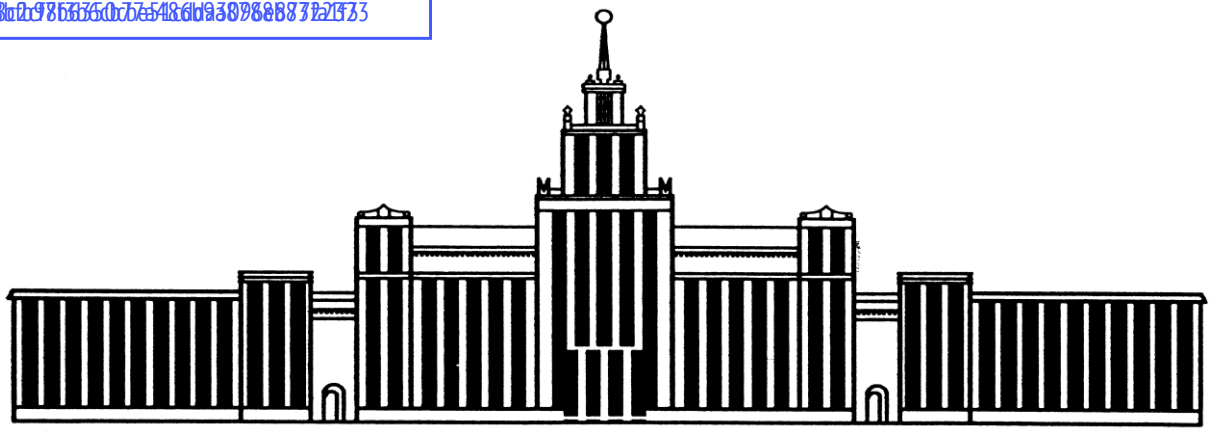
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ



ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

004.4(07)
О-535

Оленчикова Т.Ю., Сартасова М.Ю.

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ В C++ Builder 10.0

Учебное пособие

Челябинск
2019

Министерство науки и высшего образования Российской Федерации
Южно-Уральский государственный университет
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

004.4(07)
О-535

Оленчикова Т.Ю., Сартасова М.Ю.

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ В C++ Builder 10.0

Учебное пособие

Челябинск
Издательский центр ЮУрГУ
2019

УДК 004.4(07)
О535

Одобрено
учебно-методической комиссией факультета математики, механики и компьютерных технологий

Рецензенты:

Заместитель директора по научной работе Челябинского института путей сообщения – филиала УрГУПС, к.т.н., доцент А.Н. Давыдов

Доцент кафедры математики и информатики ЧОУ ВО «МИДиС», к.ф.-м.н. Чеботарев С.С.

Оленчикова, Т.Ю.
О535 Визуальное программирование в C++ Builder 10.0: учебное пособие /Т.Ю. Оленчикова, М.Ю.Сартасова. – Челябинск: Издательский центр ЮУрГУ, 2019. – 110 с.

Данное пособие содержит методический и справочный материал по визуальному программированию в интегрированной среде C++ Builder 10.0. В пособии рассмотрены возможности C++ Builder 10.0 для решения научных и прикладных задач, приведены вопросы для самоконтроля и примерные темы курсовых работ. Предложенная структура пособия помогает выделить главные аспекты изучаемых визуальных компонентов, организовать и конкретизировать учебный процесс.

Учебное пособие «Визуальное программирование в C++ Builder 10.0», подготовлено по дисциплине «Визуальное программирование» в соответствии с Федеральным государственным образовательным стандартом высшего профессионального образования для студентов, обучающихся по направлениям «Прикладная математика», «Прикладная математика и информатика».

УДК 004.4(07)

ВВЕДЕНИЕ

Настоящее пособие разработано для дисциплины «Визуальное программирование», соответствующего учебным планам направлений 01.03.02 «Прикладная математика и информатика», 01.03.04 «Прикладная математика». Цель издания – способствовать теоретической подготовке и овладению практическими навыками использования технологии визуального программирования. Формируемые компетенции – ОПК-3 (способностью к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям) и ПК-7 (способностью к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения).

Пособие содержит задачи и упражнения по изучению визуальных компонентов, их свойств и методов обработки. В каждом подразделе приведены краткие теоретические сведения по рассматриваемой теме, даны примеры решения типовых задач. Задания рекомендуется использовать для самостоятельной работы студентов для закрепления практических навыков визуального программирования. Все задания, включённые в пособие, могут быть использованы на лекционных и лабораторных занятиях по соответствующим разделам, а также для текущей и промежуточной оценки знаний обучающихся.

1 ОСНОВНЫЕ ВОЗМОЖНОСТИ C++BUILDER

1.1 Обзор основных возможностей

C++ Builder (другое название – Turbo C++) является объектно-ориентированной средой программирования, предназначенной для разработки и отладки программ в операционной системе Windows. Интегрированная среда C++ Builder обеспечивает высокую скорость визуальной разработки, продуктивность повторно используемых компонент в сочетании с мощью языковых средств C++, усовершенствованными инструментами и разномасштабными средствами доступа к базам данных. C++Builder может быть использован везде, где требуется дополнить существующие приложения расширенным стандартом языка C++, повысить быстродействие и придать пользовательскому интерфейсу качества профессионального уровня.

1.2 Типы данных

В C++Builder имеются все типы данных, которые Вам встречались в C и C++, однако существует много дополнительных типов, наиболее важные из которых мы сейчас рассмотрим.

1.2.1 Тип AnsiString

Тип AnsiString представляет собой класс для реализации текстовой строки переменной длины.

Конструкторы:

- AnsiString( ) – создает пустую строку;
- AnsiString(char *str) – создает строку, содержащую str;
- AnsiString(int n) – создает строку, содержащую число n, в виде строки;
- AnsiString(long n) – создает строку, содержащую число n, в виде строки;
- AnsiString(double n) – создает строку, содержащую число n, в виде строки.
- Методы:
- char *c_str( ) – преобразует AnsiString в char *;
- int ToInt( ) – преобразует AnsiString в int;
- double ToDouble( ) – преобразует AnsiString в double;
- int Length( ) – определяет длину;
- bool IsEmpty( ) – определяет, является ли строка пустой.

Перегруженные операторы:

- AnsiString = char * – преобразует char *в AnsiString;
- AnsiString = int – преобразует int в AnsiString;
- AnsiString = double – преобразует double в AnsiString;
- AnsiString + AnsiString – сцепляет строки;
- AnsiString + char * – сцепляет строки;
- AnsiString + int – сцепляет строку и число;
- AnsiString + double – сцепляет строку и число;
- AnsiString = = AnsiString – сравнивает строки. Аналогично перегружаются операторы !=, >=, >, <=, <.

1.2.2 Тип TStrings

Тип TStrings представляет собой класс для реализации списка текстовых строк переменной длины.

Основные свойства:

- Count – количество строк (тип int);
- Strings – массив строк (тип AnsiString[ ]).

Конструктор:

• TStrings( ) – создает пустой список строк.

Основные методы:

• Add(AnsiString s) – добавление строки в конец списка;

• Insert(int Pos, AnsiString s) – добавление строки в указанную позицию списка;

• Delete(int Pos) – удаление указанной строки из списка;

• SaveToFile(AnsiString FileName) – сохранение в текстовом файле, где FileName – имя файла;

• LoadFromFile(AnsiString FileName) – загрузка из текстового файла;

• Clear( ) – очистка списка.

1.2.3 Тип TColor

Целое число, обозначающее цвет. В таблице 1.1 приводятся предопределенные символические константы цветов.

Таблица 1.1 – Предопределенные константы цветов

Константа	Цвет	Константа	Значение
clAqua	Голубой	clBackground	Цвет фона рабочего стола Windows
clBlack	Черный	clActiveCaption	Цвет заголовка активного окна
clBlue	Синий	clMenu	Цвет фона меню
clDkGray	Темносерый	clWindow	Цвет фона окна Windows
clGray	Серый	clMenuText	Цвет текста меню
clGreen	Зеленый	clWindowText	Цвет текста окна Windows
clLime	Светлозеленый	clActiveBorder	Цвет бордюра активного окна
clLtGray	Светлосерый	clInactiveBorder	Цвет бордюра неактивного окна
clMaroon	Коричневый	clBtnFace	Цвет поверхности кнопки
clNavy	Темносиний	clBtnShadow	Цвет тени кнопки
clPurple	Фиолетовый	clBtnText	Цвет текста на кнопке
clRed	Красный	clGrayText	Цвет недоступных элементов
clWhite	Белый	clInfoBk	Цвет фона информационных объектов
clYellow	Желтый		

1.3 Начало работы с C++Builder

Запустите C++ Builder, для этого нажмите «ПУСК/ПРОГРАММЫ /Программирование/Borland Developer Studio 2006/Turbo C++» (рисунок 1.1). У вас должна появиться заставка (рисунок 1.2).

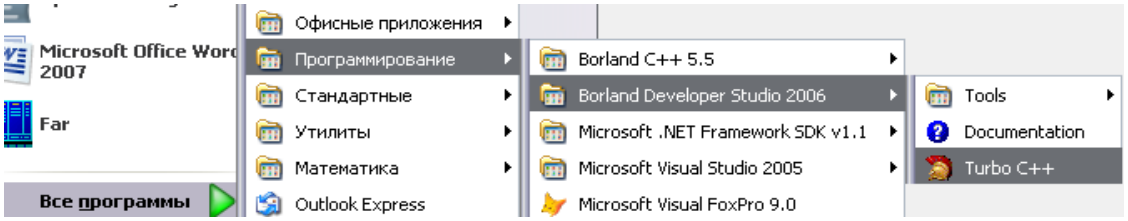


Рисунок 1.1 – Запуск Turbo C++

Рисунок 1.2 – Заставка

Не торопитесь, C++Builder 10.0 запускается относительно медленно. После запуска интегрированной среды разработки открывается главное окно (рисунок 1.3).

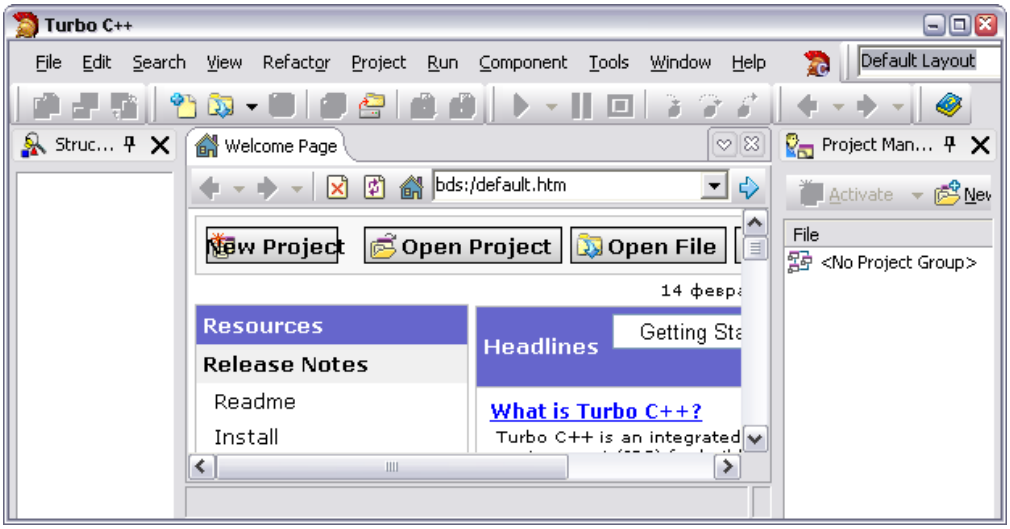


Рисунок 1.3 – Главное окно C++ Builder

Интерфейс, используемый в C++ Builder 10.0, называется средой быстрой разработки приложений RAD (Rapid Application Development). При разработке программы многие стандартные и рутинные операции выполняет за человека компьютер. Например, при создании проекта VCL автоматически

создается заготовка для функции WinMain и для функций обработки событий. Также существенным является то, что созданные приложения можно компилировать как в среде Windows, так и в среде Linux.

Для создания приложения с использованием форм выбираем пункт меню «File/New/VCL Forms Application – C++ Builder» (рисунок 1.4).

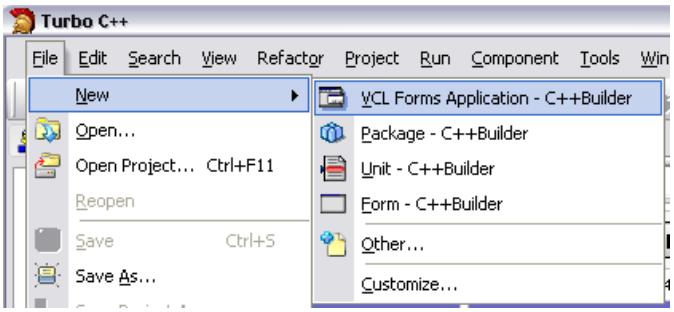


Рисунок 1.4 – Пункты меню «File/New»

Наиболее интересными являются 4 части рабочего стола (рисунок 1.5): Палитра компонентов, Редактор форм, Инспектор объектов и Редактор кода.

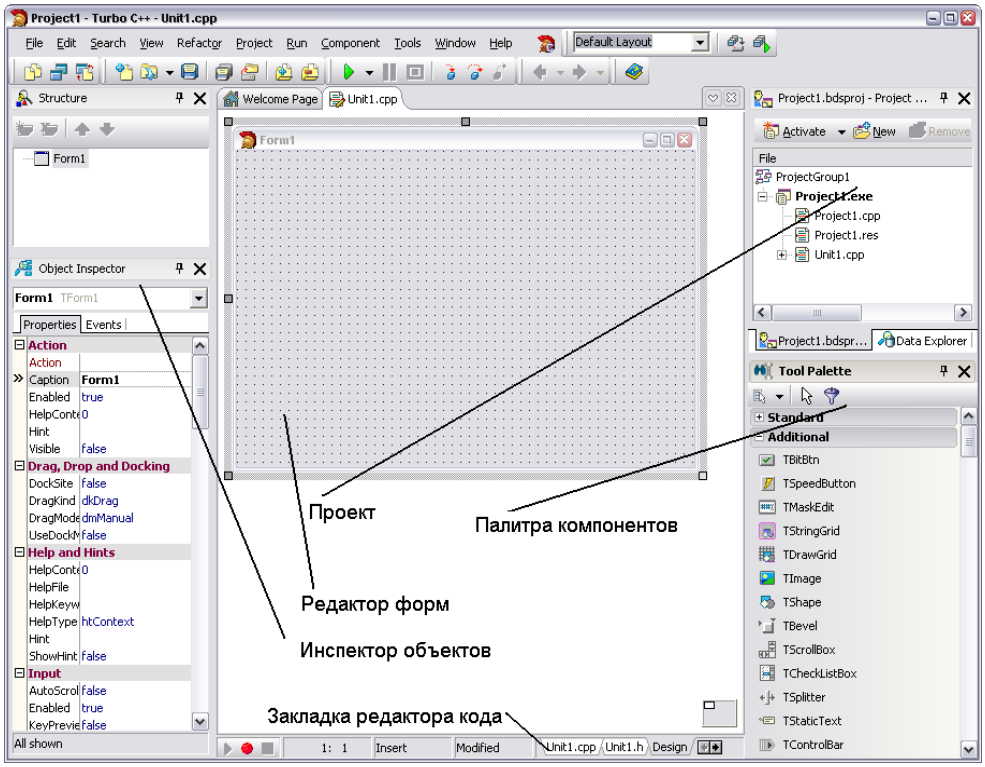


Рисунок 1.5 – Рабочий стол проекта

Палитра компонентов представляет собой набор «строительных» материалов и механизмов, из которых строится программа (в дальнейшем будем называть их *компонентами*). Палитра состоит из нескольких закладок, на каждой закладке расположены свои компоненты.

Редактор форм является своего рода «строительной площадкой», на которой из «строительных материалов» возводят будущее «здание» программы.

Инспектор объектов позволяет задавать свойства компонентам, придавая им широкое многообразие, а также приписывать им различные реакции на события – действия, которые они должны выполнять в той или иной ситуации. Инспектор объектов состоит из двух закладок Properties (Свойства) и Events (События). На обеих закладках расположены таблицы, состоящие из двух столбцов: в первом столбце записано название свойства или события, во втором – его значение. Количество строк зависит от типа объекта.

Свойства объекта подразделяются на *простые* и *составные*. *Простые* свойства имеют одно значение какого-либо типа. *Составные* представляют собой набор подсвойств, каждое из которых имеет свое значение. Отличительным признаком составного свойства является знак + или – в квадратике.

По способу задания значения свойства делятся на три группы: вводимые, выбираемые и задаваемые в диалоге. Если щелкнув мышью по значению свойства, Вы не увидите ничего, кроме поля ввода, то это вводимое свойство и его следует вводить с клавиатура. Если в конце поля вы увидите кнопку с треугольником вниз  , то это выбираемое свойство. Последующий щелчок по ней приведет к раскрытию списка для выбора одного из допустимых значений. Если в конце поля вы увидите кнопку с тремя точками  , то при щелчке по этой кнопке (или двойном щелчке по значению свойства) откроется диалоговое окно для выбора значений. Это окно может быть специфическим для данного свойства.

Перейдем на закладку Events (События). В первом столбце таблицы событий находятся название событий. Каждое событие возникает при определенных ситуациях, например, при нажатии кнопки мыши на объекте или клавиши клавиатура (для каждого типа объектов свой перечень событий). Во втором столбце – функции, которые вызываются при возникновении события. В таблицу закладки Events занесены имена функций, текст самих функций заполняется в окне Редактора кода. Двойной щелчок по значению события (имени функции в таблице) приводит к открытию окна Редактора кода в месте записи данной функции. Если поле значения события пустое, то двойной щелчок по нему приведет к генерации некоторого имени функции и создания заготовки функции вида:

```
Тип_возвращаемого_значения имя_класса имя_функции(параметры)
{
}
```

Между открывающейся и закрывающейся фигурными скобками можно писать тело функции на языке C++.

Рассмотрим процесс проектирования первой программы, которая выдает текст: Hello World! Для этого на Палитре компонент найдем компонент (рисунок 1.6), который называется Label – (Надпись, дословный перевод – Метка).

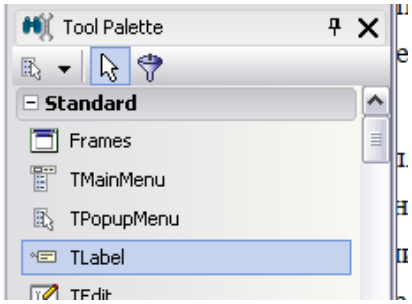


Рисунок 1.6 – Визуальный компонент Label

Щелкнем по нему мышью, а затем щелкнем в левой верхней части Редактора форм. В месте щелчка появится маленькая надпись Label1 (рисунок 1.7).

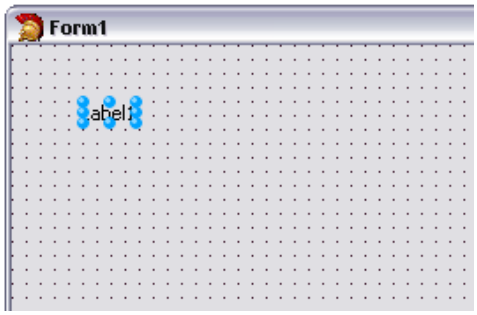


Рисунок 1.7 – Объект Label на форме

Зададим необходимые свойства. Для этого в инспекторе объектов найдем свойство Caption (Заголовок) и напротив его вместо Label1 напишем “Hello World!”.

Найдем свойство Font (Шрифт) перед которым имеется знак + в прямоугольнике (признак составного свойства т.е. свойства с подсвойствами). Щелкнем по +, подсвойства раскроются. Изменим подсвойства Color (Цвет), Name (Имя) и Size (Размер). Каждое из этих свойств изменяется по-разному.

Для свойства Size можно просто поменять значение, например, вместо стандартных 8 написать 64. При этом текст “Hello World” увеличится в размере.

Одинарный щелчок по полю Color приводит к появлению поля со списком, при раскрытии которого можно выбрать цвет, однако это не вполне удобно. Предпочтительней двойным щелчком по свойству (а не по кнопке с треугольником!) вызвать диалоговую панель для выбора цвета. Аналогично в поле Name можно выбрать название шрифта.

Для сохранения воспользуйтесь кнопкой  **.** В открывшемся диалоговом окне сохранения на диске W:\ создайте папку VP, в ней создайте папку для нашего примера Lab1, в которой и сохраните созданный проект. Обратите внимание на то, что появляются два диалоговых окна, первое для сохранения для сохранения формы и программы (Unit1.cpp), а второе проекта (project1.bdsproj). Естественно их имена можно изменить, но они должны бать **разными**. Если после сохранения посмотреть содержимое папки, то

обнаружится более чем 2 файла, причем будут файлы с именами проекта и формы, но с различными расширениями. Пока не удаляйте эти файлы.

Для запуска программы следует воспользоваться кнопкой . Если все было сделано правильно, то на экране появится окно созданной нами программы (рисунок 1.8).

Рисунок 1.8 – Приложение «Hello, World!»

После завершения работы закройте C++Builder и почистите свою папку. Безболезненно можно удалить файлы с расширениями obj, tds и файлы начинающиеся со знака ~ (тильда).

1.4 Основные свойства, методы и события формы

Для эффективной работы с формой следует рассмотреть основные свойства методы и события.

1.4.1 Основные свойства формы:

- Name – имя формы (тип свойства AnsiString);
- Caption – заголовок формы (тип AnsiString);
- Left – расстояние от левого края экрана до формы (тип int);
- Top – расстояние от верхнего края экрана до формы (тип int);
- Width – ширина формы (тип int);
- Height – высота формы (тип int);
- Menu – главное меню (см. пункт 2.1 ниже);
- Color – цвет фона формы (тип TColor);
- BorderIcons – составное свойство означающее наличие или отсутствие кнопок в строке заголовка, свойство состоит из подсвойств:
    - biSystemMenu – наличие или отсутствие системного меню и кнопки закрытия формы;
    - biMinimize – наличие или отсутствие кнопки минимизации;

- biMaximize – наличие или отсутствие кнопки максимизации;
- biHelp – наличие или отсутствие кнопки помощи;

• Position – расположение формы, может принимать одно из следующих значений:

- poDesigned – форма расположена так, как в редакторе форм;
- poScreenCenter – в центре экрана;
- poDesktopCenter – в центре рабочего стола;
- poMainFormCenter – в центре главной формы программы;
- poOwnerFormCenter – в центре родительской формы;
- poDefault – так, как разместит Windows;

• WindowState – состояние размера формы, может принимать одно из следующих значений:

- wsNormal – форма нормального размера;
- wsMinimized – форма свернута (минимизирована);
- wsMaximized – форма развернута на весь рабочий стол (максимизирована);

• PopupMenu – всплывающее меню; вызывается по нажатию правой кнопки мыши; правила составления меню будут описаны в пунктах 2.1 и 2.2;

• Handle – hwnd формы.

Остальные свойства, при желании, можно рассмотреть самостоятельно.

Основные методы формы:

• Close( ) – закрыть форму;

• Show( ) – показать форму;

• ShowModal( ) – показать форму с разрешением выполнять операции только в этой форме, такие формы называются модальными;

• Hide( ) – сделать форму невидимой.

• Основные события формы:

• OnCreate – возникает при создании формы;

• OnActivate – при активизации формы;

• OnPaint – при перерисовке формы;

• OnClose – при закрытии формы;

• OnDestroy – при разрушении формы;

• OnClick – при нажатии кнопки мыши;

• OnDblClick – при двойном щелчке левой кнопки мыши;

• OnKeyDown – при нажатии кнопки на клавиатуре.

1.4.2 Пример работы с формой

Войдите в C++Builder. Откройте проект из папки W:\VP\Lab1. Задайте свойства формы, указанные в таблице 1.2.

Задайте следующие функции для события OnCreate.

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 MessageBox(NULL, " Form1::OnCreate", "", MB_OK);
}
```

Таблица 1.2 – Свойства формы

Свойство	**Значение**
Caption	Пробная форма
Color	clInfoBk
Position	poDesktopCenter

Для события OnActivate (здесь и далее будет написано без заголовка функции и фигурных скобок, которые C++Builder создает сам):

```
MessageBox(NULL, " Form1::OnActivate", "", MB_OK);
```

Для события OnClose:

```
MessageBox(NULL, " Form1::OnClose", "", MB_OK);
```

Для события OnDestroy:

```
MessageBox(NULL, " Form1::OnDestroy", "", MB_OK);
```

Запустите программу. После окончания попробуйте изменить, описанные выше свойства и снова запустить.

Несколько усложним задачу. Для этого на форме расположите кнопку, при щелчке по которой будет вызываться вторая форма. Для ее создания в главном меню выберите пункт New / Form. Уменьшите ее размеры относительно первой формы примерно в два раза. Задайте свойства второй формы, указанные в таблице 1.3.

Таблица 1.3 – Свойства второй формы

Свойство	**Значение**
Caption	Вторая форма
Color	clPurple (если не нравится, то другой цвет)
Position	poDesktopCenter

Задайте следующие функции для события OnCreate:

```
MessageBox(NULL, " Form2::OnCreate", "", MB_OK);
```

Для события OnActivate:

```
MessageBox(NULL, " Form2::OnActivate", "", MB_OK);
```

Для события OnClose:

```
MessageBox(NULL, " Form2::OnClose", "", MB_OK);
```

Для события OnDestroy:

```
MessageBox(NULL, " Form2::OnDestroy", "", MB_OK);
```

Двойным щелчком по кнопке задайте функцию, которая будет вызываться при нажатии кнопки:

```
Form2->ShowModal();
```

В начале файла программы после строчки

```
#include "Unit1.h"
```

запишите строчку:

```
#include "Unit2.h"
```

Запустите программу. При нажатии кнопки вызовется вторая форма. Обратите внимание, в какие моменты возникают события.

1.5 Основные свойства и события компонентов

В данном пункте рассмотрим основные свойства, которыми обладает большинство компонентов. В последующих главах будем рассматривать компоненты по отдельности со их индивидуальными свойствами, методами и событиями.

К основным свойствам большинства компонентов можно отнести:

- Name – имя компонента (тип свойства AnsiString);
- Left – расстояние от левого края экрана до формы (тип int);
- Top – расстояние от верхнего края экрана до формы (тип int);
- Width – ширина формы (тип int);
- Height – высота формы (тип int);
- Color – цвет фона (тип TColor);
- Font – шрифт, составное свойство, состоящее из подсвойств:
  - Name – наименование шрифта (тип AnsiString);
  - Color – цвет шрифта (тип TColor);
  - Size – размер (тип int);
- Style – составное свойство, состоящее из подсвойств:
  - FsBold – полужирный (логический тип: true – да, false – нет);
  - FsItalic – наклонный (логический тип);
  - FsUnderline – подчеркнутый (логический тип);
- Visible – видимость (логический тип);
- Enabled – доступность (логический тип);
- PopupMenu – всплывающее меню;
- Handle – hwnd дочернего окна.
- Основные события большинства компонентов:
- OnEnter – возникает при попадании на объект;
- OnExit – возникает при покидании объекта;
- OnClick – при нажатии левой кнопки мыши;
- OnDblClick – при двойном щелчке по левой кнопке мыши;

• OnKeyDown – при нажатии клавиши на клавиатуре.

1.6 Объект Application

Объект Application – это неотображаемый объект. Он создается автоматически при инициализации нового приложения и служит для задания свойств программы в целом.

Основные свойства объекта Application:

- MainForm – указатель на главную форму;
- Icon – иконка программы.

Основные методы объекта Application:

- Initialize – инициализация программы;
- CreateForm – создание формы;
- Run – запуск программы;
- MessageBox – аналогичен функции MessageBox, но без первого параметра;
- ShowException – показ сообщения об ошибочной ситуации.

Рассмотрим главную программу последнего проекта. Для этого откроем файл с именем проекта с расширением cpp (обычно project1.cpp), увидим:

```
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
 try
 {
 Application->Initialize();
 Application->CreateForm(__classid(TForm1), &Form1);
 Application->CreateForm(__classid(TForm2), &Form2);
 Application->Run();
 }
 catch (Exception &exception)
 {
 TApplication->ShowException(&exception);
 }
 return 0;
}
```

Обратим внимание на то, что здесь используются четыре метода объекта Application.

1.7 Контрольные вопросы

1. Что представляет собой палитра компонентов?
2. Для чего нужен редактор форм?
3. Что представляет собой тип AnsiString?
4. Как преобразовать char * в AnsiString?
5. Как преобразовать int в AnsiString?
6. Как преобразовать double в AnsiString?

7. Как преобразовать AnsiString в char * ?
8. Как преобразовать AnsiString в int?
9. Как преобразовать AnsiString в double?
10. Что означает операция AnsiString + AnsiString?
11. Что означает операция AnsiString > AnsiString?
12. Что представляет собой класс TStrings.?
13. Что означает свойство Count класса TStrings?
14. Что означает свойство Strings класса TStrings?
15. Что означает метод Add класса TStrings?
16. Что означает метод Insert класса TStrings?
17. Что означает метод Delete класса TStrings?
18. Что означает метод SaveToFile класса TStrings?
19. Что означает метод LoadFromFile класса TStrings?
20. Что означает метод Clear( ) класса TStrings?
21. Что представляет собой тип TColor?
22. Какого типа свойство Name формы?
23. Что означает свойство Caption формы?
24. Что означают свойства Left, Top, Width, Height формы?
25. Что означают свойства Menu и PopupMenu формы?
26. Какого типа свойство Color формы?
27. Что означает свойство BorderIcons формы?
28. Что означает свойство Position формы?
29. Что означает свойство WindowState формы?
30. Для чего используется метод Close формы?
31. Для чего используется метод Show формы?
32. Для чего используется метод ShowModal формы?
33. Для чего используется метод Hide формы?
34. В каких случаях возникает событие OnCreate формы?
35. В каких случаях возникает событие OnActivate формы?
36. В каких случаях возникает событие OnPaint формы?
37. В каких случаях возникает событие OnClose формы?
38. В каких случаях возникает событие OnDestroy формы?
39. В каких случаях возникает событие OnClick формы?
40. В каких случаях возникает событие OnDblClick формы?
41. В каких случаях возникает событие OnKeyDown формы?
42. Что означает свойство Visible компонентов?
43. Что означает свойство Enabled компонентов?
44. В каких случаях возникает событие OnEnter?
45. В каких случаях возникает событие OnExit?
46. Что означает свойство MainForm объекта Application?
47. Что означает свойство Icon объекта Application?
48. Что означает метод Initialize объекта Application?
49. Что означает метод CreateForm объекта Application?
50. Что означает метод Run объекта Application?
51. Что означает метод MessageBox объекта Application?

52. Что означает метод ShowException объекта Application?

2 КОМПОНЕНТЫ ЗАКЛАДКИ STANDARD

На закладке Standard расположены основные компоненты Windows. Компоненты закладки показаны на рисунке 2.1.

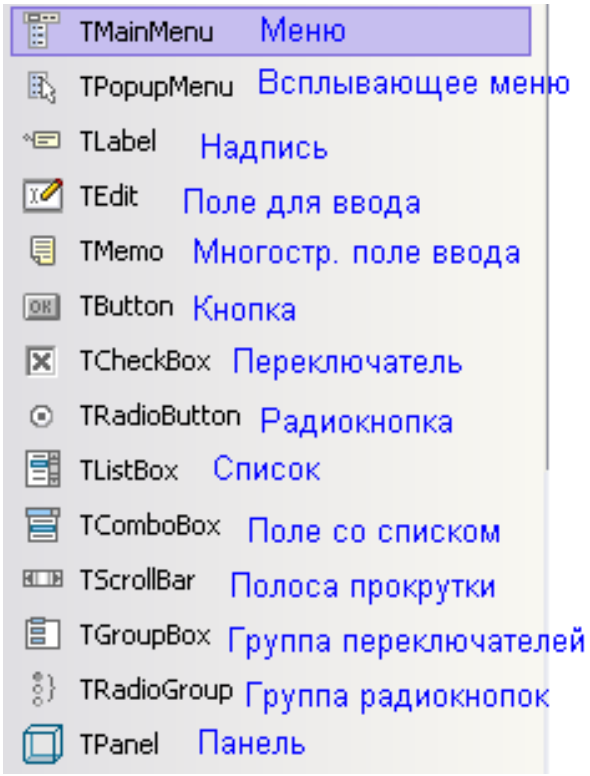
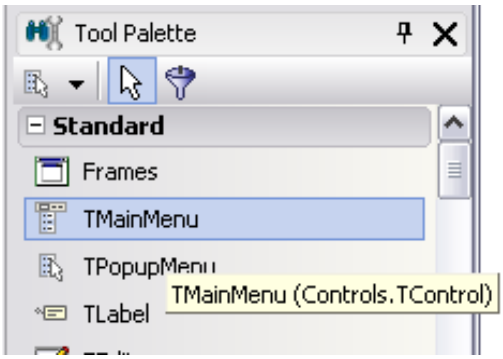


Рисунок 2.1 – Закладка Standard

Ниже будут рассмотрены основные компоненты закладки Standard и их наиболее часто встречающиеся свойства, методы и события.

2.1 Компонент MainMenu (Меню)

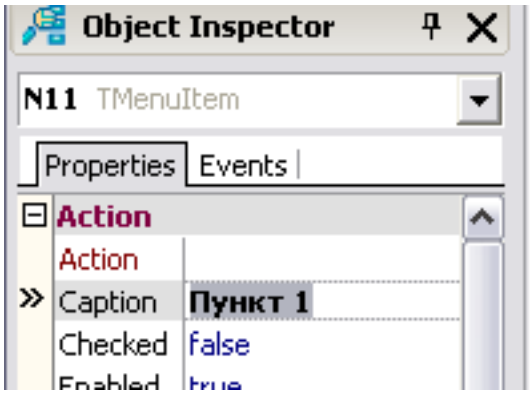
Компонент MainMenu предназначен для создания меню, связанного с формой:



Расположите компонент MainMenu в любой части Редактора форм и дважды щелкните по нему. Вашему вниманию будет предложено окно редактора меню, изображенное на рисунке 2.2.

Рисунок 2.2 – Окно редактора меню

В инспекторе объектов следует задать свойства пунктов меню. Основным свойством является Caption (Заголовок пункта меню). Тип свойства – AnsiString.



Двойным щелчком по пункту меню попадаем в редактор кода программы, в котором будет сформирован пустой шаблон функции, вызываемой при активизации пункта меню (рисунок 2.3).

```
//---------------------------------------------------
void __fastcall TForm1::N11Click(TObject *Sender)
{
19
20 }
//---------------------------------------------------
```

Рисунок 2.3 – Шаблон функции для пункта меню

На месте, где на рисунке показан курсор, следует написать текст функции. Заметим, что переход из окна Редактора форм в окно Редактора кода

для всех рассмотренных ниже компонент (или их элементов) выполняется аналогично (двойным щелчком).

Для примера создадим меню, показанное на рисунке 2.4.

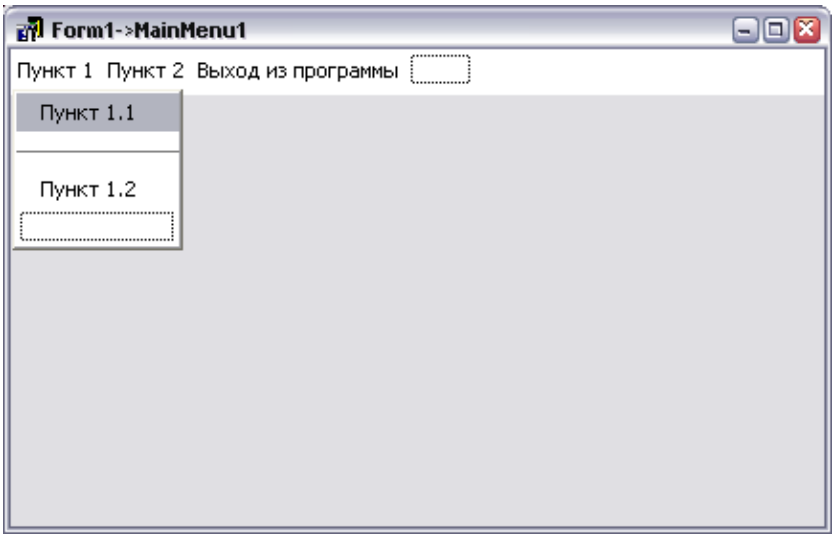


Рисунок 2.4 – Пример построенного меню

Для пунктов меню зададим функции, изображенные на рисунке 2.5.

```
//---------------------------------------------------
void __fastcall TForm1::N111Click(TObject *Sender)
{
  // Пункт 1.1
  MessageBox(NULL, "Пункт 1.1", "", MB_OK);
}
//---------------------------------------------------
void __fastcall TForm1::N121Click(TObject *Sender)
{
  // Пункт 1.2
  MessageBox(NULL, "Пункт 1.2", "", MB_OK);
}
//---------------------------------------------------
void __fastcall TForm1::N21Click(TObject *Sender)
{
// Пункт 2
  MessageBox(NULL, "Пункт 2", "", MB_OK);
}
//---------------------------------------------------
void __fastcall TForm1::N1Click(TObject *Sender)
{
// Пункт Выход из программы
  Close();
}
//---------------------------------------------------
```

Рисунок 2.5 – Функции для обработки пунктов меню

Сохраните получившийся проект в папке W:\VP\Lab2 и запустите его. По завершении не забудьте почистить папку.

Примечание. Для отделения одной группы пунктов меню от другой следует использовать пункт меню, в котором свойство Caption содержит символ ‘-‘.

2.2 Компонент PopupMenu (Всплывающее меню)

Компонент PopupMenu реализует вертикальное меню, которое во время выполнения программы вызывается щелчком по правой кнопке мыши в форме или над некоторым ее объектом. Для этого форме или объекту формы необходимо задать свойство PopupMenu, в котором следует выбрать заранее созданное PopupMenu.

Создание PopupMenu аналогично созданию MainMenu (разница лишь в отсутствии горизонтальной составляющей), поэтому этот процесс рассматривать не будем.

2.3 Компонент Label (Надпись)

Компонент Label (прямой перевод – «метка», наиболее подходит слово - «Надпись») предназначен для размещения на форме текста. Основным свойством является: Caption – заголовок надписи (тип свойства AnsiString).

Примеры использования компонента Label приведены в пунктах 1.3 и 2.5, а также во многих последующих примерах.

2.4 Компонент Edit (Поле ввода)

Компонент Edit (прямой перевод – «редактор», наиболее подходит выражение – «поле ввода и редактирования») предназначен для размещения на форме однострочного поля ввода. К основным свойствам относятся:

- Text – текст, содержащийся в поле (тип свойства AnsiString);
- ReadOnly – логическое поле: ложь – редактирование разрешено, истина – редактирование запрещено (только отображение).

Примеры использования компонента Edit приведены в пункте 2.5, а также во многих последующих примерах.

2.5 Компонент Button (Кнопка)

Компонент Button (Кнопка) предназначен для размещения на форме простой кнопки с текстом (в последующих пунктах будут рассмотрены более сложные кнопки).

Основным свойством является Caption – текст, на поверхности кнопки (тип свойства AnsiString).

Основным событием является OnClick. Это событие происходит при щелчке по левой кнопке мыши во время выполнения программы и запускает связанную с ним функцию.

Рассмотрим пример сложения двух чисел с использованием компонентов Label, Edit и Button. Для этого разместим на форме элементы, показанные на рисунке 2.6.

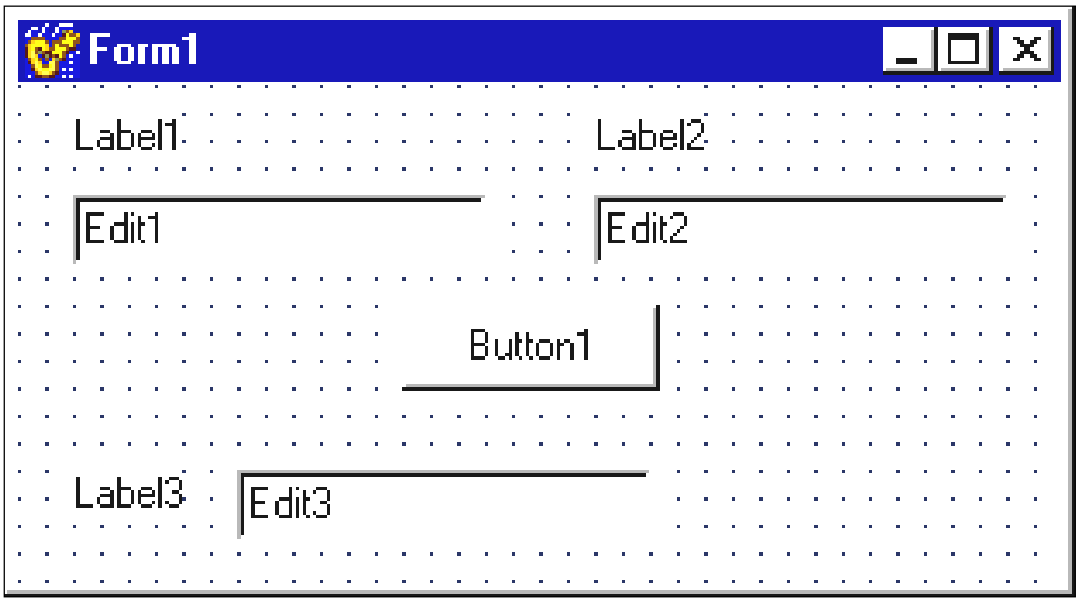


Рисунок 2.6 – Форма для сложения двух чисел

Зададим свойства формы и объектов на ней, описанные в таблице 2.1.

Таблица 2.1 – Свойства формы и объектов для сложения двух чисел

Объект	**Свойство**	**Значение**
Форма	Caption	Сложение двух чисел
Label1	Caption	Первое слагаемое
Label2	Caption	Второе слагаемое
Label3	Caption	Сумма
Edit1	Text	
Edit2	Text	
Edit3	Text	
Edit3	ReadOnly	True
Button1	Caption	Сложить

В окне Редактора кода напишем текст связанной с кнопкой функции:

```
Edit3->Text = Edit1->Text.ToInt( ) + Edit2->Text.ToInt( );
```

Программа готова, ее можно запустить. Следует отметить, что данная программа работает только с целыми числами. Для работы с дробными числами следует заменить ToInt на ToDouble.

Если Вы успешно справились с предыдущей задачей, то предлагается выполнить самостоятельно задание по разработке формы, показанной на рисунке 2.7.

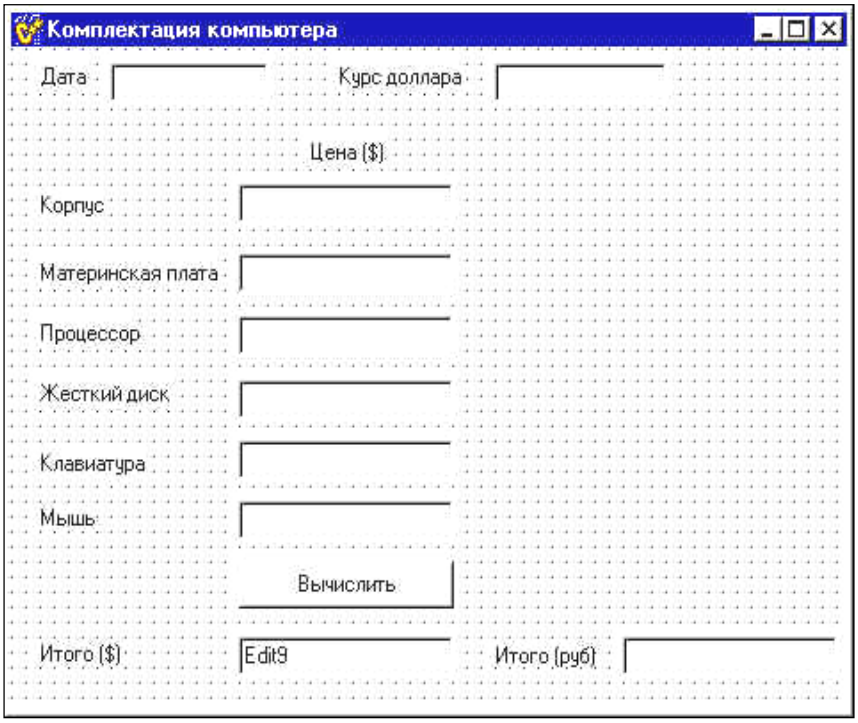


Рисунок 2.7 – Форма для самостоятельной работы

2.6 Компонент Memo (Многострочное поле ввода)

Компонент Memo (прямой перевод – сокращение от слова «память», наиболее подходящее выражение – «многострочное поле ввода и редактирования») предназначен для размещения на форме многострочного текстового редактора. Многострочный редактор имеет практически те же возможности по редактированию текста, что и однострочные редакторы. Главное отличие этих элементов управления заключается в том, что многострочный редактор содержит несколько строк текста.

К основным свойствам относятся:

- Lines – многострочный текст, содержащийся в поле (тип свойства TStrings), двойным щелчком по значению свойства в инспекторе объектов открывается редактор строк для начального ввода;
- ReadOnly – логическое поле: ложь – редактирование разрешено, истина – редактирование запрещено, только отображение;
- Align – выравнивание редактора относительно формы (тип TAlign);
- Alignment – выравнивание текста относительно редактора (тип TAlignment).

TAlign – является перечислимым типом (enum) и может принимать одно из следующих значений: alNone – без выравнивания, alTop – выравнивание

вверх, alBottom – вниз, alLeft – влево, alRight – вправо, alClient – по всему окну.

TAlignment – также является перечислимым типом и может принимать одно из следующих значений: taLeftJustify – выравнивание влево, taRightJustify – вправо, taCenter – по центру.

Основными методами являются:

- Clear( ) – очистить содержимое;
- ClearSelection( ) – удалить выделенный кусок;
- CopyToClipboard( ) – копировать выделенный участок текста в clipboard (буфер обмена);
- CutToClipboard( ) – копировать выделенный участок текста в clipboard с одновременным удалением;
- PasteFromClipboard( ) – вставка из clipboard;
- SelectAll( ) – выделить все;
- Undo( ) – отменить последнее действие;
- ClearUndo( ) – отменить отмену последнего действия.

В качестве примера рассмотрим текстовый редактор типа «блокнот». Для его создания разместим на форме объекты: MainMenu и Memo. Зададим свойства, указанные в таблицы 2.2.

Таблица 2.2 – Свойства формы и многострочного текстового редактора

Объект	**Свойство**	**Значение**
Форма	Caption	Текстовый редактор типа «блокнот»
Memo1	Lines	
Memo1	Align	AlClient

Создайте меню вида, показанного на рисунках 2.8 и 2.9.

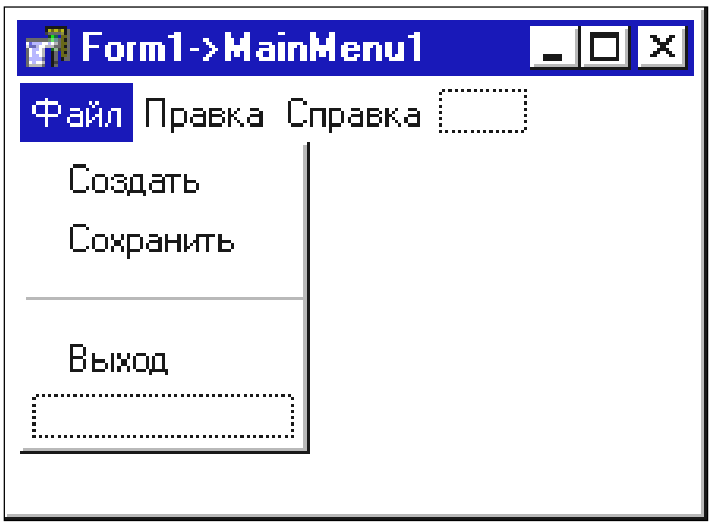


Рисунок 2.8 – Горизонтальное меню и подменю «файл»

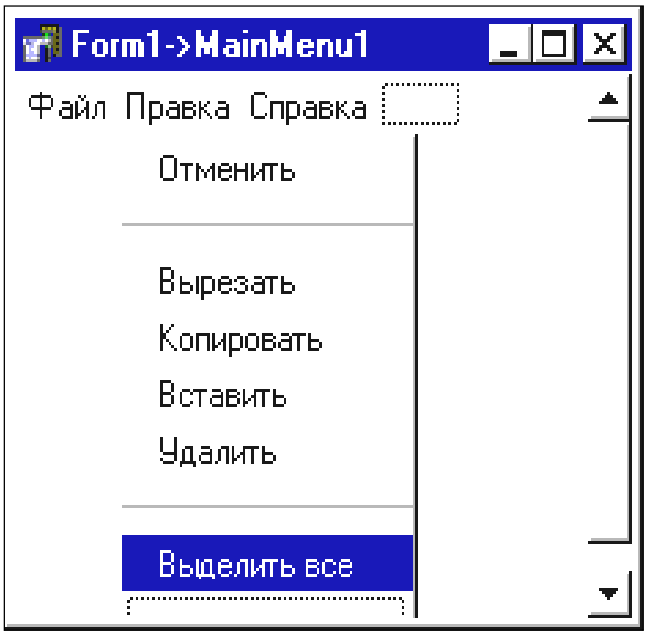


Рисунок 2.9 – Горизонтальное меню и подменю «правка»

Для каждого пункта меню следует задать действие.
Для пункта «Создать»:

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

Для пункта «Сохранить»:

```
Memo1->Lines->SaveToFile("document.txt");
```

Для пункта «Выход»:

```
Close();
```

Для пункта «Отменить»:

```
Memo1->Undo();
```

Для пункта «Вырезать»:

```
Memo1-> CutToClipboard();
```

Для пункта «Копировать»:

```
Memo1-> CopyToClipboard();
```

Для пункта «Вставить»:

```
Memo1-> PasteFromClipboard();
```

Для пункта «Удалить»:

```
Memo1-> ClearSelection ();
```

Для пункта «Выделить все»:

```
Memo1->SelectAll();
```

Для пункта «Справка»:

```
MessageBox(NULL, "Здесь можно написать о себе", "Справка", MB_OK);
```

Программа готова. Естественно это еще не полный вариант редактора «блокнот». В дальнейшем при изучении других конструкций мы усовершенствуем редактор и доведем его до полного набора функций программы «блокнот». Не забудьте сохранить проект для дальнейшей доработки.

2.7 Компонент CheckBox (Переключатель)

Компонент CheckBox предназначен для размещения на форме переключателя вида  , который может находиться в двух состояниях: включенном и выключенном. К основным свойствам относятся:

- Caption – подпись к переключателю (тип свойства AnsiString).
- Checked – состояние переключателя (true – включен, false – выключен).

Основным событием является OnClick.

В качестве примера рассмотрим форму, на которой расположим поле ввода и переключатель. Если переключатель включен, то ввод разрешен, если выключен, то запрещен.

Создайте форму с элементами, показанными на рисунке 2.10.

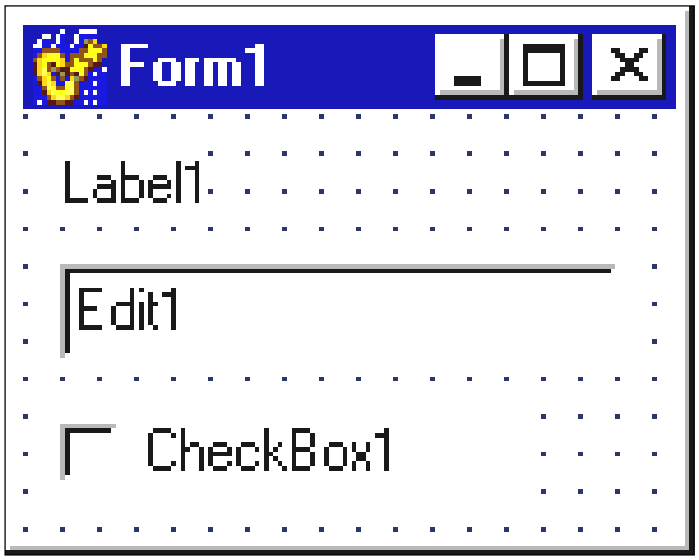


Рисунок 2.10 – Ввод с разрешением

Задайте им свойства из таблицы 2.3.

Таблица 2.3 – Свойства формы и элементов для ввода с разрешением

Объект	**Свойство**	**Значение**
Форма	Caption	Ввод с разрешением
Label1	Caption	Введите текст
Edit1	Text	
CheckBox1	Caption	Ввод разрешен
CheckBox1	Checked	True

Задайте функцию события OnClick:

```
Edit1->ReadOnly = !CheckBox1->Checked;
```

Запустите программу. Проверьте выполнение заданного функционала формы. Сохраните форму на свой диск.

2.8 Компонент RadioButton (Радио-кнопка)

Компонент RadioButton представляет собой переключатель вида . Таких переключателей должно быть несколько (два или более), причем один из них включен, остальные выключены (как в многопрограммном приемнике). При переключении всегда отмечен только один из установленной группы.

Основные свойства у компонента RadioButton такие же, как у компонента CheckBox:

- Caption – подпись к переключателю (тип свойства AnsiString);
- Checked – состояние переключателя (true – включен, false – выключен).

Основным событием также является OnClick. В качестве примера рассмотрим выбор цвета для текста. Для этого расположите на форме элементы, показанные на рисунке 2.11.

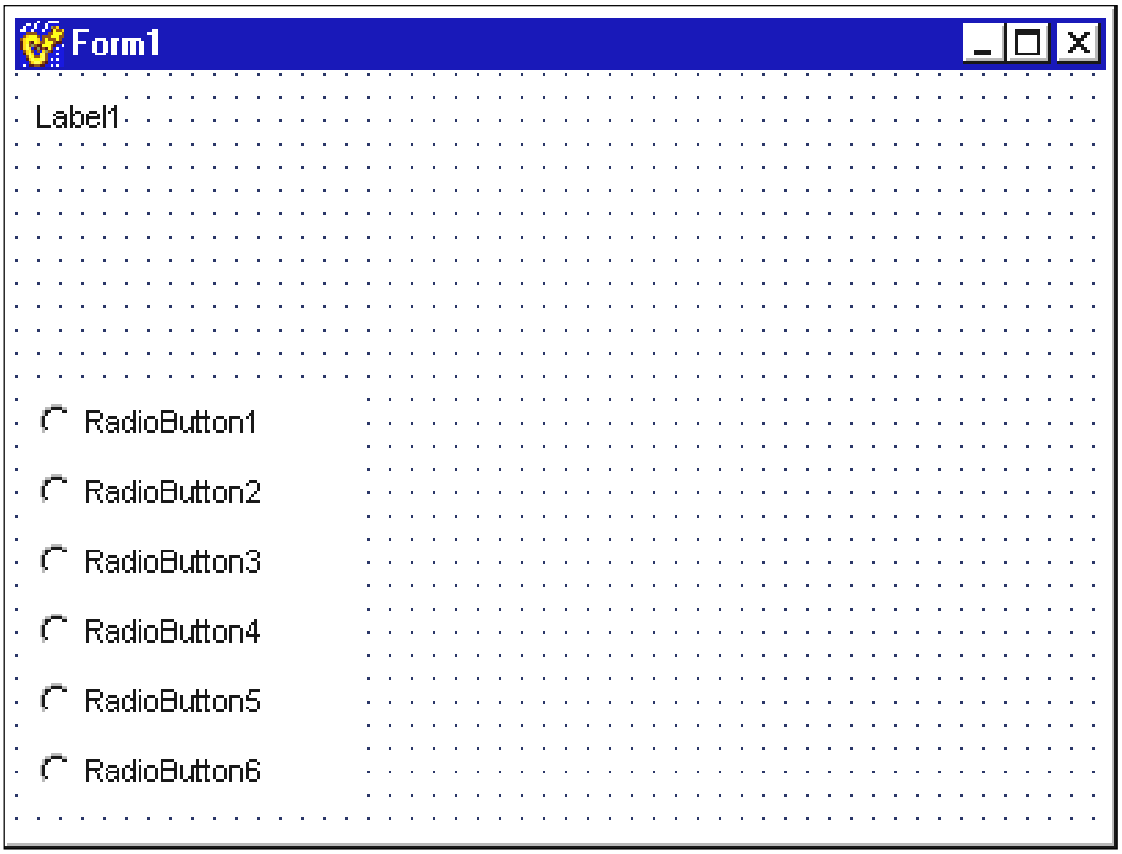


Рисунок 2.11. Выбор цвета текста

Задайте свойства формы и объектов, указанные в таблице 2.4.

Таблица 2.4 – Свойства формы и объектов для выбора цвета текста

Объект	Свойство	Значение
Форма	Caption	Выбор цвета
Label1	Caption	Демонстрация цвета
Label1	Font / Name	Times New Roman
Label1	Font / Size	64
RadioButton1	Caption	Черный
RadioButton1	Name	BlackRB
RadioButton2	Caption	Белый
RadioButton2	Name	WhiteRB
RadioButton3	Caption	Красный
RadioButton3	Name	RedRB
RadioBitton4	Caption	Зеленый
RadioBitton4	Name	GreenRB
RadioBitton5	Caption	Синий
RadioBitton5	Name	BlueRB
RadioBitton6	Caption	Желтый
RadioBitton6	Name	YellowRB

Для первой радио-кнопки создайте функцию изменения состояния кнопки, содержащую следующий текст:

```
if (BlackRB->Checked) Label1->Font->Color = clBlack;
if (WhiteRB->Checked) Label1->Font->Color = clWhite;
if (RedRB->Checked) Label1->Font->Color = clRed;
if (GreenRB->Checked) Label1->Font->Color = clGreen;
if (BlueRB->Checked) Label1->Font->Color = clBlue;
if (YellowRB->Checked) Label1->Font->Color = clYellow;
```

Для остальных кнопок в инспекторе объектов на закладке Events установите событие OnClick, выбрав его из списка. Если Вы ранее все сделали правильно, то в списке будет один элемент BlackRBClick. Его и следует выбрать.

На этом программу можно считать законченной. Запустите ее. При нажатии на радио кнопки будет изменяться цвет текста.

2.9 Компонент ListBox (Список)

Компонент ListBox предназначен для реализации списка, образец вида которого показан на рисунке 2.12.

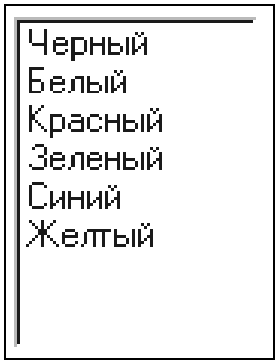


Рисунок 2.12 – Образец списка

Основные свойства ListBox:

• Items – содержимое списка (тип свойства TStrings). Двойной щелчок в инспекторе объектов по значению этого свойства приводит к открытию редактора текста для введения первоначального состояния списка;

• ItemIndex – номер текущего элемента списка (тип int). Этого свойства нет в инспекторе объектов, оно доступно только из программы. В дальнейшем такие свойства будем называть *программируемыми*;

• Sorted – логическое значение: является ли список сортированным?

Основным событием также является OnClick.

Рассмотрим пример подобный предыдущему, однако цвета будем выбирать не радио-кнопками, а с помощью списка.

Расположите на форме элементы, показанные на рисунке 2.13.

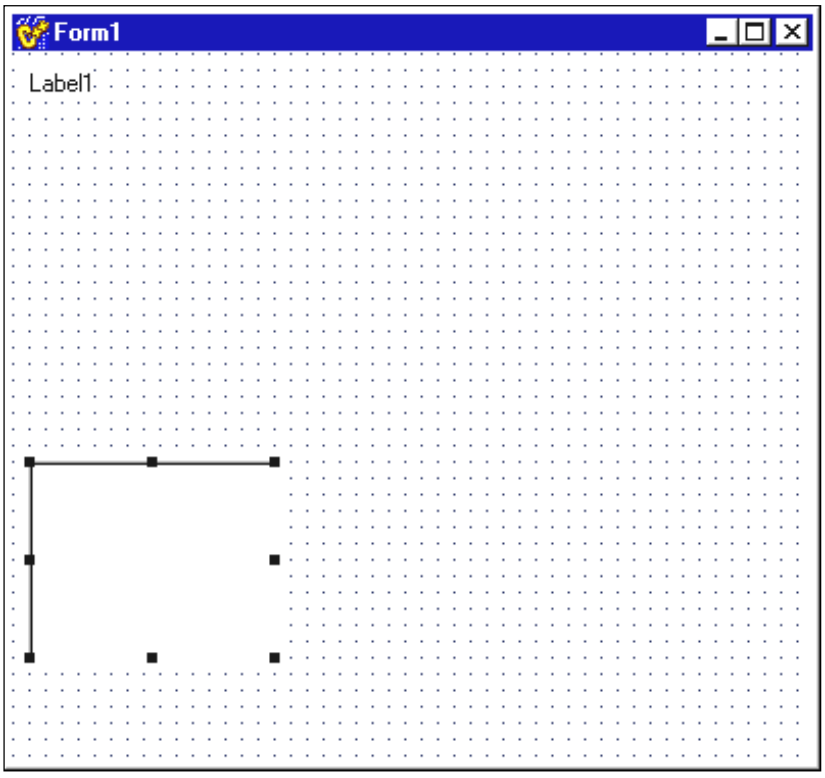


Рисунок 2.13 – Заготовка для выбора цвета текста из списка

Задайте свойства формы и объектов, указанные в таблице 2.5.

Таблица 2.5 – Свойства формы и объектов для выбора цвета текста из списка

Объект	Свойство	Значение
Форма	Caption	Выбор цвета
Label1	Caption	Демонстрация цвета
Label1	Font / Name	Times New Roman
Label1	Font / Size	64

Дважды щелкните по значению свойства Items списка, введите список цветов: “Черный”; “Белый”; “Красный”; “Зеленый”; “Синий”; “Желтый”. Запишите код функции события OnClick:

```
TColor Colors[ ] = {clBlack, clWhite, clRed, clGreen, clBlue, clY-
ellow};
Label1->Font->Color = Colors[ListBox1-> ItemIndex];
```

На этом программу можно считать законченной. Запустите ее. При нажатии на элементы списка будет изменяться цвет текста.

2.10 Компонент ComboBox (Поле со списоком)

Компонент ComboBox предназначен для реализации поля со списком, образец вида которого показан на рисунке 2.14.

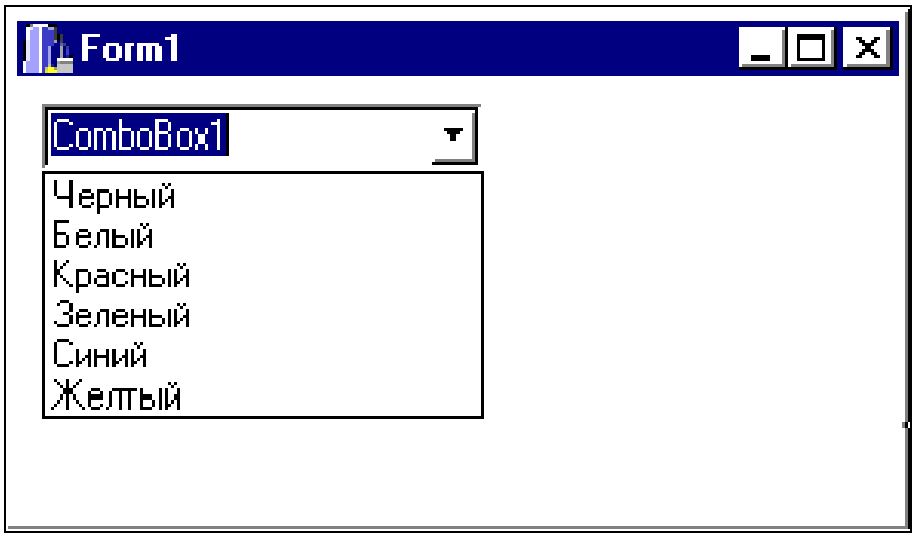


Рисунок 2.14 – Образец поля со списка

ComboBox объединяет в себе однострочный текстовый редактор и список и как следствие часто используется для выбора из списка.

Основные свойства ComboBox:

• Text – текст в поле ввода (тип свойства AnsiString);

• Items – содержимое списка (тип свойства TStrings). Двойной щелчок в инспекторе объектов по значению этого свойства приводит к открытию редактора текста для введения первоначального состояния списка;

• Sorted – логическое значение: является ли список сортированным?

Основным событием также является OnChange. Это событие запускает функцию, которая вызывается при изменении содержимого поля.

Рассмотрим пример подобный предыдущим, однако цвета будем вводить или выбирать в поле со списком.

Расположите на форме элементы, показанные на рисунке 2.15.

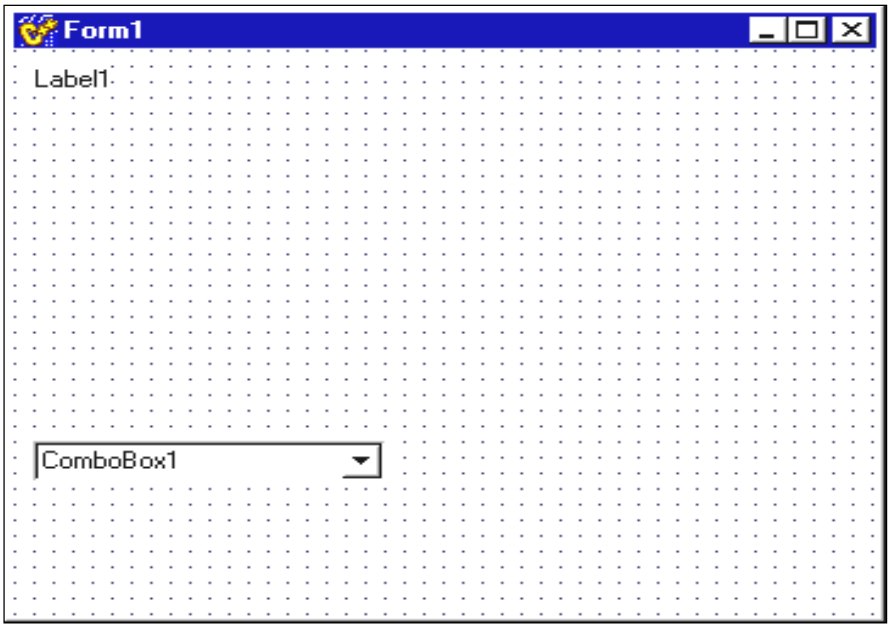


Рисунок 2.15 – Заготовка для выбора цвета текста в поле со списком

Задайте свойства формы и объектов, указанные в таблице 2.6. Дважды щелкните по значению свойства Items поля со списком и введите список цветов: “Черный”; “Белый”; “Красный”; “Зеленый”; “Синий”; “Желтый”.

Таблица 2.6 – Свойства формы и объектов для выбора цвета текста в поле со списком

Объект	**Свойство**	**Значение**
Форма	Caption	Выбор цвета
Label1	Caption	Демонстрация цвета
Label1	Font / Name	Times New Roman
Label1	Font / Size	64

Запишите код функции события OnChange:

```
TColor Colors[ ] = {clBlack, clWhite, clRed, clGreen, clBlue, clY-
ellow};
for (int i=0; i< ComboBox1->Items->Count; i++) {
if (ComboBox1->Text == ComboBox1->Items->Strings[i] {
Label1->Font->Color = Colors[i];
      } // if
} // for
```

На этом программу можно считать законченной. Запустите ее. При вводе или выборе из списка цвета будет изменяться цвет текста. При вводе цвет вводите с большой буквы.

2.11 Компонент ScrollBar (Полоса прокрутки)

Компонент ScrollBar предназначен для реализации полосы прокрутки, показанной на рисунке 2.16.

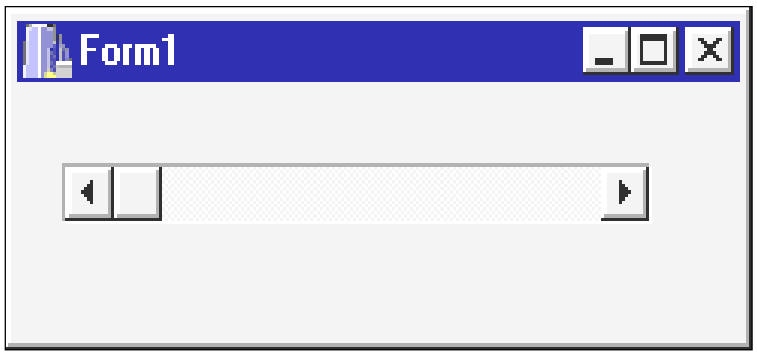


Рисунок 2.16 – Образец полосы прокрутки

Основные свойства ScrollBar:

- Kind – вид полосы прокрутки. Может принимать два значения:
- sbHorizontal - горизонтальная,
- sbVertical – вертикальная;
- Min – минимальное значение позиции ползунка (тип int);
- Max – максимальное значение позиции ползунка (тип int);
- Position – текущее значение позиции ползунка (тип int).

Основным событием является OnChange.

Рассмотрим пример, подобный предыдущим, однако цвета будем изменять плавно при помощи трех ползунков. Первый ползунок будет отвечать за красную составляющую цвета, второй – за зеленую, третий – за синюю.

Расположите на форме элементы, показанные на рисунке 2.17.

Задайте свойства формы и объектов, указанные в таблице 2.7.

Таблица 2.7 – Свойства формы и объектов для выбора цвета текста с помощью трех полос прокрутки

Объект	Свойство	Значение
Форма	Caption	Выбор цвета
Label1	Caption	Демонстрация цвета
Label1	Font / Name	Times New Roman
Label1	Font / Size	64
Label2	Caption	Красный
Label2	Font / Color	clRed
Label3	Caption	Зеленый
Label3	Font / Color	clGreen
Label4	Caption	Синий
Label4	Font / Color	clBlue
ScrollBar1	Name	RedSB
ScrollBar2	Name	GreenSB
ScrollBar3	Name	BlueSB
Все ScrollBar	Min	0
Все ScrollBar	Max	255
Все ScrollBar	Position	0

Запишите код функции события OnChange:

```
Label1->Font->Color = RGB(RedSB->Position, GreenSB->Position, BlueSB-
> Position);
```

Для остальных полос прокрутки в инспекторе объектов на закладке Events установите событие OnChange, выбрав его из списка. Если Вы ранее все сделали правильно, то в списке будет один элемент RedSBChange, который и следует выбрать. На этом программу можно считать законченной. Запустите ее. При перемещении ползунков будет плавно изменяться цвет текста.

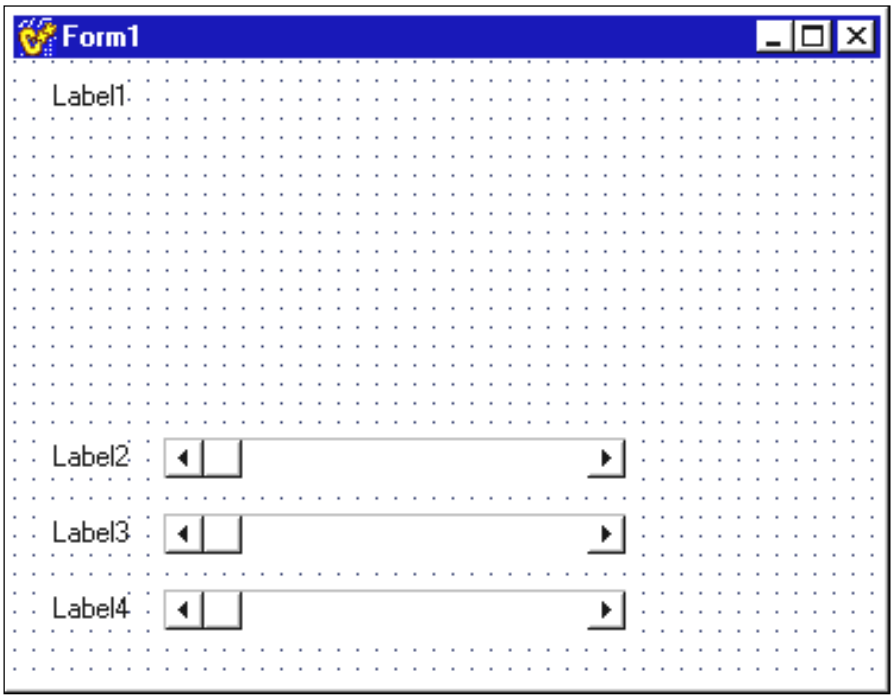


Рисунок 2.17 – Заготовка для выбора цвета с помощью трех полос прокрутки

2.12 Компонент GroupBox (Группа переключателей)

Компонент GroupBox предназначен для реализации групп переключателей, радио кнопок и т.п. Пример группы показан на рисунке 2.18.

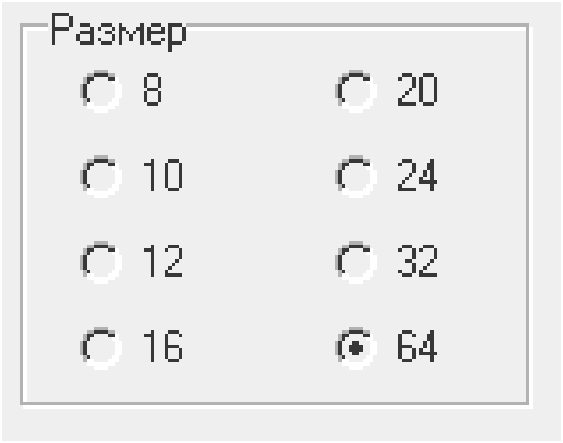


Рисунок 2.18 – Образец группы переключателей

Основное свойство компонента Caption – заголовок группы (тип AnsiString).

Основным событием является OnClick. Это событие запускает функцию, которая вызывается при щелчке мыши в пределах группы переключателей, но вне любого конкретного переключателя.

В качестве примера рассмотрим изменение шрифта, размера и цвета текста. Форма для такого изменения показана на рисунке 2.19.

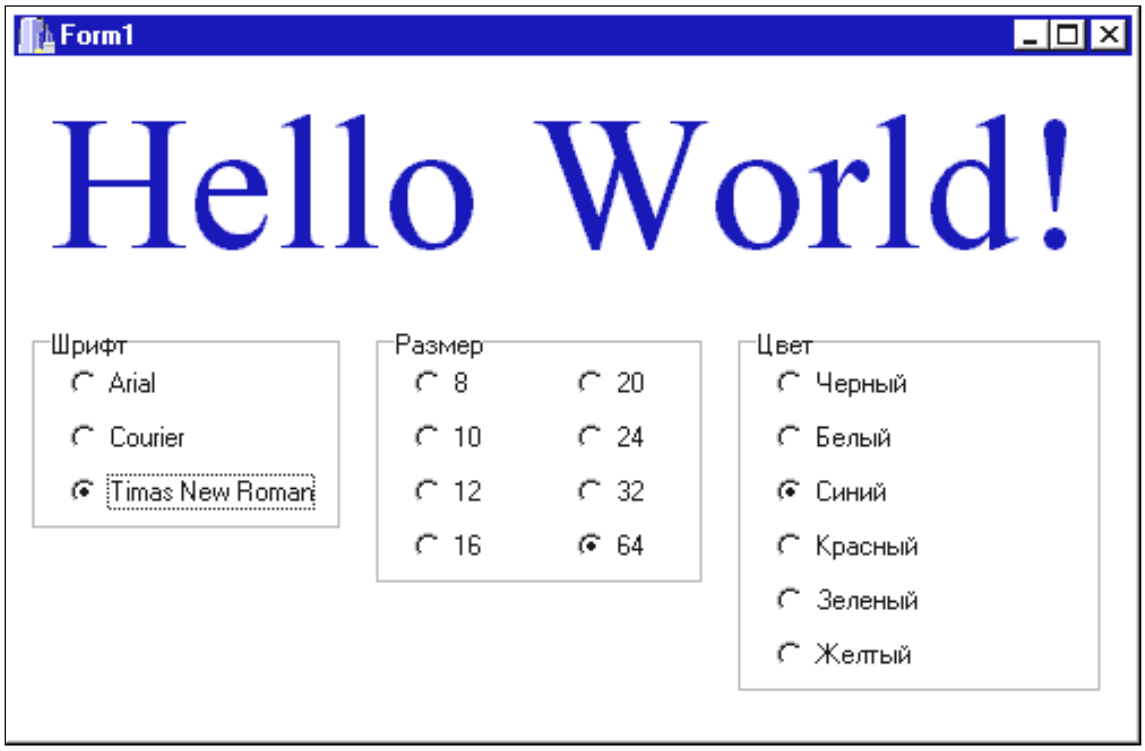


Рисунок 2.19 – Управление шрифтом, размером и цветом текста

Для этого расположите на форме элементы, показанные на рисунке 2.20.

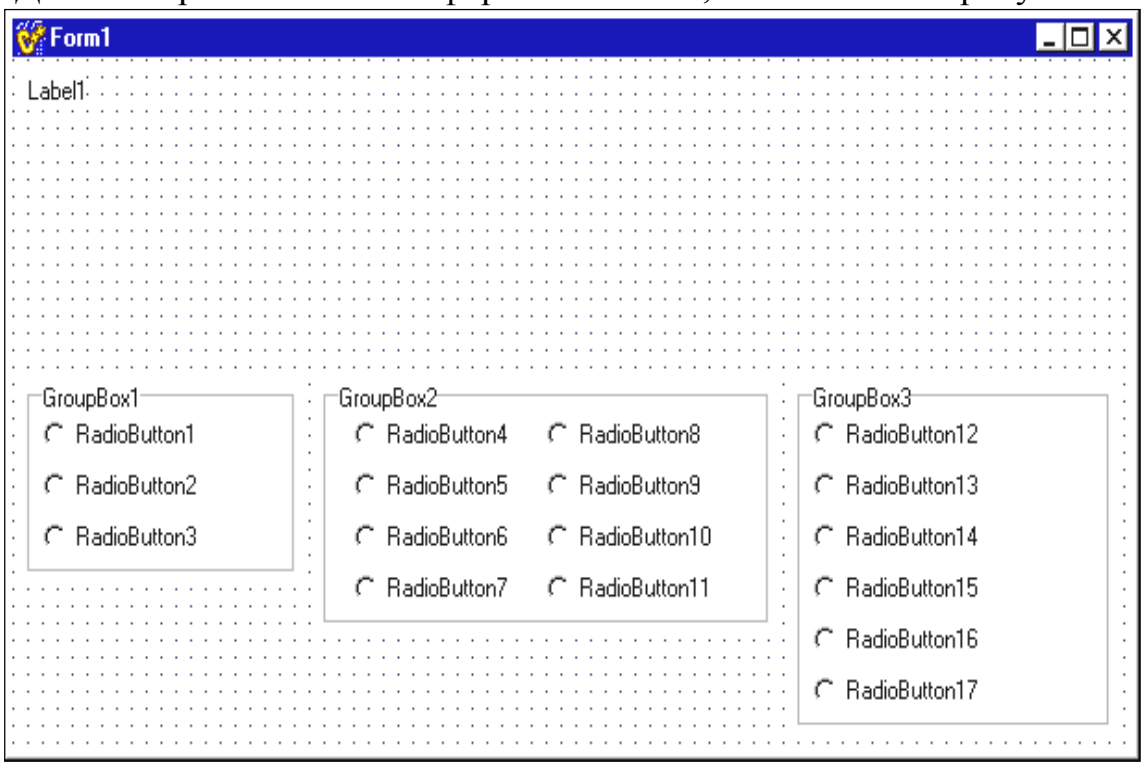


Рисунок 2.20 – Заготовка для управления шрифтом, размером и цветом текста

Задайте свойства формы и объектов, указанные в таблице 2.8.

Таблица 2.8 – Свойства формы и объектов для управления шрифтом, размером и цветом текста

Объект	**Свойство**	**Значение**
Форма	Caption	Управление шрифтом, размером и цветом текста
Label1	Caption	Hello World!
GroupBox1	Caption	Шрифт
GroupBox2	Caption	Размер
GroupBox3	Caption	Цвет
RadioButton1	Name	ArialRB
RadioButton1	Caption	Arial
RadioButton1	Checked	True
RadioButton2	Name	CourierRB
RadioButton2	Caption	Courier
RadioButton3	Name	TimesNewRomanRB
RadioButton3	Caption	Times New Roman
RadioButton4	Name	Size8RB
RadioButton4	Caption	8
RadioButton4	Checked	True
RadioButton5	Name	Size10RB
RadioButton5	Caption	10
RadioButton6	Name	Size12RB
RadioButton6	Caption	12
RadioButton7	Name	Size16RB
RadioButton7	Caption	16
RadioButton8	Name	Size20RB
RadioButton8	Caption	20
RadioButton9	Name	Size24RB
RadioButton9	Caption	24
RadioButton10	Name	Size32RB
RadioButton10	Caption	32
RadioButton11	Name	Size64RB
RadioButton11	Caption	64
RadioButton12	Name	BlackRB
RadioButton12	Caption	Черный
RadioButton12	Checked	True
RadioButton13	Name	WhiteRB
RadioButton13	Caption	Белый
RadioButton14	Name	BlueRB
RadioButton14	Caption	Синий
RadioButton15	Name	RedRB
RadioButton15	Caption	Красный
RadioButton16	Name	GreenRB
RadioButton16	Caption	Зеленый
RadioButton17	Name	YellowRB
RadioButton17	Caption	Желтый

Для RadioButton1 (с надписью «Arial») напишем следующий программный код функции ArialRBClick:

```
if (ArialRB->Checked) Label1->Font->Name = "Arial";
if (CourierRB->Checked) Label1->Font->Name = "Courier";
if (TimesNewRomanRB->Checked) Label1->Font->Name="Times New Roman";
```

На закладке Events выберем для остальных кнопок группы «Шрифт» в качестве функции обработки события OnClick функцию ArialRBClick.

Для RadioButton4 (первая в группе «Размер» с надписью «8») напишем следующий программный код функции Size8RBClick:

```
if (Size8RB->Checked) Label1->Font->Size = 8;
if (Size10RB->Checked) Label1->Font->Size = 10;
if (Size12RB->Checked) Label1->Font->Size = 12;
if (Size16RB->Checked) Label1->Font->Size = 16;
if (Size20RB->Checked) Label1->Font->Size = 20;
if (Size24RB->Checked) Label1->Font->Size = 24;
if (Size32RB->Checked) Label1->Font->Size = 32;
if (Size64RB->Checked) Label1->Font->Size = 64;
```

Выберем для остальных кнопок группы «Размер» в качестве функции обработки события OnClick функцию Size8RBClick.

Для RadioButton12 (первая в группе «Цвет» с надписью «черный») напишем следующий программный код функции BlackRBClick:

```
if (BlackRB->Checked) Label1->Font->Color = clBlack;
if (WhiteRB->Checked) Label1->Font->Color = clWhite;
if (BlueRB->Checked) Label1->Font->Color = clBlue;
if (RedRB->Checked) Label1->Font->Color = clRed;
if (GreenRB->Checked) Label1->Font->Color = clGreen;
if (YellowRB->Checked) Label1->Font->Color = clYellow;
```

Выберем для остальных кнопок группы «Цвет» в качестве функии обработки события OnClick функцию BlackRBClick. На этом программу можно считать законченной. Запустите ее.

2.13 Компонент RadioGroup (Группа радио-кнопок)

Компонент RadioGroup предназначен для реализации группы радио-кнопок. В отличии от GroupBox данная группа не может содержать ничего кроме радио кнопок, причем не нужно создавать отдельно каждую кнопку и задавать ей свойства, достаточно задать свойства всей группы.

Основные свойства RadioGroup:

- Caption – заголовок группы (тип AnsiString);
- Items – заголовки отдельных кнопок группы (тип TStrings). Двойной щелчок в инспекторе объектов по значению этого свойства приводит к открытию редактора текста для ввода заголовков кнопок;

• ItemIndex – номер нажатой кнопки;

• Columns – количество столбцов кнопок.

Основным событием является OnClick. Это событие запускает функцию, которая вызывается при изменении состояния какой-либо кнопки.

Рассмотрим предыдущий пример, где вместо GroupBox будем использовать RadioGroup.

Для этого расположите на форме элементы, показанные на рисунке 2.21.

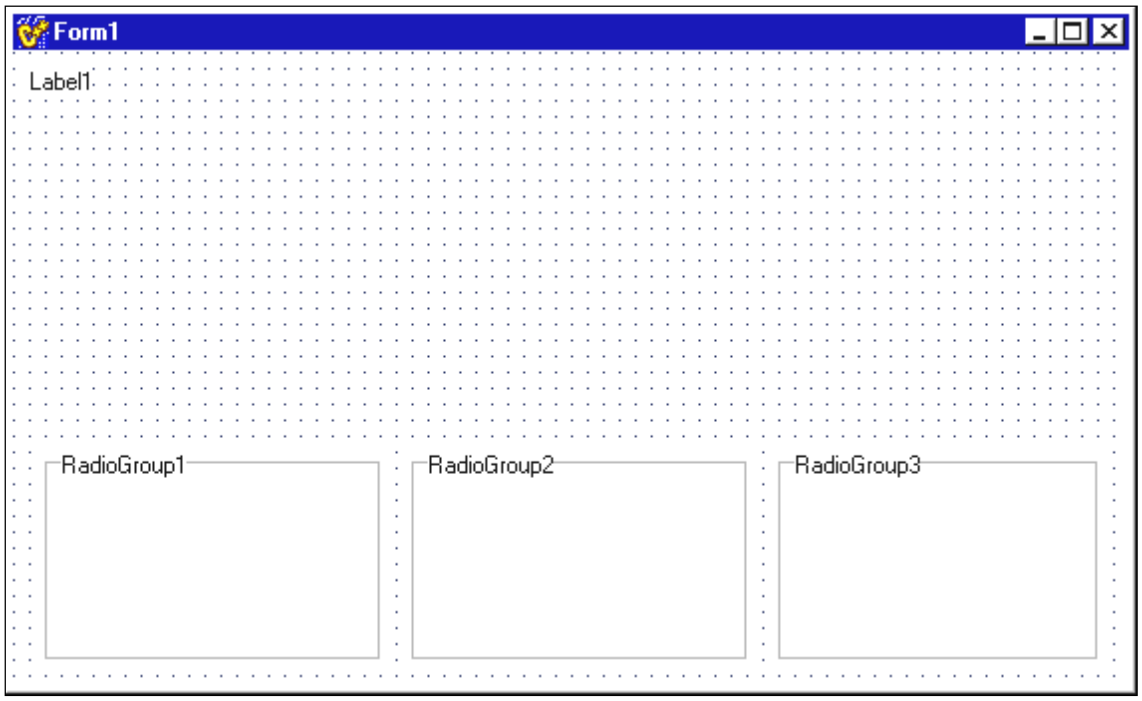


Рисунок 2.21 – Заготовка для управления шрифтом, размером и цветом текста с помощью RadioGroup

Задайте свойства формы и объектов, указанные в таблице 2.9.

Для RadioGroup1 задайте функцию обработки события OnClick:

```
Label1->Font->Name = FontRG ->Items->Strings[FontRG ->ItemIndex];
```

Для RadioGroup2:

```
Label1 -> Font -> Size = SizeRG -> Items -> Strings [SizeRG ->ItemIn-
dex]. ToInt( );
```

Для RadioGroup3:

```
TColor Colors[ ] = {clBlack, clWhite, clRed, clGreen, clBlue, clY-
ellow};
Label1->Font->Color = Colors[ColorRG->ItemIndex];
```

На этом программу можно считать законченной. Обратите внимание на сколько проще она получилась по сравнению с предыдущей. Запустите программу.

Таблица 2.9 – Свойства формы и объектов для управления шрифтом, размером и цветом текста с помощью RadioGroup

Объект	**Свойство**	**Значение**
Форма	Caption	Управление шрифтом, размером и цветом текста
Label1	Caption	Hello World!
RadioGroup1	Name	FontRG
RadioGroup1	Caption	Шрифт
RadioGroup1	Items	Arial Courier Times New Roman
RadioGroup2	Name	SizeRG
RadioGroup2	Caption	Размер
RadioGroup2	Items	8 10 12 16 20 24 32 64
RadioGroup2	Columns	2
RadioGroup3	Name	ColorRG
RadioGroup3	Caption	Цвет
RadioGroup3	Items	Черный Белый Красный Зеленый Синий Желтый

2.14 Комплексный пример

Упражнение 1. Расположите на форме элементы, показанные на рисунке 2.22 (элемент Shape находится на закладке Addition).

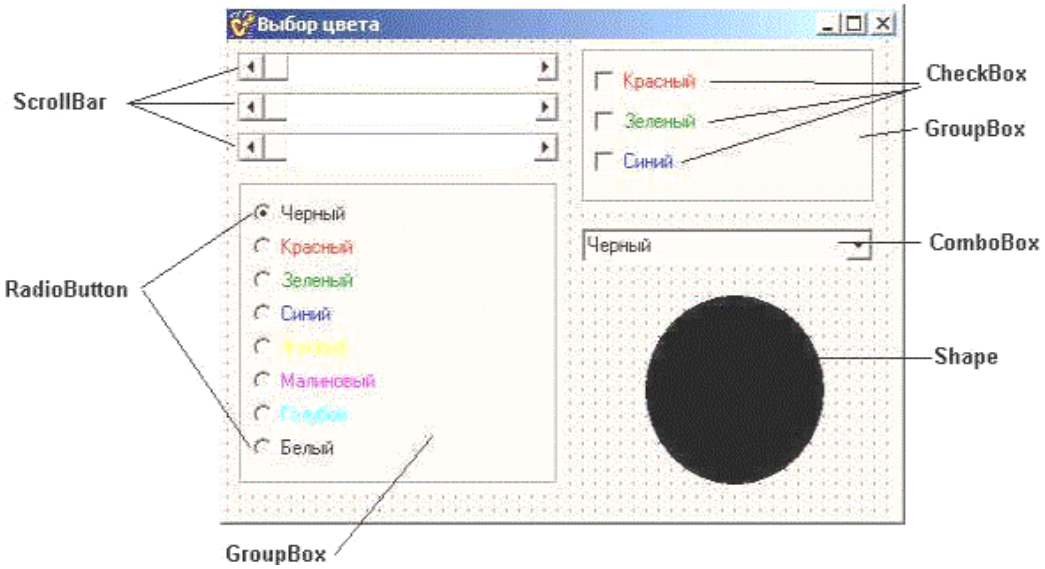


Рисунок 2.22 – Комплексный пример

Установите следующие свойства элементов:

Все ScrollBar: Max = 255.
Все RadioButton: Caption = <цвет>, Font / Color = <цвет>, для первого элемента Checked = true.
Все CheckBox: Caption = <цвет>, Font / Color = <цвет>.
ComboBox : Text = черный, Items = список цветов, каждый с новой строчки, в списке: Черный / Красный / Зеленый / Синий / Желтый / Малиновый / Голубой / Белый.
Shape: Brush / Color = clBlack.

Для того, чтобы цвет фигуры менялся в зависимости от положений ползунков полос прокрутки, следует установить обработчик события OnChange для всех полос прокрутки с текстом:

```
Shape1->Brush->Color = RGB(ScrollBar1->Position, ScrollBar2->Posi-
tion, ScrollBar3->Position);
```

Для того, чтобы цвет фигуры менялся в зависимости от нажатия кнопок RadioButton, следует установить обработчик события OnClick для первой кнопки переключателя:

```
Shape1->Brush->Color = RGB(0, 0, 0);
```

Для остальных кнопок подумайте самостоятельно.

Для того, чтобы цвет фигуры менялся в зависимости от нажатия кнопок CheckBox, следует установить обработчик события OnClick для всех кнопок:

```
Shape1->Brush->Color = RGB(255 *CheckBox1->Checked, 255 * CheckBox2->
Checked, 255*CheckBox3->Checked);
```

Подумайте почему?

Для того чтобы цвет фигуры менялся в зависимости от выбора значения в поле со списком, следует установить для него обработчик события OnChange:

```
if (ComboBox1->Text == "Черный") Shape1->Brush->Color = RGB(0, 0, 0);
```

Остальные 7 строчек придумайте самостоятельно.

В упражнении 1 различные элементы изменяли цвет фигуры, однако они не были синхронизированы друг с другом. Следующее упражнение должно исправить этот недочет.

1. При нажатии RadioButton следует менять:
- положение ползунков полос прокрутки,
- состояние кнопок CheckBox,
- текст в ComboBox.
2. При нажатии CheckBox следует менять:
- положение ползунков полос прокрутки,
- состояние кнопок RadioButton,
- текст в ComboBox.
3. При изменении текста в ComboBox следует менять:
- положение ползунков полос прокрутки,
- состояние кнопок RadioButton,
- состояние кнопок CheckBox.
4. При изменении положения ползунков полос прокрутки следует менять:
- состояние кнопок RadioButton,
- состояние кнопок CheckBox,
- текст в ComboBox,

Выполните эти изменения самостоятельно и покажите преподавателю.

2.15 Контрольные вопросы

1. Для чего используется компонент MainMenu?
2. Что означает свойство Caption пункта меню?
3. Какой тип свойства Caption пункта меню?
4. Для чего используется компонент PopupMenu?
5. Для чего используется компонент Label?
6. Что означает свойство Caption компонента Label?
7. Какой тип свойства Caption компонента Label?
8. Для чего используется компонент Edit?
9. Что означает свойство Text компонента Edit?
10. Какой тип свойства Text компонента Edit?
11. Что означает свойство ReadOnly компонента Edit?
12. Какой тип свойства ReadOnly компонента Edit?
13. Для чего используется компонент Button?

14. Что означает свойство Caption компонента Button?
15. Какой тип свойства Caption компонента Button?
16. Что означает событие OnClick компонента Button?
17. Для чего используется компонент Memo?
18. Что означает свойство Lines компонента Memo?
19. Какой тип свойства Lines компонента Memo?
20. Что означает свойство Align компонента Memo?
21. Какой тип свойства Align компонента Memo?
22. Что означает свойство Alignment компонента Memo?
23. Какой тип свойства Alignment компонента Memo?
24. Что означает метод Clear компонента Memo?
25. Что означает метод ClearSelection компонента Memo?
26. Что означает метод ClearUndo компонента Memo?
27. Что означает метод CopyToClipboard компонента Memo?
28. Что означает метод CutToClipboard компонента Memo?
29. Что означает метод PasteFromClipboard компонента Memo?
30. Что означает метод SelectAll компонента Memo?
31. Что означает метод Undo компонента Memo?
32. Для чего используется компонент CheckBox?
33. Что означает свойство Caption компонента CheckBox?
34. Какой тип свойства Caption компонента CheckBox?
35. Что означает свойство Checked компонента CheckBox?
36. Какой тип свойства Checked компонента CheckBox?
37. Что означает событие OnClick компонента CheckBox?
38. Для чего используется компонент RadioButton?
39. Для чего используется компонент ListBox?
40. Что означает свойство Items компонента ListBox?
41. Какой тип свойства Items компонента ListBox?
42. Что означает свойство ItemIndex компонента ListBox?
43. Какой тип свойства ItemIndex компонента ListBox?
44. Что означает свойство Sorted компонента ListBox?
45. Какой тип свойства Sorted компонента ListBox?
46. Что означает событие OnClick компонента ListBox?
47. Для чего используется компонент ComboBox?
48. Что означает свойство Text компонента ComboBox?
49. Какой тип свойства Text компонента ComboBox?
50. Что означает событие OnChange компонента ComboBox?
51. Для чего используется компонент ScrollBar?
52. Что означает свойство Min компонента ScrollBar?
53. Какой тип свойства Min компонента ScrollBar?
54. Что означает свойство Max компонента ScrollBar?
55. Какой тип свойства Max компонента ScrollBar?
56. Что означает свойство Position компонента ScrollBar?

57. Какой тип свойства Position компонента ScrollBar?
58. Что означает событие OnChange компонента ScrollBar?
59. Для чего используется компонент GroupBox?
60. Что означает свойство Caption компонента GroupBox?
61. Какой тип свойства Caption компонента GroupBox?
62. Что означает событие OnClick компонента GroupBox?
63. Для чего используется компонент RadioGroup?
64. Что означает свойство Caption компонента RadioGroup?
65. Какой тип свойства Caption компонента RadioGroup?
66. Что означает свойство Items компонента RadioGroup?
67. Какой тип свойства Items компонента RadioGroup?
68. Что означает свойство ItemIndex компонента RadioGroup?
69. Какой тип свойства ItemIndex компонента RadioGroup?
70. Что означает свойство Columns компонента RadioGroup?
71. Какой тип свойства Columns компонента RadioGroup?
72. Что означает событие OnClick компонента RadioGroup?

3 КОМПОНЕНТЫ ЗАКЛАДКИ ADDITIONAL

На закладке Additional расположены дополнительные компоненты Windows. Закладка изображена на рисунке 3.1.

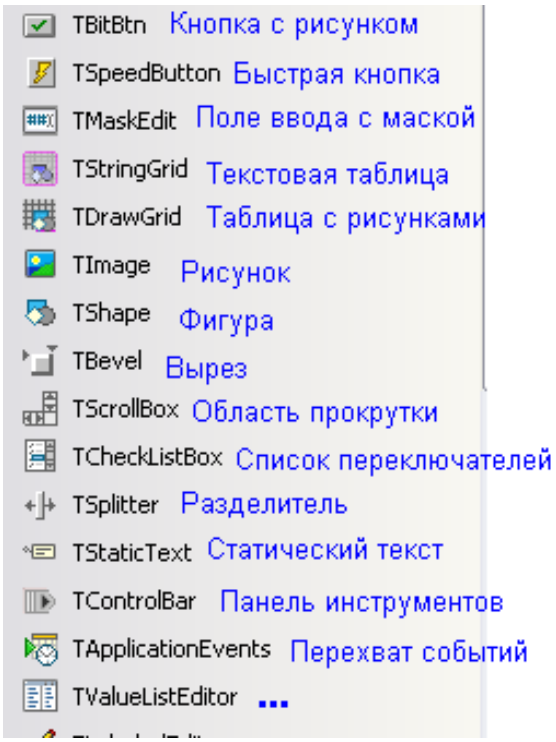


Рисунок 3.1 – Закладка Additional

Объем данного пособия не позволяет подробно рассмотреть все компоненты закладки, поэтому рассмотрим лишь основные.

3.1 Компонент BitBtn (Кнопка с рисунком)

Компонент BitBtn (Кнопка с рисунком) предназначен для размещения на форме кнопки аналогичной компоненту Button, только с рисунком.

К основным свойствам относятся:

- Caption – текст на кнопке (тип свойства AnsiString);
- Kind – вид рисунка; может принимать значения, приведенные в таблице 3.1;
- Layout – взаимное расположение текста и рисунка; может принимать значения, приведенные в таблице 3.2;
- Glyph – рисунок (тип свойства TBitmap).

Основным событием является OnClick.

Таблица 3.1 – Перечень значений свойства Kind

Значение свойства Kind	**Рисунок в кнопке**
bkCustom	Определяется свойством Glyph
bkOK	
bkCancel	
bkYes	
bkNo	
bkHelp	
bkClose	
bkAbort	
bkRetry	
bkIgnore	
bkAll	

Таблица 3.2 – Перечень значений свойства Layout

Значение свойства Layout	Расположение рисунка и текста
blGlyphLeft	Рисунок слева от текста
blGlyphRight	Рисунок справа от текста
blGlyphTop	Рисунок сверху от текста
blGlyphBottom	Рисунок снизу от текста

3.2 Компонент SpeedButton (Быстрая кнопка)

Данный вид кнопки имеет практически такие же основные свойства и методы, как и предыдущий, поэтому рассмотрение пропустим.

3.3 Компонент MaskEdit (Поле ввода с маской)

В набор свойств компонента по сравнению компонентом Edit добавлено свойство EditMask, задающее ограничение на тип символа в каждой позиции ввода. Тип свойства – AnsiString. Правила разрешения ввода символов приведены в таблице 3.3.

Таблица 3.3 – Правила разрешения ввода символов

Символ маски	Разрешенные символы ввода
L	A-Z, a-z
A	A-Z, a-z, 0-9
9	0-9 с подавлением незначащих нулей
0	0-9 без подавления незначащих нулей
\символ	Только указанный символ
_	Позиция пропускается

Например, для ввода номера телефона в виде (3512) 33-33-33 следует использовать маску: \(999\)_99\-99\-99. Если дважды щелкнуть по значению свойства EditMask, то откроется редактор масок, который существенно упростит задание маски.

3.4 Компонент StringGrid (Текстовая таблица)

Данный компонент предназначен для реализации таблицы, содержащей текстовую или числовую информацию. Таблица имеет вид, приведенный на рисунке 3.2.

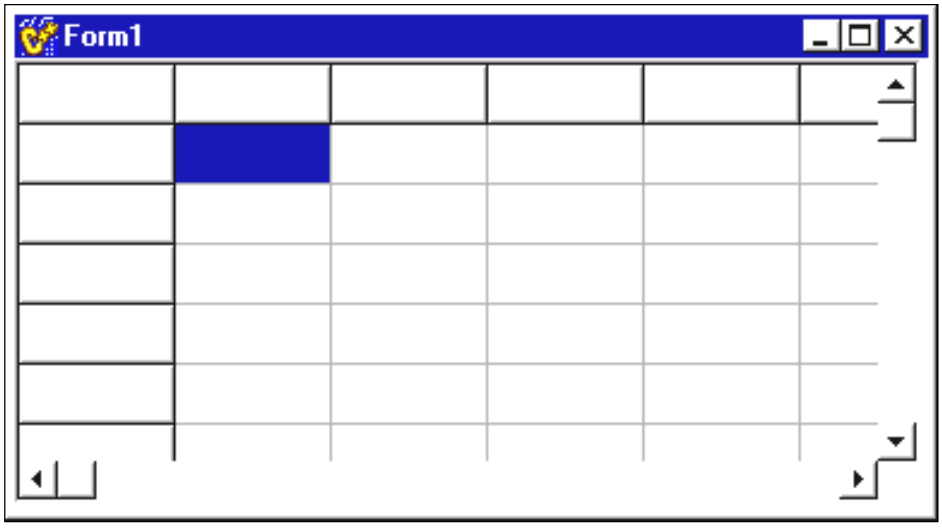


Рисунок 3.2 – Вид текстовой таблицы

К основным свойствам компонента StringGrid относятся:

- RowCount – количество строк (тип int);
- ColCount – количество столбцов (тип int);
- FixedRows – количество фиксированных строк (тип int), на рис. 3.2 фиксированными являются первая строка и первый столбец, фиксированные строки и столбцы возвышаются над остальными, при запуске программы пользователь не может вводить в них информацию, обычно их используют для обозначения заглавий строк и столбцы;
- FixedCols – количество фиксированных столбцов (тип int);
- RowHeights – целочисленный массив высот строк, это свойство является программируемым, оно отсутствует в инспекторе объектов и его значение можно изменять только в программе;
- ColWidths – целочисленный массив ширин столбцов, также программируемое свойство;
- Cells – двумерный массив типа AnsiString, содержащий значения элементов таблицы, также программируемое свойство;
- Options – параметры таблицы, это составное свойство, наиболее важным подсвойством которого является goEditing – разрешение на редактирование.

В качестве примера рассмотрим математическую задачу перемножения матриц.

Расположите на форме элементы, показанные на рисунке 3.3.

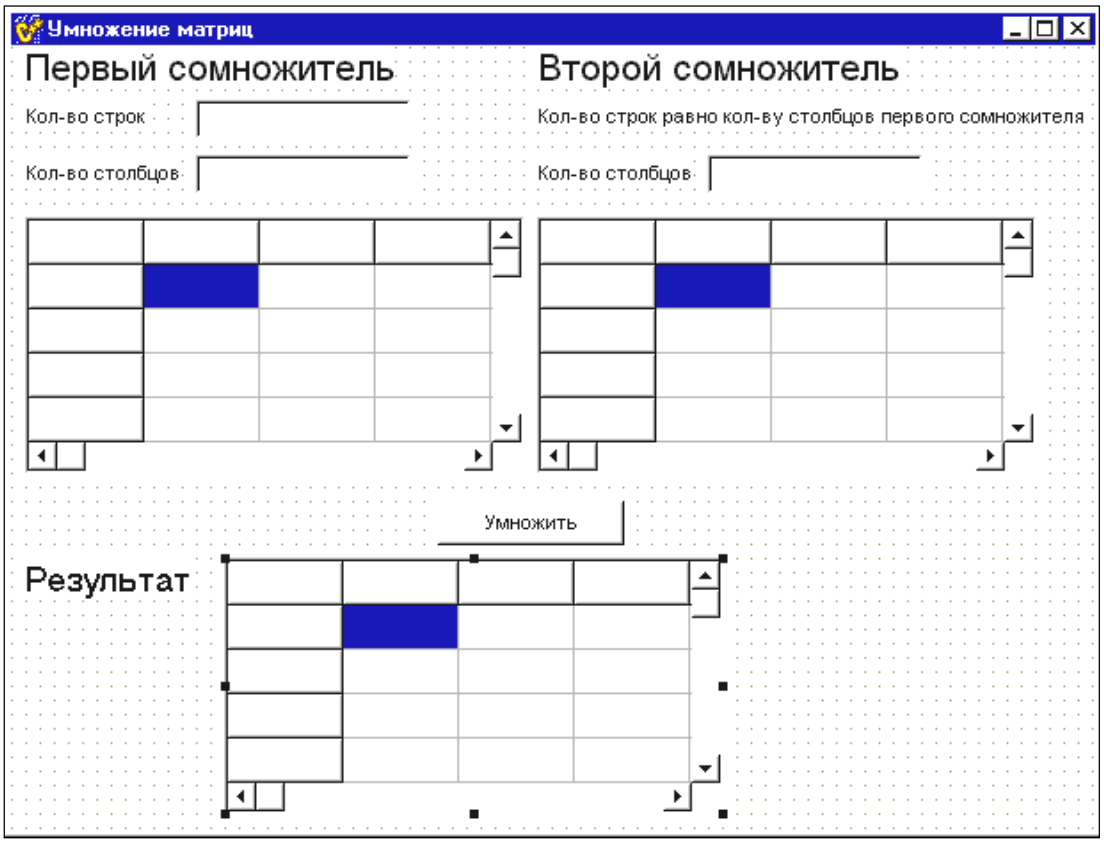


Рисунок 3.3 – Форма для перемножения матриц

Задайте свойства Caption и Font для текста, Text – для полей ввода, Caption – для кнопки, для первых двух таблиц задайте свойство Options, подсвойство goEditing = true, все остальные свойства таблиц будем задавить программно.

Для всех полей зададим обработчик события OnExit. Это событие происходит при покидании поля ввода. Для первого оно будет выглядеть так:

```
int n, i;
n = Edit1->Text.ToInt( );
StringGrid1->RowCount = n+1;
StringGrid3->RowCount = n+1;
for (i=1; i<n+1; i++) {
      StringGrid1->Cells[0][i] = i;
      StringGrid3->Cells[0][i] = i;
} // for
```

Для второго поля ввода обработчик события OnExit будет выглядеть так:

```
int m, i;
m = Edit2->Text.ToInt( );
StringGrid1->ColCount = m+1;
StringGrid2->RowCount = m+1;
for (i=1; i<m+1; i++) {
      StringGrid1->Cells[i][0] = i;
      StringGrid2->Cells[0][i] = i;
} // for
```

Для третьего поля ввода обработчик события OnExit будет выглядеть так:

```
int l, i;
l = Edit3->Text.ToInt( );
StringGrid2->ColCount = l+1;
StringGrid3->ColCount = l+1;
for (i=1; i<l+1; i++) {
     StringGrid2->Cells[i][0] = i;
     StringGrid3->Cells[i][0] = i;
} // for
```

Для кнопки следует задать обработчик события OnClick :

```
int n, m, l, i, j, k;
double a;
n = Edit1->Text.ToInt( );
m = Edit2->Text.ToInt( );
l = Edit3->Text.ToInt( );
for (i = 1; i <= n; i++) {
 for (j = 1; j <= l; j++) {
 a = 0;
 for (k = 1; k <= m; k++) {
 a += StringGrid1->Cells[k][i].ToDouble( ) *
 StringGrid2->Cells[j][k].ToDouble( );} // for k
 StringGrid3->Cells[j][i] = a;} // for j
} // for i
```

На этом программу можно считать законченной. Запустите ее. Не забудьте сохранить проект на диск W.

3.5 Компонент DrawGrid (Таблица с рисунками)

Таблица с рисунками аналогична текстовой таблице, однако в ее ячейках можно рисовать геометрические фигуры и помещать другие рисунки. В отличие от текстовой таблицы у этого компонента нет свойства Cells, однако есть свойство Canvas, с помощью которого можно рисовать фигуры и рисунки. Что это за свойство и как с помощью него рисовать, будет рассмотрено в 6.2.

3.6 Компонент Image (Рисунок)

Данный компонент предназначен для размещения на форме рисунка. Рисунок может быть загружен из файла или скопирован через буфер обмена (clipboard). Основное свойство Picture – собственно рисунок (тип TPicture). Рассмотрим это свойство более подробно.

В процессе конструирования программы можно выбрать рисунок следующим образом: в инспекторе объектов дважды щелкнем по значению свойства Picture, при этом откроется окно, показанное на рисунке 3.4.

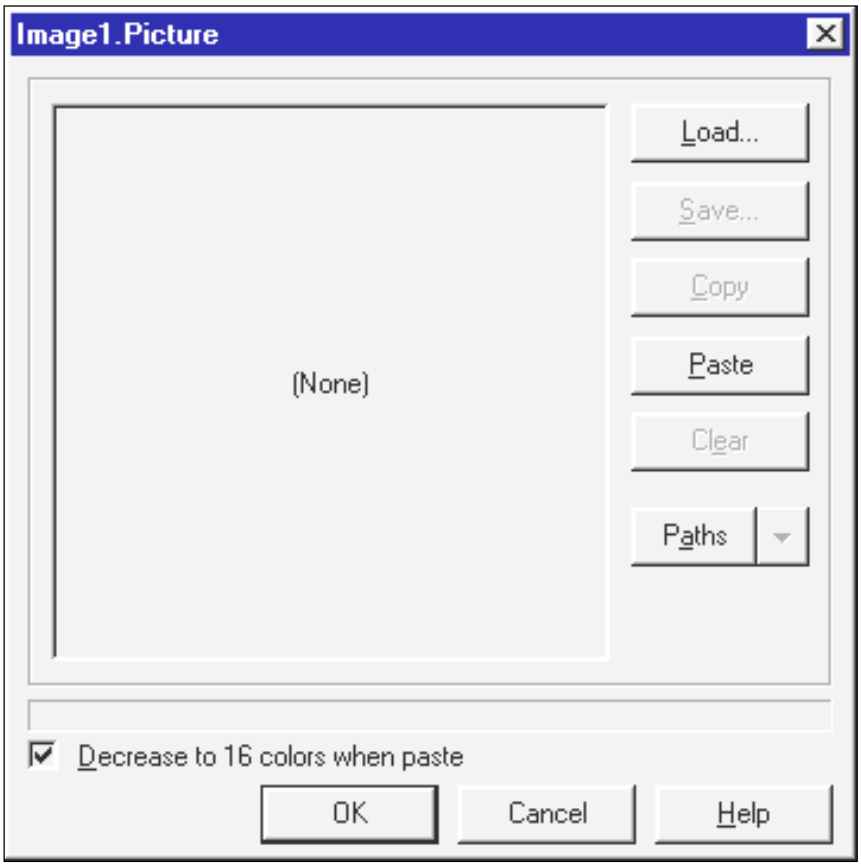


Рисунок 3.4 – Окно для выбора рисунка

Если рисунок загружается из файла, то следует нажать кнопку Load и в следующем окне выбрать файл. Если рисунок вставляется из буфера обмена, то следует в какой-либо другой программе поместить его в буфер, а затем нажать кнопку Paste.

Рисунок можно добавлять и изменять в процессе работы программы. Для этого следует воспользоваться методом LoadFromFile(AnsiString FileName) класса TPicture, который загружает рисунок из указанного файла. Например:

Image1->Picture->LoadFromFile(“C:\\Windows\\Облака.bmp”);

3.7 Компонент Shape (Фигура)

Компонент Shape (Фигура) позволяет размещать на форме некоторые геометрические фигуры: прямоугольник, квадрат, круг и эллипс. Основные свойства:

- Name – имя фигуры (тип свойства AnsiString);
- Shape – вид фигуры (перечислимый тип, возможные значения приведены в таблице 3.4.);
- Pen – перо для рисования границы фигуры, составной тип, состоящий из:
    - Color – цвет (тип TColor);

- Style – стиль границы (перечислимый тип, возможные значения приведены в таблице 3.5.);
- Width – ширина границы (тип int);

• Brush – кисть для закрашивания внутренней части фигуры, составной тип состоящий из следующих подсвойств:

- Color – цвет (тип TColor);
- Style – стиль закрашивания (перечислимый тип, возможные значения приведены в таблице 3.6.).

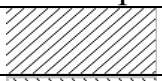
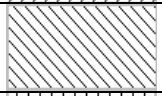
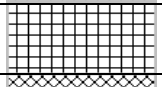
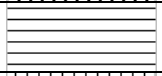
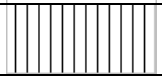
Таблица 3.4 – Перечень видов фигур

Наименование вида фигуры	**Описание**
stCircle	Круг
stEllipse	Эллипс
stRectangle	Прямоугольник
stRoundRect	Прямоугольник со скругленными углами
stSquare	Квадрат
stRoundSquare	Квадрат со скругленными углами

Таблица 3.5 – Перечень стилей пера

Наименование стиля	**Вид**
psSolid	Сплошная линия
psDash	Штриховая линия
psDot	Пунктирная линия
psDashDot	Штрих пунктирная линия
psClear	Невидимая линия

Таблица 3.6 – Перечень стилей кисти

Наименование стиля	**Вид**
bsSolid	Сплошное закрашивание
bsClear	Без закрашивания
bsBDiagonal	
bsFDiagonal	
bsCross	
bsDiagCross	
bsHorizontal	
bsVertical	

3.8 Компонент ScrollBox (Область прокрутки)

Если в некоторой прямоугольной области окна требуется разместить больше элементов, чем туда может войти, то следует пользоваться компонентом ScrollBox. В случае, если элементы не помещаются по вертикали, то автоматически создается вертикальная полоса прокрутки (если помещаются, то она не создается). Аналогично по горизонтали. Иллюстрация области прокрутки показана на рисунке 3.5.

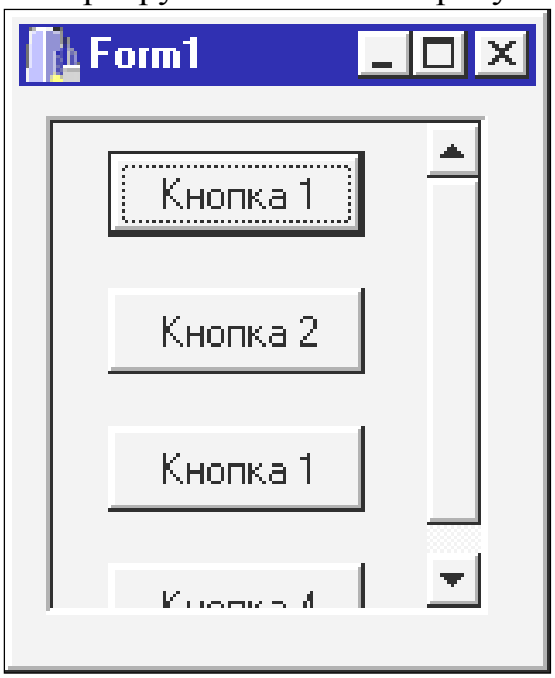


Рисунок 3.5 – Иллюстрация области прокрутки

3.9 Компонент CheckListBox (Список переключателей)

Компонент CheckListBox предназначен для реализации групп переключателей. В отличие от GroupBox данная группа не может содержать ничего кроме переключателей. При этом не нужно создавать отдельно каждый переключатель и задавать ему свойства, достаточно задать свойства всего списка.

Основные свойства CheckListBox:

- Caption – заголовок группы (тип AnsiString);
- Items – заголовки отдельных переключателей группы (тип TStrings). Двойной щелчок в инспекторе объектов по значению этого свойства приводит к открытию редактора текста для ввода заголовков;
- Checked – массив логических значений, каждое значение означает включен или нет соответствующий переключатель;
- Columns – количество столбцов переключателей (тип int).

Основным событием является OnClick.

3.10 Компонент StaticText (Статический текст)

Компонент StaticText аналогичен компоненту Label, однако вокруг текста может быть рамка. Для ее реализации добавлено свойство BorderStyle, которое имеет перечислимый тип, и может принимать одно из значений:

- sbsNone – нет рамки,
- sbsSingle – простая рамка,
- sbsSunken – утопленная рамка.

3.11 Контрольные вопросы

1. Для чего используется компонент BitBtn?
2. Что означает свойство Kind компонента BitBtn?
3. Что означает свойство Layout компонента BitBtn?
4. Что означает свойство Glyph компонента BitBtn?
5. Какой тип свойства Glyph компонента BitBtn?
6. Для чего используется компонент MaskEdit?
7. Какие элементы управления лучше использовать для таблиц с картинками?
8. Для чего используется компонент StringGrid?
9. Для чего используется компонент DrawGrid?
10. Что означает свойство RowCount компонента StringGrid?
11. Какой тип свойства RowCount компонента StringGrid?
12. Что означает свойство ColCount компонента StringGrid?
13. Какой тип свойства ColCount компонента StringGrid?
14. Что означает свойство FixedRows компонента StringGrid?
15. Какой тип свойства FixedRows компонента StringGrid?
16. Что означает свойство FixedCols компонента StringGrid?
17. Какой тип свойства FixedCols компонента StringGrid?
18. Что означает свойство RowHeights компонента StringGrid?
19. Какой тип свойства RowHeights компонента StringGrid?
20. Что означает свойство ColWidths компонента StringGrid?
21. Какой тип свойства ColWidths компонента StringGrid?
22. Что означает свойство Cells компонента StringGrid?
23. Какой тип свойства Cells компонента StringGrid?
24. Для чего используется компонент DrawGrid?
25. Для чего используется компонент Image?
26. Что означает свойство Picture компонента Image?
27. Какой тип свойства Picture компонента Image?
28. Для чего используется метод LoadFromFile класса TPicture?
29. Что означает параметр метода LoadFromFile?
30. Для чего используется компонент Shape?
31. Что означает свойство Shape компонента Shape?

32. Что означает свойство Pen компонента Shape?
33. Что означает подсвойство Color свойства Pen?
34. Что означает подсвойство Style свойства Pen?
35. Что означает подсвойство Width свойства Pen?
36. Что означает свойство Brush компонента Shape?
37. Что означает подсвойство Color свойства Brush?
38. Что означает подсвойство Style свойства Brush?
39. Для чего используется компонент ScrollBox?
40. Для чего используется компонент CheckListBox?
41. Что означает свойство Items компонента CheckListBox?
42. Какой тип свойства Items компонента CheckListBox?
43. Что означает свойство Checked компонента CheckListBox?
44. Какой тип свойства Checked компонента CheckListBox?
45. Что означает свойство Columns компонента CheckListBox?
46. Какой тип свойства Columns компонента CheckListBox?
47. Для чего используется компонент StaticText?
48. Что означает свойство BorderStyle компонента StaticText?

4 КОМПОНЕНТЫ ЗАКЛАДКИ DIALOGS

На закладке Dialogs расположены компоненты, которые не видны в месте их расположения на форме во время выполнения программы, их действие осуществляется при вызове метода Execute. Этот метод запускает другую форму для выбора некоторых параметров, которые зависят от конкретного компонента.

Закладка изображена на рисунке 4.1.

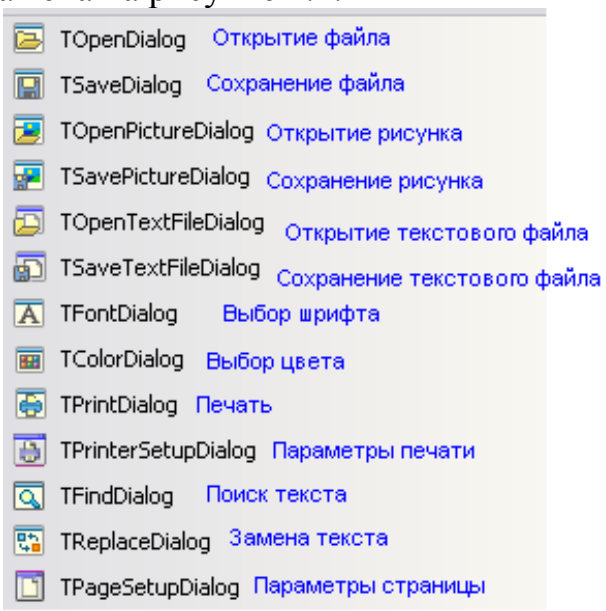


Рисунок 4.1 – Закладка Dialogs

4.1 Компонент OpenDialog (Диалог для открытия файла)

Компонент OpenDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор имени файла, предназначенного для открытия документа. Диалоговая форма изображена на рисунке 4.2.

Основные свойства OpenDialog:

- FileName – имя выбранного файла (тип AnsiString);
- DefaultExt – расширение файла по умолчанию (тип AnsiString);
- InitialDir – начальная папка для показа (тип AnsiString);
- Title – заголовок панели (тип AnsiString);
- Filter –задает расширения, с которыми видны файлы (тип AnsiString); двойной щелчок по значению свойства Filter вызывает редактор фильтра, в котором можно задать ограничения на видимость (рис 4.3);
- FilterIndex – номер действующего фильтра.

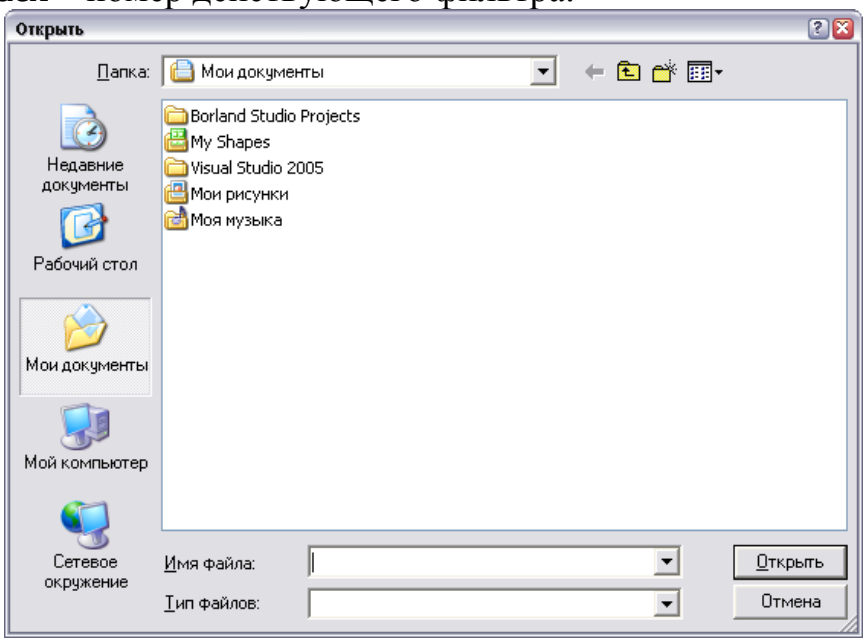


Рисунок 4.2 – Диалоговая панель для открытия файла

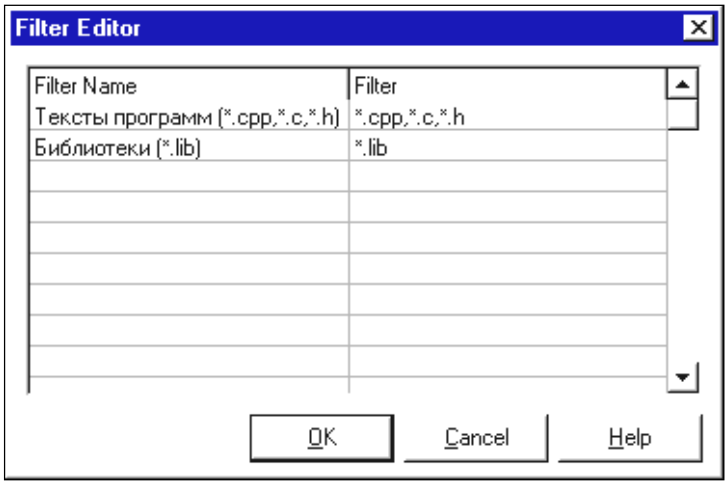


Рисунок 4.3 – Фильтр на видимость файлов

Фильтров может быть несколько. Приведенный на рис 4.3 пример делает 3 фильтра, первый разрешает показ файлов с расширениями cpp, c и h, второй только с расширениями lib. Выбор нужного фильтра производится в поле диалоговой панели "Тип файлов".

Основной метод Execute запускает диалоговую панель. Метод не имеет параметров, возвращает логическое значение: true – если в диалоговой панели щелкнули по кнопке «Открыть», false – если «Отмена».

В качестве примера рассмотрим текстовый редактор из пункта 2.6. В конце этого пункта предлагалось сохранить этот редактор для дальнейшего усовершенствования, теперь это время настало.

В любом месте формы расположите компонент OpenDialog и задайте его свойства, указанные в таблице 4.1.

Таблица 4.1 – Свойства OpenDialog

<table>
<tr><th>Свойство</th><th>Значение</th></tr>
<tr><td>FileName</td><td>*.txt</td></tr>
<tr><td>DefaultExt</td><td>txt</td></tr>
<tr><td>Title</td><td>Открытие файла</td></tr>
<tr><td>Filter</td><td>
<table>
<tr><th>FilterName</th><th>Filter</th></tr>
<tr><td>Текстовые документы</td><td>*.txt</td></tr>
<tr><td>Все файлы</td><td>*.*</td></tr>
</table>
</td></tr>
</table>

Дважды щелкните по компоненте “меню”, в открывшемся окне вставьте пункт “Открыть” между пунктами “Создать” и “Сохранить”. Для пункта “Открыть” задайте функцию:

```
int  Rc;
Rc = OpenDialog1->Execute();
if (Rc) {
     Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
} // if
```

Запустите программу. Выберите файл для редактирования. Закройте программу и сохраните ее, в следующем пункте рассмотрим выбор имени файла для сохранения документа.

4.2 Компонент SaveDialog (Диалог для сохранения в файле)

Компонент SaveDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор имени файла, предназначенного для сохранения документа. Диалоговая форма имеет практически такой же вид, как и у компонента OpenDialog (рис. 4.2). Внешнее отличие состоит в том, что в кнопке вместо слова «открыть» написано слово «сохранить». Внутренним отличием является то, что в OpenDialog всегда выбирается существующий

файл, а в SaveDialog либо новый, либо существующий. Если файл существует, то возможны следующие варианты действий:

- отказ от сохранения,
- сохранение с предупреждением о перезаписи этого файла,
- сохранение без предупреждения.
- Основные свойства SaveDialog:
- Свойства FileName, DefaultExt, InitialDir, Title, Filter и FilterIndex аналогичны соответствующим свойствам компонента OpenDialog (см. 4.1).
- Свойство Options содержит параметры работы диалоговой панели. Параметры состоят из отдельных составляющих. Например, составляющая ofOverwritePrompt говорит о том, что при попытке выбрать существующий файл будет выдано предупреждение с запросом о продолжении.

Основной метод Execute полностью аналогичен компоненту OpenDialog.

Усовершенствуем программу из предыдущего пункта.

В любом месте формы расположите компонент SaveDialog, задайте его свойства, указанные в таблице 4.2.

Таблица 4.2 – Свойства SaveDialog

<table>
<tr><th>Свойство</th><th>Значение</th></tr>
<tr><td>FileName</td><td>Безымянный</td></tr>
<tr><td>DefaultExt</td><td>txt</td></tr>
<tr><td>Title</td><td>Сохранение файла</td></tr>
<tr><td>Options</td><td>ofOverwritePrompt</td></tr>
<tr><td>Filter</td><td>
<table>
<tr><th>FilterName</th><th>Filter</th></tr>
<tr><td>Текстовые документы</td><td>*.txt</td></tr>
<tr><td>Все файлы</td><td>*.*</td></tr>
</table>
</td></tr>
</table>

В меню, после пункта «Сохранить», добавьте пункт «Сохранить как …». Для этого пункта задайте функцию:

```
int  Rc;
Rc = SaveDialog1->Execute( );
if (Rc) {
     Memo1->Lines->SaveToFile(SaveDialog1->FileName);
} // if
```

Запустите программу. Выберите файл для редактирования, измените его и сохраните его с помощью пункта «Сохранить как».

В данной программе есть несколько недостатков:

1) неадекватно работает кнопка «Сохранить». Она предлагает сохранить отредактированный текст в файле с заданным именем FileName, а следует с тем именем, с которым его открывали или сохраняли предыдущий раз. Если

документ только что создан и не был сохранен, то следует запросить имя файла;
2) если документ был изменен, то при выборе пунктов «Создать» или «Открыть» следует выдавать предупреждение о том, что текущий документ не сохранен, и запрашивать, следует ли его сохранить.

Для устранения этих недостатков внесем в программу изменения.

В секции public файла описания класса TForm1 Unit1.h опишем 4 прототипа функций:

```
void NewDocument(void);
void OpenDocument(void);
void SaveDocument(void);
void SaveAsDocument(void);
```

и глобальную переменную типа **AnsiString FileName**, в которой будем хранить имя файла, с которым последний раз открывали или сохраняли документ. Для этого необходимо щелкнуть правой кнопкой мыши по корешку закладки файла Unit1.cpp в Редакторе кода и в открывшемся меню выбрать пункт Open Source/Header File.

В обработчиках события OnClick для пунктов меню «Создать», «Открыть», «Сохранить» и «Сохранить как» запишем вызовы функций:

```
NewDocument(); // В функции пункта меню «Создать»
OpenDocument(); // В функции пункта меню «Создать»
SaveDocument();// В функции пункта меню «Сохранить»
SaveAsDocument();// В функции пункта меню «Сохранить как»
```

В конце файла программы добавим их реализации, которые и будут выполнять все действия с файлами.

```
void TForm1::NewDocument(void)
{
int Rc;
if (Memo1->Modified) {
Rc = Application->MessageBox("Текст был изменен\n"
"Сохранить изменение?", "Вопрос", MB_YESNOCANCEL | MB_ICONQUESTION);
          if (Rc == IDCANCEL) return;
          if (Rc == IDYES) {
               SaveDocument( );
          } // if
     } // if
     Memo1->Clear( );
     FileName = "";
} // NewDocument

void TForm1::OpenDocument(void)

{
int Rc;
if (Memo1->Modified) {
Rc = Application->MessageBox("Текст был изменен\n" "Сохранить измене-
ние?", "Вопрос", MB_YESNOCANCEL | MB_ICONQUESTION);
```

```
      if (Rc == IDCANCEL) return;
      if (Rc == IDYES) {
            SaveDocument( );
      } // if
} // if
Rc = OpenDialog1->Execute( );
if (Rc) {
            Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
            FileName = OpenDialog1->FileName;
} // if
} // OpenDocument

void TForm1::SaveDocument(void)

{
if (FileName.IsEmpty( )) {
      SaveAsDocument( );
} else {
            Memo1->Lines->SaveToFile(FileName);
} // else
} // SaveDocument


void TForm1::SaveAsDocument(void)

{
int   Rc;
SaveDialog1->FileName = FileName;
Rc = SaveDialog1->Execute( );
if (Rc) {
            Memo1->Lines->SaveToFile(SaveDialog1->FileName);
            FileName = SaveDialog1->FileName;
} // if
} // SaveAsDocument
```

Запустите программу и проверьте ее работоспособность. Сохраните проект, т.к. в дальнейших пунктах мы продолжим его усовершенствование.

4.3 Компонент OpenPictureDialog (Диалог для открытия рисунка)

Компонент OpenPictureDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор имени файла, предназначенного для открытия рисунка. Компонент похож на OpenDialog. Точка Его отличительные особенности перечислены ниже.

В диалоговой панели добавлено поле для предварительного просмотра рисунка (рисунок 4.4).

В фильтр автоматически подставляются все зарегистрированные на данном компьютере графические форматы.

4.4 Компонент SavePictureDialog (Диалог для сохранения рисунка)

Компонент SavePictureDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор имени файла, предназначенного для сохранения рисунка. Диалоговая форма имеет практически такой же вид, как и у компонента OpenPictureDialog (рис. 4.4). Отличие состоит в том, что в кнопке вместо слова «открыть» написано «сохранить».

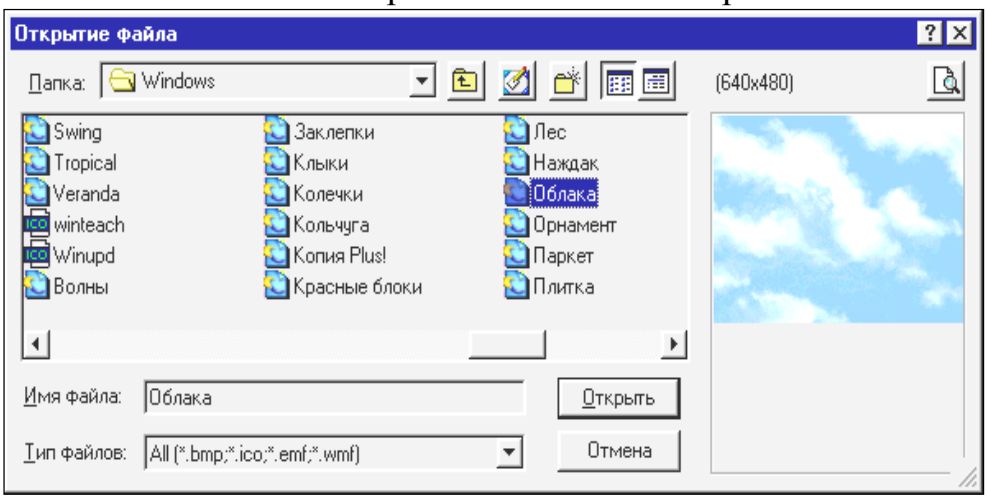


Рисунок 4.4 – Диалоговая панель для выбора рисунка

4.5 Компонент FontDialog (Диалог для выбор шрифта)

Компонент FontDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор шрифта. Диалоговая форма имеет вид, изображенный на рисунке 4.5.

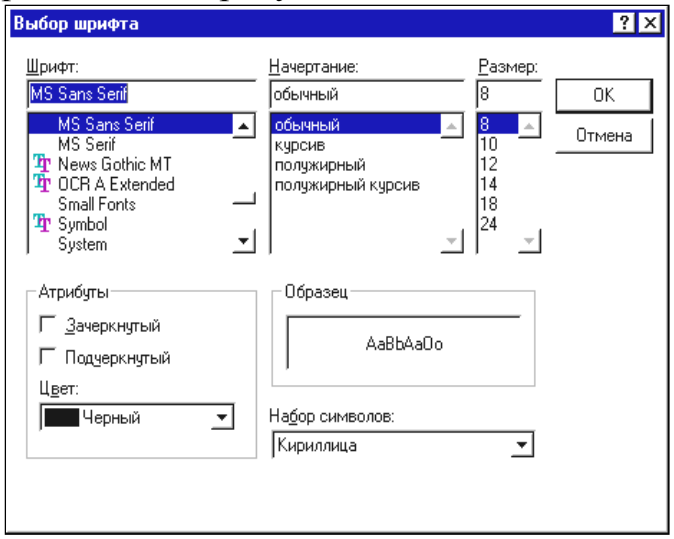


Рисунок 4.5 – Диалоговая панель для выбора шрифта

Основные свойства FontDialog:

• Font – выбранный шрифт (тип TFont), состоит из нескольких подсвойств, основными из которых являются:

- Color – цвет текста (тип TColor);
- Name – название шрифта (тип AnsiString);
- Size – размер шрифта (тип int);

• Device – тип устройств, поддерживающий шрифты. Данное свойство может принимать следующие значения:

- fdScreen – шрифты для экрана;
- fdPrinter – шрифты для принтера;
- fdBoth – оба типа.

Основным методом, как и для предыдущих компонент, является метод Execute, который запускает диалоговую панель.

В качестве примера продолжим рассматривать текстовый редактор.

В любом месте формы расположите компонент FontDialog. В подменю «Правка» ниже пункта «Выделить все» добавьте пункт «Шрифт…», задайте для него функцию следующего вида:

```
int Rc;
Rc = FontDialog1->Execute( );
if (Rc) {
      Memo1->Font = FontDialog1->Font;
} // if
```

Запустите программу и проверьте ее работоспособность, выбрав и изменив несколько раз шрифты. Сохраните проект, т.к. в дальнейших пунктах мы продолжим его усовершенствование.

4.6 Компонент ColorDialog (Диалог для выбор цвета)

Компонент ColorDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор цвета. Диалоговая форма имеет вид, изображенный на рисунке 4.6.

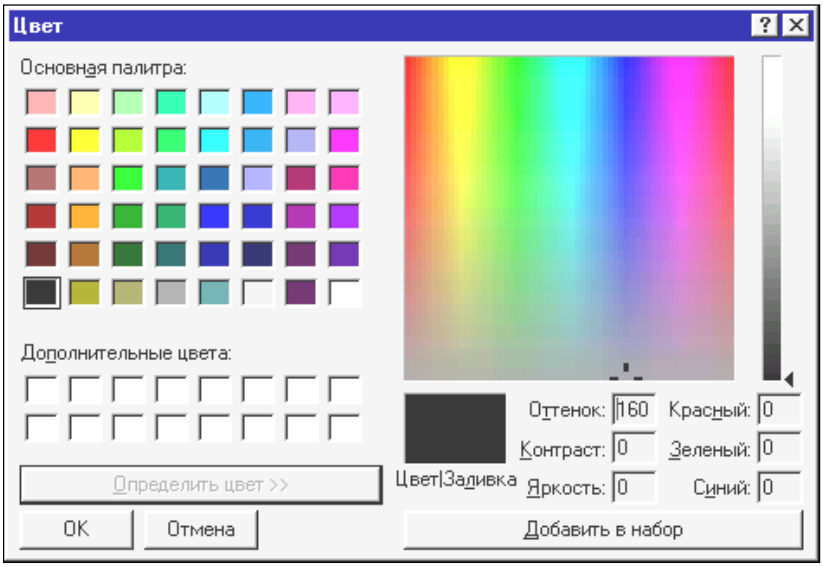


Рисунок 4.6 – Диалоговая панель для выбора цвета

Основным свойством компонента является Color – выбранный цвет (тип TColor).

Основным методом, как и для предыдущих компонент, будет Execute, который запускает диалоговую панель.

В качестве примера рассмотрим выбор цвета круга.

Создайте новый проект и расположите на нем 3 компонента: Shape (Фигура), Button (Кнопка) и ColorDialog (рисунок 4.7).

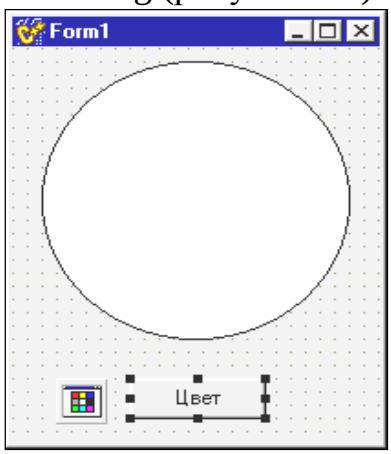


Рисунок 4.7 – Выбор цвета круга

Для кнопки задайте функцию:

```
int Rc;
Rc = ColorDialog1->Execute( );
if (Rc) {
      Shape1->Brush->Color = ColorDialog1->Color;
} // if
```

Запустите программу и проверьте ее работоспособность.

4.7 Компонент PrintDialog (Диалог печати)

Компонент PrintDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор некоторых параметров печати. Диалоговая форма имеет вид, изображенный на рисунке 4.8.

Основные свойства компонента.

- PrintRange – область печати, может принимать 3 значения:
  prAllPages – печатать все страницы;
  prSelection – печатать выделенный фрагмент;
  prPageNums – печатать в указанном диапазоне страниц;
- FromPage – начальная страница диапазона печати (тип int);
- ToPage – конечная страница диапазона печати (тип int);
- Collate – разобрать по страницам? (логический тип);
  Copies – количество копий.

Основным методом, как и для предыдущих компонент, является Execute, который запускает диалоговую панель. Однако, для того, чтобы напечатать этих знаний недостаточно, поэтому рассмотрим класс TPrinter, .который реализует работу с текущим принтером. Сначала мы должны получить указатель на класс TPrinter, для этого используется функция Printer().

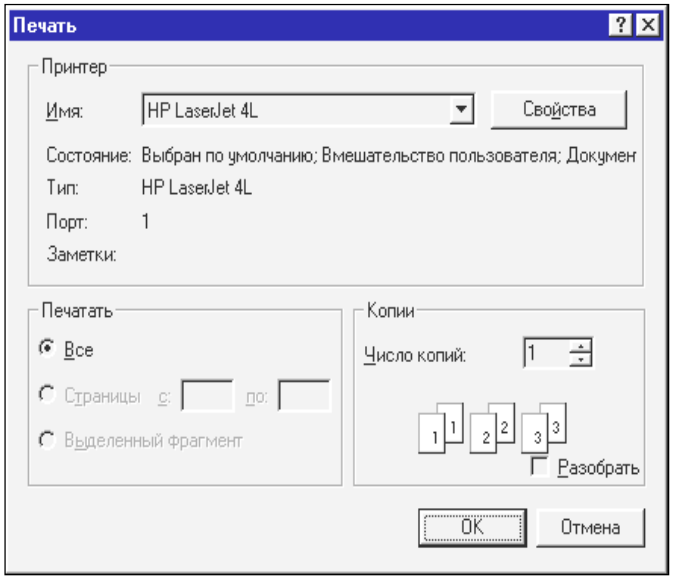


Рисунок 4.8 – Форма диалога печати

Основные свойства класса TPrinter:

- PageHeight – высота страницы (тип int);
- PageWidth – ширина страницы (тип int);
- PageNumber – кол-во страниц (тип int);
- Orientation – ориентация (poPortrait – вертикальная, poLandscape – горизонтальная);
- Canvas – холст для изображения (тип TCanvas, подробно этот тип рассмотрен в пункте 6.2; здесь рассмотрим один метод: TextOut(int X, int Y, AnsiString Text), который в данном случае выводит на печать текст (указанный в 3-ем параметре), начиная с позиции (X,Y)).

Основные методы класса TPrinter:

- BeginDoc( ) – начало печати документа;
- EndDoc( ) – окончание печати документа;
- NewPage( ) – переход на новую страницу.

В качестве примера продолжим рассматривать текстовый редактор. В любом месте формы расположите компонент PrintDialog. После пункта меню «Сохранить как…» сделайте разделитель. После разделителя создайте пункт «Печать», для которого напишите функцию:

```
int Rc;
```

```
Rc = PrintDialog1->Execute( );
if (Rc) {
TPrinter *Prntr = Printer( );
Prntr->BeginDoc( );
Prntr->Canvas->TextOut(200, 200, Memo1->Lines->Text);
Prntr->EndDoc( );
} // if
```

В начале текста напишите строчку:

```
#include "printers.hpp"
```

4.8 Компонент PrinterSetupDialog (Параметры печати)

Компонент PrinterSetupDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор параметров печати, показанных на рисунке 4.9.

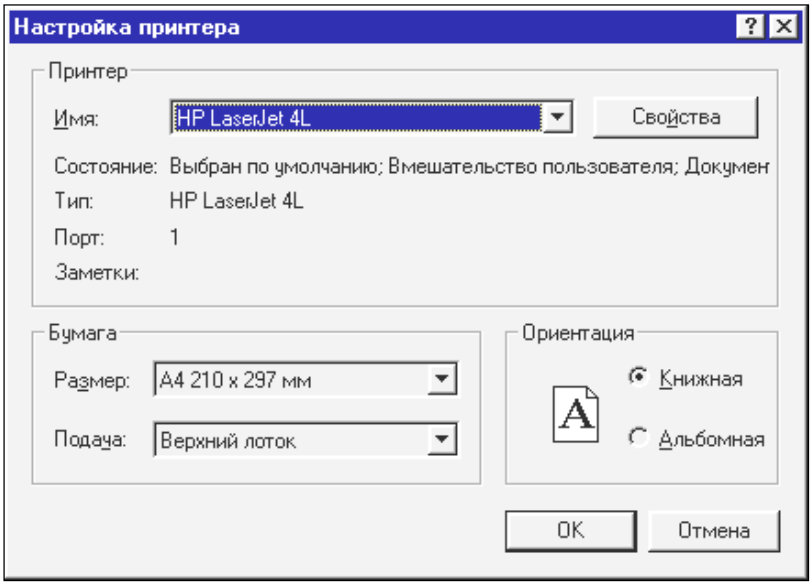


Рисунок 4.9 – Форма диалога параметров печати

Основным методом, как и для предыдущих компонент, является Execute, который запускает диалоговую панель.

В качестве примера продолжим рассматривать текстовый редактор. В любом месте формы расположите компонент PrinterSetupDialog. После пункта меню «Печать» создайте пункт «Параметры печати», для которого напишите функцию:

```
PrinterSetupDialog1->Execute( );
```

Установленные параметры без нашего участия попадут в класс TPrinter и будут задействованы при печати.

4.9 Компонент FindDialog (Диалог поиска)

Компонент FindDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор параметров поиска текста. Диалоговая форма имеет вид, изображенный на рисунке 4.10.

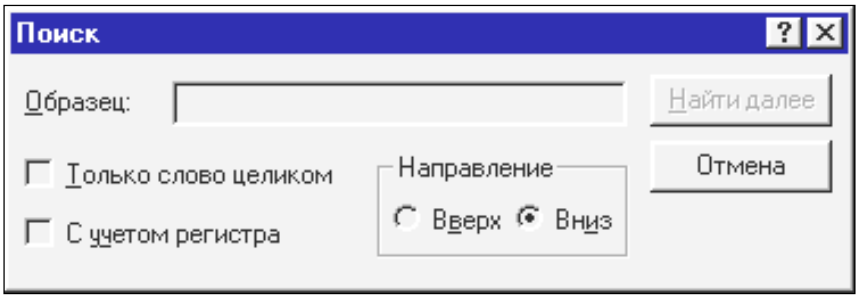


Рисунок 4.10 – Диалог поиска

Основные свойства компонента:

- FindText – текст для поиска (тип AnsiString);
- Option – параметры поиска. Данное свойство состоит из флагов, основными из которых являются:
  - frDown – искать вниз;
  - frMatchCase – с учетом регистра;
  - frWholeWord – только слово целиком.

Основным событием является OnFind, которое возникает в случае нахождения текста.

Основным методом, как и для предыдущих компонент, является Execute, который запускает диалоговую панель.

В качестве примера продолжим рассматривать текстовый редактор. В любом месте формы расположите компонент FindDialog. В горизонтальной части меню после пункта «Правка» создайте пункт «Поиск», для которого создайте подменю с двумя пунктами «Найти» и «Найти далее». Для пункта «Найти» напишите функцию:

```
FindDialog1-> Execute();
```

Для пункта «Найти далее» напишите:

```
FindText();
```

В начале файла программы опишите прототип функции FindText:

```
void FindText(void);
```

Для компонента FindDialog задайте событие OnFind и напишите вызов следующей функции:

```
FindText();
```

В конце файла напишите текст функции FindText:

```
// Поиск введенной строки в тексте
```

```
void FindText(void)
{
 TPoint    CaretPos;
 int       x, y, i, j, l, beg, end;
 AnsiString FindText, Str;
char       *SStr, *SfindText;
 CaretPos = Form1->Memo1->CaretPos;
 x = CaretPos.x;
 y = CaretPos.y;
 FindText = Form1->FindDialog1->FindText;
 if (FindText.IsEmpty( )) return;
 SfindText = FindText.c_str( );
 SStr = Form1->Memo1->Lines->Text.c_str( );
 l = x;
 for (i = 0; i < y; i++)
l += Form1->Memo1->Lines->Strings[i].Length( ) + 2;
if (Form1->FindDialog1->Options.Contains(frDown))
{
      for (i = l+1; i < Form1->Memo1->Lines->Text.Length( ); i++)
      {
            if (strncmp(SStr+i, SfindText, FindText.Length( )) == 0)
            {
                  Form1->Memo1->SetFocus( );
                  Form1->Memo1->SelStart = i;
                  Form1->Memo1->SelLength = FindText.Length( );
                  return;
            } // if
      } // for
 } else {
      for (i = l-1- FindText.Length( ); i >= 0; i--) {
            if (strncpy(SStr+i, SfindText, FindText.Length()) == 0) {
                  Form1->Memo1->SetFocus();
                  Form1->Memo1->SelStart = i;
                  Form1->Memo1->SelLength = FindText.Length();
                  return;
            } // if
      } // for
 } // else
 Application->MessageBox("Не найдено", "Сообщение", MB_OK);
} // FindText
```

До полной реализации функций блокнота нам осталось несколько пунктов. Пункт «Перенос по словам» реализуется следующей функцией:

```
 Memo1->WordWrap = ! Memo1->WordWrap;
```

4.10 Компонент ReplaceDialog (Диалог замены)

Компонент ReplaceDialog предназначен для вызова диалоговой формы, в которой осуществляется выбор параметров замены текста. Диалоговая форма имеет вид, изображенный на рисунке 4.11.

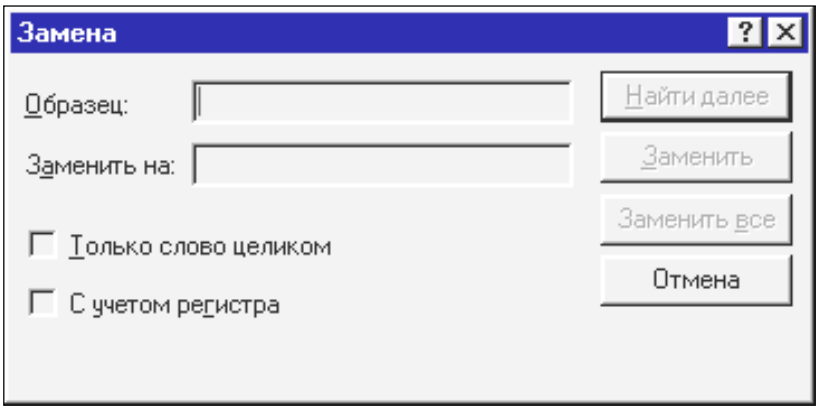


Рисунок 4.11 – Диалог замены

Основные свойства ReplaceDialog:

- FindText – текст для поиска (тип AnsiString);
- ReplaceText – текст для замены (тип AnsiString);
- Option – параметры поиска. Данное свойство состоит из флагов, основными из которых являются:
  - frDown – искать вниз;
  - frMatchCase – с учетом регистра;
  - frWholeWord – только слово целиком;
  - frReplaceAll – заменить все.

Как и для предыдущего компонента, основным событием является OnFind, которое возникает в случае нахождения текста, а основным методом является Execute, который запускает диалоговую панель.

4.11 Контрольные вопросы

1. Для чего используется компонент OpenDialog?
2. Что означает свойство FileName компонента OpenDialog?
3. Какой тип свойства FileName компонента OpenDialog?
4. Что означает свойство DefaultExt компонента OpenDialog?
5. Какой тип свойства DefaultExt компонента OpenDialog?
6. Что означает свойство InitialDir компонента OpenDialog?
7. Какой тип свойства InitialDir компонента OpenDialog?
8. Что означает свойство Title компонента OpenDialog?
9. Какой тип свойства Title компонента OpenDialog?
10. Что означает свойство Filter компонента OpenDialog?
11. Какой тип свойства Filter компонента OpenDialog?
12. Что означает свойство FilterIndex компонента OpenDialog?
13. Какой тип свойства FilterIndex компонента OpenDialog?
14. Для чего используется метод Execute компонента OpenDialog?
15. Для чего используется компонент SaveDialog?
16. Для чего используется компонент OpenPictureDialog?

17. Для чего используется компонент SavePictureDialog?
18. Для чего используется компонент FontDialog?
19. Что означает свойство Font компонента FontDialog?
20. Что означает свойство Device компонента FontDialog?
21. Для чего используется компонент ColorDialog?
22. Что означает свойство Color компонента ColorDialog?
23. Для чего используется компонент PrintDialog?
24. Что означает свойство PrintRange компонента PrintDialog?
25. Что означает свойство FromPage компонента PrintDialog?
26. Что означает свойство ToPage компонента PrintDialog?
27. Что означает свойство Collate компонента PrintDialog?
28. Что означает свойство Copies компонента PrintDialog?
29. Что реализует класс TPrinter?
30. Что означает свойство PageHeight класса TPrinter?
31. Что означает свойство PageWidth класса TPrinter?
32. Что означает свойство PageNumber класса TPrinter?
33. Что означает свойство Orientation класса TPrinter?
34. Что означает свойство Canvas класса TPrinter?
35. Для чего используется метод TextOut компонента свойства Canvas класса TPrinter?
36. Для чего используется метод BeginDoc класса TPrinter?
37. Для чего используется метод EndDoc класса TPrinter?
38. Для чего используется метод NewPage класса TPrinter?
39. Для чего используется компонент PrinerSetuptDialog?
40. Для чего используется компонент FindDialog?
41. Что означает свойство FindText компонента FindDialog?
42. Что означает свойство Option компонента FindDialog?
43. Для чего используется компонент ReplaceDialog?
44. Что означает свойство ReplaceText компонента ReplaceDialog?

5 КОМПОНЕНТЫ ЗАКЛАДКИ WIN32

На закладке Win32 расположены компоненты, которые были добавлены в Windows, начиная с 1995 года (Windows 95 и NT 3). Эти версии стали использовать 32-х разрядные регистры, поэтому их и последующие версии называются Win32, также называется и закладка. На закладке расположены разнородные компоненты. Они, как правило, более сложные, чем те, которые встречались на предыдущих закладках. Их использование приводит к современному внешнему виду программы.

Закладка изображена на рисунке 5.1.

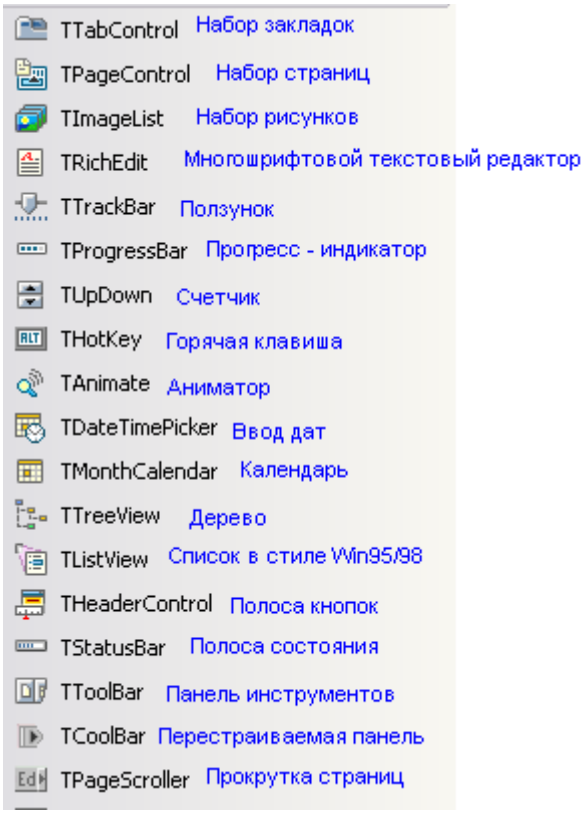


Рисунок 5.1 – Закладка Win32

5.1 Компонент TabControl (Набор закладок)

Компонент TabControl предназначен для реализации управляющего элемента, изображенного на рисунке 5.2.

Особенностью данного компонента является то, что элементы рабочей области являются общими для всех закладок. Если нужно, чтобы элементы различались на различных закладках, то следует использовать компонент PageControl, который будет рассмотрен в следующем пункте.

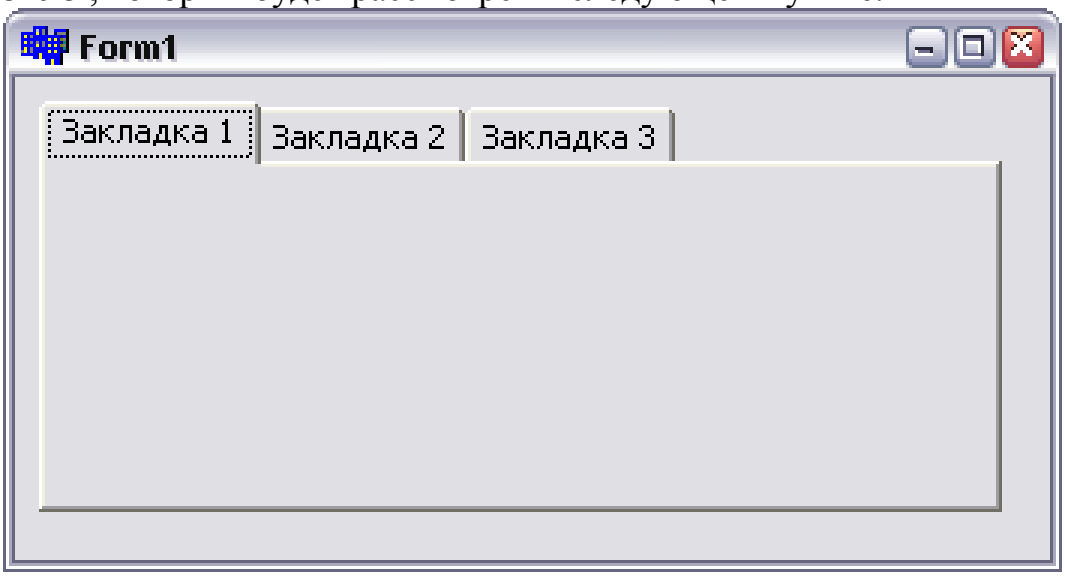
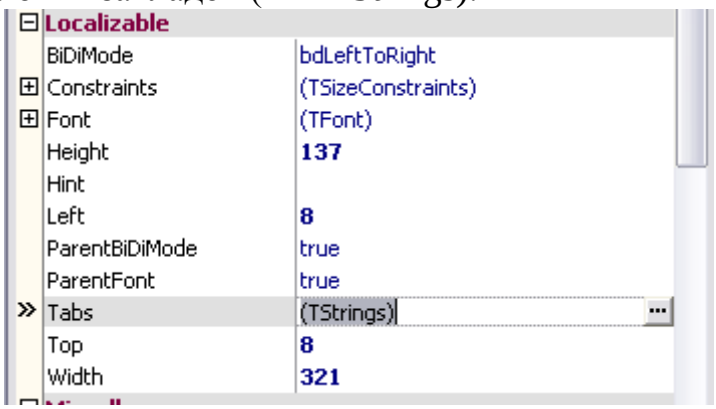


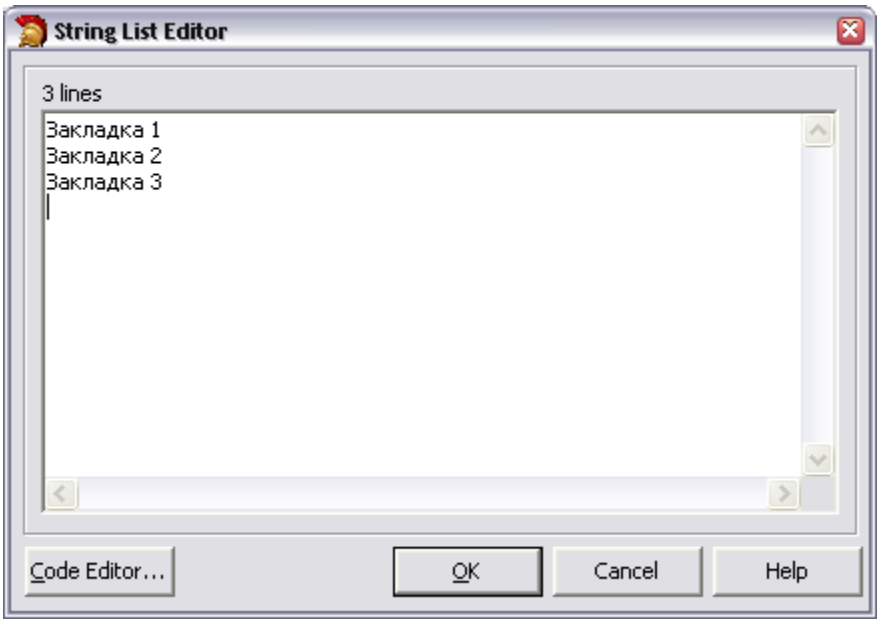
Рисунок 5.2 – Компонент TabControl

Основные свойства компонента:

• **Tabs** – заголовки закладок (тип TStrings).



Двойным щелчком по значению свойства открывается редактор для ввода значений:



Каждая строка задает заголовок одной закладки. Количество закладок определяется количеством строк.

• **Images** – рисунки перед заголовками закладок (тип TCustomImageList). Рисунки должны быть объединены в набор рисунков, а наборы рисунков будут рассмотрены в пункте 5.3;

• TabIndex – номер текущей закладки (тип int).

• Основные события:

• OnChanging – возникает перед покиданием закладки;

• OnChange – возникает при попадании на закладку.

В качестве примера рассмотрим записную книжку с закладками.

Для этого расположите на форме компоненты TabControl и ListBox согласно рисунку 5.3. В свойстве **Tabs** задайте 3 строки: А, Б, В.

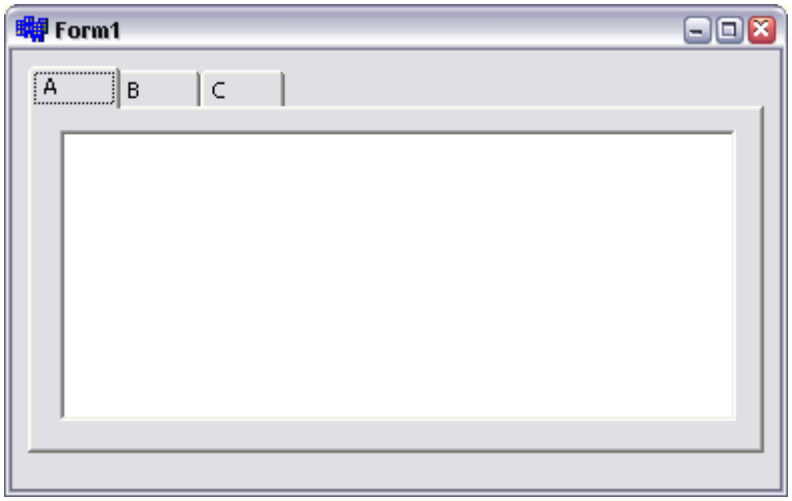


Рисунок 5.3 – Заготовка для записной книжки

В начале файла программы (после строчки TForm1 *Form1) запишите:

```
AnsiString A[ ] = {
 "Андреев Андрей Андреевич 11-11-11",
 "Алексеев Алексей Алексеевич 22-22-22",
 "Александров Александр Александрович 33-33-33"};
AnsiString B[ ] = {
 "Борисов Борис Борисович 44-44-44"};
AnsiString C[ ] = {
 "Викторов Виктор Викторович 55-55-55",
 "Владимиров Владимир Владимирович 66-66-66"};
```

В конструкторе формы:

```
ListBox1->Items->Add(A[0]);
ListBox1->Items->Add(A[1]);
ListBox1->Items->Add(A[2]);
```

В событии OnChange:

```
ListBox1->Items->Clear( );
if (TabControl1->TabIndex == 0) {
 ListBox1->Items->Add(A[0]);
 ListBox1->Items->Add(A[1]);
 ListBox1->Items->Add(A[2]);
} // if
if (TabControl1->TabIndex == 1) {
 ListBox1->Items->Add(B[0]);
} // if
if (TabControl1->TabIndex == 2) {
 ListBox1->Items->Add(C[0]);
 ListBox1->Items->Add(C[1]);
} // if
```

Запустите программу. Если Вы все сделали верно, то на экране появится окно записной книжки, в котором при изменении закладки будет видна информация о людей, чьи фамилии начинаются с соответствующей буквы.

5.2 Компонент PageControl (Набор страниц)

Компонент PageControl предназначен для реализации управляющего элемента внешне похожего на TabControl, однако на каждой странице можно размещать различные элементы. Например, из пункта меню «Файл / Параметры страницы» программы MS Word вызывается форма с четырьмя страницами. Первые две страницы изображены на рисунках 5.4 и 5.5.

И так далее.

Основные свойства компонента:

- PageCount – кол-во страниц (тип int);
- Pages – массив указателей на страницы (тип TTabSheet * Pages[]);
- ActivePageIndex – номер текущей страницы (тип int).

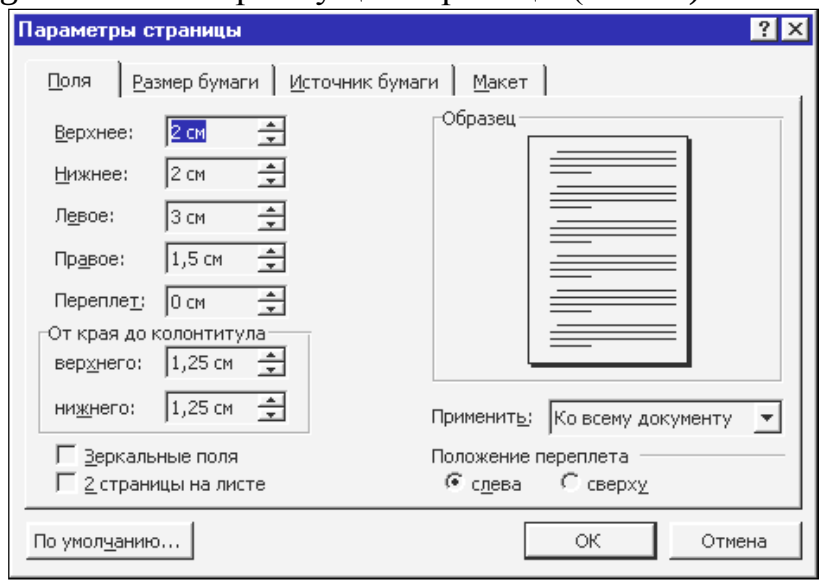


Рисунок 5.4 – Первая страница формы «Параметры страницы»

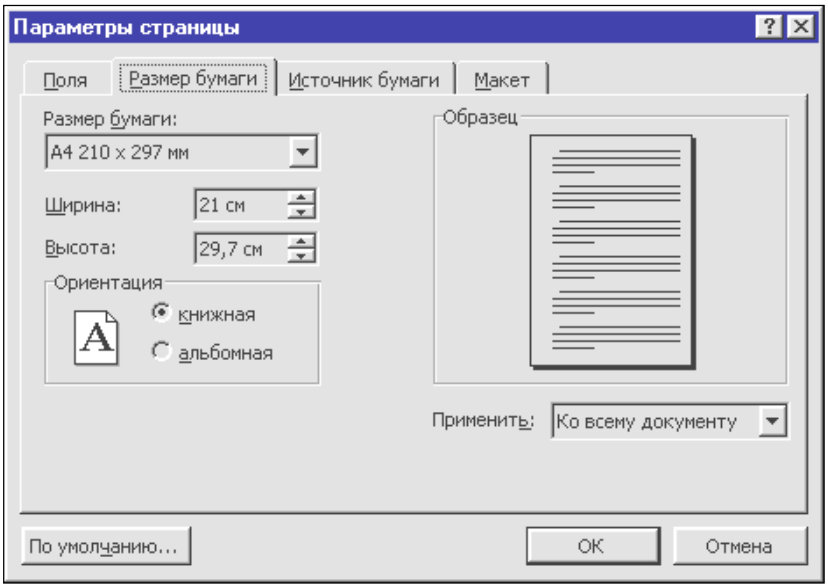


Рисунок 5.5 – Вторая страница формы «Параметры страницы»

Основные события те же, что и для компонента TabControl.

Если в Редакторе форм вызвать контекстное меню на элементе PageControl, то можно добавлять, удалять и менять местами страницы. Если существует хотя бы одна страница, то щелчок по корешку страницы приводит к тому, что она становится текущей. Щелчок в области корешков приводит к показу свойств компонента PageControl, а щелчок в рабочей области – к показу свойств страницы (TTabSheet).

Основные свойства TTabSheet:

- Name – имя страницы (тип AnsiString).
- Caption – заголовок страницы (тип AnsiString).
- PageIndex – номер страницы (тип int).

В качестве примера самостоятельно создайте три страницы, на которых разместите элементы, изображенные на рисунках 5.6, 5.7 и 5.8.

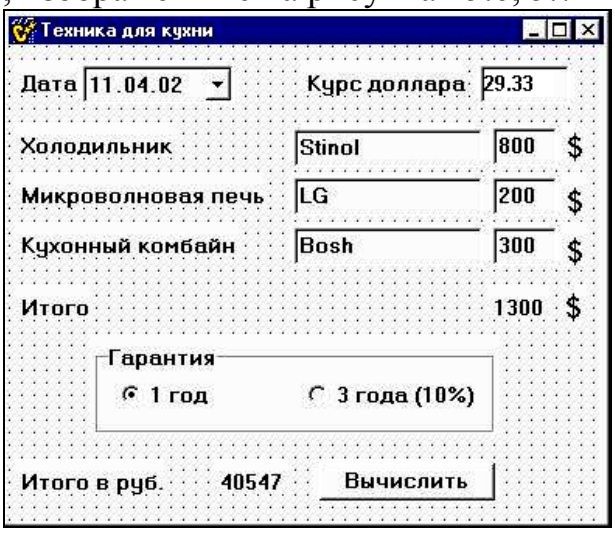


Рисунок 5.6 – Первая страница
для самостоятельного упражнения

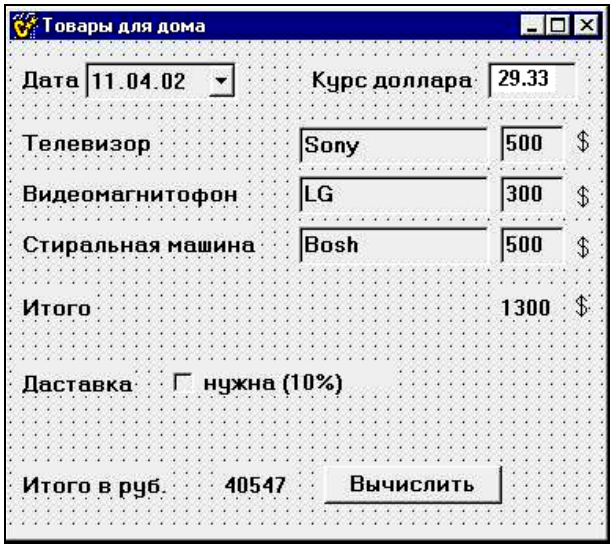


Рисунок 5.7 – Вторая страница
для самостоятельного упражнения

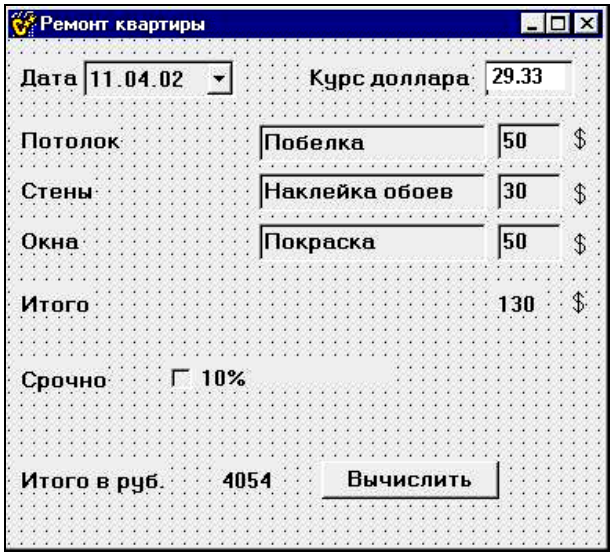


Рисунок 5.8 – Третья страница
для самостоятельного упражнения

5.3 Компонент ImageList (Набор рисунков)

Компонент ImageList предназначен для объединения рисунков, используемых в других компонентах. Компонент не отображается во время выполнения программы и может располагаться в любом месте формы.

Для заполнения набора рисунков следует дважды щелкнуть по нему. В результате откроется окошко, изображенное на рисунке 5.9.

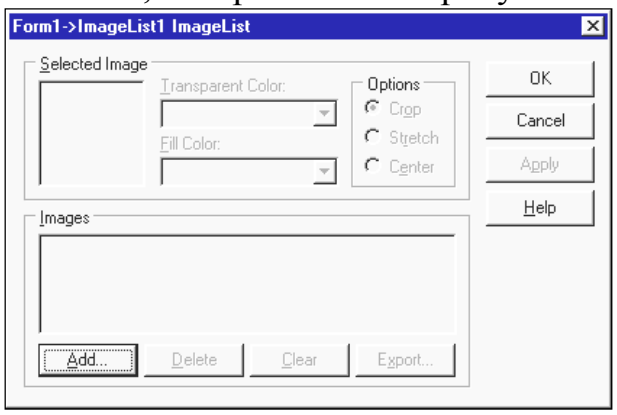


Рисунок 5.9 – Окно заполнения набора рисунков

При щелчке по кнопке Add откроется окно для выбора папки, содержащей файлы с рисунками. Таким образом, можно набрать необходимое количество рисунков.

В качестве примера усовершенствуем программу из пункта 5.1 (про записную книжку).

Откройте проект той программы. Расположите в любом месте компонент ImageList. Дважды щелкните по нему. В открывшемся окне нажмите кнопку Add. Выберите имя файла с рисунком. Рисунки имеются, например, в ката-

логе C: \ Programs Files \ Common Files \ Borland Shared \ Images \ Icons. Добавьте еще два рисунка. В свойстве Images компонента TabControl1 выберите значение ImageList1. Если все сделано правильно, то получится форма, изображенная на рисунке 5.10.

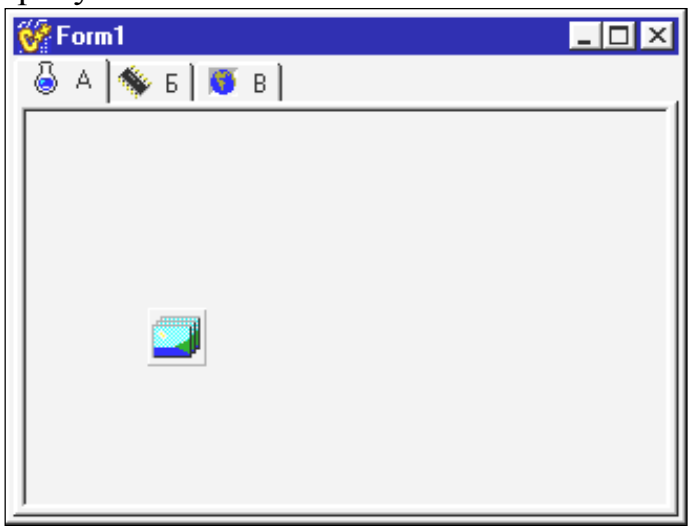


Рисунок 5.10 – Закладки с рисунками

5.4 Компонент RichEdit (Многошрифтовой текстовый редактор)

Компонент RichEdit предназначен для ввода и корректировки текста с использованием различных шрифтов и возможностью оформления параграфов. Этот компонент похож на компонент Memo (см. пункт 2.6), однако обладает большими возможностями, которые иллюстрируются на рисунке 5.11.

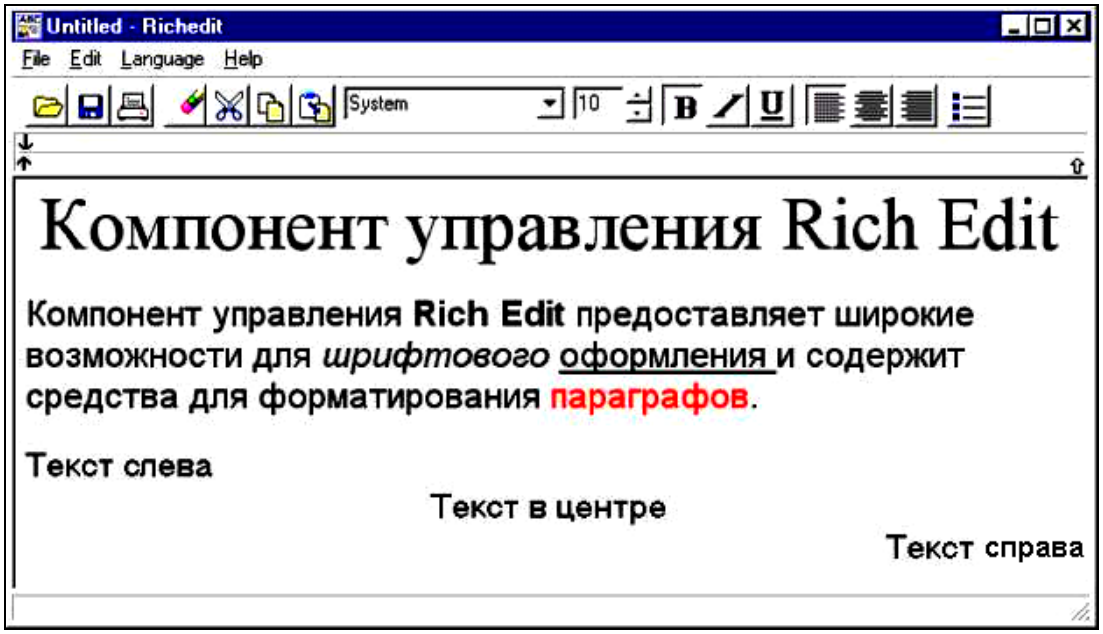


Рисунок 5.11 – Иллюстрация возможностей компонента RichEdit

Основными свойствами компонента RichEdit являются как те, что были у компонента Memo, так дополнительные:

- Lines – многострочный текст, содержащийся в поле (тип свойства TStrings), двойным щелчком по значению свойства в инспекторе объектов открывается редактор строк для начального ввода;
- ReadOnly – логическое поле: ложь – редактирование разрешено, истина – редактирование запрещено, только отображение;
- Align – выравнивание редактора относительно формы (тип TAlign);
- Alignment – стандартное выравнивание текста относительно редактора (тип TAlignment);
- SelAttributes – атрибуты выделенного текста (тип TTextAttributes);
- Paragraph – атрибуты параграфа (тип TParaAttributes).

Два последних свойства – программируемые и доступны только во время выполнения.

Тип TTextAttributes имеет основные свойства:

- Color – цвет выделенного участка (тип TColor);
- Name – имя шрифта (тип AnsiString);
- Size – размер (тип int);
- Style – стиль, составной тип, составленный побитно из констант:
  - fsBold – полужирный;
  - fsItalic – наклонный;
  - fsUnderline – подчёркнутый.

Тип TParaAttributes имеет основные свойства:

- Alignment – выравнивание параграфа, может принимать значения:
  - taLeftJustify – выровнен влево;
  - taCenter – выровнен по центру;
  - taRightJustify – выровнен вправо;
- FirstIndent – сдвиг первой строки параграфа относительно левого края в пикселях (тип int);
- LeftIndent – сдвиг всех остальных строк относительно левого края в пикселях (тип int);
- RightIndent – сдвиг строк относительно правого края в пикселях (тип int);
- Numbering – является или нет нумерованным списком? Может принимать два значения:
  - nsNone – .не является;
  - nsBullet – является;

Основные методы аналогичны методам компонента Memo с той лишь разницей, что работают не с текстовыми файлами, а с файлами формата rtf.

Пример работы с компонентом RichEdit имеется в стандартной поставке C++Builder. Он расположен в каталоге «C:\Program files\ Borland\ CBuilder5\Examples\Apps\RichEdit». Пример достаточно громоздкий, поэтому с целью экономии места здесь не приведен.

5.5 Компонент TrackBar (Ползунок)

Компонент TrackBar аналогичен компоненту ScrollBar (полоса прокрутки) из пункта 2.11, однако имеет несколько иной дизайн, показанный на рис. 5.12.

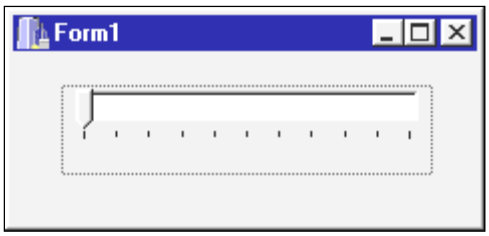


Рисунок 5.12 – Ползунок

Основные свойства компонента:

- **Orientation** – ориентация ползунка:
  - tbHorizontal – горизонтальная;
  - tbVertical – вертикальная;
- **Min** – минимальное значение позиции ползунка (тип int).
- **Max** – максимальное значение позиции ползунка (тип int).
- **Position** – текущее значение позиции ползунка (тип int).

Основным событием является OnChange. Это событие возникает при изменении положения ползунка. Функцию обработчика события OnChange можно задать двойным щелчком мыши по ползунку.

Самостоятельно измените программу из пункта 2.11 так, чтобы вместо полосы прокрутки использовался ползунок.

5.6 Компонент ProgressBar (Прогресс индикатор)

Компонент ProgressBar предназначен для отображения хода выполнения «длинной» операции. Прогресс индикатор имеет дизайн, показанный на рисунке 5.13.

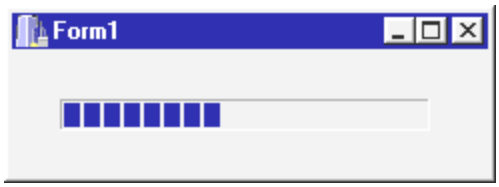


Рисунок 5.13 – Прогресс индикатор

Основные свойства компонента:

- **Orientation** – ориентация прогресс индикатора:
  - pbHorizontal – горизонтальная;
  - pbVertical – вертикальная;
- **Min** – минимальное значение (тип int);
- **Max** – максимальное значение (тип int);
- **Step** – шаг изменения состояния (тип int);

• **Position** – текущее значение состояния (тип int).

В качества примера рассмотрим прогресс индикатор, который изменяет свое состояние от 1 до 10, с интервалом 1 сек.

Для этого расположите на форме прогресс индикатор и таймер (Timer) с закладки System. К сожалению, таймер будем рассматривать подробно только в следующей главе. Сейчас рассмотрим кратко. Основное свойство таймера Interval – интервал времени, через который наступает событие OnTimer. Зададим свойства компонентов, указанные в таблице 5.1.

Таблица 5.1 – Свойства компонентов: прогресс индикатор и таймер

Компонент	Свойство	Значение
Прогресс индикатор	Min	0
Прогресс индикатор	Max	10
Прогресс индикатор	Step	1
Прогресс индикатор	Position	0
Таймер	Interval	1000

Дважды щелкните по таймеру, откроется редактор кода для введения функции обработки события OnTimer. Запишите:

```
if (ProgressBar1->Position < 10) ProgressBar1->Position++;
```

Запустите программу.

5.7 Компонент UpDown (Счетчик)

Компонент UpDown предназначен для уменьшения или увеличения значения в поле ввода, связанном с данным счетчиком. Дизайн поля ввода со счетчиком показан на рис. 5.14.

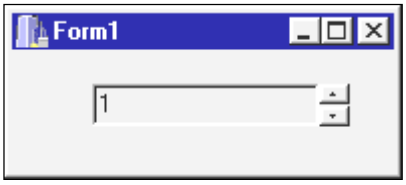


Рисунок 5.14 – Поле ввода со счетчиком

Основные свойства компонента:

• **Associate** – связанное со счетчиком поле ввода;
• **Min** – минимальное значение счетчика (тип int);
• **Max** – максимальное значение счетчика (тип int);
• **Increment** – шаг изменения значение счетчика (тип int);
• **Position** – текущее значение счетчика (тип int).

Основное событие счетчика OnChange возникает при изменении значения поля ввода с помощью счетчика.

В качестве примера расположите на форме компоненты согласно рисунка 5.14. Задайте свойства счетчика, указанные в таблице 5.2.

Таблица 5.2 – Свойства компонента счетчик

Свойство	**Значение**
Associate	Edit1
Max	10
Min	1
Position	1

Запустите программу. Ваш счетчик будет считать от 1 до 10.

5.8 Компонент HotKey (Горячая клавиша)

Компонент HotKey отображает комбинацию нажатых клавиш, включая shift, ctrl, alt и т.д.

5.9 Компонент Animate (Аниматор)

Компонент Animate предназначен для воспроизведения небольших видео клипов без звука.

Основные свойства компонента:

- CommonAVI – один из встроенных видео клипов. Перечень допустимых значений приведен в таблице 5.3;
- FileName – имя файла, предназначенного для воспроизведения, расширение файла avi (тип свойства AnsiString); двойной щелчок по значению свойства приводит к открытию диалога для выбора файла;
- StartFrame – номер начального кадра клипа (тип short);
- StopFrame – номер конечного кадра клипа (тип short);
- Active – Если это свойство – истина, то видео клип воспроизводится, даже если задача не запущена.

Основным методом аниматора является Play(Word FromFrame, Word ToFrame, int Count) – запустить воспроизведение видео клипа, где:

- FromFrame – начальный кадр воспроизведения,
- ToFrame- конечный кадр воспроизведения,
- Count – количество воспроизведений (если Count = 0, то крутится бесконечное количество раз).

В качестве примера расположите на форме 8 аниматоров, согласно рисунка 5.15.

Таблица 5.3 – Допустимые значения свойства CommonAVI

Значение свойства CommonAVI	**Кадр из видео клипа**
aviFindFolder	
aviFindFile	
aviFindComputer	
aviCopyFile	
aviCopyFiles	
aviRecycleFile	
aviEmptyRecycle	
aviDeleteFile	
aviNone	Определяется свойством FileName

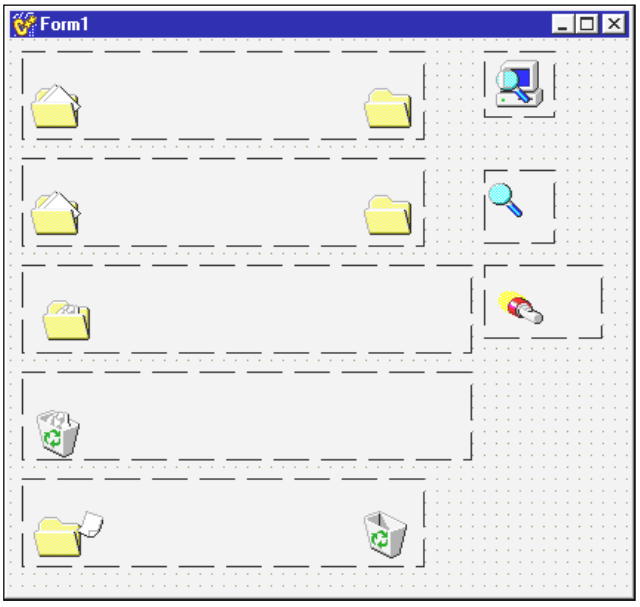


Рисунок 5.15 – Пример расположения аниматоров

Задайте свойство CommomAVI, согласно таблице 5.3. Всем аниматорам задайте свойство Active = true. Собственно программу можно не запускать,

все уже начнет двигаться и летать. Однако запустим и посмотрим на бесконечную анимацию.

Для того, чтобы анимация происходила 1 раз выполним следующее:

- Зададим всем аниматорам свойство Active = false;
- Разместим на свободном месте формы кнопку;
- Дважды щелкнув по кнопке введем следующий код программы:

```
Animate1->Play(Animate1-> StartFrame, Animate1-> StopFrame, 1);
Animate2->Play(Animate2-> StartFrame, Animate2-> StopFrame, 1);
Animate3->Play(Animate3-> StartFrame, Animate3-> StopFrame, 1);
Animate4->Play(Animate4-> StartFrame, Animate4-> StopFrame, 1);
Animate5->Play(Animate5-> StartFrame, Animate5-> StopFrame, 1);
Animate6->Play(Animate6-> StartFrame, Animate6-> StopFrame, 1);
Animate7->Play(Animate7-> StartFrame, Animate7-> StopFrame, 1);
Animate8->Play(Animate8-> StartFrame, Animate8-> StopFrame, 1);
```

5.10 Компонент DateTimePicker (Ввод даты)

Компонент DateTimePicker предназначен для ввода даты и времени. Дата вводится в виде поля с выпадающим календарем на месяц, время в поле со счетчиком. Дату и время можно естественно вводить и вручную, при этом осуществляется проверка допустимости введенных значений. Внешний вид компонента показан на рисунке 5.16.

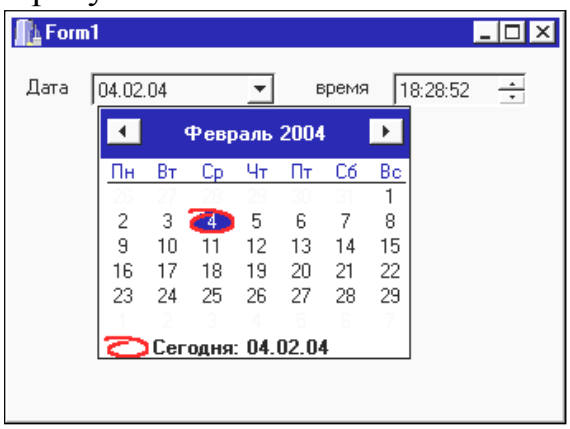


Рисунок 5.16 – Внешний вид компонента DateTimePicker

Основные свойства компонента:

- Kind – вид компонента, может принимать два значения:
  - dtkDate – ввод даты;
  - dtkTime – ввод времени;
- Date – дата (тип свойства TDateTime);
- Time – время (тип TDateTime).

Основное событие компонента DateTimePicker – OnChange возникает при изменении значения даты или времени.

Рассмотрим тип данных TDateTime. Этот тип создан для хранения даты и времени. Он представляет собой число с плавающей точкой (тип double) в

котором записано число суток, прошедших с 12:00 30 декабря 1899 года. Естественно число может быть дробным и отрицательным. Число является дробным в любое время, отличное от 12:00 (считается часть суток, 1 час = 1/24), отрицательным – до 12:00 30 декабря 1899.

Основные конструкторы TDateTime:

- TDateTime( ) – создает нулевую дату и время (12:00 30 декабря 1899);
- TDateTime(const TDateTime& src) – конструктор копирования создает дату и время, равную scr;
- TDateTime(unsigned short year, unsigned short month, unsigned short day) – создает дату, где: day – день, month – месяц, year – год;
- TDateTime(unsigned short hour, unsigned short min, unsigned short sec, unsigned short msec) – создает время, где hour – час, min – минута, sec – сек, msec – миллисекунда;

Основные методы типа TDateTime:

- AnsiString DateString( ) – преобразует TDateTime в AnsiString в виде строки даты, например “04/02/04”;
- AnsiString TimeString( ) – преобразует TDateTime в AnsiString в виде строки времени, например “18:58:52”;
- AnsiString DateTimeString( ) – преобразует TDateTime в AnsiString в виде строки даты и времени, например “04/02/04 18:58:52”.

Примечание: формат даты и времени зависят от региональных установок.

Перегруженные операции

- TDateTime + int – прибавление к дате и времени целого кол-ва суток;
- TDateTime + double – прибавление к дате и времени дробного кол-ва суток;
- TDateTime – int – вычитание из даты и времени целого кол-ва суток;
- TDateTime – double – вычитание из даты и времени дробного кол-ва суток;
- TDateTime – TDateTime – определение промежутка времени между двумя событиями;
- TDateTime = = TDateTime – сравнение двух дат, времен. Аналогично !=, >, <, >=, <=;

В качестве примера расположите на форме два компонента DateTimePicker, согласно рисунку 5.15. У второго компоненты установите свойство Kind = dtkTime. Ниже расположите кнопку, для которой задайте следующее действие:

```
Application->MessageBox((DateTimePicker1->Date.DateString( )+" "+
DateTimePicker2->Time.TimeString( )).c_str(), "", MB_OK);
```

Запустите программу, задайте произвольные дату и время и нажмите кнопку.

5.11 Компонент MonthCalendar (Календарь)

Компонент MonthCalendar предназначен для выбора даты из календаря, который имеет вид, изображенный на рис. 5.17.

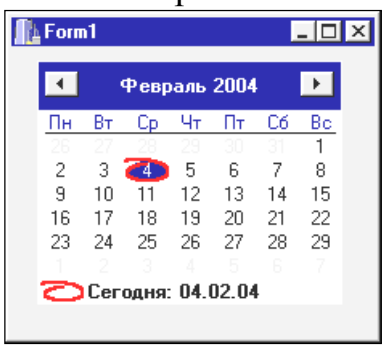


Рисунок 5.17 – Календарь

Основным свойством компонента является Date – выбранная дата (тип свойства TDateTime). Основное событие OnClick возникает при выборе даты.

В качестве примера расположите на форме календарь и кнопку. Для кнопки задайте следующее действие:

```
Application->MessageBox((MonthCalendar1->Date.DateString()). c_str(),
"", MB_OK);
```

Запустите программу, выберите интересующую дату и нажмите кнопку.

5.12 Компонент TreeView (Дерево)

Компонент TreeView предназначен для представления данных в виде иерархической структуры (рис. 5.18).

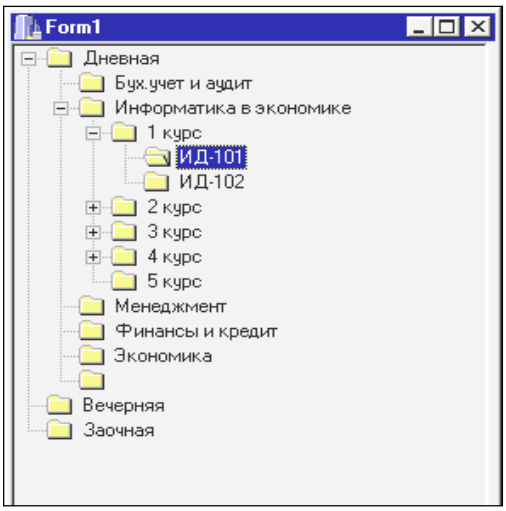


Рисунок 5.18 – Дерево

Двойной щелчок по дереву приводит к открытию редактора для первоначального ввода элементов дерева. Дизайн редактора показан на рис. 5.19.

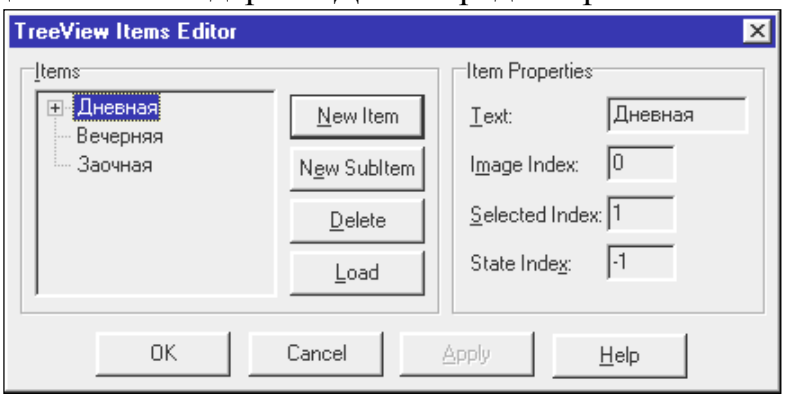


Рисунок 5.19 – Дизайн Редактора деревьев

С помощью этого редактора можно добавлять, изменять и удалять узлы и элементы дерева.

Основные свойства компонента:

- Items – указатель на список элементов дерева (тип TTreeNodes);
- Selected – указатель на выделенный элемент (тип TTreeNode);
- Images – указатель на набор рисунков, расположенных перед текстом. Наборы рисунков описаны в пункте 5.3;
- ReadOnly – только для чтения (логический тип); если ReadOnly = false, то, щелкнув мышью по элементу можно редактировать текст, если ReadOnly = true – то нельзя;
- Align – выравнивание дерева относительно формы.

Основные методы компонента:

- FullCollapse( ) – закрыть все узлы;
- FullExpand( ) – открыть все узлы;
- SaveToFile(имя_файла) – сохранить дерево в файле;
- LoadFromFile(имя_файла) – загрузить дерево из файла;
- Основные события компонента:
- OnChanging – возникает перед изменении текущего элемента;
- OnChange – возникает после изменения текущего элемента;
- OnCollapsing – возникает перед закрытием узла;
- OnCollapsed – возникает после закрытия узла;
- OnExpanding – возникает перед открытием узла;
- OnExpanded – возникает после открытия узла;
- OnEditing – возникает перед началом редактирования элемента;
- OnEdited – возникает после окончания редактирования элемента;
- OnDeletion – возникает при удалении элемента.

При описании свойств дерева нам встретились типы данных TTreeNodes и TTreeNode. Рассмотрим их более подробно.

Тип TTreeNodes – класс данных, реализующий список элементов дерева.

Основные свойства TTreeNodes:

- Count – общее количество элементов дерева (тип int);
- Item – указатель на массив элементов дерева (тип TTreeNode *).
- Основные методы TTreeNodes:
- Clear( ) – удалить все элементы дерева;
- AddFirst(TTreeNode *Node, AnsiString S) – добавить элемент в начало уровня, где находится элемент Node;
- Add(TTreeNode* Node, AnsiString S) – добавить элемент после Node в тот же уровень;
- AddChild(TTreeNode* Node, AnsiString S) – добавить элемент в поддерево Node;
- Delete(TTreeNode* Node) – удалить узел.

Примечание. Методы AddFirst, Add и AddChild возвращают указатель на добавленный элемент.

Тип TTreeNode – класс данных, реализующий элемент дерева.

Основные свойства TTreeNodes:

- Text – текст элемента дерева (тип AnsiString);
- Parent – указатель на родительский узел (тип TTreeNode);
- Level – уровень (тип int);
- Selected – является ли текущим (логический тип);
- ImageIndex – номер рисунка в наборе, который будет изображен в случае неактивного элемента;
- SelectedIndex – номер рисунка в наборе, который будет изображен в случае активного элемента.

Основные методы TTreeNode:

- Expand (bool Recurse) – открыть узел; если Recurse = true, то открываются все подъузлы, иначе подъузлы не открываются;
- Collapse (bool Recurse) – закрыть узел; если Recurse = true, то закрываются все подъузлы, иначе подъузлы не закрываются;
- Delete( ) – удалить узел;
- DeleteChildren( ) – удалить все подчиненные элементы;
- AlphaSort( ) – отсортировать подчиненные элементы по алфавиту;
- CustomSort(PFNTVCOMPARE SortProc, int Data) – отсортировать подчиненные элементы, порядок сортировки определяется функцией SortProc.

В качестве примера рассмотрим построение дерева, изображенного на рисунке 5.18. Дерево можно заполнить с помощью построителя или программно. С помощью построителя заполнить дерево очень просто, поэтому мы заполним его программно.

Расположите на форме дерево и список рисунков. Для дерева задайте свойства, указанные в таблице 5.4.

Таблица 5.4 – Свойства дерева

Свойство	Значение
Align	alClient
Images	ImageList1

Дважды щелкните по списку рисунков и добавьте два рисунка в виде закрытой и открытой папки (рис 5.20).

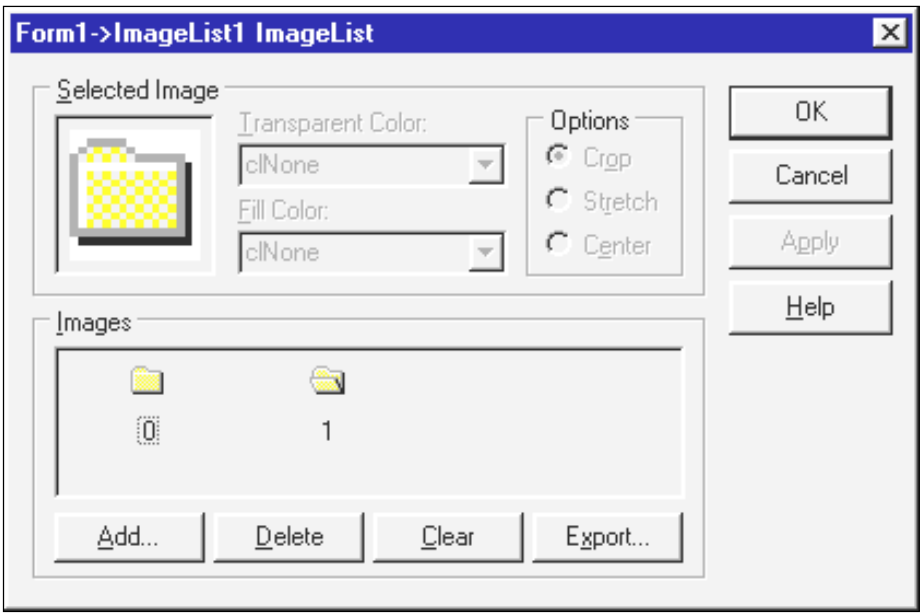


Рисунок 5.20 – Список рисунков для дерева

Для событие OnActivate формы напишите функцию:

```
TTreeNode *Tn1, *Tn2;
Tn1 = TreeView1->Items->Add(NULL, "Дневная");
Tn1->SelectedIndex = 1;
 Tn2 = TreeView1->Items->AddChild(Tn1, "Бух.учет и аудит");
 Tn2->SelectedIndex = 1;
 Tn2 = TreeView1->Items->AddChild(Tn1, "Информатика в экономике");
 Tn2->SelectedIndex = 1;
 Tn2 = TreeView1->Items->AddChild(Tn1, "Менеджмент");
 Tn2->SelectedIndex = 1;
 Tn2 = TreeView1->Items->AddChild(Tn1, "Финансы и кредит");
 Tn2->SelectedIndex = 1;
 Tn2 = TreeView1->Items->AddChild(Tn1, "Экономика");
 Tn2->SelectedIndex = 1;
Tn1 = TreeView1->Items->Add(NULL, "Вечерняя");
Tn1->SelectedIndex = 1;
Tn1 = TreeView1->Items->Add(NULL, "Заочная");
Tn1->SelectedIndex = 1;
```

Приведенный выше фрагмент программы реализует только 2 уровня вложенности дерева. Самостоятельно реализуйте дерево с рис. 5.18 в полном объеме.

5.13 Компонент ListView (Список в стиле Win 95/98)

Компонент ListView предназначен для реализации списков в одном из четырех видов:

- Горизонтально расположенный список с крупными значками (рис 5.21).
- Горизонтально расположенный список с мелкими значками (рис 5.22).
- Вертикально расположенный список с мелкими значками (рис 5.23).
- Таблица с мелкими значками (рис 5.24).

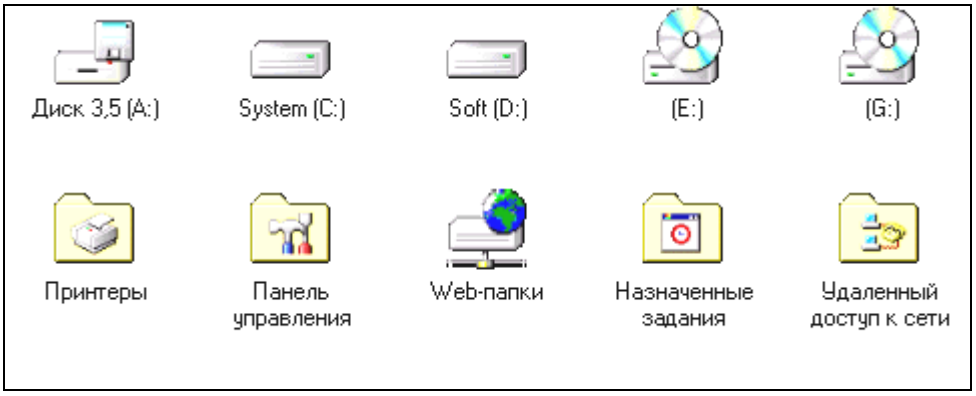


Рисунок 5.21 – Горизонтально расположенный список с крупными значками

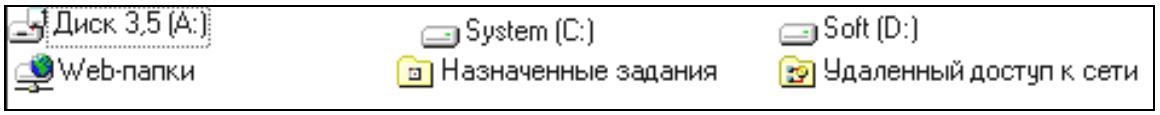


Рисунок 5.22 – Горизонтально расположенный список с мелкими значками

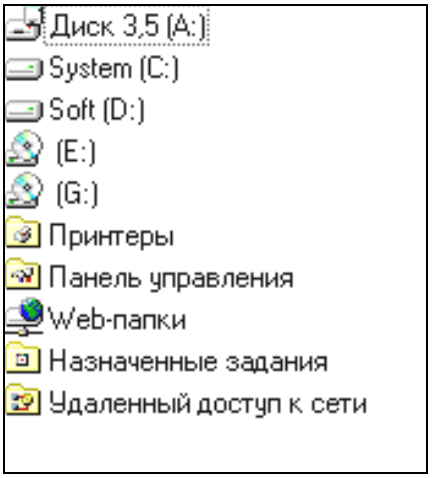


Рисунок 5.23 – Вертикально расположенный список с мелкими значками

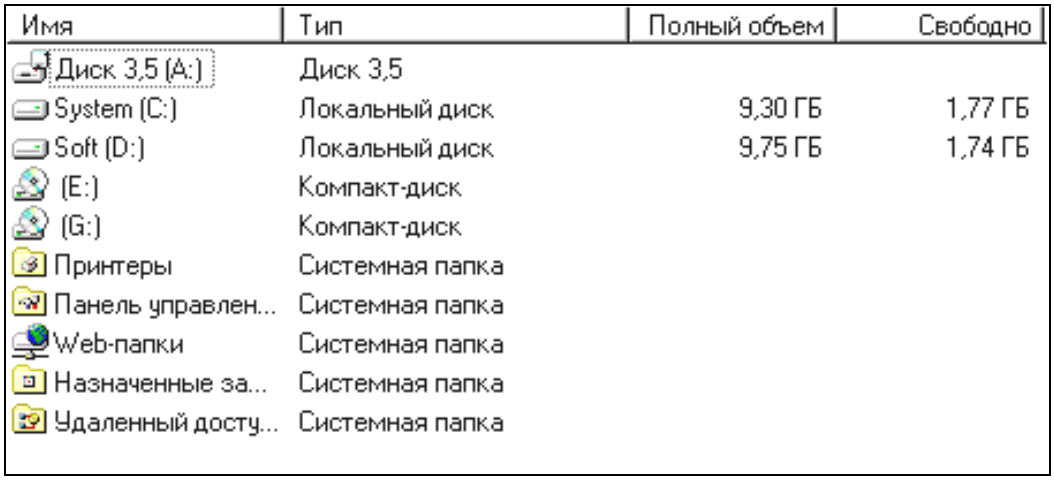


Рисунок 5.24 – Таблица с мелкими значками

Основные свойства компонента:

- ViewStyle – вид списка, может принимать следующие значения:
  - - vsIcon – горизонтально расположенный список с крупными значками;
  - - vsSmallIcon-горизонтально расположенный список с мелкими значками;
  - - vsList – вертикально расположенный список с мелкими значками;
  - - vsReport – таблица с мелкими значками;
- Items – содержимое списка (тип TListItems *); двойной щелчок по значению этого свойства открывает редактор для первоначального ввода элементов списка (рис 5.25);
- Selected – указатель на выделенный элемент списка (тип TListItem *);
- Columns – описание столбцов таблицы, действует только в том случае, если список имеет вид таблицы с мелкими значками (тип TListColumns); двойной щелчок по значению этого свойства открывает редактор для определения свойств столбцов (рис 5.26);
- LargeImages – набор крупных значков;
- SmallImages – набор мелких значков;
- SortType – способ сортировки списка, может принимать значения:
  - - stNone – нет сортировки;
  - - stText – сортировка по тексту элементов;
  - - stData – сортировка по данным;
- ReadOnly – только для чтения (логический тип); если ReadOnly = false – то, щелкнув мышью по элементу, можно редактировать текст, если ReadOnly = true, то – нельзя.
- Align – выравнивание списка относительно формы.

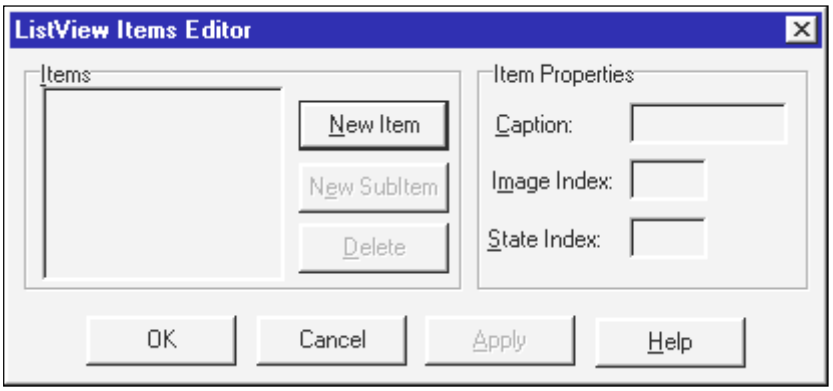


Рисунок 5.25 – Редактор для первоначального ввода элементов списка

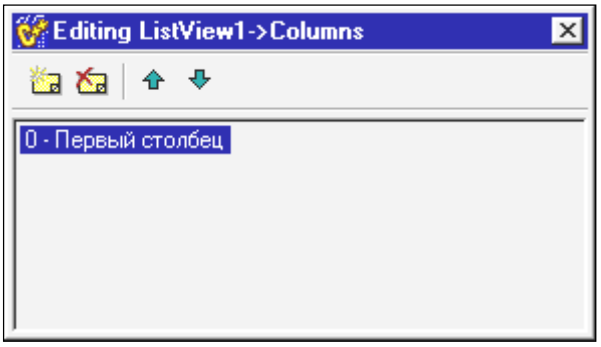


Рисунок 5.26 – Редактор для определения свойств столбцов

Основные методы компонента:

- AlphaSort( ) – сортировка списка по алфавиту;
- CustomSort(PFNLVCOMPARE SortProc, int lParam) – сортировка, порядок которой определяется функцией SortProc.

Основные события компонента:

- OnChanging – возникает перед изменением текущего элемента;
- OnChange – возникает после изменения текущего элемента;
- OnEditing – возникает перед началом редактирования элемента;
- OnEdited – возникает после окончания редактирования элемента;
- OnDeletion – возникает при удалении элемента;
- OnInsert – возникает при вставке элемента.

При описании свойств списка нам встретились типы данных TListItems, TListItem и TListColumns. Рассмотрим их более подробно.

Тип TListItems – это класс данных, реализующий содержимое списка.
Основные свойства TListItems:

- Count – количество элементов списка (тип int);
- Items – указатель на массив элементов списка (тип TListItem *).

Основные методы TListItems:

- Clear( ) – очистить список;

- Add( ) – добавить элемент в конец списка;
- Insert(int Index) – добавить элемент в заданное место списка;
- Delete(int Index) – удаление элемента списка.

Тип TListItem – это класс данных, реализующий элемент списка.

Основные свойства TListItem:

- Caption – текст (тип AnsiString); в табличном виде – текст первого столбца;
- ImageIndex – номер в наборе значков (тип int);
- Selected – является ли выбранным (логический тип);
- Subitems – В табличном виде – текст последующих столбцов (тип TStrings);
- Data – произвольные данные, связанные с элементом.

Тип TListColumns – это класс данных, реализующий набор столбцов таблицы (для табличного вида списка).

Основные свойства TListColumns:

- Count – количество столбцов таблицы (тип int);
- Items – указатель на массив столбцов таблицы (тип TListColumn *).

Основные методы TListColumns:

- Clear( ) – очистить набор столбцов;
- Insert(int Index) – добавить столбец;
- Delete(int Index) – удалить столбец.

Тип TListColumn – это класс данных, реализующий столбец таблицы.

Основные свойства TListColumn:

- Caption – заголовок столбца таблицы (тип AnsiString);
- Width – ширина столбца (тип int);
- AutoSize – автоподбор ширины (логический тип);
- Alignment – выравнивание.

Пример использования списка находится в папке: C: \ Program Files \ Borland \ CBuilder5 \ Examples \ VirtualListView. Откройте проект VirtualListView.bpr и запустите программу.

5.14 Компонент StatusBar (Полоса состояния)

Компонент StatusBar предназначен для реализации полосы, в нижней части окна, в которой можно выводить различную информацию о функционировании программы. Расположение StatusBar показано на рис 5.27.

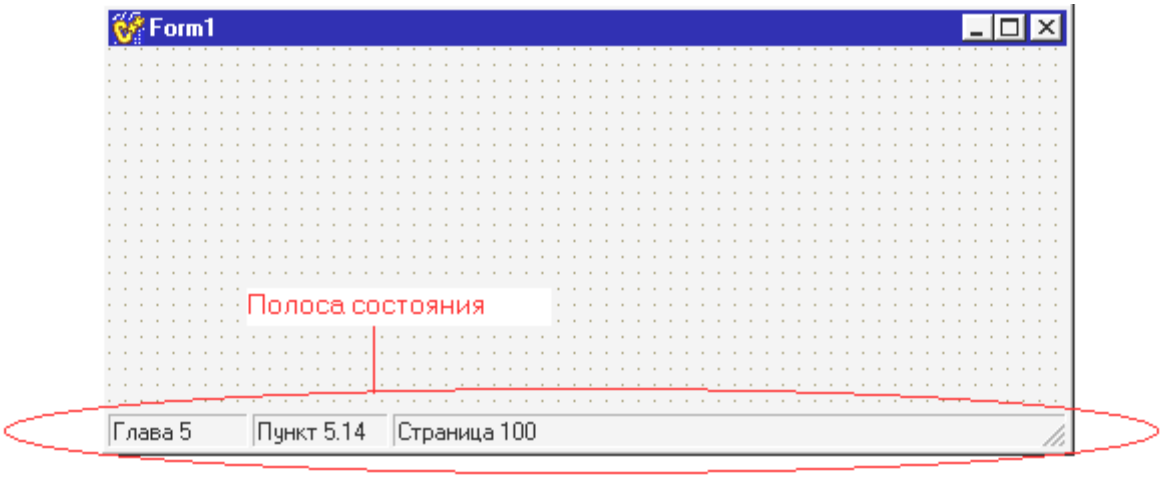


Рисунок 5.27 – Расположение полосы состояния

Основные свойства компонента:

• SimplePanel – делится ли полоса состояния на секции; SimplePanel = false – делится, SimplePanel = true – не делится (логический тип);

• SimpleText – текст внутри полосы состояния, при SimplePanel = true (тип AnsiString);

• Panels – секции при SimplePanel = false (тип TStatusPanels); двойной щелчок на значении этого свойства приводит к открытию редактора для ввода свойств секций (рис. 5.28);

• SizeGrip – наличие или отсутствие в правом нижнем углу рельефного объекта для удобства изменения размеров окна (логический тип).

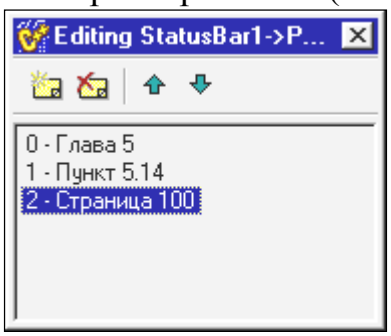


Рисунок 5.28 – Редактор для ввода свойств секций

Тип TStatusPanels – это класс данных для реализации набора секций. Основные свойства класса TStatusPanels:

• Count – количество секций (тип int);

• Items – указатель на массив описания секций (тип TStatusPanel *).

Тип TStatusPanel – это класс данных для реализации описания секции. Основные свойства класса TStatusPanel:

• Text – текст внутри секции (тип AnsiString);

• Width – размер секции (тип int);

• Alignment – выравнивание текста в секции (тип TAlignment).

5.15 Компонент ToolBar (Панель инструментов)

Компонент ToolBar предназначен для реализации полосы, в верхней части окна, в которой находятся кнопки для быстрого вызова важных функций. Расположение ToolBar показано на рис 5.29.

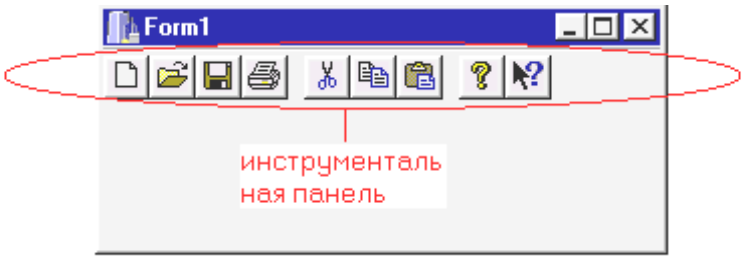


Рисунок 5.29 – Расположение инструментальной панели

Основные свойства компонента:

- Images – список рисунков в кнопках (тип TImageList см. пункт 5.3);
- ButtonCount – количество кнопок (тип int);
- Buttons – указатель на массив кнопок (тип TToolButton).

Тип TToolButton – это класс данных для реализации описания кнопок. Основным свойством класса TToolButton является TImageIndex – номер рисунка в списке. Основным событием класса TToolButton является OnClick – действие, выполняемое при нажатии кнопки. Двойной щелчок по кнопке приводит к открытию редактора кода программы для ввода функции обработки события OnClick.

Примечание. Для добавления кнопки в ToolBar следует щелкнуть по нему правой кнопкой мыши и в контекстном меню выбрать пункт New Button (новая кнопка).

5.16 Контрольные вопросы

1. Для чего используется компонент TabControl?
2. Что означает свойство Tabs компонента TabControl?
3. Какой тип свойства Tabs компонента TabControl?
4. Что означает свойство TabIndex компонента TabControl?
5. Какой тип свойства TabIndex компонента TabControl?
6. В каком случае возникает событие OnChanging компонента TabControl?
7. В каком случае возникает событие OnChange компонента TabControl?
8. Для чего используется компонент PageControl?
9. Что означает свойство Pages компонента PageControl?
10. Какой тип свойства Pages компонента PageControl?
11. Что означает свойство ActivePageIndex компонента PageControl?
12. Какой тип свойства ActivePageIndex компонента PageControl?

13. Для чего используется компонент ImageList?
14. Для чего используется компонент RichEdit?
15. Что означает свойство Lines компонента RichEdit?
16. Какой тип свойства Lines компонента RichEdit?
17. Что означает свойство ReadOnly компонента RichEdit?
18. Какой тип свойства ReadOnly компонента RichEdit?
19. Что означает свойство Align компонента RichEdit?
20. Что означает свойство Alignment компонента RichEdit?
21. Что означает свойство SelAttributes компонента RichEdit?
22. Какой тип свойства SelAttributes компонента RichEdit?
23. Что означает свойство Paragraph компонента RichEdit?
24. Какой тип свойства Paragraph компонента RichEdit?
25. Для чего используется компонент TrackBar?
26. Что означает свойство Orientation компонента TrackBar?
27. Что означает свойство Min компонента TrackBar?
28. Что означает свойство Max компонента TrackBar?
29. Что означает свойство Position компонента TrackBar?
30. Для чего используется компонент ProgressBar?
31. Что означает свойство Step компонента ProgressBar?
32. Для чего используется компонент UpDown?
33. Что означает свойство Associate компонента UpDown?
34. Что означает свойство Increment компонента UpDown?
35. Для чего используется компонент Animate?
36. Что означает свойство CommonAVI компонента Animate?
37. Что означает свойство FileName компонента Animate?
38. Какой тип свойства FileName компонента Animate?
39. Что означает свойство StartFrame компонента Animate?
40. Что означает свойство StopFrame компонента Animate?
41. Что означает свойство Active компонента Animate?
42. Для чего используется метод Play класса TAnimate?
43. Сколько параметров у метода Play класса TAnimate?
44. Что означает первый параметр метода Play класса TAnimate?
45. Что означает второй параметр метода Play класса TAnimate?
46. Что означает третий параметр метода Play класса TAnimate?
47. Для чего используется компонент DateTimePicker?
48. Что означает свойство Kind компонента DateTimePicker?
49. Что означает свойство Date компонента DateTimePicker?
50. Какой тип свойства Date компонента DateTimePicker?
51. Что означает свойство Time компонента DateTimePicker?
52. Какой тип свойства Time компонента DateTimePicker?
53. Для чего используется компонент MonthCalendar?
54. Для чего используется компонент TreeView?
55. Что означает свойство Items компонента TreeView?

56. Какой тип свойства Items компонента TreeView?
57. Что означает свойство Selected компонента TreeView?
58. Какой тип свойства Selected компонента TreeView?
59. Что означает свойство Images компонента TreeView?
60. Что означает свойство ReadOnly компонента TreeView?
61. Что означает свойство Align компонента TreeView?
62. Для чего используется метод FullCollapse компонента TreeView?
63. Для чего используется метод FullExpand компонента TreeView?
64. Для чего используется метод SaveToFile компонента TreeView?
65. В каком случае возникает событие OnChanging компонента TreeView?
66. В каком случае возникает событие OnCollapsing компонента TreeView?
67. В каком случае возникает событие OnCollapsed компонента TreeView?
68. В каком случае возникает событие OnExpanding компонента TreeView?
69. В каком случае возникает событие OnExpanded компонента TreeView?
70. В каком случае возникает событие OnEditing компонента TreeView?
71. В каком случае возникает событие OnEdited компонента TreeView?
72. В каком случае возникает событие OnDeletion компонента TreeView?
73. Что реализует класс TTreeNodes?
74. Что означает свойство Count класса TTreeNodes?
75. Что означает свойство Items класса TTreeNodes?
76. Для чего используется метод Clear класса TTreeNodes?
77. Для чего используется метод AddFirst класса TTreeNodes?
78. Для чего используется метод Add класса TTreeNodes?
79. Для чего используется метод AddChild класса TTreeNodes?
80. Для чего используется метод Delete класса TTreeNodes?
81. Что реализует класс TTreeNode?
82. Что означает свойство Text класса TTreeNode?
83. Что означает свойство Parent класса TTreeNode?
84. Что означает свойство Level класса TTreeNode?
85. Что означает свойство Selected класса TTreeNode?
86. Что означает свойство ImageIndex класса TTreeNode?
87. Что означает свойство SelectedIndex класса TTreeNode?
88. Для чего используется метод Expand класса TTreeNode?
89. Для чего используется метод Collapse класса TTreeNode?
90. Для чего используется метод Delete класса TTreeNode?
91. Для чего используется метод DeleteChildren класса TTreeNode?
92. Для чего используется метод AlphaSort класса TTreeNode?
93. Для чего используется метод CustomSort класса TTreeNode?

94. Для чего используется компонент ListView?
95. Что означает свойство ViewStyle компонента ListView?
96. Что означает свойство Items компонента ListView?
97. Какой тип свойства Items компонента ListView?
98. Что означает свойство Selected компонента ListView?
99. Какой тип свойства Selected компонента ListView?
100. Что означает свойство Columns компонента ListView?
101. Какой тип свойства Columns компонента ListView?
102. Что означает свойство LargeImages компонента ListView?
103. Что означает свойство SmallImages компонента ListView?
104. Что означает свойство SortType компонента ListView?
105. Что означает свойство ReadOnly компонента ListView?
106. Что означает свойство Align компонента ListView?
107. Для чего используется метод AlphaSort компонента ListNode?
108. Для чего используется метод CustomSort компонента ListNode?
109. Что реализует класс TListItems?
110. Что означает свойство Count класса TListItems?
111. Что означает свойство Items класса TListItems?
112. Для чего используется метод Clear класса TListItems?
113. Для чего используется метод Add класса TListItems?
114. Для чего используется метод Insert класса TListItems?
115. Для чего используется метод Delete класса TListItems?
116. Что реализует класс TListItem?
117. Что означает свойство Caption класса TListItem?
118. Что означает свойство ImageIndex класса TListItem?
119. Что означает свойство Selected класса TListItem?
120. Что означает свойство SubItems класса TListItem?
121. Что означает свойство Data класса TListItem?
122. Что реализует класс TListColumns?
123. Что означает свойство Count класса TListColumns?
124. Что означает свойство Items класса TListColumns?
125. Для чего используется метод Clear класса TListColumns?
126. Для чего используется метод Insert класса TListColumns?
127. Для чего используется метод Delete класса TListColumns?
128. Что реализует класс TListColumn?
129. Что означает свойство Caption класса TListColumn?
130. Что означает свойство Width класса TListColumn?
131. Что означает свойство AutoSize класса TListColumn?
132. Что означает свойство Alignment класса TListColumn?
133. Для чего используется компонент StatusBar?
134. Что означает свойство SimplePanel компонента StatusBar?
135. Что означает свойство SimpleText компонента StatusBar?
136. Что означает свойство Panels компонента StatusBar?

137. Что означает свойство SizeGrip компонента StatusBar?
138. Что реализует класс TStatusPanels
139. Что означает свойство Count класса TStatusPanels?
140. Что означает свойство Items класса TStatusPanels?
141. Какой тип свойства Items класса TStatusPanels?
142. Что реализует класс TStatusPanel?
143. Что означает свойство Text класса TStatusPanel?
144. Что означает свойство Width класса TStatusPanel?
145. Что означает свойство Alignment класса TStatusPanel?
146. Для чего используется компонент ToolBar?
147. Что означает свойство Images компонента ToolBar?
148. Какой тип свойства Items класса ToolBar?
149. Что означает свойство ButtonCount компонента ToolBar?
150. Что означает свойство Buttons компонента ToolBar?
151. Какой тип свойства Buttons класса ToolBar?
152. Что реализует класс TToolButton?
153. Что означает свойство TImageIndex класса TToolButton?
154. В каком случае возникает событие OnClick класса TToolButton?

6 ПРОЧИЕ ПОЛЕЗНЫЕ КОМПОНЕНТЫ

6.1 Компонент Timer (Таймер)

Компонент Timer (таймер)  расположен на закладке System. Он предназначен для реализации процессов, выполняемых многократно через заданный промежуток времени.

Основные свойства компонента:

- Interval – интервал срабатывания таймера в миллисекундах (тип свойства int);
- Enabled – доступность таймера (логический тип); если Enabled = true – то через указанный в свойстве Interval промежуток времени генерируется событие OnTimer; В противном случае не генерируется.

Основное событие OnTimer возникает через указанный в свойстве Interval промежуток времени при Enabled = true. Двойной щелчок по таймеру приводит к открытию Редактора кода программы для записи функции обработки события OnTimer.

В качестве примера рассмотрим электронные часы. Расположите на форме элементы, согласно рис. 6.1.

Рисунок 6.1 – Заготовка для электронных часов

Задайте свойства формы и элементов, в соответствии с таблицей 6.1.

Таблица 6.1 – Свойства формы и элементов для электронных часов

Элемент	Свойство	Значение
Форма	Caption	Электронные часы
Edit1	Text	
Timer1	Interval	1000

В функции обработки события OnTimer напишите:

```
Edit1->Text = Time().TimeString();
```

Запустите программу.

6.2 Компонент PaintBox (Блок рисования)

Компонент PaintBox  (закладка System) предназначен для рисования графических объектов. Основное свойство компонента – Canvas. Это свойство предназначено для вызова графических функций рисования (тип TCanvas). Основной метод Paint генерирует событие OnPaint. Основное событие OnPaint возникает при необходимости перерисовки блока. В функции обработки этого события необходимо рисовать все графические объекты.

Тип **TCanvas** – класс данных имеющий следующие основные свойства:

- **Pen** – перо для рисования линий составной тип, состоит из:
  - Color – цвет (тип TColor);
  - Style – стиль границы (перечислимый тип, возможные значения приведены в таблице 3.5);
  - Width – ширина границы (тип int);
- **Brush** – кисть для закрашивания поверхностей составной тип, состоит из:
  - Color – цвет (тип TColor);
  - Style – стиль закрашивания (перечислимый тип, возможные значения приведены в таблице 3.6);
- **Font** – шрифт для вывода текста составной тип, состоит из:

- Name – наименование шрифта (тип AnsiString);
- Color – цвет шрифта (тип TColor);
- Size – размер (тип int);

• **Pixels** – двумерный массив цветов пикселей (тип каждого пикселя TColor).

• **Handle** – hdc блока.

Основные методы класса TCanvas (в алфавитном порядке):

• Arc (int X1, int Y1, int X2, int Y2, int X3, int Y3, int X4, int Y4); // рисование дуги;

• Chord(int X1, int Y1, int X2, int Y2, int X3, int Y3, int X4, int Y4); // рисование сегмента;

• Ellipse (int X1, int Y1, int X2, int Y2); // рисование эллипса;

• FloodFill(int X, int Y, TColor Color, TFillStyle FillStyle); // заливка цветом Color из точки (X, Y);

• LineTo(int X, int Y); // рисование прямой линии из текущей точки до точки (X,Y);

• MoveTo(int X, int Y); // перемещение текущей точки в (X,Y);

• Pie(int X1, int Y1, int X2, int Y2, int X3, int Y3, int X4, int Y4); // рисование сектора;

• Polygon (TPoint * Points, int TPoints); // рисование многоугольника;

• Rectangle(int X1, int Y1, int X2, int Y2); // рисование прямоугольника;

• RoundRect(int X1, int Y1, int X2, int Y2); // рисование прямоугольника с закругленными краями;

• TextOut(int X, int Y, AnsiString Text); // вывод текста;

В качестве примера рассмотрим рисование андреевского флага.

Расположите на форме PaintBox. Свойство Align установите в alClient. Задайте обработчик события OnPaint следующего вида:

```
PaintBox1->Canvas->Pen->Color = clBlue;
PaintBox1->Canvas->Pen->Width = 5;
PaintBox1->Canvas->MoveTo(0, 0);
PaintBox1->Canvas->LineTo(PaintBox1->Width, PaintBox1->Height);
PaintBox1->Canvas->MoveTo(PaintBox1->Width, 0);
PaintBox1->Canvas->LineTo(0, PaintBox1->Height);
```

Запустите программу.

Самостоятельно с помощью линий нарисуйте красную звезду с серпом и молотом внутри.

6.3 Мышь и блок рисования

Для обработки действий, связанных с мышью, используют следующие события формы или компонента:

• OnMouseDown – нажатие одной из кнопок мыши;
• OnMouseUp – отпускание одной из кнопок мыши;
• OnMouseMove – перемещение мыши.

В функциях обработки всех трех событий есть параметры для определения местоположения события: X – координата по горизонтали, Y- координата по вертикали.

В первых двух событиях имеется необходимость определить, какая из клавиш мыши была нажата или отпущена. Для этого имеется параметр Button, который может принимать одно из трех значений:

• mbLeft – нажата или отпущена левая клавиша;
• mbRight – правая;
• mbMiddle – средняя.

В качестве первого примера рассмотрим определение нажатой клавиши и места нажатия. Для этого на чистой форме разместите PaintBox. Задайте свойство: Align = alClient и функцию обработки события OnMouseDown:

```
char Buf[100];
 if (Button == mbLeft) {
 sprintf(Buf, "Левая (%d,%d)", X, Y);
 PaintBox1->Canvas->TextOut(X, Y, Buf);
 } // if
 if (Button == mbRight) {
 sprintf(Buf, "Правая (%d,%d)", X, Y);
 PaintBox1->Canvas->TextOut(X, Y, Buf);
 } // if
 if (Button == mbMiddle) {
 sprintf(Buf, "Средняя (%d,%d)", X, Y);
 PaintBox1->Canvas->TextOut(X, Y, Buf);
 } // if
```

В начале программы не забудьте написать:

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

Запустите программу и понажимайте левой и правой кнопкой в пределах формы. Для этого в предыдущей программе задайте функцию обработки события

В качестве второго примера рассмотрим рисование трассы ведения мыши. Для этого в предыдущей программе задайте функцию обработки события OnMouseMove:

```
static int n=0;
 if (n == 0)
 PaintBox1->Canvas->MoveTo(X, Y);
 else
 PaintBox1->Canvas->LineTo(X, Y);
 n = 1;
```

Запустите программу и поводите мышью в пределах формы.

Третий пример будет посвящен рисованию синего крестика в месте нажатия левой кнопки. Для этого на чистой форме разместите PaintBox. Задайте свойство Align = alClient и функцию обработки события OnMouseDown:

```
if (Button == mbLeft) {
 PaintBox1->Canvas->Pen->Color = clBlue;
 PaintBox1->Canvas->MoveTo(X-10, Y-10);
 PaintBox1->Canvas->LineTo(X+10, Y+10);
 PaintBox1->Canvas->MoveTo(X-10, Y+10);
 PaintBox1->Canvas->LineTo(X+10, Y-10);
 } // if
```

Запустите программу и понажимайте в разных частях окна левой и правой клавишами мыши. Перекройте окно с крестиками окном от другой программы. После открытия окна, перекрытые другим окном крестики исчезнут. Для устранения этого отрицательного явления сделаем следующее.

1. Заменим функцию обработки события OnMouseDown:

```
if (Button == mbLeft) {
 x[n] = X;
 y[n] = Y;
 n++;
 PaintBox1->Invalidate( );
 } // if
```

Зададим функцию обработки события OnPaint:

```
for (int i=0; i<n; i++) {
 PaintBox1->Canvas->Pen->Color = clBlue;
 PaintBox1->Canvas->MoveTo(x[i]-10, y[i]-10);
 PaintBox1->Canvas->LineTo(x[i]+10, y[i]+10);
 PaintBox1->Canvas->MoveTo(x[i]-10, y[i]+10);
 PaintBox1->Canvas->LineTo(x[i]+10, y[i]-10);
 } // for
```

2. В секции public класса TForm1 добавим описание массивов x, y и переменной n:

```
int x[1000], y[1000], n;
```

Модифицируйте программу так, чтобы по нажатию правой кнопки рисовался красный кружок, а по нажатию левой продолжал рисоваться синий крестик.

6.4 Клавиатура и блок рисования

Для работы с клавиатурой у формы и некоторых элементов имеются события:

- OnKeyDown – нажатие клавиши;

• OnKeyUp – отпускание клавиши.

В функциях обработки этих событий имеется параметр key – код клавиши.

В качестве примера рассмотрим перемещение окружности с помощью стрелок на клавиатуре.

Разместите на чистой форме PaintBox. Задайте свойство: Align = alClient и функцию обработки события OnPaint:

```
PaintBox1->Canvas->Pen->Color = clBlue;
PaintBox1->Canvas->Pen->Width = 5;
PaintBox1->Canvas->Brush->Color = clYellow;
PaintBox1->Canvas->Ellipse(x-50, y-50, x+50, y+50);
```

Задайте две функции обработки событий формы. Для события OnKeyDown:

```
if (Key == VK_LEFT) {
 if (x >= 65) x -= 10;
 } // if
 if (Key == VK_RIGHT) {
 if (x <= cx-65) x += 10;
 } // if
 if (Key == VK_UP) {
 if (y >= 65) y -= 10;
 } // if
 if (Key == VK_DOWN) {
 if (y <= cy-65) y += 10;
 } // if
 Invalidate( );
```

Для события OnResize:

```
// Сохраняем координаты нижнего правого угла окна
cx = ClientWidth;
cy = ClientHeight;
// Окружность в центре
x = cx/2;
y = cy/2;
```

В начале текста программы опишите переменные:

```
static int cx, cy; // размеры области вывода данных
static int x, y; // центр окружности
```

Запустите программу и переместите окружность с помощью нажатия стрелок.

Модифицируйте предыдущую программу так, чтобы окружность перемещалась с помощью стрелок, крестик с помощью других клавиш (например F2, F3, F4, F5 коды VK_F2, VK_F3, VK_F4, VK_F5)

6.5 Таймер и блок рисования

Рассмотрим пример движущегося круга. Для этого разместим на форме таймер и PaintBox. Для PaintBox зададим свойство: Align = alClient и функцию обработки события OnPaint:

```
PaintBox1->Canvas->Ellipse(x-10, y-10, x+10, y+10);
```

В начале текста программы опишите переменные:

```
static int x=11, y=11, dx=1, dy=1, x1, y1;
```

Для таймера запишите функцию обработки события OnTimer:

```
x1 = x + 10 * dx;
y1 = y + 10 * dy;
if (x1 >= 10 && x1 <= Width-10)
{
      x = x1;
} else
  {
      dx = -dx;
      x = x + 10 * dx;
  } // else
if (y1 >= 10 && y1 <= Height-10)
{
      y = y1;
} else
  {
      dy = -dy;
      y = y + 10 * dy;
  } // else
```

Запустите программу.

Измените программу так, чтобы круг начинал двигаться из левого верхнего угла, а крестик из правого верхнего.

6.6 Компонент MediaPlayer (Аудио- и видео проигрыватель)

Компонент MediaPlayer  (закладка System) предназначен для воспроизведения аудио и видео клипов. Компонент MediaPlayer реализует систему управления показанную на рис. 6.2. В качестве экрана для проигрывателя можно использовать любой другой компонент большого размера, например, Memo.

Основные свойства проигрывателя:

- FileName – имя файла для проигрывания (тип AnsiString);
- Display – компонент для экрана.

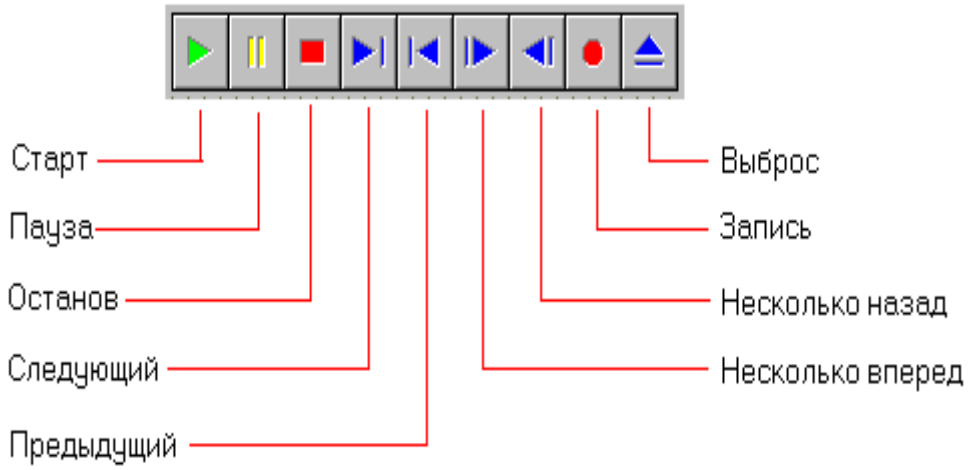


Рисунок 6.2 – Управление проигрывателем

6.7 Компоненты ClientSocket и ServerSocket

Компоненты ClientSocket  и ServerSocket  с закладки Internet предназначены для организации обмена информацией между программами, запущеными на одном или нескольких компьютерах в пределах сети (как локальной, так и глобальной). В глобальной сети могут быть ограничения, которые следует узнать у администратора сети. ServerSocket ожидает соединение, ClientSocket инициирует. После того, как соединение между ClientSocket и ServerSocket установлено, один из компонентов может передавать и получать информацию.

Основные свойства ClientSocket:

• Host – имя компьютера, с которым следует установит соединение (тип AnsiString). Для соединения с программой на своем компьютере можно использовать имя localhost;

• Address – IP-адрес компьютера (тип AnsiString); о том как узнать этот адрес будет написано ниже; IP адрес собственного компьютера: 127.0.0.1;

• Port – номер порта ServerSocket, с которым устанавливается соединение (тип int). Port назначается самостоятельно и должен быть не менее 1200;

• Active – логическое значение, показывающее подключен ClientSocket к ServerSocket или нет;

• Socket – свойство предназначенное для передачи и приема данных, будет описано ниже (тип TClientWinSocket).

Основные методы компонента ClientSocket:

• Open( ) – установить соединение с ServerSocket;

• Close( ) – разорвать соединение с ServerSocket.

• Основные события компонента ClientSocket. Возникают:

• OnConnect – при подключении к ServerSocket;

• OnDisconnect – при отключении от ServerSocket;

• OnError – при ошибке подключения, приема или передачи данных;
• OnRead – при получении данных;
• OnWrite – при отправке данных.

Класс TClientWinSocket нужен для приема и передачи данных. Перечислим его методы:

• SendBuf(void *Buf, int Count) – передать данные. Buf – указатель на область передаваемых данных, Count – количество передаваемых байтов;
• SendText(AnsiString S) -передать строку S;
• ReceiveLength( ) – количество байтов, которые следует принять;
• ReceiveBuf(void *Buf, int Count) – принять данные; Buf – указатель на область, в которую будут записаны принимаемые данных, Count – размер области;
• ReceiveText( ) – принять строку, возвращает принятую строку.

Основные свойства ServerSocket:

• Port – номер порта (тип int). Port назначается самостоятельно и должен быть не менее 1200;
• Active – логическое значение, показывающее, готов ли ServerSocket к подключению;
• Socket – свойство, предназначенное для передачи и приема данных, будет описано ниже (тип TServerWinSocket).

Основные методы компонента ServerSocket:

• Open( ) – установить соединение с ClientSocket;
• Close( ) – разорвать соединение с ClientSocket.

Основные события компонента ServerSocket:

• OnClientConnect – возникает при подключении ClientSocket;
• OnClientDisconnect – при отключении от ClientSocket;
• OnClientError – при ошибке подключения, приема или передачи данных;
• OnClientRead – при получении данных;
• OnClientWrite – при отправке данных.

Класс TServerWinSocket аналогичен классу TClientWinSocket.

6.7.1 Определение своего IP-адреса и доменного имени компьютера

Прежде, чем передавать и принимать данные необходимо знать систему адресов. Определим свои IP-адрес и доменное имя компьютера. Для этого на форме расположим компоненты с закладки Standard, согласно рис 6.3.

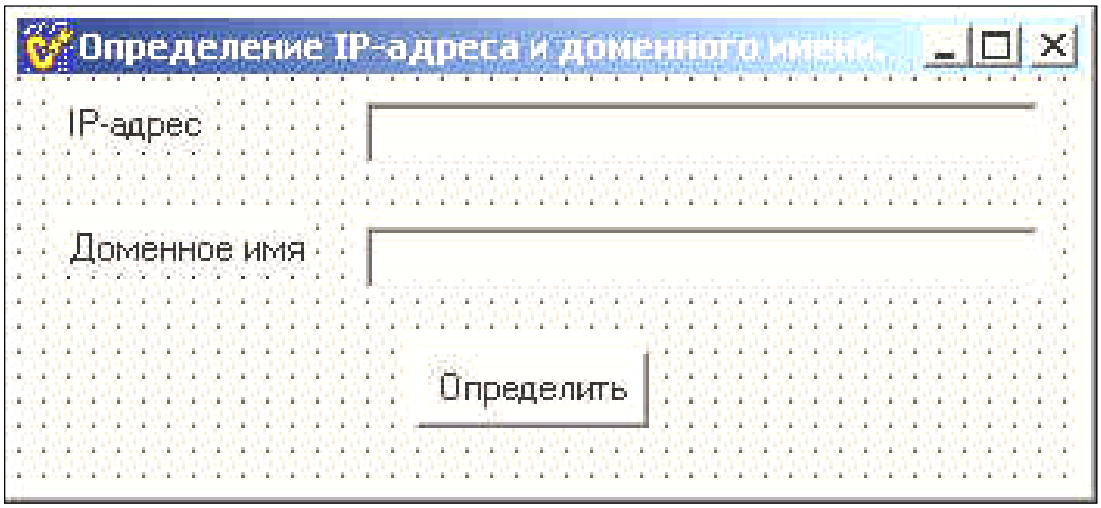


Рисунок 6.3 – Форма для определения имени компьютера и IP-адреса

Двойным щелчком по кнопке откройте окно для записи кода программы и введите:

```
int rc;
WSADATA WSAData;
char name[80], *lpAddr;
PHOSTENT phe;
in_addr Addr;

rc = WSAStartup(0x0101, &WSAData);
if (rc != 0) {
MessageBox(NULL, "TCP/IP не установлен", "", MB_OK);
return;
} // if

rc = gethostname(name, 80);
if (rc == -1) {
MessageBox(NULL, "Ошибка при определении имени компьютера", "Ошибка",
MB_OK);
return;
} // if

phe = gethostbyname(name);
if (phe == NULL) {
MessageBox(NULL, "Ошибка в имени компьютера", "Ошибка", MB_OK);
return;
} // if

memcpy((char FAR *)&(Addr), phe->h_addr, phe->h_length);
lpAddr = inet_ntoa(Addr);

Edit1->Text = lpAddr;
Edit2->Text = name;
```

В начале программы рядом со строчкой **#include "Unit1.h"** напишите строчку **#include "winsock.h"**

Сохраните и запустите программу. Если у Вас нет ошибок, то Вам станут известны IP-адрес и доменное имя Вашего компьютера. Обменяйтесь этими адресами с теми, с кем хотите осуществлять обмен сообщениями в дальнейшем.

6.7.2 Приложение для получения сообщений.

Создайте новый проект.

Выберите закладку Internet. В данном упражнении будем использовать компоненты  ClientSocket и  ServerSocket. Программа, имеющая компонент ServerSocket, ожидает, когда другая программа свяжется с ней и передаст информацию. Программа, имеющая компонент ClientSocket, может организовать связь с другой программой, которая имеет компонент ServerSocket и ожидает связь. Таким образом, первой должна быть запущена программа, которая имеет компонент ServerSocket и ожидает установление связи. С нее и начнем.

Расположите на форме компоненты  ServerSocket и  ListBox (Список) с закладки Standard. ServerSocket будет принимать сообщения, а ListBox – отображать их. Для компонента ServerSocket установите свойства Port = 2000, Active = true. Форму расположите так, чтобы она занимала правую половину экрана. ListBox растяните на всю форму. Щелкните один раз по компоненту ServerSocket, в инспекторе объектов перейдите на закладку Events. Дважды щелкните напротив события OnClientRead, в открывшемся окне кода наберите следующую строчку текста программы:

```
ListBox1->Items->Add(Socket->ReceiveText( ));
```

Сохраните проект и запустите его. Т.к. Вам в этот момент времени никто сообщения не посылает, то окно останется пустым. Теперь начнем создавать программу для посылки сообщений. Для этого необходимо завершить программу приема сообщение (чтобы освободится C++Builder), а затем войти в папку, в которой был сохранен проект и запустить полученный exe-файл. Запустится та же самая задача и при этом C++Builder будет свободен.

6.7.3 Приложение для отправки сообщений.

Создайте новый проект.

На форме расположите компоненты согласно рис 6.4.

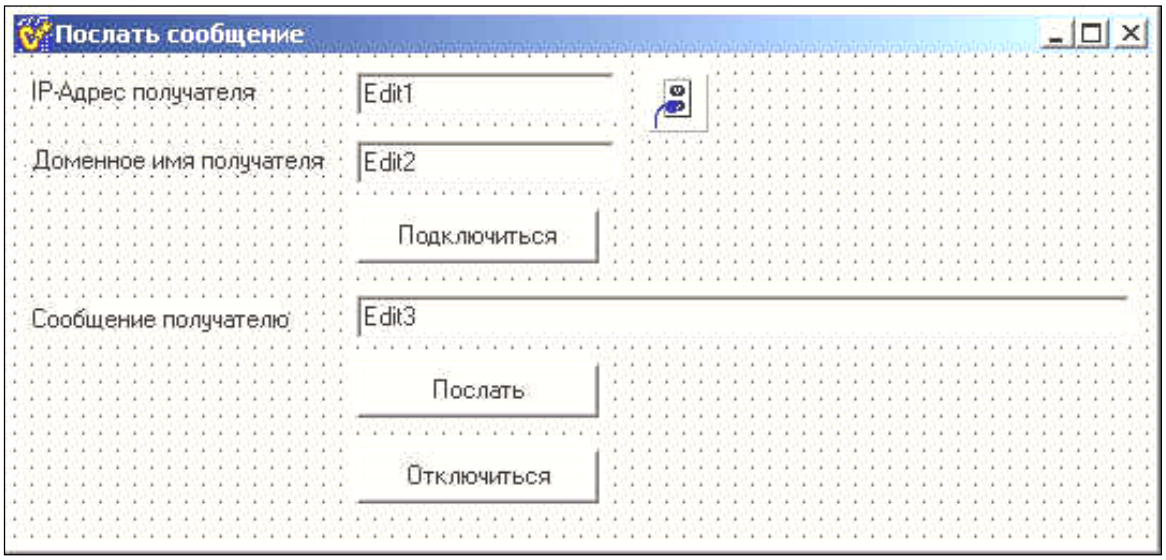


Рисунок 6.4 – Форма для отправки сообщений

В свойствах ClientSocket укажите Port = 2000.

Дважды щелкнув по кнопке "Подключиться" запишите следующий код:

```
if (Edit1->Text != "") {
ClientSocket1->Address = Edit1->Text;
} // if
if (Edit2->Text != "") {
ClientSocket1->Host = Edit2->Text;
} // if
ClientSocket1->Open( );
```

Дважды щелкнув по кнопке "Послать" запишите следующий код:

```
ClientSocket1->Socket->SendText(Edit3->Text);
```

Дважды щелкнув по кнопке "Отключиться" запишите следующий код:

```
ClientSocket1->Close( );
```

Сохраните проект и запустите его.

В одном из двух верхних полей введите IP-адрес или доменное имя товарища, с которым Вы обменялись адресами в конце первого упражнения. Нажмите кнопку "Подключиться". Напишите сообщение, которое хотите передать товарищу. Нажмите поочередно кнопки "Послать" и "Отключиться". Если все сделано правильно, Ваш товарищ получит сообщение. Следите затем, какие Вам приходят сообщения. К сожалению, в данной программе приема неизвестно, от кого пришло сообщение. Для того чтобы определить адрес отправителя необходимо изменить программу, принимающую сообщения. Вместо единственной строки в тексте программы нужно ввести следующую строку:

```
ListBox1->Items->Add(Socket->RemoteHost+" "+Socket->ReceiveText( ));
```

6.8 Компонент CppWebBrowser

Компонент CppWebBrowser  с закладки Internet предназначен для создания средств просмотра Web страниц интернета.

Основные методы компонента CppWebBrowser:

- Navigate(wchar_t WebPage) – переход к указанному Web сайту или странице.
- Refresh( ) – обновление страницы.
- Stop( ) – останов загрузки страницы.
- GoHome( ) – переход к домашней странице.
- GoBack( ) – переход к предыдущей странице.
- GoForward( ) – переход к последующей странице.

В качестве примера рассмотрим простой интернет-навигатор.

Для этого расположите на чистой форме следующие элементы: 5 кнопок, ComboBox, Panel. На Panel поместите CppWebBrouser. Рекомендуемое расположение показано на рис. 6.5.

Задайте свойства: для элемента Panel: Align-alBottom, BorderStyle – bsSingle, Top – 60, для элемента CppWebBrouser: Align-alClient.

Для элемента ComboBox1 задайте функцию обработки события OnClick:

```
wchar_t s1[100];
StringToWideChar(ComboBox1->Text, s1, 100);
if (ComboBox1->Text != "") {
CppWebBrowser1->Navigate(s1);
} // if
```

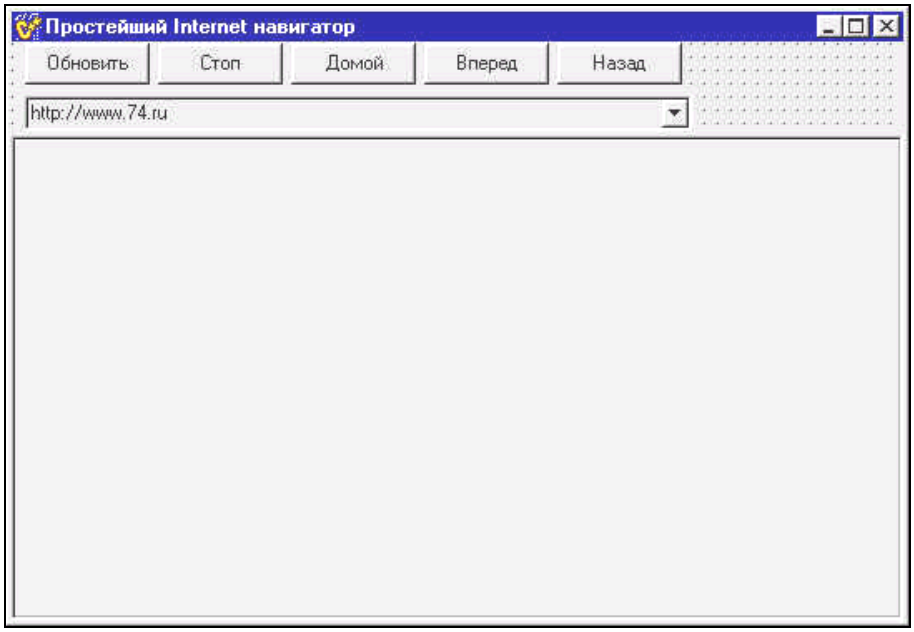


Рисунок 6.5 – Простейший интернет-навигатор

Для элемента ComboBox1 задайте функцию обработки события OnKeyDown:

```
if (Key == VK_RETURN) {
      Key = 0;
      if (ComboBox1->Text == "") return;
      ComboBox1->Items->Insert(0, ComboBox1->Text);
      ComboBox1Click(Sender);
} // if
```

Для элемента ComboBox1 задайте функцию обработки события OnDblClick:

```
ComboBox1Click(Sender);
```

Для кнопки "Обновить" задайте действие:

```
CppWebBrowser1->Refrech( );
```

Для кнопки "Стоп" задайте действие:

```
CppWebBrowser1->Stop( );
```

Для кнопки "Домой" задайте действие:

```
CppWebBrowser1->GoHome( );
```

Для кнопки "Вперед" задайте действие:

```
CppWebBrowser1->GoForward( );
```

Для кнопки "Назад" задайте действие:

```
CppWebBrowser1->GoBack( );
```

Запустите программу.

6.9 Контрольные вопросы

1. Для чего используется компонент Timer?
2. Что означает свойство Interval компонента Timer?
3. Что означает свойство Enabled компонента Timer?
4. В каком случае возникает событие OnTimer компонента Timer?
5. Для чего используется компонент PaintBox?
6. Для чего используется свойство Canvas компонента PaintBox?
7. В каком случае возникает событие OnPaint компонента PaintBox?
8. Что означает свойство Pen класса TCanvas?
9. Что означает свойство Brush класса TCanvas?
10. Что означает свойство Font класса TCanvas?
11. Что означает свойство Pixels класса TCanvas?
12. Что означает свойство Handle класса TCanvas?
13. Для чего используется метод Arc класса TCanvas?
14. Для чего используется метод Chord класса TCanvas?
15. Для чего используется метод Ellipse класса TCanvas?

16. Для чего используется метод FloodFill класса TCanvas?
17. Для чего используется метод LineTo класса TCanvas?
18. Для чего используется метод MoveTo класса TCanvas?
19. Для чего используется метод Pie класса TCanvas?
20. Для чего используется метод Polygon класса TCanvas?
21. Для чего используется метод Rectangle класса TCanvas?
22. Для чего используется метод RoundRect класса TCanvas?
23. В каком случае возникает событие OnMouseDown?
24. В каком случае возникает событие OnMouseUp?
25. В каком случае возникает событие OnMouseMove?
26. В каком случае возникает событие OnKeyDown?
27. В каком случае возникает событие OnKeyUp?
28. Для чего используется компонент MediaPlayer?
29. Что означает свойство FileName компонента MediaPlayer?
30. Что означает свойство Display компонента MediaPlayer?
31. Для чего используется компонент ClientSocket?
32. Для чего используется компонент ServerSocket?
33. Что означает свойство Host компонента ClientSocket?
34. Что означает свойство Address компонента ClientSocket?
35. Что означает свойство Port компонента ClientSocket?
36. Что означает свойство Active компонента ClientSocket?
37. Для чего используется метод Open компонента ClientSocket?
38. Для чего используется метод Close компонента ClientSocket?
39. В каком случае возникает событие OnConnect компонента ClientSocket?
40. В каком случае возникает событие OnDisconnect компонента ClientSocket?
41. В каком случае возникает событие OnError компонента ClientSocket?
42. В каком случае возникает событие OnRead компонента ClientSocket?
43. В каком случае возникает событие OnWrite компонента ClientSocket?
44. Для чего используется метод SendBuf класса TClientWinSocket?
45. Для чего используется метод SendText класса TClientWinSocket?
46. Для чего используется метод ReceiveLength класса TClientWinSocket?
47. Для чего используется метод ReceiveBuf класса TClientWinSocket?
48. Для чего используется метод ReceiveTex класса TClientWinSocket?
49. Что означает свойство Port компонента ServerSocket?
50. Что означает свойство Active компонента ServerSocket?
51. В каком случае возникает событие OnClientConnect компонента ServerSocket?
52. В каком случае возникает событие OnClientDisconnect компонента ServerSocket?
53. В каком случае возникает событие OnClientError компонента ServerSocket?

54. В каком случае возникает событие OnClientRead компонента ServerSocket?
55. В каком случае возникает событие OnClientWrite компонента ServerSocket?
56. Для чего используется компонент CppWebBrowser?
57. Для чего используется метод Navigate класса CppWebBrowser?
58. Для чего используется метод Refresh класса CppWebBrowser?
59. Для чего используется метод Stop класса CppWebBrowser?
60. Для чего используется метод GoHome класса CppWebBrowser?
61. Для чего используется метод GoBack класса CppWebBrowser?
62. Для чего используется метод GoForward класса CppWebBrowser?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Архангельский, А.Я. Программирование в C++Builder 6 / А.Я. Архангельский. – М.: «Издательство «БИНОМ», 2003. – 1152 с.: ил.

2 Керниган, Б.И., Ритчи Д.М. Язык программирования С / Б.И. Керниган, Д.М. Ритчи. – 2-е изд.: пер. с англ. – М.: Издат. дом «Вильямс», 2006. – 304 с.

3 Липман, С. Язык программирования C++. Полное руководство. / С. Липман, Ж. Лажойе. – М.: ДМК Пресс, 2016. – 1105 с.

4 Пахомов, Б.И. C/C++ и MS Visual C++ 2012 для начинающих / Б.И. Пахомов. – СПб.: БХВ-Петербург, 2015. – 512 с.

5 Русакова, З.Н. Динамические структуры данных и вычислительные алгоритмы. Visual C++/ З.Р. Русакова. – СПб.: Образовательные проекты, 2013. – 272 с.

ОГЛАВЛЕНИЕ

Учебное издание

Оленчикова Татьяна Юрьевна,
Сартасова Марина Юрьевна

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ
В C ++ Builder 10.0

Учебное пособие

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 20.06.2019. Формат 60×84 1/16. Печать цифровая.
Усл. печ. л. 6,51. Тираж 30 экз. Заказ 213/528.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.