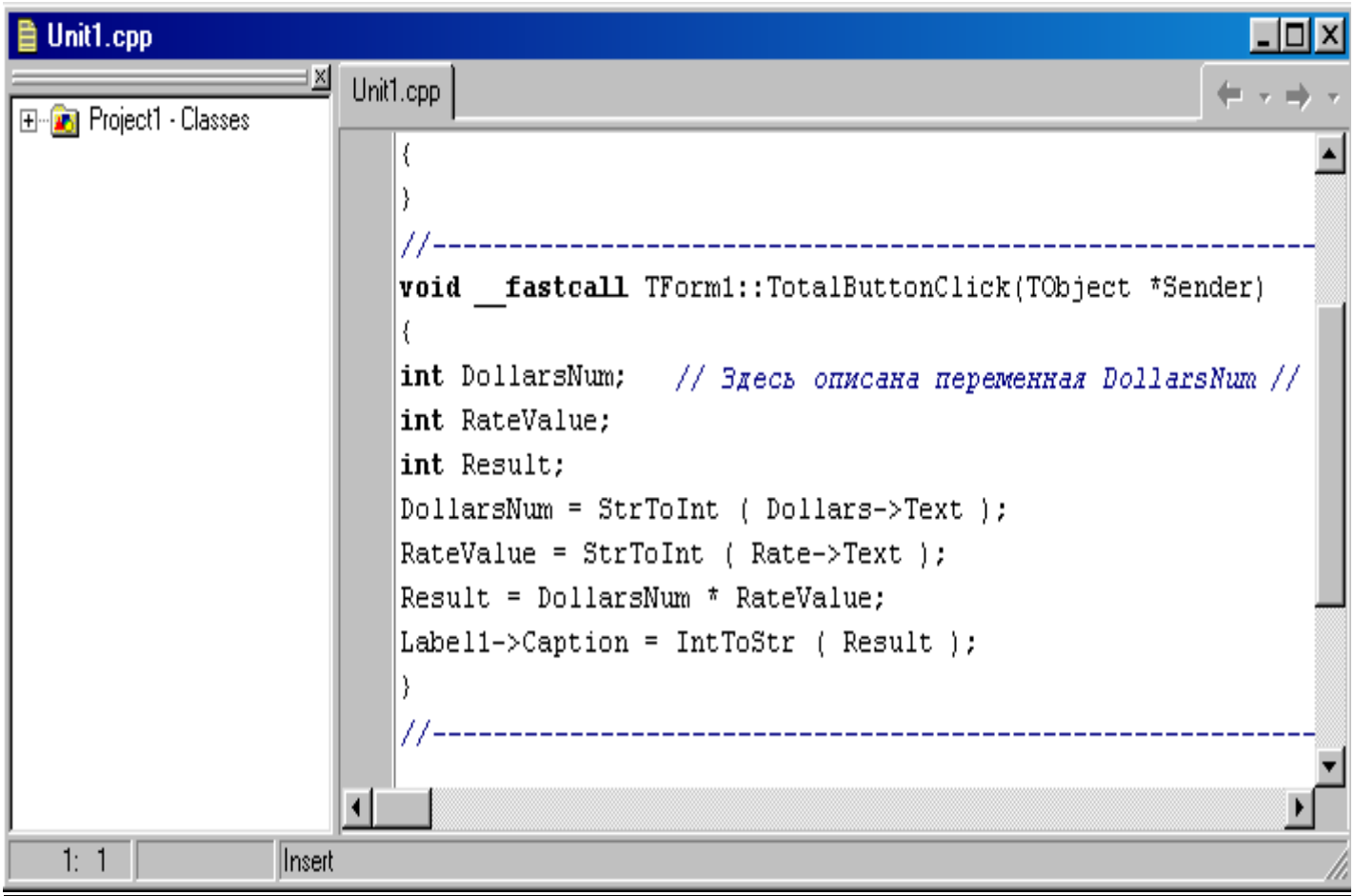
А так же присвоим 0 в качестве начального значения полей Edit1 и Edit2.

Если даже сейчас мы «от компилируем» (соберем) нашу программу то она не будет выполнять никаких действий т.к. мы не придали действующее значения наших компонентов, проще говоря, не запрограммировали ее.

Начинаем программировать.

На этом этапе создания программы мы будем в ручную вводить код на языке C++, который будет производить необходимые вычисления - перемножение двух чисел: числа долларов и текущего курса. Операцию по перемножению двух чисел о с выводом результата на экран будет выполнят код кнопки TotalButton.

Программный код мы будем писать в окне редактора кнопки TotalButton:



И так мы видим окно редактора кода кнопки TotalButton, *и в нем уже написанный код:*

```
//---------------------------------------------------------------------------
void __fastcall TForm1::TotalButtonClick(TObject *Sender)
{
```

```
int DollarsNum;   // Здесь описана переменная DollarsNum //
int RateValue;
int Result;
DollarsNum = StrToInt ( Dollars->Text );
RateValue = StrToInt ( Rate->Text );
Result = DollarsNum * RateValue;
Label1->Caption = IntToStr ( Result );
}
//---------------------------------------------------------------------------
```

1) Первая строчка это автоматически созданный текст C++, в нем написано что этот программный код является кодом кнопки TotalButton расположенной на визуальном проектировщике рабочих окон, форме№1 (Form1).

2)Фигурные скобки, одна в начале другая в конце {…} обязательны. Эти скобки определяют смысловые границы, внутри которых должен располагаться наш программный код.

3)В третьей строке мы видим: ***int*** *DollarsNum; это переменная, которая хранит в себе число долларов, обратим внимание как она написана:* ***int (****integer****)–****это тип, который обозначает что переменная, которая будет следом будет хранить число, обратим внимание что в конце переменной стоит ; это принято в C++, что в конце логически законченной части текста ставится точка с запятой. Строка выделенная синим цветом // Здесь описана переменная DollarsNum // это комментарий, когда программа становится все больше и больше, запоминать, что делается в той или иной ее части, становится все сложнее и сложнее. Через месяц можно полностью забыть, что мы напрограммировали в каком-то проекте, а уж постороннему человеку разобраться в чужом тексте, даже аккуратно написанном, крайне сложно. Поэтому профессиональные программисты очень подробно комментируют свои тексты.*

*4) Переменная-****int*** *RateValue;*

*5****)*** *Переменная-****int*** *Result;*

6) DollarsNum= StrToInt (Dollars->Text);

Функция преобразования строки в число называется StrToInt (StrToInt-это сокращение от английских слов String To Integer. Обратим внимание на использование строчных и заглавных

букв. Каждая стандартная функция имеет свой тип, точно так же, как и переменная. Тип функции StrToInt-целое число.

И так на надо передать функции StrToInt текстовую строку из поля ввода Dollars. Доступом к содержимому этого поля записывается конструкцией Dollars->Text, значит вызов StrToInt будет выглядеть так: StrToInt (Dollars->Text);

7) RateValue = StrToInt (Rate->Text);

В этой строке мы так же как и в шестой, передаем функции StrToInt текстовую строку из поля ввода Rate.

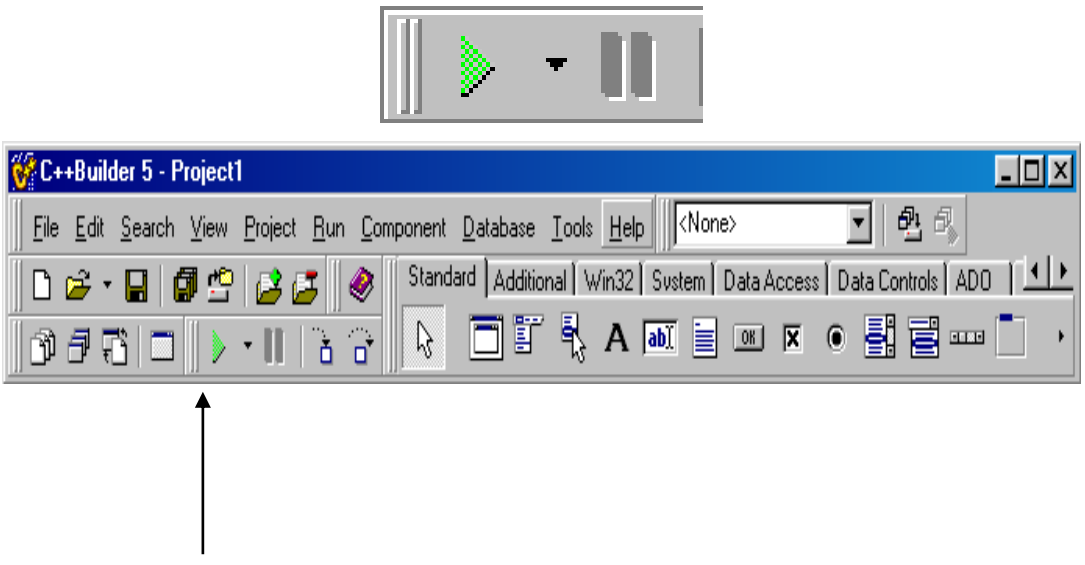
*8)Result = DollarsNum * RateValue;*

Эта строка перемножает значения, полученные из полей DollarsNum и RateValue.
Result-результат, который будет выводится на компонент Lable1.

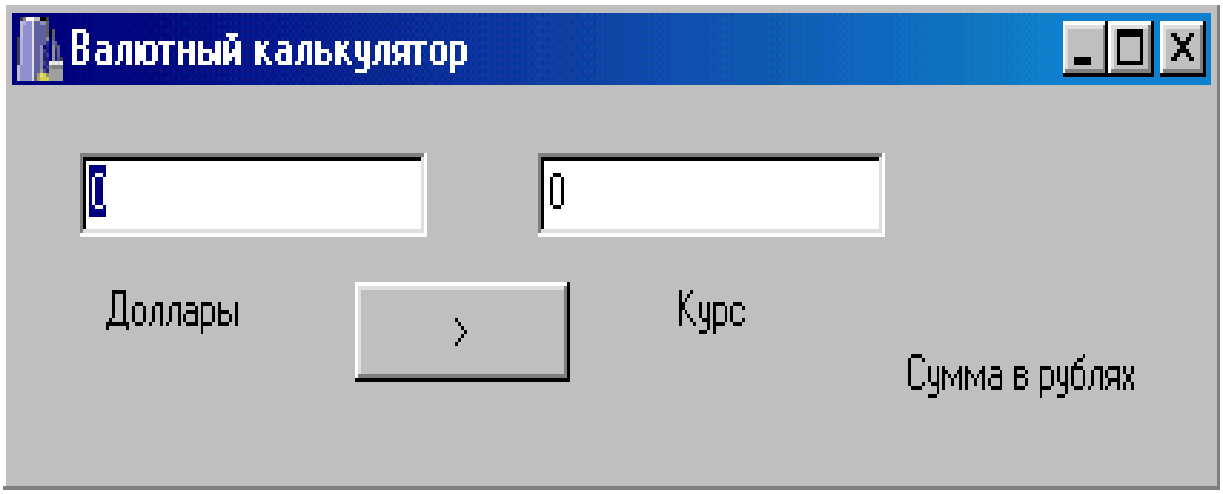
9) Label1->Caption = IntToStr (Result)

Здесь написано, что значение полученное в результате вычисления в восьмой строке (Результата), передается на поле Lable с помощью функции IntToStr.

И так текст готов, теперь мы с можем собрать нашу программу (откомпилировать) и если
в ней не будет никаких ошибок то программа соберется и будет работать. Что бы собрать программу, нужно запустить компилятор, запустить его можно при помощи клавиши F9 или значка на панели инструментов, выглядит он так:



После сборки программа выглядит так:



она выполняет все задачи которые мы задумали, то есть переводит доллары в рубли.

В первое поле мы вводим количество долларов, а во второе рублей и при нажатии кнопки, перемножаем два значения, выводя результат на экран.

Обычный блокнот

Запустите Borland C++ Builder 6 и создайте новый проект с именем edit. Поскольку текстовый редактор предназначен для работы с текстовыми строками, нам понадобится компонент Memo из вкладки Standard. Поместите его в центре формы проекта. Теперь поместите на форму компонент MainMenu из той же вкладки. Наконец, добавьте на форму три стандартные кнопки BitButton из вкладки Additional. Они понадобятся для формирования в программе органов управления для открытия и сохранения файлов и для других функций. Поскольку редактор должен при открытии и сохранении файлов обращаться к дисковым накопителям, необходимо добавить на форму диалоги Open Dialog и Save Dialog из вкладки Dialogs. Кроме того, добавьте на форму диалог Font Dialog из этой же вкладки. Он облегчит нам задачу настройки шрифтов из программы. Эти компоненты являются невидимыми при работе программы, поэтому место их размещения на форме не имеет значения. Но все же лучше располагать такие компоненты без наложения, для того чтобы они не скрывали друг друга. Последний компонент, который необходимо поместить на форму, — PopupMenu из вкладки Standard. Он позволит создать в программе контекстное меню, появляющееся при нажатии правой кнопки мыши.

Теперь разместим выбранные компоненты на форме и настроим их свойства. Начнем с главного компонента — формы приложения. Выберите в инспекторе объектов в поле селектора объектов Form1. *Измените свойство* Caption *этого объекта на название программы редактора Edit. Размеры формы установите с помощью мыши, захватив любой угол или сторону формы и перемещая их в любом направлении. Размер формы можно будет поправить при необходимости на любом этапе разработки программы. В свойстве* Position *установите значение* poDesigned. *Это обеспечит центровку нашего приложения при запуске по центру экрана. Щелкните по компоненту* MainMenu *левой кнопкой мыши. При этом откроется окно дизайнера меню. В свойстве* Caption *инспектора объектов введите название* &File *и нажмите клавишу <Enter>. На форме приложения при этом появится строка главного меню. Таким образом, на форме можно будет видеть поле главного меню и учитывать его размеры. Теперь можно временно закрыть окно дизайнера меню и приступить к размещению остальных компонентов на форме. Вначале разместите на форме компонент* Memo *так, чтобы он занимал всю ее свободную область за исключением правой полосы для кнопок. Кнопки расположите друг под другом, не изменяя при этом их размеров. В оставшейся свободной части формы можно расположить оставшиеся невидимые компоненты приложения. Не забывайте, что перемещать компоненты можно группами, предварительно выделив эти компоненты с помощью мыши. В результате после размещения всех компонентов должна получиться форма, подобная приведенной на рис. 21.1.*

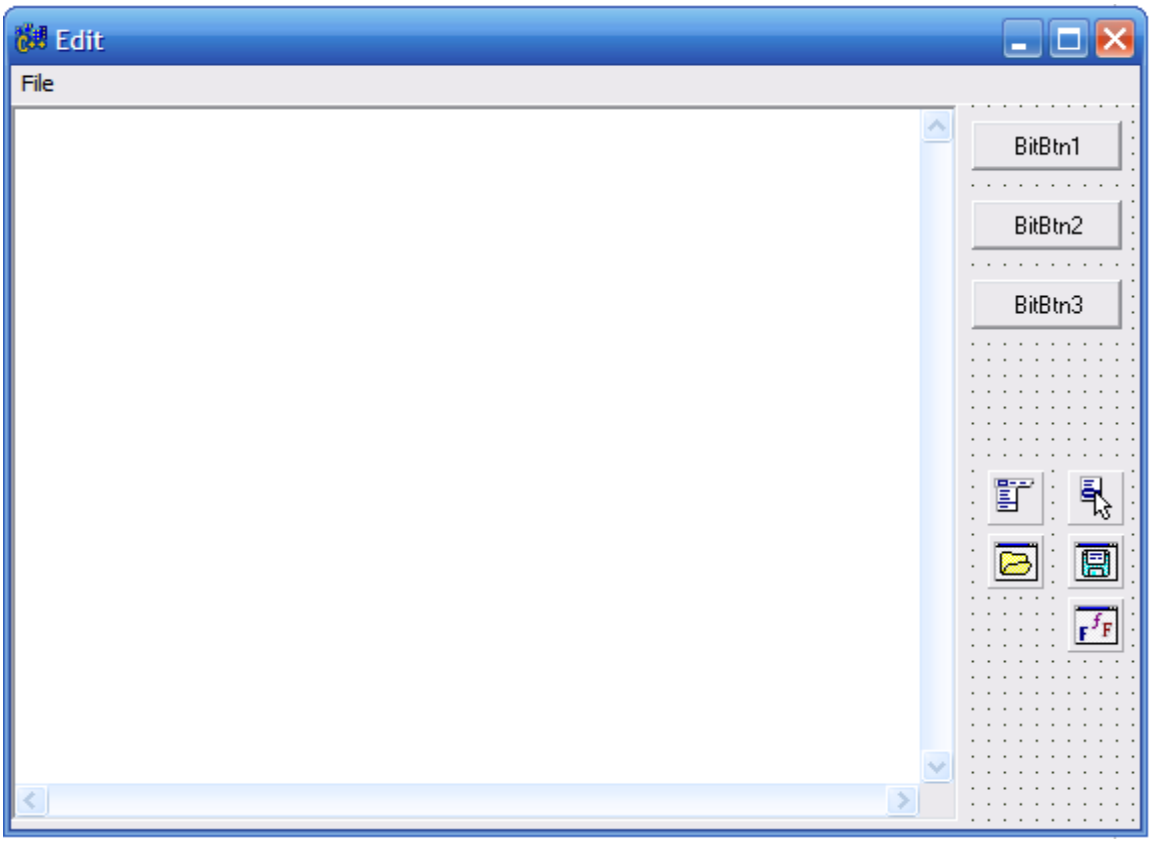


Рис. 21.1. Окно формы программы

Теперь приступим к изменению свойств остальных компонентов. Начнем с компонента Memo, выбрав его в инспекторе объектов. Вначале удалим надпись Memo1 с компонента, для того чтобы она не появлялась при запуске приложения. Для этого изменим свойство Lines, щелкнув по кнопке с тремя точками правее надписиTstrings этого свойства. При этом откроется окно редактора строки String List Editor, в котором необходимо удалить строку Memo1 и нажать кнопку ОК. Теперь добавим для этого компонента линейки прокрутки для обеспечения возможности просмотра и редактирования больших областей текста с помощью мыши. Для этого в свойстве ScrollBars необходимо выбрать значение ssBoth. Для того чтобы работало контекстное меню, необходимо в свойстве PopupMenu выбрать значение PopupMenu1. Это значение стало доступным после помещения на форму проекта диалога PopupMenu. Настройку данного меню произведем позже.

Настроим свойства кнопок в соответствии с выполняемыми ими функциями. Для этого изменим поочередно свойство Caption для всех трех кнопок на Open, Save и Font соответственно. Для того чтобы тип курсора изменялся при попадании на эти кнопки, изменим еще одно групповое свойство кнопок. Для этого с помощью мыши выделите все три кнопки, и в появившемся объекте 3 items selected в инспекторе объектов измените свойство Cursor на значение crHandPoint.

Настройку главного меню произведем с помощью дизайнера меню, вызываемого двойным щелчком левой кнопки мыши по компоненту MainMenu. Ранее мы уже ввели один пункт меню под названием &File. Символ "ампер-санд" перед File производит выделение первой буквы названия в меню для быстрого доступа, путем ее подчеркивания. Следующий по горизонтали пункт меню назовем &Edit. Теперь заполним пункты меню по вертикали. Щелкните левой кнопкой мыши по пункту меню File, а затем ниже него по пустому полю. СвойствоCaption этого пункта меню измените на Open и нажмите клавишу <Enter>. Следующий пункт назовите Save. Аналогично расширим по вертикали пункты меню Edit, добавив пункты Font и Clear.
На этом этап создания главного меню программы редактора завершен. Все введенные пункты меню теперь будут присутствовать в приложении, но пока не будут выполняться, поскольку не созданы функции для выполнения соответствующих команд. Создадим их. Для этого воспользуемся компонентами диалогов, помещенных ранее на форму приложения. С

помощью диалогов работы с файлами можно задать направленность приложения путем определения типов открываемых файлов. Это делается с помощью свойстваFilter.

Выберите в инспекторе объектов компонент OpenDialog1 и нажмите кнопку с тремя точками правее свойстваFilter. При этом появится диалоговое окно Filter Editor, в котором производится настройка типов файлов. Окно разбито на два поля. В левом поле Filter Name вводятся строки пояснения. В правом поле Filter — расширения файлов. Для указания нескольких типов файлов необходимо записать их через точку с запятой. Если задать несколько строк с пояснениями и типами файлов, то при работе приложения их можно будет выбирать с помощью выпадающего списка. Заполните оба поля диалогового окна Filter Editor в соответствии с рис. 21.2 и нажмите кнопку OK.

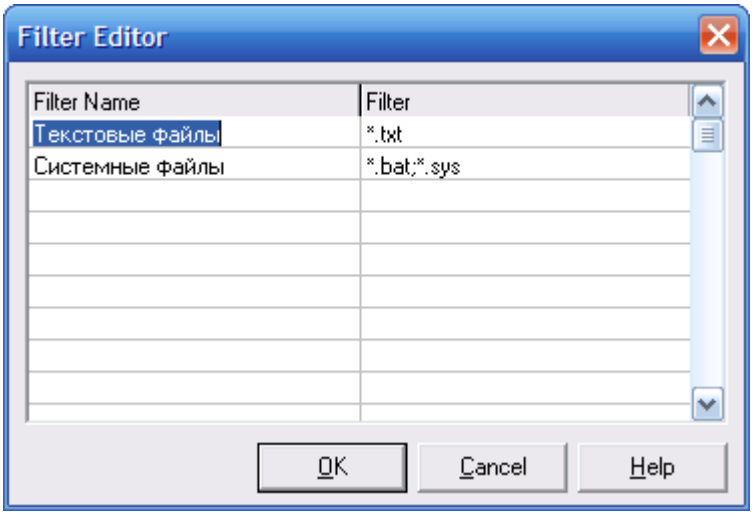


Рис. 21.2. Окно Filter Editor

Аналогично выполните настройку компонента SaveDialog1. Для корректности работы приложения оба эти диалога должны иметь одинаковую настройку свойства Filter. Чтобы облегчить эту работу и избежать ошибок, можно использовать два способа. Первый способ уже был показан ранее, когда несколько компонентов кнопок выделялось в группу и задавалось свойство одновременно для всех выделенных компонентов. Второй способ заключается в том, что, задав свойство для одного из компонентов, можно скопировать и вставить его для других компонентов с помощью комбинации клавиш <Ctrl>+<C> и <Ctrl>+<V> соответственно. Так, после задания свойства Filter компонента OpenDialog1 с помощью Filter Editor в поле этого свойства появится запись: Текстовые файлы |.txt| Системные файлы |*.bat; *.sys. Выделив эту строку с помощью клавиш <Home> и <Shift>+<End>, скопируйте ее с помощью клавиш <Ctrl>+<C>. Теперь можно вставить эту строку в свойство Filter для компонента SaveDialog1. Замечательно то, что работу этих компонентов можно проверить еще до выполнения программы. Дело в том, что после*

задания свойства Filter этих компонентов двойной щелчок по компонентам приводит к открытию диалога на основе заданных свойств. Например, после двойного щелчка по компоненту SaveDialog1 откроется окно, изображенное на рис. 21.3.

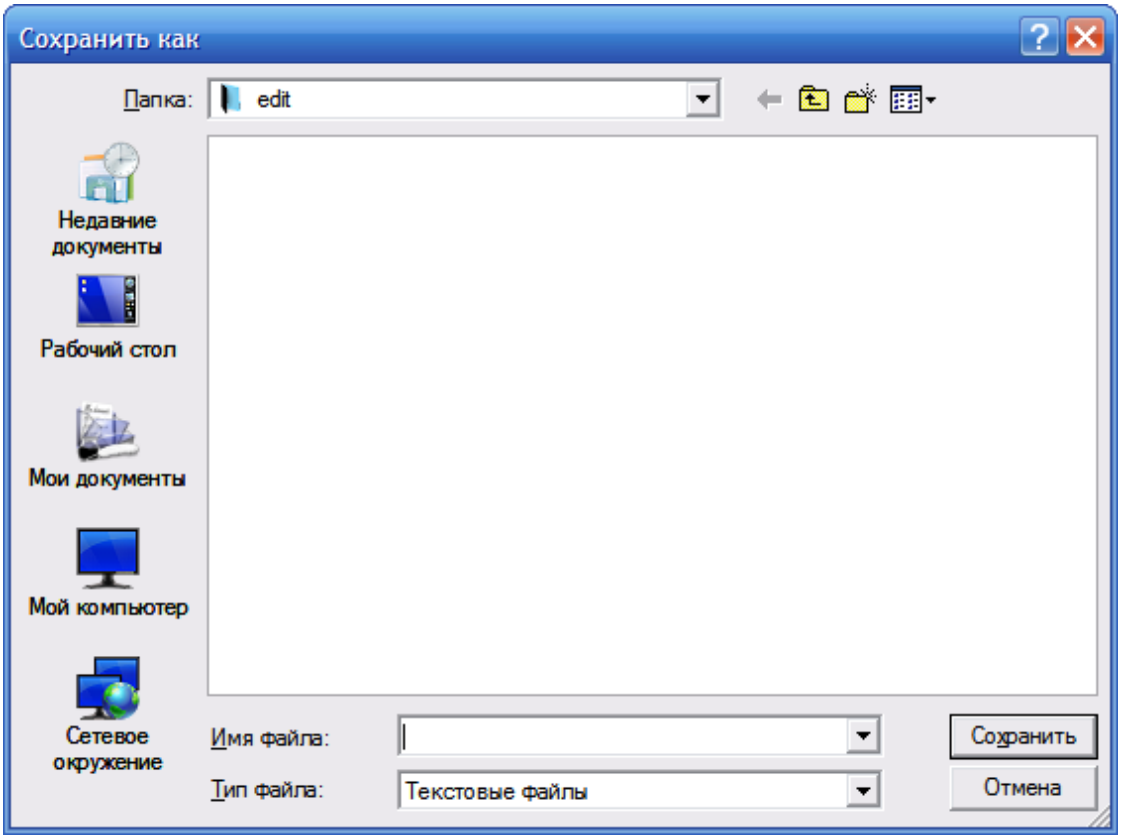


Рис. 21.3. Окно компонента Save Dialog

Теперь настроим компонент FontDialog1. Основным свойством данного компонента является Font, с помощью которого можно выбрать тип шрифта, его цвет, размер, стиль и т. п. Эти установки будут использоваться в программе редактора по умолчанию. С помощью свойства Options можно настроить вид и поведение диалога. Например, можно добавить в него кнопку справки, установив для подсвойства fdShowHelp значение true. Нам осталось произвести настройку контекстного меню PopupMenu. Она производится аналогично настройке главного меню MainMenu. Добавим в это меню пункты с названиями Font и Clear через свойство Caption, как это показано на рис. 21.4.

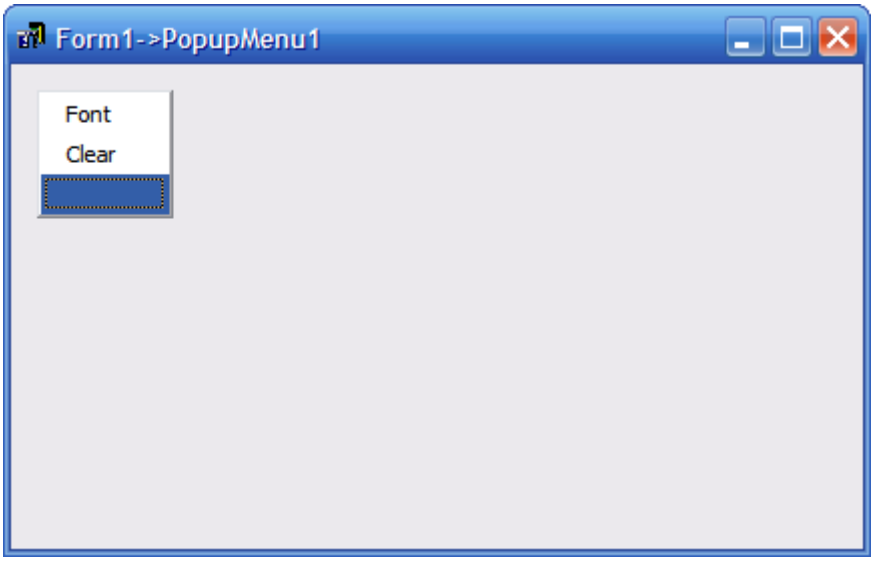


Рис. 21.4. Настройка контекстного меню PopMenu

Кроме того, добавьте название PopupMenu1 *в свойство* PopupMenu *главной формы приложения.*

На этом настройка компонентов завершена и получена визуальная оболочка программы редактора. Однако без программных строк это приложение не будет работать. При запуске приложения будут открываться пункты меню, вводиться и редактироваться текст, но только потому, что эти функции заложены в сами компоненты. Нажатие на кнопки и вызов команд из главного меню не будут приводить к каким-либо действиям программы, поскольку не заданы функции обработки событий. Вдохнем в приложение жизнь. В программе редактора имеется четыре типа операций. Это открытие и сохранение файлов, настройка шрифтов и очистка окна редактора. Все эти операции можно вызвать разными способами. Например, с помощью главного меню, кнопок или контекстного меню. Можно написать функции для каждого способа, но это усложнит программу и внесет трудности при ее создании и отладке. Правильнее будет создать такую функцию один раз, а затем задать ее выполнение другим компонентам с помощью инспектора объектов. Создайте функцию для команды главного меню Open. Для этого откройте на главной форме пункт меню File и щелкните левой кнопкой мыши по команде Open. При этом активизируется редактор кода с заготовкой функции обработки. Введите между фигурными скобками следующий текст программы:

```
if (OpenDialog1->Execute() )
Memo1->Lines->LoadFromFile (OpenDialog1->FileName) ;
```

Обратите внимание на то, что среда разработки сама предоставляет выбрать функцию из множества разрешенных для компонента функций после введения символа стрелки. Эти

программные строки позволят при запуске диалога открытия файла загрузить содержимое открываемого файла в свойство Lines *компонентаMemo1 для отображения на экране. Теперь задайте выполнение этой функции кнопке команды* Open *с помощью инспектора объектов. Для этого щелкните левой кнопкой мыши по кнопке* Open *и выберите в инспекторе объектов вкладку* Events *(События). Раскройте список в событии* OnClick *и выберите в нем строкуOpen1Click. Таким образом будет задано, что при нажатии кнопки в программе редактора запустится та же функция открытия файла, что и для команды* Open *из пункта главного меню* File. *Запустите программу с помощью команды* Run *и убедитесь, что команды открытия файла работают и выбирают заданные нами типы файлов. Аналогично создадим функции для оставшихся трех типов команд. Начнем с команды сохранения файлов Save. Откройте на главной форме пункт меню* File *и щелкните левой кнопкой мыши по команде Save. При этом активизируется редактор кода с заготовкой функции обработки. Введите между фигурными скобками следующий текст программы:*

```
if (SaveDialog1->Execute() )
Memo1->Lines-> SaveToFile (SaveDialog1->FileName) ;
```

Эти программные строки позволят при запуске диалога сохранения файла загрузить содержимое свойство Linesкомпонента Memo1 *в сохраняемый файл. Задайте выполнение этой функции кнопке с командой Save с помощью инспектора объектов. Для этого щелкните левой кнопкой мыши по кнопке* Save *и выберите в инспекторе объектов вкладку Events. Раскройте список в событии* OnClick *и выберите в нем строку Save1Click. Таким образом будет задано, что при нажатии кнопки* Save *в программе редактора запустится та же функция сохранения файла, что и для команды* Save *из пункта главного меню* File*. Для команды* Font *из пункта* Edit *главного меню введите с помощью инспектора кодов строки:*

```
if (FontDialog1->Execute() )
Memo1-> Font=FontDialog1-> Font;
```

Это позволит изменять шрифт текста компонента Memo1 *при открытии диалога* FontDialog1 *Задайте выполнение этой функции кнопке с командой* Font *с помощью инспектора объектов, для чего раскройте список в событии* OnClick *и выберите в нем строку Font1Click.*

Теперь нужно добавить вызов этой функции в контекстное меню. Выделите на форме компонент PopupMenu1 *и в инспекторе объектов щелкните по кнопке с тремя точками в*

строке свойства Items. При этом откроется окноForm1 | PopupMenu. В этом окне щелкните на поле Font и в инспекторе объектов на закладке Events выберите для события OnClick строку Font1Click.

Наконец создадим функцию обработки очистки редактора. Для команды Clear из пункта Edit главного меню введите с помощью инспектора кодов одну-единственную строку:

```
Memo1->Clear ();
```

Эта программная строка будет производить очистку содержимого компонента Memo1. Аналогично предыдущему описанию назначьте выполнение данной функции через контекстное меню, выбрав строку Clear1Click для события OnClick.

На этом создание программы редактора заканчивается, и можно приступить к его проверке. Предварительно сохраните весь проект в отдельном каталоге, например, с именем Edit, а затем запустите его на выполнение. На экране должно появиться окно, аналогичное изображенному на рис. 21.5.

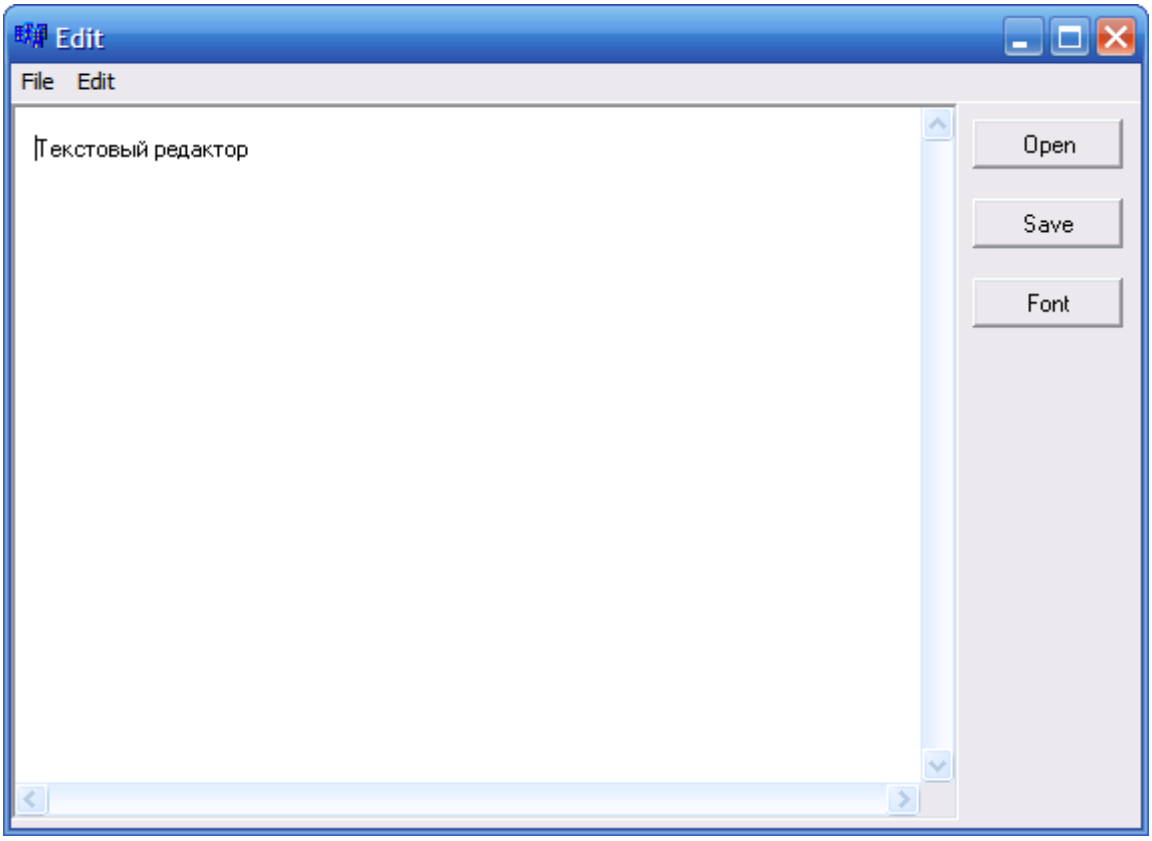


Рис. 21.5. Окно работающей программы

Нажмите кнопку Open и выберите какой-либо файл на диске. После чего загрузите его в редактор. Измените с помощью клавиатуры текст файла и сохраните его с новым именем, нажав кнопку Save. Нажмите кнопку Font и измените шрифт текста редактора. Убедитесь

в том, что текст изменился в соответствии с вашими установками. Из главного меню выполните команду Clear *и убедитесь, что поле редактирования очистилось. Проверьте работу остальных пунктов главного меню. Далее проверьте работу контекстного меню, нажав правую кнопку мыши на поле редактора и выбрав одну из двух команд. Обратите внимание на то, что при наведении курсора на кнопки он должен менять свой вид.*

Отличие между этими двумя программами (валютный калькулятор ,обычный блокнот) в том, что как вы заметили если в калькуляторе мы работаем с цифрами, в блокноте же мы просто пишем текст или читаем текст, т.е работаем с текстом. В калькуляторе мы обошлись всего тремя компонентами (TotalButton,Label,Edit),а в блокноте их довольно больше (Memo1, PopurMenu, MainMenu, BitButton, OpenDialog, SaveDialog, FontDialog). И их составляющий код слишком отличается друг от друга.

Но всё же если же подитожитьто то эти две программы сойдутся в одном, что хорошо подойдут для начинающих программистов ООП.

Заключение.

Мы живем в мире объектов. Стол, автомобиль, ручка, классная доска - все это объекты. Объекты характеризуются атрибутами. Так атрибутами автомобиля являются максимальная скорость, мощность двигателя, цвет кузова и т.д.

Помимо атрибутов объекты обладают некоторыми функциональными возможностями, которые в объектно-ориентированном программировании (ООП) называют операциями или методами. Так автомобиль может ездить, корабль - плавать.

Таким образом, объект инкапсулирует атрибуты и методы, скрывая от других объектов взаимодействующих с ним и использующих его функциональность, свою реализацию. Объект - это экземпляр некоторого класса объектов или просто класса. Так автомобиль Audi 6 является экземпляром класса автомобилей данной модели, приемник Sony SW-7600G так же будет представителем класса одноименных приемников.

Классы могут быть связаны друг с другом различными отношениями. Одним из основных таких отношений является отношение класс - подкласс, известный в объектно-ориентированном программировании как наследование. Например, класс автомобилей Audi 6 является подклассом легковых автомобилей, который в свою очередь входит в более крупный класс автомобилей, а последний является подклассом класса транспортных средств, который помимо автомобилей включает в себя самолеты, корабли поезда и т.д. Примером подобных отношений, являются системы классификации в ботанике и зоологии. Отношением, обратным наследованию, является обобщение или генерализация. Она указывает, что некий

класс, является более общим (обобщенным) классом другого класса. Класс транспортных средств, к примеру, является генерализацией классов автомобилей, самолетов и кораблей.

При наследовании все атрибуты и методы родительского класса наследуются классом-потомком. Наследование может быть многоуровневым, и тогда классы, находящиеся на нижних уровнях иерархии, унаследуют все свойства (атрибуты и методы) всех классов, прямыми или косвенными потомками которых они являются.

Помимо одиночного, существует и множественное наследование, когда класс наследует сразу нескольким классам. При этом он унаследует свойства всех классов, потомком которых он является. При использовании множественного наследования необходимо быть особенно внимательным, так как возможны коллизии, когда класс-потомок может унаследовать одноименные свойства, с различным содержанием.

При наследовании одни методы класса могут замещаться другими. Так, класс транспортных средств будет обладать обобщенным методом движения. В классах-потомках этот метод будет конкретизирован: автомобиль будет ездить, самолет - летать, корабль - плавать. Такое изменение семантики метода называется полиморфизмом. Полиморфизм - это выполнение методом с одним и тем же именем различных действий в зависимости от контекста, в частности, от принадлежности тому или другому классу. В разных языках программирования полиморфизм реализуется различными способами. Например, в C++ он реализован с помощью механизма виртуальных функций.

В ходе выполнения курсовой работы были закреплены теоретические знания в области объектно-ориентированного программирования на языке C++ . Также я повторил свои знания при создании программ "Валютный калькулятор" и "Обычный блокнот".

Применение объектно-ориентированного подхода к программированию и широкие возможности языка C++ в среде визуального программирования C++ Builder является мощным и гибким средством при создании программ имитирующих поведение объектов реального мира.

Использованная литература:

1. «Специальный справочник C++» Борис Карпов, Татьяна Баранова (издательство Питер).

2.«C++ для Чайников» Стефан Р.Дэвис (издательство Диалектика).

3. «Самоучитель программирования на Borland C++ Bilder 5.0» С. Бобровский (издательство Десс Ком).