Никита Культин

bhv

C++ Builder
в задачах и примерах

Готовые решения и тексты программ

Использование базовых компонентов

Программирование графики, игр, мультимедиа и баз данных

Справочник по компонентам и функциям

+ CD-ROM

Никита Культин

C++ Builder
в задачах и примерах

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.068+800.92С++
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 С++ Builder в задачах и примерах. — СПб.: БХВ-Петербург, 2005. — 336 с.: ил.

ISBN 5-94157-631-5

Книга представляет собой сборник программ и задач для самостоятельного решения в среде разработки С++ Builder. Примеры различной сложности — от простейших до приложений работы с графикой, мультимедиа и базами данных — демонстрируют назначение компонентов и раскрывают тонкости процесса программирования в С++ Builder. Справочник содержит описания базовых компонентов и наиболее часто используемых функций. На прилагаемом компакт-диске находятся исходные тексты программ.

Для начинающих программистов

УДК 681.3.068+800.92С++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Шишигин Игорь*
Зав. редакцией	*Григорий Добин*
Редактор	*Андрей Смышляев*
Компьютерная верстка	*Татьяны Олоновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.08.05.
Формат 60×90$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 21.
Тираж 5000 экз. Заказ № 1241
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-631-5

Содержание

Предисловие

В последнее время резко возрос интерес к программированию. Это связано с развитием и внедрением в повседневную жизнь информационных и коммуникационных технологий. Если человек имеет дело с компьютером, то, рано или поздно, у него возникает желание, а иногда и необходимость, программировать.

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую "быструю разработку". В основе систем быстрой разработки или RAD-систем (Rapid Application Development — среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть рутины, оставляя программисту работу по конструированию диалоговых окон и созданию функций обработки событий. Производительность программиста при использовании RAD-систем — фантастическая!

Одной из широко используемых RAD-систем является Borland C++Builder, которая позволяет создавать различные программы: от простейших однооконных приложений до программ управления распределенными базами данных. В качестве языка программирования в среде Borland C++Builder используется C++.

Чтобы научиться программировать, надо программировать — писать программы, решать конкретные задачи. Для этого надо изучить язык программирования и среду разработки. И здесь хорошим подспорьем могут быть программы, которые демонстрируют назначение компонентов и особенности их использования.

В книге, которую вы держите в руках, собраны разнообразные примеры, которые демонстрируют технологию создания программ, возможности среды разработки, назначение компонентов,

знакомят с принципами работы с графикой, звуком, базами данных. Следует обратить внимание на то, что большинство примеров не являются учебными — это вполне работоспособные программы.

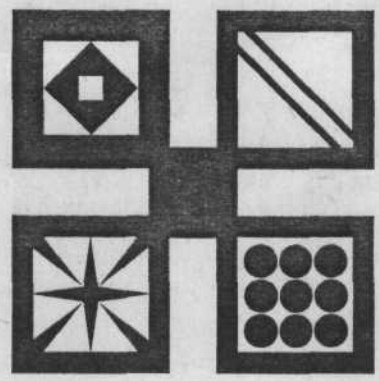
Состоит книга из двух частей.

Первая часть содержит примеры, представленные в виде краткого описания, диалоговых окон и прокомментированных текстов программ.

Вторая часть книги — это краткий справочник, в котором можно найти описание базовых компонентов и наиболее часто используемых функций.

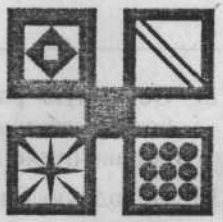
Научиться программировать можно, только решая конкретные задачи. При этом успехи, достигнутые в программировании, в значительной степени зависят от опыта. Поэтому чтобы получить максимальную пользу от книги, вы должны активно с ней работать. Изучайте листинги, старайтесь понять, как работают программы. Не бойтесь экспериментировать — вносите изменения в программы.

Если что-то не понятно, обратитесь к справочнику в конце книге, к справочной системе C++Builder или к литературе, например, к книге **Культин Н. Б. Самоучитель C++Builder. — СПб.: БХВ-Петербург, 2004**. В ней помимо описания среды разработки, компонентов, процессов создания и отладки программ вы найдете ответы на многие вопросы, в том числе: как создать базу данных и зарегистрировать ее в системе, как, при помощи Microsoft Help Workshop, создать справочную систему, как, используя InstallShield Express, создать дистрибутив (пакет для установки программы).



ЧАСТЬ 1

Примеры и задачи



Базовые компоненты

В этом разделе приведены примеры, которые должны продемонстрировать назначение и технологию работы с базовыми компонентами.

Общие замечания

- Процесс создания программы в C++Builder состоит из двух шагов: сначала нужно создать форму программы (диалоговое окно), а затем функции обработки *событий*. Форма *приложения* (так принято называть прикладные программы, работающие в Windows) создается путем добавления в нее *компонентов* и последующей их настройки.
- В форме практически любого приложения есть компоненты, которые обеспечивают интерфейс (взаимодействие) между программой и пользователем. Такие компоненты называют базовыми. К базовым компонентам можно отнести:
  - `Label` — поле вывода текста;
  - `Edit` — поле редактирования текста;
  - `Button` — командная кнопка;
  - `CheckBox` — независимая кнопка выбора;
  - `RadioButton` — зависимая кнопка выбора;
  - `ListBox` — список выбора;
  - `ComboBox` — комбинированный список выбора.
- Вид компонента, его размер и поведение определяют значения *свойств* (характеристик) компонента (описание свойств базовых компонентов можно найти в справочнике во второй части книги).

- Основную работу в программе выполняют функции обработки *событий* (описание основных событий можно найти в справочнике во второй части книги).
- Исходную информацию программа может получить из полей редактирования (компонент `Edit`), списка выбора (компонент `ListBox`) или комбинированного списка (компонент `ComboBox`). Для ввода значений логического типа можно использовать компоненты `CheckBox` и `RadoiButton`.
- Результат программа может вывести в поле вывода текста (компонент `Label`) или в окно сообщения (функции `ShowMessage`, `MessageDlg`).
- Для преобразования текста, например, находящегося в поле редактирования, в целое число нужно использовать функцию `StrToInt`, а в дробное — функцию `StrToFloat`. Для преобразования целого, например, значения переменной, в строку нужно использовать функцию `IntToStr`, а для преобразования дробного — функцию `FloatToStr` или `FloatToStrF`.

Конвертор

Программа **Конвертор** пересчитывает цену из долларов в рубли. Демонстрирует использование компонентов `TextBox` и `Label` для ввода и отображения числовых данных. Программа спроектирована таким образом, что пользователь может ввести в поля редактирования только правильные данные (число). Форма программы приведена на рис. 1.1.

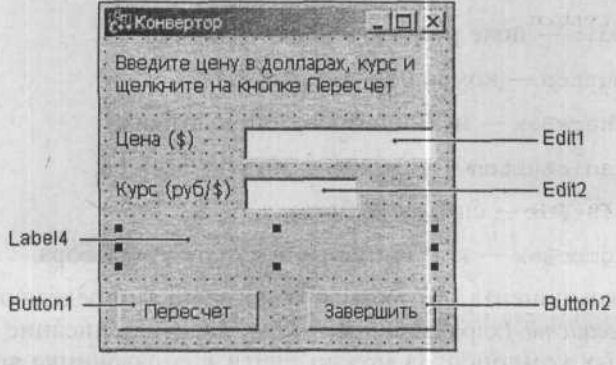


Рис. 1.1. Форма программы **Конвертор**

```
// нажатие клавиши в поле Цена
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char
&Key)
{
     // код запрещенного символа заменим нулем, в результате
    // символ в поле редактирования не появится

    // Key - код нажатой клавиши
    // проверим, является ли символ допустимым
    if ((Key >= '0') && (Key <= '9')) //цифра
        return;

    // глобальная переменная DecimalSeparator
    // содержит символ, используемый в качестве разделителя
    // при записи дробных чисел
    if (Key == DecimalSeparator)
    {
       if ((Edit1->Text).Pos(DecimalSeparator) != 0)
           Key = 0; // разделитель уже введен
       return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if (Key == VK_RETURN) // клавиша <Enter>
    {
        Edit2->SetFocus();
        return;
    }

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}
```

```
// нажатие клавиши в поле Курс
void __fastcall TForm1::Edit2KeyPress(TObject *Sender,
                                      char &Key)
{
   if ((Key >= '0') && (Key <= '9')) //цифра
      return;

    if (Key == DecimalSeparator)
    {
     if ((Edit2->Text).Pos(DecimalSeparator) != 0)
         Key = 0; // разделитель уже введен
      return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
       return;

    if (Key == VK_RETURN) // клавиша <Enter>
    {
       Button1->SetFocus();// переход к кнопке Вычислить
       // повторное нажатие клавиши <Enter>
       // активизирует процесс вычисления денег
        return;
    };

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// щелчок на кнопке Пересчет
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float usd; // цена в долларах
```

```
    float k;    // курс
    float rub; // цена в рублях
    // проверим, введены ли данные в поля Цена и Курс
    if (((Edit1->Text).Length() == 0) ||
            ((Edit2->Text).Length() == 0))
    {
       MessageDlg("Надо ввсети цену и курс",
          mtInformation, TMsgDlgButtons() << mbOK, 0);
        if ((Edit1->Text).Length() == 0)
          Edit1->SetFocus(); // курсор в поле Цена
        else
          Edit2->SetFocus(); // курсор в поле Курс
        return;
    };

    // ввод исходных данных
    usd = StrToFloat(Edit1->Text);
    k = StrToFloat(Edit2->Text);

    // вычисление
    rub = usd * k;

    // вывод результата
    Label4->Caption = FloatToStrF(usd,ffGeneral,7,2) +
            "$ = "+FloatToStrF(rub,ffGeneral,7,2) + " руб." ;
}

//щелчок на кнопке Завершить
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Form1->Close(); // закрыть форму приложения
}
```

Фунты-килограммы

Программа **Фунты-килограммы**, форма которой приведена на рис. 1.2, позволяет пересчитать вес из фунтов в килограммы. Программа спроектирована таким образом, что кнопка **Пересчет** доступна только в том случае, если пользователь ввел исходные данные.

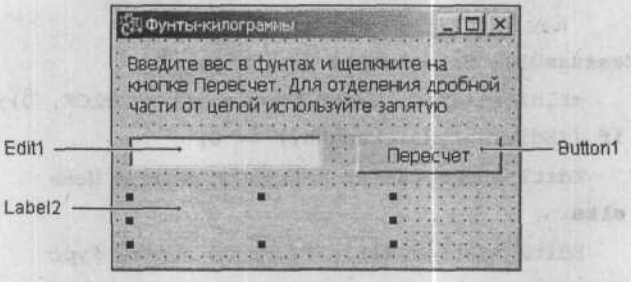


Рис. 1.2. Кнопка **Пересчет** доступна только тогда, когда в поле редактирования есть данные

```
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
    /* так как поле Edit1 пустое (пользователь
    еще не ввел исходные данные), то
    сделаем кнопку Пересчет недоступной */
    Button1->Enabled  =  False;
}

// нажатие клавиши в поле Edit1
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char
&Key)
{
    // код запрещенного символа заменим нулем, в результате
    // символ в поле редактирования не появится

    // Key - код нажатой клавиши
    // проверим, является ли символ допустимым
    if ((Key >= '0') && (Key <= '9')) //цифра
        return;
```

```
    // глобальная переменная DecimalSeparator
    // содержит символ, используемый в качестве разделителя
    // при записи дробных чисел
    if (Key == DecimalSeparator)
    {
        if ((Edit1->Text).Pos(DecimalSeparator) != 0)
            Key = 0; // разделитель уже введен
        return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if (Key == VK_RETURN) // клавиша <Enter>
    {
        Button1->SetFocus();
        return;
    }
    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// Содержимое поля Edit1 изменилось
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    // проверим, есть ли в поле Edit1 исходные данные
    if ( (Edit1->Text).Length() == 0)
        Button1->Enabled  =  False; // кн. Пересчет недоступна
    else Button1->Enabled  =  True; // кн. Пересчет доступна

   Label2->Caption = "";
}
```

```
// щелчок на кнопке Пересчет
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   double funt; // вес в фунтах
   double kg;   // вес в килограммах

   // кнопка Пересчет доступна только в том случае,
   // если в поле Edit1 есть данные. Поэтому,
   // наличие в поле информации можно не проверять.
   funt  =  StrToFloat(Edit1->Text);
   kg  =  funt * 0.4995;
   Label2->Caption  =  FloatToStrF(funt,ffGeneral,5,2) +
                       " ф. - это " +
                       FloatToStrF(kg,ffGeneral,5,2) + " кг";
}
```

Сила тока

Программа **Сила тока** демонстрирует использование компонентов TextBox и Label, а также обработку исключения "деление на ноль". Форма программы показана на рис. 1.3.

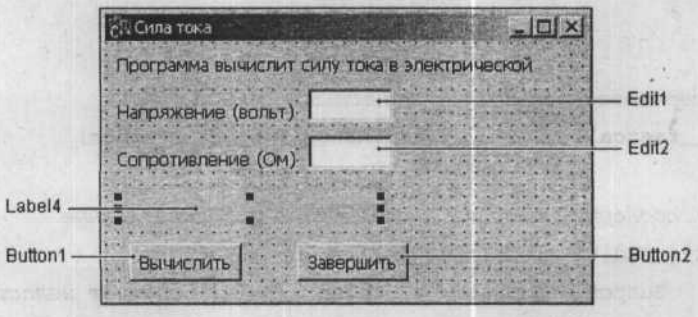


Рис. 1.3. Форма программы **Сила тока**

```
// щелчок на кнопке Вычислить
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
    float u; // напряжение
    float r; // сопротивление
    float i; // ток

    // проверим, введены ли данные в поля Напряжение и
    // Сопротивление
    if ( ((Edit1->Text).Length() == 0) ||
         ((Edit2->Text).Length() == 0))
    {
        MessageDlg("Надо ввести напряжение и сопротивление",
                mtInformation, TMsgDlgButtons() << mbOK, 0);
        if ((Edit1->Text).Length() == 0)
            Edit1->SetFocus(); // курсор в поле Напряжение
        else
            Edit2->SetFocus(); // курсор в поле Сопротивление
        return;
    };

    // получить данные из полей ввода
    u = StrToFloat(Edit1->Text);
    r = StrToFloat(Edit2->Text);

    // вычислить ток
    try
    {
       i = u/r;
    }
    catch (EZeroDivide &e)
    {
        ShowMessage("Величина сопротивления не должна быть"
                    "равна нулю");
        Edit2->SetFocus();    // курсор в поле Сопротивление
        return;
    }
```

```
    // вывести результат в поле Label4
   Label4->Caption = "Ток : " +
                     FloatToStrF(i,ffGeneral,7,2) + " A";
}

// нажатие клавиши в поле Напряжение
// коды запрещенных клавиш заменим нулем, в результате
// символы этих клавиш в поле редактирования не появятся
void __fastcall TForm1::Edit1KeyPress(TObject *Sender,
                                      char &Key)
{
    // Key - код нажатой клавиши
    // проверим, является ли символ допустимым

    if ( ( Key >= '0') && ( Key <= '9' ) ) // цифра
        return;

    // Глобальная переменная Decimalseparator
    // содержит символ, используемый в качестве разделителя
    // при записи дробных чисел
    if ( Key == DecimalSeparator)
    {
       if ( (Edit1->Text).Pos(DecimalSeparator) != 0 )
           Key = 0; // разделитель уже введен
    return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if ( Key == VK_RETURN) // клавиша <Enter>
    {
       Edit2->SetFocus();
       return;
    };
```

```
    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// нажатие клавиши в поле Сопротивление
void __fastcall TForm1::Edit2KeyDown(TObject *Sender,
                        WORD &Key,TShiftState Shift)
{
    if ( ( Key >= '0') && ( Key <= '9' ) ) // цифра
        return;

    if ( Key == DecimalSeparator) {
       if ( (Edit2->Text).Pos(DecimalSeparator) != 0 )
          Key = 0; // разделитель уже введен
       return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if ( Key == VK_RETURN) // клавиша <Enter>
    {
       Button1->SetFocus(); // переход к кнопке Вычислить
                            // повторное нажатие клавиши <Enter>
                            // активизирует процесс вычисления тока
                            // см. Button1Click
       return;
    };

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// щелчок на кнопке Завершить
void __fastcall TForm1::Button2Click(TObject *Sender)
```

```
{
    Form1->Close(); // закрыть форму приложения
}

/* Процедура Edit1Change обрабатывает событие Change
   как поля Edit1, так и поля Edit2.
   Сначала надо создать процедуру обработки события Change
   для поля Edit1, затем - в строке события Change компонента
   Edit2 щелкнуть на значке раскрывающегося списка и выбрать
   Edit1Change. */
void __fastcall TForm1::EditChange(TObject *Sender)
{
    Label4->Caption = "";
}
```

Сопротивление

Программа **Сопротивление**, ее форма приведена на рис. 1.4, вычисляет сопротивление электрической цепи, состоящей из двух резисторов, которые могут быть соединены последовательно или параллельно. Демонстрирует использование компонента `RadioButton`.

Рис. 1.4. Форма программы **Сопротивление**

```
// щелчок на кнопке Вычислить
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float r1,r2,r;

    r1 = StrToFloat(Edit1->Text);
    r2 = StrToFloat(Edit2->Text);

   /* Переключатели RadioButton1 и RadioButton2
      зависимые, поэтому о типе соединения можно
      судить по состоянию одного из них */
    if ( RadioButton1->Checked )
    {
        // выбран переключатель "последовательно"
        r = r1 + r2;
    }
    else
    {
        // выбран переключатель "параллельно"
        // при вычислении сопротивления возможно
        // исключение EInvalidOp
        try
        {
            r = (r1 * r2) / (r1 + r2);
        }
        catch ( EInvalidOp &e)
        {
            ShowMessage("Необходимо задать величину"
                        "сопротивлений");
            return;
        }
    }
    Label4->Caption = FloatToStrF(r,ffGeneral,6,2) + " Ом";
}
```

```
// щелчок на переключателе "последовательно"
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
    Label4->Caption = "";
}

// щелчок на переключателе "параллельно"
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
    Label4->Caption = "";
}
```

Кафе

Программа **Кафе**, ее форма приведена на рис. 1.5, демонстрирует использование компонента CheckBox.

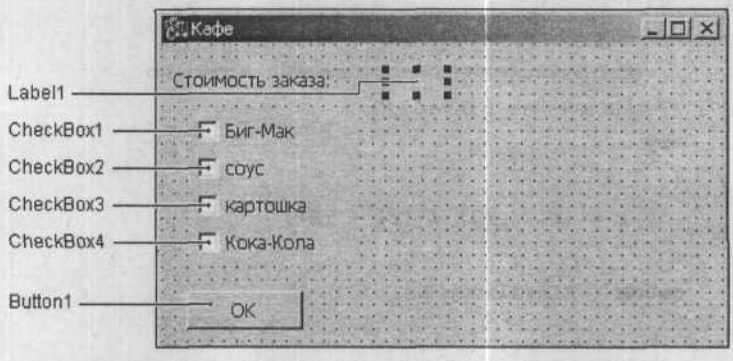


Рис. 1.5. Форма программы **Кафе**

```
float summ; // сумма заказа

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
```

```
{
    // сделать недоступным переключатель "соус"
    CheckBox2->Enabled = false;
}

// щелчок на переключателе "Биг-Мак"
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
    if ( CheckBox1->Checked )
    {
        /* переключатель был сброшен,
           пользователь установил его */
        summ += 54;

        // сделать доступным переключатель "соус"
        CheckBox2->Enabled = true;
    }
    else
    {
        /* переключатель был установлен,
           пользователь сбросил его */

        summ -= 54;
        // сбросить и сделать недоступным переключатель "соус"
        if (CheckBox2->Checked)
            CheckBox2->Checked = false;
        CheckBox2->Enabled = false;
    }

    // отобразить измененную сумму в поле
    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}
```

```
// щелчок на переключателе "соус"
void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
    if ( CheckBox2->Checked)
        summ += 10.5;
    else
        summ -= 10.5;

    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}

// щелчок на переключателе "картошка"
void __fastcall TForm1::CheckBox3Click(TObject *Sender)
{
    if ( CheckBox3->Checked)
        summ += 18.5;
    else
        summ -= 18.5;

    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}

// щелчок на переключателе "Кока-Кола"
void __fastcall TForm1::CheckBox4Click(TObject *Sender)
{
    if ( CheckBox4->Checked)
        summ += 14;
    else
        summ -= 14;
    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}

// щелчок на кнопке OK
```

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if ( (CheckBox1->Checked)&& (CheckBox2->Checked)&&
         (CheckBox3->Checked)&&(CheckBox4->Checked) )
    {
        /* пользователь заказал полный набор
           предоставить скидку 5% */
        summ = summ * 0.95;
        ShowMessage("Вам предоставляется скидка 5%.\n"
        "Сумма заказа: " + FloatToStrF(summ,ffCurrency,6,2)+
        " руб.");
    }
    else
        if ( (CheckBox1->Checked)||
             (CheckBox3->Checked)||
             (CheckBox4->Checked))
               ShowMessage("Сумма заказа: " +
                 FloatToStrF(summ,ffGeneral,6,2)+ " руб.");
        else   ShowMessage("Вы ничего не заказали");
}
```

Любимый напиток

Программа **Любимый напиток**, ее форма приведена на рис. 1.6, демонстрирует использование компонента ComboBox.

Списки компонентов ComboBox2 и ComboBox3 формируются во время работы программы (делает это конструктор формы). Пользователь может добавить элементы в списки компонентов ComboBox2 и ComboBox3, однако элемент в список компонента ComboBox3 добавляется только в том случае, если такого элемента в списке нет.

Значения свойств компонентов ComboBox приведены в табл. 1.1.

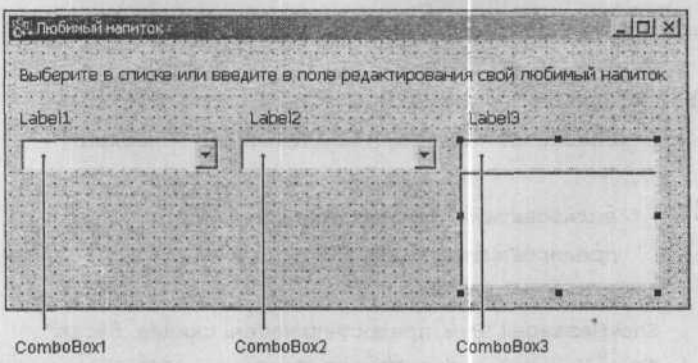


Рис. 1.6. Программа **Любимый напиток**

Таблица 1.1. *Значения свойств компонентов* `ComboBox`

Свойство	Значение	Комментарий
`ComboBox1.Style`	`csDropDownList`	Раскрывающийся список (добавить элемент в список нельзя)
`ComboBox2.Style`	`csDropDown`	Раскрывающийся комбинированный список. Пользователь может ввести текст в поле редактирования
`ComboBox3.Style`	`csSimple`	Поле редактирования и список

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
    // сформировать список компонента ComboBox3
    ComboBox2->Sorted = true; // список упорядочен
    ComboBox2->Items->Add("Кока-Кола");
    ComboBox2->Items->Add("Меринда");
    ComboBox2->Items->Add("Пепси-Кола");
    ComboBox2->Items->Add("Спрайт");
    ComboBox2->Items->Add("Фанта");
```

```
// сформировать список компонента ComboBox3
    ComboBox2->Sorted = true; // список упорядочен
    ComboBox3->Items->Add("Чай");
    ComboBox3->Items->Add("Чай с лимоном");
    ComboBox3->Items->Add("Кофе черный");
    ComboBox3->Items->Add("Кофе со сливками");
    ComboBox3->Items->Add("Какао");
}

// выбор элемента в списке ComboBox1
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{
    Label1->Caption = ComboBox1->Text;
}

// щелчок на элементе списка компонента ComboBox2
void __fastcall TForm1::ComboBox2Click(TObject *Sender)
{
    Label2->Caption = ComboBox2->Items->
                          Strings[ComboBox2->ItemIndex];
}

// щелчок на элементе списка компонента ComboBox3
void __fastcall TForm1::ComboBox3Click(TObject *Sender)
{
    Label3->Caption = ComboBox3->Items->
                        Strings[ComboBox3->ItemIndex];
}

// нажатие клавиши в поле редактирования компонента ComboBox2
void __fastcall TForm1::ComboBox2KeyPress(TObject *Sender,
char &Key)
{
    if  (Key == VK_RETURN)
    {
```

```
        // Пользователь ввел в поле редактирования строку
        // и нажал <Enter>. Добавим строку в список.
        int n = ComboBox2->Items->Add(ComboBox2->Text);
        ComboBox2->ItemIndex = n;
        Label2->Caption = ComboBox2->Items->Strings[n];
    }
}

// нажатие клавиши в поле редактирования компонента ComboBox3
void __fastcall TForm1::ComboBox3KeyPress(TObject *Sender,
                                          char &Key)
{
    AnsiString st;  // строка, которую ввел пользователь
                    // в поле редактирования компонента
                    // ComboBox

    if (Key == VK_RETURN)
    {
        // Пользователь ввел в поле редактирования строку
        // и нажал <Enter>. Если такой строки в списке нет,
        // добавим ее в список

        st = ComboBox3->Text.Trim(); // удалить пробелы

        if ( ComboBox3->Items->IndexOf(st) == -1 )
        {
            // добавить
            int n = ComboBox3->Items->Add(st);
            ComboBox3->ItemIndex = n;

            Label3->Caption = ComboBox3->Items->Strings[n];
        }
    }
}
```

Электроэнергия

Программа **Электроэнергия** (рис. 1.7) показывает, как одна функция может обрабатывать события разных, но однотипных компонентов.

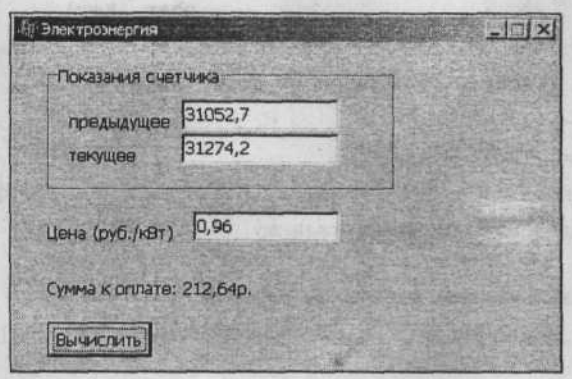


Рис. 1.7. Программа **Электроэнергия**

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
                        : TForm(Owner)
{
     /* задать, что событие KeyPress
      для компонентов Edit2 и Edit3
      обрабатывает функция EditKeyPress */

   Edit2->OnKeyPress = EditKeyPress;
   Edit3->OnKeyPress = EditKeyPress;

    /* чтобы процедура обработки события KeyPress могла
       определить, в каком поле пользователь нажал клавишу,
       запишем в свойство Tag каждого компонента Edit
       целую константу */
   Edit1->Tag = 1;
   Edit2->Tag = 2;
```

```
   Edit3->Tag = 3;
}

// нажатие клавиши в поле редактирования
void __fastcall TForm1::EditKeyPress(TObject *Sender,
                                     char &Key)
{
    TEdit * Edit;  // компонент Edit

    Edit = (TEdit*)Sender;
    /* теперь ed - это компонент Edit, в поле которого
       пользователь нажал клавишу */

    /* Реакция компонентов на нажатие всех клавиш,
       за исключением <Enter> одинаковая. */

   if ( Key == VK_RETURN) // нажате клавиша <Enter>
   {
        switch (Edit->Tag) {
            case 1 : /* клавиша нажата в поле Edit1
                        переместить курсор в поле Edit3 */
                     Edit2->SetFocus() ; break;

            case 2 : /* клавиша нажата в поле Edit2
                        переместить курсор в поле Edit3 */
                     Edit3->SetFocus() ; break;

            case 3 : /* клавиша нажата в поле Edit3
                        сделать активной кнопку Вычислить */
                     Button1->SetFocus() ; break;
        }
        return;
   }

    if ( (( Key >= '0') && ( Key <= '9' ) ) ||
                   (Key == VK_BACK))
```

```
    {
        // цифра или <>Backspace>
        return;
    }

    if ((Key == ',') || (Key == '.'))
    {
       Key = DecimalSeparator;
       if ( (Edit->Text).Pos(DecimalSeparator) != 0 )
           Key = 0;
       return;
    }

    if ( Key == VK_BACK )
        return;

    // остальные символы запрещены
    Key = 0;
}

// щелчок на кнопке Вычислить
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float prior, curr; // предыдущее и текущее показания
                       // счетчика
    float tariff;      // тариф - цена 1 кВт/час

    float summ;        // сумма к оплате

    prior = StrToFloat(Edit1->Text);
    curr = StrToFloat(Edit2->Text);
    tariff = StrToFloat(Edit3->Text);

  // проверить исходные данные
    if ( curr < prior)
```

```
    {
        MessageDlg("Текущее значение показания счетчика не"
                   " может быть меньше предыдущего.",
                     mtWarning, TMsgDlgButtons() << mbOK,0);
        return;
    }
    summ = (curr - prior) * tariff;
    Label4->Caption = "Сумма к оплате: " +
                        FloatToStrF(summ, ffCurrency, 6,2);
}
```

ОСАГО

Программа **ОСАГО** (рис. 1.8), позволяет рассчитать размер страховой премии (рис. 1.9), подлежащей уплате по договору обязательного страхования гражданской ответственности. Демонстрирует использование компонента `ComboBox`, обработку одной функцией событий от нескольких компонентов. Программа спроектирована таким образом, что кнопка **ОК** доступна только в том случае, если введены все данные, необходимые для расчета.

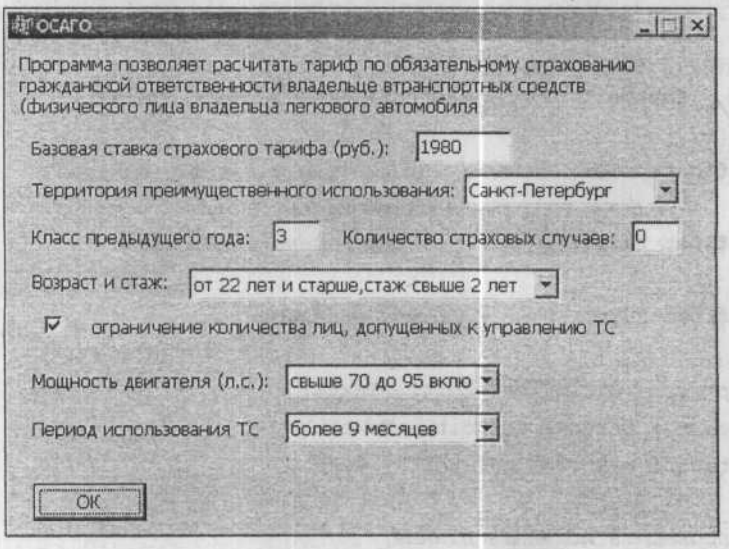


Рис. 1.8. Окно программы **ОСАГО**

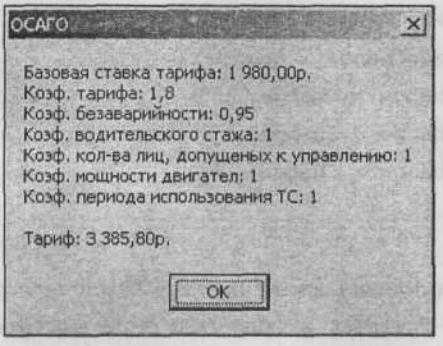


Рис. 1.9. Результат расчета

```
AnsiString reg[8] = {"Москва","Московская обл.",
                     "Санкт-Петербург",
                     "Нижний новгород",
                     "Ленинградская обл.",
                     "Ростов-наДону", "Самара",
                     "Мурманск"};

// коэф., учитывающий регион
float Kt [8] = {1.8,1.6, 1.8,1.3,1,1,1,1};

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    int i,n;
    n = sizeof (Kt) / sizeof (float);
    for (i=0; i<n; i++)
        ComboBox1->Items->Add(reg[i]);

    /* событие Change всех компонентов обрабатывает
       функция TForm1::Change */
    ComboBox1->OnChange = Change;
    ComboBox2->OnChange = Change;
```

```
    ComboBox4->OnChange = Change;
    ComboBox4->OnChange = Change;
    Edit1->OnChange = Change;
    Edit2->OnChange = Change;
    Edit3->OnChange = Change;
}

/* таблица определения коэффициента страхового тарифа
Первый год - 3-й класс, второй год (если не было страховых
случаев) 4-й класс и т.д.). Если страховой случай был,
то класс ячейка таблицы: строка - класс предыдущего
года, столбец - кол-во страховых случаев. */

// класс безаварийности
int Cb[6][5] = {{1,-1,-1,-1,-1},
                {2,-1,-1,-1,-1},
                {3,1,-1,-1,-1},
                {4,1,-1,-1,-1},
                {5,2,1,-1,-1},
                {6,3,1,-1,-1}};

// коэффициент безаварийности
float Kb[7] = {2.3, 1.55, 1.4, 1, 0.95, 0.9};

// щелчок на кнопке OK
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float aTb;  // базовая ставка тарифа
    float aKt;  // коэф. тарифа
    float aKb;  // коэф. безаварийности
    float aKvs; // коэф. водительского стажа
    float aKo;  // коэф., учитывающий количество лиц,
                // допущеных к управлению
```

```
float aKm;    // коэф. мощности двигателя
float aKs;    // коэф., учитывающий период использования ТС

int pcb,ccb; // предыдущий и текущий класс безаварийности
int nss;     // количество страховых случаев предыдущего
             // периода

aTb = StrToFloat(Edit3->Text);
aKt = Kt[ComboBox1->ItemIndex];

pcb = StrToInt(Edit1->Text);
nss = StrToInt(Edit2->Text);
ccb =   Cb[pcb][nss];
if ( ccb != -1)
    aKb = Kb[ccb];
else aKb = 2.45;
aKb = Kb[ccb];

// коэф. водительского стажа
switch (ComboBox2->ItemIndex)
{
    case 0: aKvs = 1.3;  break;
    case 1: aKvs = 1.2;  break;
    case 2: aKvs = 1.15; break;
    case 3: aKvs = 1.0;  break;
}

// коэф., учитывающий количество лиц, допущенных к
// управлению
if (CheckBox1->Checked)
    aKo = 1;
else
    aKo = 1.5;
```

```
// коэф. мощности двигателя
switch (ComboBox3->ItemIndex)
{
    case 0 : aKm = 0.5; break;
    case 1 : aKm = 0.7; break;
    case 2 : aKm = 1.0; break;
    case 3 : aKm = 1.3; break;
    case 4 : aKm = 1.5; break;
    case 5 : aKm = 1.7; break;
    case 6 : aKm = 1.9; break;
}

// коэф., учитывающий период использования ТС
switch (ComboBox4->ItemIndex)
{
    case 0 : aKs = 0.7; break;
    case 1 : aKs = 0.8; break;
    case 2 : aKs = 0.9; break;
    case 3 : aKs = 0.95; break;
    case 4 : aKs = 1.0; break;
}

// все коэффициенты определены

float T; // тариф
AnsiString st;

T = aTb * aKt * aKb * aKvs * aKo * aKm *aKs;

st = "Базовая ставка тарифа: " +
     FloatToStrF(aTb,ffCurrency,5,2)+
      "\nКоэф. тарифа: " + FloatToStrF(aKt,ffGeneral,2,2)+
      "\nКоэф. безаварийности: " +
```

```
    FloatToStrF(aKb,ffGeneral,2,2)+
    "\nКоэф. водительского стажа: " +
    FloatToStrF(aKvs,ffGeneral,2,2)+
    "\nКоэф. кол-ва лиц, допущеных к управлению: " +
    FloatToStrF(aKo,ffGeneral,2,2)+
    "\nКоэф. мощности двигател: " +
    FloatToStrF(aKm,ffGeneral,2,2)+
    "\nКоэф. периода использования ТС: " +
    FloatToStrF(aKs,ffGeneral,2,2)+
    "\n\nТариф: " + FloatToStrF(T,ffCurrency,5,2);

    ShowMessage(st);
}

// пользователь изменил состояние какого-либо из компонентов
формы
void __fastcall TForm1::Change(TObject *Sender)
{
    Button1->Enabled =
        (ComboBox1->ItemIndex != -1) &&
        (ComboBox2->ItemIndex != -1) &&
        (ComboBox3->ItemIndex != -1) &&
        (ComboBox4->ItemIndex != -1) &&
        (Edit1->Text.Length() != 0) &&
        (Edit2->Text.Length() != 0) &&
        (Edit3->Text.Length() != 0);
}
```

Просмотр иллюстраций

Программа **Просмотр иллюстраций** (рис. 1.10) демонстрирует использование компонентов `ListBox` и `OpenDialog`. Выбор каталога (папки) выполняется в стандартном окне **Открыть файл**, которое становится доступным в результате щелчка по кнопке **Выбор**. Отображение диалога осуществляет компонент `OpenDialog`.

Рис. 1.10. Выбор фотографии выполняется в списке компонента ListBox

```
#include <jpeg.hpp>      // обеспечивает работу
                         // с  иллюстрациями в формате JPEG

AnsiString aPath;        // каталог, в котором находится
                         // иллюстрация
TSearchRec aSearchRec; // рез-т поиска файла

// конструктор
__fastcall TForm1::TForm1(TComponent* Owner)
                        : TForm(Owner)
{
    Image1->Proportional = true;
    if (FindFirst(aPath+ "*.jpg", faAnyFile, aSearchRec) == 0)
    {
       ListBox1->Items->Add(aSearchRec.Name);
       while ( FindNext(aSearchRec) == 0 )// найти след. илл.
```

```
        {
            ListBox1->Items->Add(aSearchRec.Name);
        }
        // отобразить первую иллюстрацию
        ListBox1->ItemIndex = 0;
        Label1->Caption = ListBox1->Items->Strings[0];
        Image1->Picture->LoadFromFile(aPath +
                                ListBox1->Items->Strings[0]);
    }
}

// щелчок в строке компонента ListBox
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
    int n = ListBox1->ItemIndex; // номер выбранного эл-та
                                 // списка
    Label1->Caption = ListBox1->Items->Strings[n];
    Image1->Picture->LoadFromFile(aPath + ListBox1->
                                    Items->Strings[n]);
}

// щелчок на кнопке Выбор
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    if ( OpenDialog1->Execute() )
    {
        // пользователь выбрал файл
        ListBox1->Clear(); // очистить список

        aPath = ExtractFilePath(OpenDialog1->FileName);
        Form1->Caption = "Просмотр иллюстраций - " + aPath;
        if ( FindFirst(aPath+ "*.jpg", faAnyFile, aSearchRec)
                == 0)
```

```
        {
            ListBox1->Items->Add(aSearchRec.Name);
            while (FindNext(aSearchRec) == 0)//найти след.илл.
            {
                ListBox1->Items->Add(aSearchRec.Name);
            }
          // определим позицию выбранного пользователем файла
          // в списке ListBox и отобразим его
          int n = ListBox1->Items->
            IndexOf(ExtractFileName(OpenDialog1->FileName));
          ListBox1->ItemIndex = n;
          Label1->Caption = ListBox1->Items->Strings[n];
          Image1->Picture->LoadFromFile(aPath +
                                ListBox1->Items->Strings[n]);
        }
    }
}
```

Калькулятор

Программа **Калькулятор** (рис. 1.11) позволяет выполнить простейшие расчеты. Следует обратить внимание, что в качестве индикатора используется компонент `StaticText`.

Рис. 1.11. Простейший калькулятор

```
float accum; // аккумулятор (рез-т выполнения операции)
int op;      /* операция:
                1 - "+";
                2 - "-";
                0 - "выполнить" (нажата кнопка "=" */
int f;       /* f == 0 - ждем первую цифру нового числа,
                например, после выполнения
                операции, когда на индикаторе результат.
                f == 1 - ждем остальные цифры */

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    f =0;      // ждем первую цифру
    op = 0;    // предыдущая операция выполнена
    StaticText1->Caption = 0;
}

// кнопка "0"
void __fastcall TForm1::Btn0Click(TObject *Sender)
{
    if ( f != 0)
          StaticText1->Caption = StaticText1->Caption + "0";
}

// кнопка "1"
void __fastcall TForm1::Btn1Click(TObject *Sender)
{
 if ( f == 0)
    {
       StaticText1->Caption = "1";
       f = 1;
    }
```

```
    else
        StaticText1->Caption = StaticText1->Caption + "1";
}

// кнопка "2"
void __fastcall TForm1::Btn2Click(TObject *Sender)
{
 if ( f == 0)
    {
       StaticText1->Caption = "2";
       f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "2";
}

// кнопка "3"
void __fastcall TForm1::Btn3Click(TObject *Sender)
{
    if ( f == 0)
    {
       StaticText1->Caption = "3";
       f = 1;
    }
    else
       StaticText1->Caption = StaticText1->Caption + "3";
}

// кнопка "4"
void __fastcall TForm1::Btn4Click(TObject *Sender)
{
if ( f == 0)
    {
       StaticText1->Caption = "4";
```

```
        f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "4";
}

// кнопка "5"
void __fastcall TForm1::Btn5Click(TObject *Sender)
{
    if ( f == 0)
    {
        StaticText1->Caption = "5";
        f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "5";
}

// кнопка "6"
void __fastcall TForm1::Btn6Click(TObject *Sender)
{
    if ( f == 0)
    {
        StaticText1->Caption = "6";
        f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "6";
}

// кнопка "7"
void __fastcall TForm1::Btn7Click(TObject *Sender)
{
    if ( f == 0)
```

```
    {
        StaticText1->Caption = "7";
        f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "7";
}

// кнопка "8"
void __fastcall TForm1::Btn8Click(TObject *Sender)
{
    if ( f == 0)
    {
        StaticText1->Caption = "8";
        f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "8";
}

// кнопка "9"
void __fastcall TForm1::Btn9Click(TObject *Sender)
{
    if ( f == 0)
    {
        StaticText1->Caption = "9";
        f = 1;
    }
    else
        StaticText1->Caption = StaticText1->Caption + "9";
}

// кнопка "," (десятичная точка)
void __fastcall TForm1::BtnkClick(TObject *Sender)
```

```
{
    if ( f == 0)
    {
       StaticText1->Caption = "0,";
       f = 1;
    }
    else
    {
        if ( StaticText1->Caption.Pos(",") == 0)
            StaticText1->Caption = StaticText1->Caption + ",";
    }
}

// кнопка "C" (сброс)
void __fastcall TForm1::BtnCClick(TObject *Sender)
{
    StaticText1->Caption = "0";
    accum = 0;
    op = 0;
    f = 0; // ждем первую цифру числа
}

// выполнить операцию
void __fastcall TForm1::DoOp(void)
{
    /* accum содержит результат предыдущей операции.
       Сейчас надо выполнить операцию, код которой op.
       Операнд находится на индикаторе*/

    float op2 =  StrToFloat(StaticText1->Caption);
    switch ( op )
    {
        case 0 : accum = op2;  break;
        case 1 : accum += op2; break;
```

```
        case 2 : accum -= op2; break;
    }
    StaticText1->Caption = FloatToStrF(accum,ffGeneral,6,3);
}

// кнопка "+"
void __fastcall TForm1::BtnPClick(TObject *Sender)
{
    /* надо выполнить предыдущую операцию,
       вывести результат на индикатор, запомнить
       текущую операцию и установить режим ввода
       нового числа */
    if ( f != 0)
    {
        // на индикаторе есть число
        DoOp(); // выполнить предыдущую операцию
        f = 0;  // ждем первую цифру нового числа
    }
    op =1;
}

// кнопка "-"
void __fastcall TForm1::BtnMClick(TObject *Sender)
{
    if ( f != 0)
    {
        // на индикаторе есть число
        DoOp(); // выполнить предыдущую операцию
        f = 0;  // ждем первую цифру нового числа
    }
    op = 2; // запомнить текущую операцию
}

// кнопка "="
```

```
void __fastcall TForm1::BtnEClick(TObject *Sender)
{
    if ( f != 0)
    {
        DoOp(); // выполнить операцию
        f = 0;  // ждем первую цифру нового числа
    }
    op = 0;
}
```

Калькулятор-2

Программа **Калькулятор-2**, ее форма и окно приведены на рис. 1.12, демонстрирует создание компонентов во время работы программы или, как иногда говорят, "в коде". Кнопки калькулятора — это объединенные в массив компоненты `SpeedButton`. Создание и настройку кнопок выполняет конструктор формы, он же определяет (задает) процедуру обработки события `Click`. Следует обратить внимание, что объявление массива компонентов `SpeedButton` и функции обработки события `Click` находится в объявлении типа формы, в заголовочном файле.

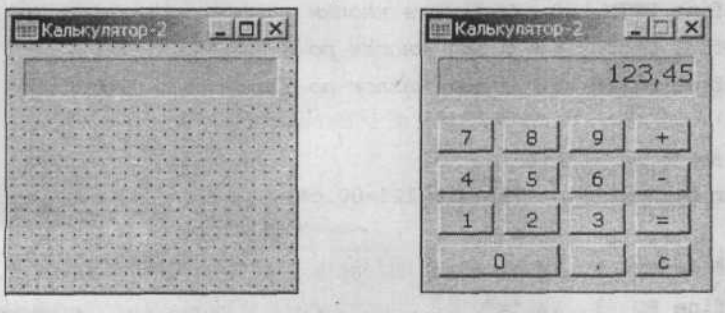


Рис. 1.12. Форма и окно программы **Калькулятор-2**

```
// объявление формы - фрагмент h-файла
class TForm1 : public TForm
{
__published:
    TStaticText *StaticText1; // индикатор
```

```
private:
    // ** эти объявления вставлены сюда вручную **
    TSpeedButton * btn[16];    // кнопки
    // процедура обработки события Click на кнопке
    void __fastcall btnClick(TObject *Sender);

public:
    __fastcall TForm1(TComponent* Owner);
};

// *** модуль формы ***
float ac;  // аккумулятор (первый операнд)
int op;    // операция
int fd;    /* fd == 0 - ждем первую цифру числа, например,
              после того, как была нажата клавиша "+";
              fd = 1 - ждем ввода следующей цифры или
              нажатия клавиши операции */

#define WBTN  35  // ширина кнопки
#define HBTN  20  // высота кнопки
#define DXBTN  6  // шаг кнопок по X
#define DYBTN  6  // шаг кнопок по Y

// текст на кнопке
Char btnText[] = "789+456-123=00.c";

#define CM -1  // запятая
#define EQ -2  // "="
#define PL -3  // "+"
#define MN -4  // "-"
#define CL -5  // "C"

// идентификатор кнопки
int  btnTag[]  = {7,8,9,PL,4,5,6,MN,1,2,3,EQ,0,0,CM,CL};
```

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    int left, top; // положение кнопки
    top = 48;

    int k = 0; // индекс массива btn
    for ( int i = 0; i < 4; i++ )  // четыре ряда кнопок
    {
        left = 8;
        for ( int j = 0;  j < 4; j++) // по 4 в каждом ряду
        {
            btn[k] = new TSpeedButton(Form1);
            btn[k]->Parent = Form1; // "посадить" кнопку на
                                    // форму
            // настройка кнопки
            btn[k]->Left = left;
            btn[k]->Top = top;
            btn[k]->Width = WBTN;
            btn[k]->Height = HBTN;
            //btn[k]->Flat = true;
            btn[k]->Font->Color = clNavy;

            btn[k]->Caption = btnText[k]; // текст на кнопке
            btn[k]->Tag = btnTag[k];    // идентификатор кнопки

            // задать процедуру обработки события Click
            btn[k]->OnClick = btnClick; // см. объявление в
                                        // h-файле формы
            k++;
            left = left + WBTN+ DXBTN;
        }
```

```
            top = top + HBTN + DYBTN;
        }
        // объединить две кнопки "0" (btn[12] и btn[13]) в одну
        btn[13]->Visible = false;
        btn[12]->Width = 2* WBTN + DXBTN;

        op = EQ;
}

// Щелчок кнопке btn[i]
// ( одна процедура для всех кнопок )
void __fastcall TForm1::btnClick(TObject *Sender)
{
    TSpeedButton *btn;
    int id; // идентификатор нажатой кнопки
    btn = (TSpeedButton*)Sender;

    // свойство Tag кнопки содержит ее идентификатор
    id = btn->Tag;
    // кнопка "1" .. "9"
    if ( id > 0 ) {
        if ( fd == 0 ) {
            StaticText1->Caption = btn->Tag;
            fd = 1;
        }
        else
            StaticText1->Caption =
                            StaticText1->Caption + btn->Tag;
        return;
    }
    // кнопка "0"
    if ( id == 0) {
        if ( StaticText1->Caption != "0" )
            StaticText1->Caption =
```

```
                        StaticText1->Caption + btn->Tag;
        return;
    }
    // кнопка ","
    if ( id == CM) {
        if ( StaticText1->Caption.Pos(",") == 0 )
        {
            StaticText1->Caption = StaticText1->Caption + ",";
            fd = 1;
        }
        return;
    }
    // кнопка "C"
    if ( id == CL) {
      ac = 0;
      id = EQ;
      fd = 0;
      StaticText1->Caption = 0;
      return;
  }
    // остальные кнопки: "+", "-" и "="
    float op2; // операнд (число на индикаторе )
    op2 = StrToFloat(StaticText1->Caption);

    /* так как нажата клавиша операции, то это значит,
       что надо выполнить предыдущую операцию, код которой
       находится в переменной op */
    switch (op) {
        case EQ : ac = op2;  break;
        case PL : ac = ac + op2; break;
        case MN : ac = ac - op2; break;
     }
     StaticText1->Caption = FloatToStrF(ac,ffGeneral,15,6);

    op = id;  // запомнить текущую операцию
```

```
    fd = 0;     // ждем новое число
}
// деструктор формы
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
    // уничтожить компоненты, созданные программой
    for ( int i = 0; i < 16; i++)
        delete btn[i];
}
```

Секундомер

Программа **Секундомер**, форма и окно которой приведены на рис. 1.13, демонстрирует использование компонента `Timer`.

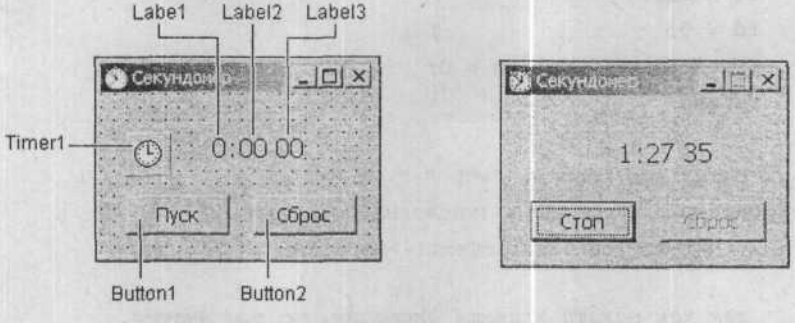


Рис. 1.13. Форма и окно программы **Секундомер**

```
// время
int min;    // минуты
int sec;    // секунды
int msec;   // миллисекунды

// конструктор
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
```

```
{
    // настройка таймера
    Timer1->Enabled = false;
    Timer1->Interval = 10;  // период сигналов от таймера
                            // 1 сек
}

// щелчок на кнопке Пуск/Стоп
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if ( ! Timer1->Enabled )
    {
        // запустить таймер
        Timer1->Enabled = true;
        Button1->Caption = "Стоп";
        Button2->Enabled = false;
    }
    else
    {
        // остановить таймер
        Timer1->Enabled = false;
        Button1->Caption = "Пуск";
        Button2->Enabled = true;
    }
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    if ( msec < 99)
        msec++;
    else
    {
```

```
        msec = 0;
        if ( sec < 59 )
            sec++;
        else
        {
            sec = 0;
            min++;
            Label1->Caption = IntToStr(min);
        }
        if ( sec <= 9 )
            Label2->Caption = "0" + IntToStr(sec);
        else
            Label2->Caption = IntToStr(sec);

        // двоеточие
        Label4->Visible = ! Label4->Visible;
    }
    if ( msec <= 9 )
            Label3->Caption = "0" + IntToStr(msec);
        else
            Label3->Caption = IntToStr(msec);
}

// щелчок на кнопке Сброс
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    min = 0;
    sec = 0;
    msec = 0;
    Label1->Caption = "0";
    Label2->Caption = "00";
    Label3->Caption = "00";
}
```

Угадай число

Программа **Угадай число** (рис. 1.14) демонстрирует использование компонента `ProgressBar`. Значения свойств компонента `ProgressBar` приведены в табл. 1.2.

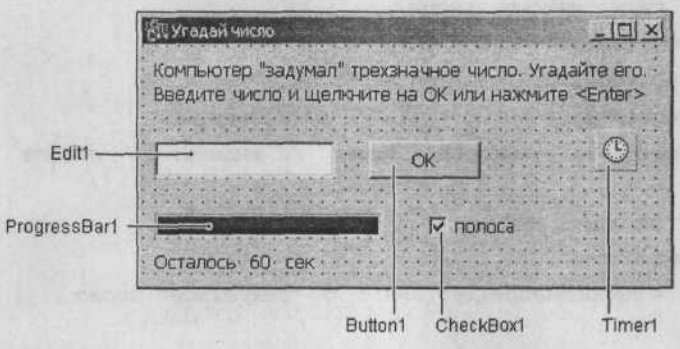


Рис. 1.14. Форма программы **Угадай число**

Таблица 1.2. *Значения свойств компонета* `ProgressBar`

Свойство	Значение
`Min`	0
`Max`	60
`Step`	1
`Position`	60
`Smooth`	`true`

```
#define TR  60 // время, отведенное на решение задачи

int pw;        // "секретное" число
int rem = TR; // остаток времени на выполнения задания

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
```

```
{
    Label4->Caption = IntToStr(TR);

    // ** настройка индикатора **
    // обратный отсчет, поэтому начальное значение
    // равно максимальному
    ProgressBar1->Max = TR;
    ProgressBar1->Position = TR;
    ProgressBar1->Step = 1;  // шаг изменения
    ProgressBar1->Smooth = true;  // индикатор - полоса

    // загадать число
    Randomize();
    pw = RandomRange(100,999); // "секретное" число
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    rem--;
    ProgressBar1->Position--;
    Label4->Caption = IntToStr(rem);

    if (rem == 0 )
    {
        // время, отведенное на решение задачи истекло
        Timer1->Enabled = false;
        Edit1->Enabled = false;
        Button1->Enabled = false;

        ShowMessage("К сожалению, Вы не справились с"
         " поставленной задачей\n \"Секретное\" число:" +
         IntToStr(pw) );
    }
}
```

```
// щелчок на переключателе Полоса
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
    if ( CheckBox1->Checked )
    {
        // полоса
        ProgressBar1->Step = 1;         // шаг изменения
        ProgressBar1->Smooth = true;    // индикатор - полоса
    }
    else
    {
        // сегменты
        ProgressBar1->Step = 6;         // шаг изменения
        ProgressBar1->Smooth = false;   // индикатор - сегменты
    }
}

// проверяет, правильное ли число ввел игрок
void __fastcall TForm1::isRight(void)
{
    if ( StrToInt(Edit1->Text) == pw )
        {
            Timer1->Enabled = false;
            Button1->Enabled = false;
            Edit1->Enabled = false;

            ShowMessage("Поздравляю!\nВы угадали число за " +
                        IntToStr(TR - rem)+ " сек");
        }
}
// Нажатие клавиши в поле редактирования
void __fastcall TForm1::Edit1KeyPress(TObject *Sender,
                                      char &Key)
{
    if ( ( Edit1->Text.Length() < 3) &&
```

```
            ((Key >= '0') &&
             (Key <= '9')))
            return;
        if ( Key == VK_RETURN)
        {
            isRight(); // проверить, правильное ли число ввел
                       // пользователь
            return;
        }
        if ( Key == VK_BACK) return;
        // остальные символы запрещены
        Key = 0;
}

// Содержимое поля редактирования изменилось
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
        // кнопка OK доступна только в том случае,
        // если в поле редактирования три символа (цифры)
        if ( Edit1->Text.Length() == 3)
            Button1->Enabled = true;
        else
            Button1->Enabled = false;
}

// щелчок на кнопке OK
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        isRight();  // проверить, правильное ли число ввел
                    // пользователь
}
```

Угадай число-2

Программа **Угадай число-2**, окно которой приведено на рис. 1.15, демонстрирует использование компонента `StatusBar`. В панелях

строки состояния отображается количество символов, введенных в поле редактирования, количество попыток угадать число и время, оставшееся на решение поставленной задачи.

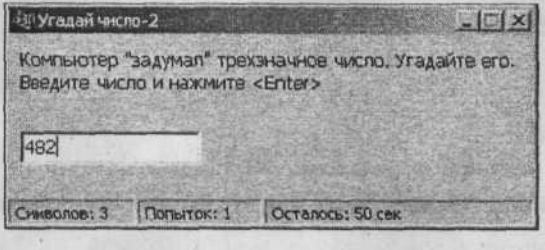


Рис. 1.15. В стоке состояния отображается информация

```
#define TR  60 // время, отведенное на решение задачи

int pw;        // "секретное" число
int rem = TR; // остаток времени на выполнения задания
int p = 0;     // количество попыток

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
    // загадать число
    Randomize();
    pw = RandomRange(100,999); // "секретное" число
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    rem--;
    StatusBar1->Panels->Items[2]->Text = " Осталось: " +
                                    IntToStr(rem) + " сек";
    if (rem == 0 )
    {
        // время, отведенное на решение задачи истекло
```

```
        Timer1->Enabled = false;
        Edit1->Enabled = false;
        ShowMessage("К сожалению, Вы не справились с"
        " поставленной задачей\n\"Секретное\" число:" +
        IntToStr(pw) );
    }
}

// Нажатие клавиши в поле редактирования
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char
&Key)
{
    if ( ( Edit1->Text.Length() < 3) && ( ( Key >= '0') &&
         ( Key <= '9') ) )
        return;

    if (( Key == VK_RETURN) && (Edit1->Text.Length() == 3))
    {
        // проверить, правильное ли число ввел пользователь
         if ( StrToInt(Edit1->Text) == pw )
        {
            Timer1->Enabled = false;
            Edit1->Enabled = false;
            ShowMessage("Поздравляю!\nВы угадали число за " +
                        IntToStr(TR - rem)+ " сек");
        }
        else
        {
            // увеличить счетчик попыток
            p++;
            StatusBar1->Panels->Items[1]->Text =
                " Попыток: " + IntToStr(p);
        }
        return;
    }
```

```
    if ( Key == VK_BACK) return;
    // остальные символы запрещены
    Key = 0;
}

// Содержимое поля редактирования изменилось
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    StatusBar1->Panels->Items[0]->Text =
        " Символов: " + IntToStr(Edit1->Text.Length());
}
```

Запуск Internet Explorer

Программа **Доступ в Internet** показывает, как запустить Internet Explorer или другой браузер для доступа к веб-странице. Форма программы приведена на рис. 1.16. Браузер запускается в результате щелчка в поле `Label1`. Значения свойств компонента `Label` приведены в табл. 1.3.

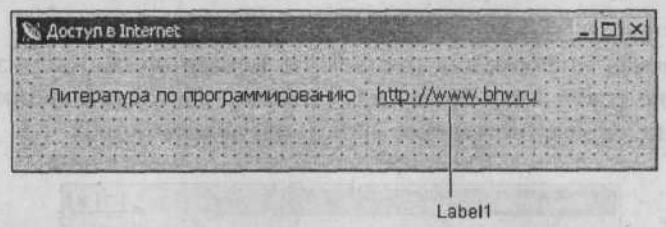


Рис. 1.16. Форма программы **Доступ в Internet**

Таблица 1.3. *Значения свойств компонента* `Label1`

Свойство	Значение
`Font.Color`	`clBlue`
`Font.Style.fsUnderline`	`true`
`Cursor`	`crHandPoint`

```
// щелчок в поле Label1
void __fastcall TForm1::Label1Click(TObject *Sender)
{
    // Открыть файл, имя которого находится в поле Label1
    ShellExecute(Form1->Handle,"open",Label1->Caption.c_str(),
                 NULL,NULL,SW_RESTORE);
}
```

Вывод справочной информации

Конвертор

Программа **Конвертор** (рис. 1.17) демонстрирует различные способы отображения справочной информации в формате Winhelp (файл с расширением hlp). Окно справки (рис. 1.18) появляется в результате нажатия клавиши <F1> или щелчка на кнопке **Справка**. Следует обратить внимание, что при нажатии клавиши <F1> в тот момент, когда курсор находится в одном из полей ввода, в окне справочной системы отображается раздел, поясняющий назначение того поля, в котором находится курсор.

Справочная информация программы **Конвертор** состоит из пяти разделов (табл. 1.4). Значения свойств, обеспечивающих отображение справочной информации, приведены в табл. 1.5.

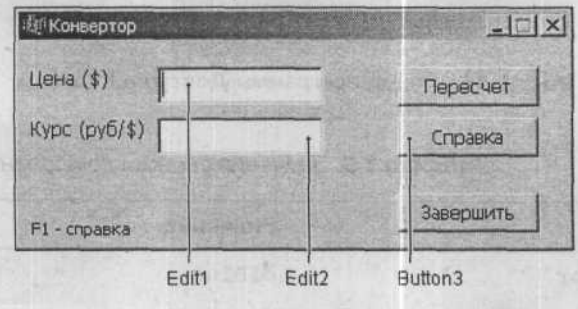


Рис. 1.17. Справочная информация отображается в результате щелчка на кнопке **Справка** или нажатия клавиши <F1>

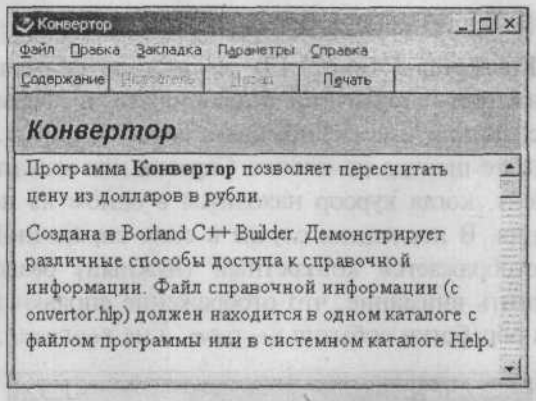


Рис. 1.18. Справочная информация в формате Winhelp

Таблица 1.4. *Разделы справочной информации программы* ***Конвертор***

Раздел	Идентификатор раздела
Конвертор	1
Исходные данные	2
Цена	3
Курс	4
Ввод чисел	5

Таблица 1.5. *Свойства, обеспечивающие вывод справочной информации*

Компонент	Свойство	Значение
Form1	HelpFile	conv.hlp
Form1	HelpContext	1
Edit1	HelpContext	3
Edit2	HelpContext	4

```
// щелчок на кнопке "Справка"
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    WinExec("winhlp32.exe conv.hlp", SW_RESTORE);
}
```

Конвертор-2

Программа **Конвертор-2** (рис. 1.19) демонстрирует различные способы отображения справочной информации в формате HTML Help (файл с расширением chm). Окно справки (рис. 1.20) появляется в результате щелчка на кнопке **Справка** или нажатия клавиши <F1> в момент, когда курсор находится в одном из полей ввода/редактирования. В последнем случае в окне справочной информации сразу отображается конкретный (нужный) раздел справки. Следует обратить внимание, что отображение справки активизирует процедура обработки события `KeyDown`, а не `KeyPress`.

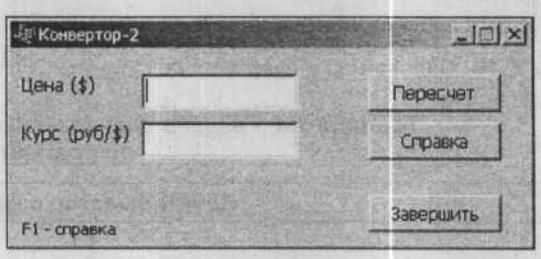


Рис. 1.19. Справочная информация отображается в результате щелчка на кнопке **Справка** или нажатия клавиши <F1>

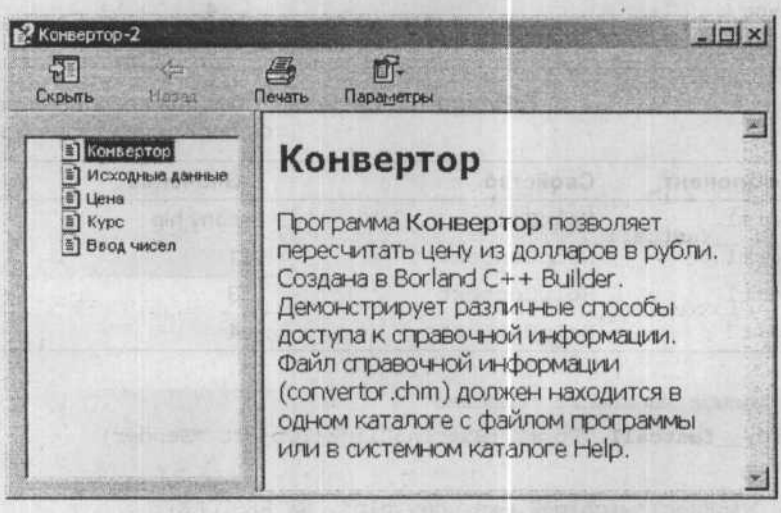
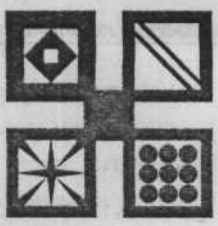


Рис. 1.20. Справочная информация в формате HTML Help

```
/* Нажатие клавиши в поле "Цена". Процедура
   обработки события KeyDown позволяет обрабатывать
   нажатие всех клавиш, в том числе и функциональных
   (при нажатии функциональной клавиши событие KeyPress
   не возникает) */
void __fastcall TForm1::Edit1KeyDown(TObject *Sender,
                          WORD &Key, TShiftState Shift)
{
    if (Key == VK_F1 )
     /* Отображение справочной информации
        обеспечивает утилита hh.exe, входящая
        в состав Windows. Ключ mapid задает
        отображаемый раздел справочной информации */
        WinExec("hh.exe -mapid 3 convertor.chm", SW_RESTORE);
}

// нажатие клавиши в поле "Курс"
void __fastcall TForm1::Edit2KeyDown(TObject *Sender,
                          WORD &Key, TShiftState Shift)
{
    if (Key == VK_F1 )
        WinExec("hh.exe -mapid 4 convertor.chm", SW_RESTORE);
}

// щелчок на кнопке "Справка"
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    WinExec("hh.exe -mapid 1 convertor.chm", SW_RESTORE);
}
```



Файлы

В этом разделе приведены программы, которые демонстрируют выполнение различных операций с файлами.

Погода

Программа **Погода**, форма которой приведена на рис. 1.21, добавляет в базу данных, представляющую собой текстовый файл, информацию о температуре воздуха. Каждая строка файла данных meteo.txt содержит дату и значение температуры. Если файла данных в текущем каталоге нет, то программа создает его. Программа спроектирована так, что кнопка **OK** доступна только в том случае, если поле **Температура** содержит данные.

Рис. 1.21. Форма программы **Погода**

```
int f; // дескриптор файла

// конструктор
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    MonthCalendar1->ShowTodayCircle = false;
    MonthCalendar1->Date = Now();
    Button1->Enabled = false;
}
// изменилось содержимое поле редактирования
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    // кнопка "Добавить" доступна, если в поле редактирования
    // находится число. Если в поле символов нет или первый
    // символ - "минус", кнопка недоступна
    if ( (Edit1->Text.Length() == 0 ) ||
          ( (Edit1->Text.Length() == 1) &&
            (Edit1->Text[1] == '-') ) )
        Button1->Enabled = false;
    else
        Button1->Enabled = true;
}
// щелчок на кнопке "Добавить"
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    AnsiString st;

    /* файл можно открыть в режиме fmCreate, тогда, если
       файл существует, он будет открыт для записи,
       если файла нет, то он будет создан */
```

```
    st = MonthCalendar1->Date.DateString() + " " +
                                    Edit1->Text + "\r\n";

    /* открыть для записи или создать файл meteo.txt */
    if ( FileExists("meteo.txt") )
        f = FileOpen("meteo.txt",fmOpenWrite);
    else
       f = FileCreate("meteo.txt"); // создать файл

    if ( f != -1 )
    {
        // файл открыт для записи
        FileSeek(f,0,2); // установить указатель на конец
                         // файла
        FileWrite(f,st.c_str(),st.Length());
        FileClose(f);
        Button1->Enabled = false;
    }
    else
    {
        /* ошибка доступа к файлу: ни открыть,
           ни создать не получилось */
        ShowMessage("Ошибка доступа к файлу: ни открыть,"
                    "ни создать не получилось");
    }
}

// в поле редактирования можно ввести только
// положительное или отрицательное число
void __fastcall TForm1::Edit1KeyPress(TObject *Sender,
                                      char &Key)
{
```

```
    if (( Key >= '0') && (Key <= '9'))
        return;

    // десятичная точка (запятая)
    if ( ( Key == '.') || (Key == ','))
    {
        Key = ',';
        if ( Edit1->Text.Pos(',') != 0 )
            Key = 0;
        return;
    }

   if ( Key == 8)
        return;
   if ((Key == '-') && (Edit1->Text.Length() == 0))
        // "минус" может быть только первым символом
        return;
   // все остальные символы запрещены
   Key = 0;
}
// щелчок в поле компонента MonthCalendar
void __fastcall TForm1::MonthCalendar1Click(TObject *Sender)
{
    Edit1->Text = "";     // очистить поле ввода температуры
    Edit1->SetFocus();    // установить курсор в поле ввода
                          // температуры
}
```

Средняя температура

Программа **Среднемесячная температура** (рис. 1.22) демонстрирует чтение данных из текстового файла. Программа выводит в поле компонента Memo содержимое, сформированного программой **Погода** файла meteo.txt, а также выполняет его обработку — вычисляет среднемесячную температуру.

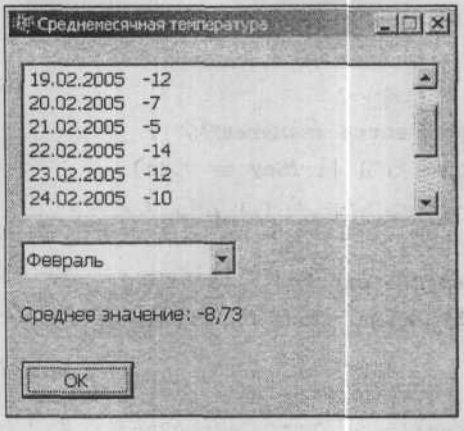


Рис. 1.22. Окно программы **Среднемесячная температура**

```
#include "DateUtils.hpp"  // для доступа к MonthOf

/* Функция GetString читает из файла строку символов
   от текущей позиции до первого пробела.
   значение функции - кол-во прочитанных символов */
int GetString(int f, AnsiString *st);

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    ComboBox1->Style = csDropDownList;
    ComboBox1->Items->Add("Январь");
    ComboBox1->Items->Add("Февраль");
    ComboBox1->Items->Add("Март");
    ComboBox1->Items->Add("Апрель");
    ComboBox1->Items->Add("Май");
    ComboBox1->Items->Add("Июнь");
    ComboBox1->Items->Add("Июль");
    ComboBox1->Items->Add("Август");
```

```
    ComboBox1->Items->Add("Сентябрь");
    ComboBox1->Items->Add("Октябрь");
    ComboBox1->Items->Add("Ноябрь");
    ComboBox1->Items->Add("Декабрь");
    // эл-ты списка нумеруются с нуля
    ComboBox1->ItemIndex = MonthOf( Now() ) -1 ;
}
// щелчок на кнопке OK
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int h; // дескриптор файла
    h = FileOpen("meteo.txt",fmOpenRead);
    if ( h == -1)
    {
        ShowMessage("Файл данных meteo.txt не найден.");
        return;
    }

    Memo1->Lines->Clear();
    // Обработка файла
    AnsiString st; // строка, прочитанная из файла
    int ls;          // длина строки

    TDateTime aDate;  // дата
    float temp;       // температура
    int nMonth;       // месяц

    int n = 0;      // количество дней (прочитанных строк)
    float sum = 0; // сумма температур
    float sred;     // среднее значение

    AnsiString buf;

    /* каждая строка файла имеет вид:
```

```
    ДД.ММ.ГГГГ Т
    где: ДД.ММ.ГГГГ - дата; Т - температура */

  do
  {
     ls = GetString(h,&st); // дата
     if ( ls != 0)
     {
        nMonth = MonthOf (StrToDate(st)) - 1; // месяц
        buf = st;

        ls = GetString(h,&st); // температура
        temp = StrToFloat(st);
        if ( nMonth == ComboBox1->ItemIndex )
        {
           n++;
           sum = sum + temp;
           buf = buf + "    " + st;
           Memo1->Lines->Add(buf);
        }
     }
  } while ( ls != 0);
  FileClose(h);
  if ( n != 0 )
  {
     sred = sum / n;
     Label1->Caption = "Среднее значение: " +
                  FloatToStrF(sred,ffGeneral,3,2);
  }
  else
     Label1->Caption = "В БД нет информации о "
                  "температуре за " + ComboBox1->Text;
}
```

```
// пользователь выбрал другой месяц
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
    Label1->Caption = "";
    Memo1->Text = "";
}

// Функция GetString читает из файла строку символов
// от текущей позиции до первого пробела.
// значение функции - кол-во прочитанных символов
int GetString(int f, AnsiString *st)
{
    unsigned char buf[256]; // строка (буфер)
    unsigned char *p = buf;  // указатель на строку

    int n;  // кол-во прочитанных байт (значение ф-и FileRead)
    int len = 0; // длина строки

    // удалить ведущие пробелы
    do
        n = FileRead(f, p, 1);
    while ((n != 0) && (*p == ' '));

    while (( n != 0 ) && ( *p != ' '))
    {
        if ( *p == '\r')
        {
            n = FileRead(f, p, 1); // прочитать '\n'
            break;
        }
        len++;
        p++;
        n = FileRead(f, p, 1);
    }
```

```
    *p = '\0';
    if ( len !=0 )
        st->printf("%s", buf);
    return len;
}
```

Простая база данных

Программа **Расходы** обеспечивает работу с базой данных, которая представляет собой текстовый файл. Для редактирования и просмотра данных используется компонент `StringGrid` (рис. 1.23). В начале работы данные автоматически загружаются в компонент `StringGrid` из файла, в конце — записываются в файл.

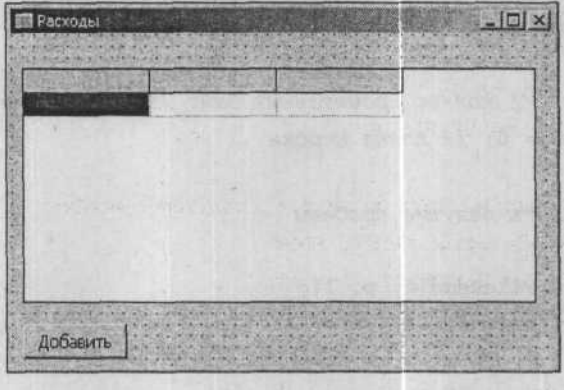


Рис. 1.23. Компонент `StringGrid` используется для просмотра и редактирования данных, которые находятся в текстовом файле

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    // *** настроить таблицу ***
    StringGrid1->Options
            << goEditing  // разрешить редактировать
            << goTabs;    // <Tab> - переход к следующей ячейке
```

```
    // заголовки столбцов
    StringGrid1->Cells[0][0] = "Дата";
    StringGrid1->Cells[1][0] = "Сумма";
    StringGrid1->Cells[2][0] = "Комментарий";

    // ширина столбцов
    StringGrid1->ColWidths[0] = 70;
    StringGrid1->ColWidths[1] = 70;
    StringGrid1->ColWidths[2] = 200;
}

// нажатие клавиши в ячейке таблицы
void __fastcall TForm1::StringGrid1KeyPress(TObject *Sender,
char &Key)
{
    int cRow, cCol;
    if ( Key == VK_RETURN )
    {
        // переместить курсор в следующую ячейку
        cRow = StringGrid1->Row;
        cCol = StringGrid1->Col;
        if ( cCol < StringGrid1->ColCount - 1 )
            StringGrid1->Col = StringGrid1->Col + 1;
        else
            if ( cRow < StringGrid1->RowCount - 1 )
            {
                StringGrid1->Row = StringGrid1->Row + 1;
                StringGrid1->Col = 1;
            }
    }
}

// добавить строку
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
    // добавить строку в таблицу
    StringGrid1->Row = StringGrid1->RowCount-1;  // перейти к
                                                 // последней строке
    StringGrid1->RowCount++;                     // добавить строку
    StringGrid1->Row =
       StringGrid1->RowCount-1;  // перейти к последней строке
}

// завершение работы программы
void __fastcall TForm1::FormCloseQuery(TObject *Sender, bool
&CanClose)
{
    int f; // дескриптор файла

      /* файл можно открыть в режиме fmCreate, тогда, если
       файл существует, он будет открыт для записи,
       если файла нет, то он будет создан */

    if ( FileExists("tabl.grd") )
        f = FileOpen("tabl.grd",fmOpenWrite);
    else
       f = FileCreate("tabl.grd");

    if ( f != -1 )
    {
        // сохранить таблицу в файле
        for (int i = 1; i < StringGrid1->RowCount; i++)
        {
             AnsiString st = StringGrid1->
                    Rows[i]->DelimitedText +"\r\n";
            FileWrite(f,st.c_str(), st.Length());
        }
        FileClose(f);
    }
```

```
    else
        // ошибка доступа к файлу
        ShowMessage("Ошибка доступа к файлу");
}

int GetLine(int f, AnsiString *st); // читает строку из файла

// загружает таблицу из файла
void __fastcall TForm1::FormActivate(TObject *Sender)
{
    int f;          // дескриптор файла
    AnsiString st;  // прочитанная строка
    bool fl = true; // true - чтение первой строки

    if (( f = FileOpen("tabl.grd",fmOpenRead)) == -1 )
        return;

    // файл открыт
    // загрузить таблицу
    while ( GetLine(f, &st) != 0)
    {
        // добавить строку в таблицу
        if ( fl )
        {
            StringGrid1->Rows[StringGrid1->Row]->
                                    DelimitedText = st;
            fl = false;
        }
        else
        {
            StringGrid1->RowCount++;
            StringGrid1->Row = StringGrid1->RowCount-1;
            StringGrid1->Rows[StringGrid1->Row]->
                                    DelimitedText = st;
```

```
        }
    }
    FileClose(f);
}

// читает из файла строку символов
// от текущей позиции до символа "новая строка"
// значение функции - кол-во прочитанных символов
int GetLine(int f, AnsiString *st)
{
    unsigned char buf[256]; // строка (буфер)
    unsigned char *p = buf;  // указатель на строку

    int n;  // кол-во прочитанных байт (значение ф-и FileRead)
    int len = 0; // длина строки

    n = FileRead(f, p, 1);
    while ( n != 0 )
    {
        if ( *p == '\r')
        {
            n = FileRead(f, p, 1); // прочитать '\n'
            break;
        }
        len++;
        p++;
        n = FileRead(f, p, 1);
    }

    *p = '\0';
    if ( len !=0)
        st->printf("%s", buf);
    return len;
}
```

Редактор текста

Программа **MEdit** (Micro Editor) демонстрирует использование компонентов `RichEdit`, `MainMenu`, `ToolBar`, `SpeedButton`, `OpenDialog` и `SaveDialog`, а также отображение информации о программе в отдельном окне. Главная форма программы приведена на рис. 1.24, значения свойств компонента `RichEdit` — в табл. 1.6. Информация о программе отображается в окне **О программе MEedit**, которое появляется в результате выбора соответствующей команды в меню **Справка**. Форма **О программе MEedit** приведена на рис. 1.25.

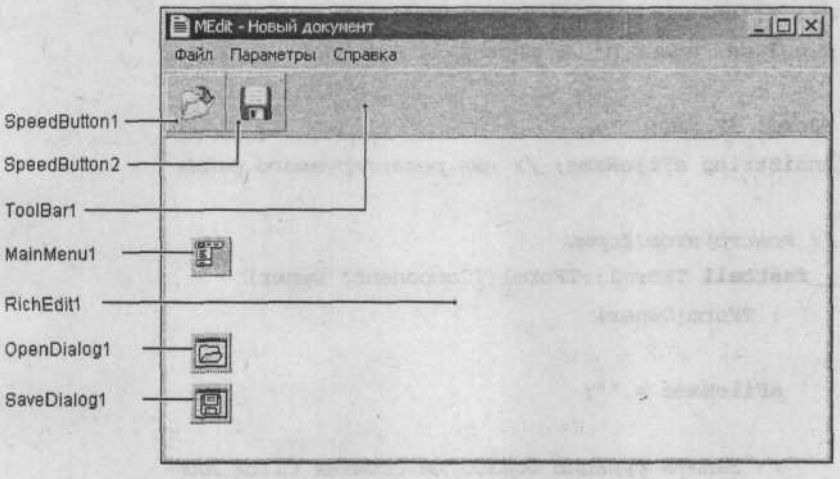


Рис. 1.24. Форма программы **MEdit**

Таблица 1.6. *Значения свойств компонента* `RichEdit`

Свойство	Значение
`Align`	`alClient`
`BorderStyle`	`bsNone`
`ScrollBars`	`ssVertical`
`WordWrap`	`true`

Рис. 1.25. Форма **О программе MEedit**

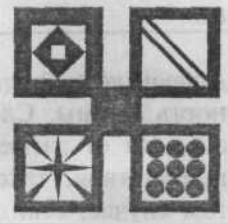
```
//  ***модуль главной формы (MEditFrm.cpp) ***
#include "about.h" // директива вставлена вручную

TForm1 *Form1;
AnsiString aFileName; // имя редактируемого файла

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    aFileName = "";

    /* Задать функцию обработки события Click для
       кнопок панели инструментов
       Внимание! Сначала надо создать функции обработки
       события Click для команд Файл/Открыть и Файл/Сохранить,
       затем - вставить в текст следующие две инструкции. */
    SpeedButton1->OnClick = imOpenClick;
    SpeedButton2->OnClick = imSaveClick;
}

// сохранить текст в файле
bool __fastcall TForm1::SaveText()
{
```

```
    // значение функции SaveText равно false, если
    // в окне Сохранить пользователь сделал щелчок
    // на кнопке Отменить
    if ( aFileName == "" )
    {
        // получить у пользователя имя файла
        //SaveDialog1->FileName = "Text.txt";
        SaveDialog1->Options << ofPathMustExist
                             << ofOverwritePrompt;
        if ( SaveDialog1->Execute() )
            aFileName = SaveDialog1->FileName;
        else
           return false; // пользователь закрыл окно
                         // щелчком на кнопке Отменить
    }
    // записать текст в файл
    RichEdit1->Lines->SaveToFile(aFileName);
    return true;
}

// команда Файл/Открыть
void __fastcall TForm1::imOpenClick(TObject *Sender)
{
    int r;

    // если текст изменен, то его надо сохранить
    if ( RichEdit1->Modified )
    {
        // текст изменен
        r = MessageDlg("Текст был изменен.\n"
                       "Сохранить изменения?",
                   mtWarning, TMsgDlgButtons() << mbYes
                   << mbNo << mbCancel, 0);

          if ( r == mrCancel )
```

```
            // в окне сообщения пользователь Cancel
            // щелкнул на кнопке return;

        // здесь пользователь выбрал Yes
        if ( ( r == mrYes ) && ( ! SaveText() ))
            return;
    }

    // открыть файл
    OpenDialog1->FileName = "*.txt";
    OpenDialog1->Options << ofPathMustExist
                         << ofFileMustExist;

    if ( OpenDialog1->Execute() )
    {
        RichEdit1->Lines->LoadFromFile(OpenDialog1->FileName);
        RichEdit1->Modified = false;
        Form1->Caption = "MEdit - " +  OpenDialog1->FileName;
        aFileName = OpenDialog1->FileName;
    }
}

// команда Файл/Сохранить
void __fastcall TForm1::imSaveClick(TObject *Sender)
{
    // записать текст в файл
    if ( SaveText() )
    {
        RichEdit1->Modified = false;
        Form1->Caption = "MEdit - " + aFileName;
    }
}

// команда Файл/Выход
```

```
void __fastcall TForm1::imExitClick(TObject *Sender)
{
     Form1->Close();
     /* возникает событие FormCloseQuery,
        функция которого проверяет, внесены ли
        изменения в текст */
}

// пользователь сделал щелчок на кнопке "Закрыть окно"
void __fastcall TForm1::FormCloseQuery(TObject *Sender,
                                       bool &CanClose)
{
    int r;
    if ( RichEdit1->Modified ) // текст изменен
    {
        r = MessageDlg("Текст был изменен.\n"
                       "Сохранить изменения?",
                        mtWarning, TMsgDlgButtons()
                        << mbYes << mbNo << mbCancel, 0);

        switch ( r )
        {
            case mrYes : // пользователь щелкнул на кнопке Yes
                if ( ! SaveText() )
                    CanClose = false;  // окно не закрывать
                break;

            case mrCancel : // пользователь щелкнул на Cancel
                    CanClose = false; // продолжить работу
                                      // с текстом
        }
     }
}
```

```
// команда Параметры/Панель инструментов
void __fastcall TForm1::imToolBarClick(TObject *Sender)
{
    // скрыть/отобразить панель инструментов
    ToolBar1->Visible = ! ToolBar1->Visible;

    // поставить/убрать "галочку" в меню Параметры/Панель
    // инструментов
    imToolBar->Checked = ! imToolBar->Checked;
}

// команда Справка/О программе
void __fastcall TForm1::imAboutClick(TObject *Sender)
{
    TAboutForm *AboutForm;
     // свойству Position формы AboutForm надо присвоить
     // значение poOwnerForm
    AboutForm = new TAboutForm (this);
    AboutForm->ShowModal();
    delete AboutForm;
}

/*  Модуль формы "О программе" (about.cpp).
    Ссылку на этот модуль (директиву #include "about.h")
    надо поместить в модуль главной формы (MEditFrm.cpp) */
// щелчок на кнопке OK
void __fastcall TAboutForm::Button1Click(TObject *Sender)
{
    ModalResult = mrOk;
}
```



Графика

В этом разделе собраны программы, которые демонстрируют работу с графикой.

Общие замечания

- Программа может выводить графику на *поверхность* объекта (формы или компонента `Image`), которой соответствует свойство (объект) `Canvas`.
- Для того чтобы на поверхности объекта появился графический элемент (линия, окружность, прямоугольник и т. д.) или картинка, необходимо применить соответствующий метод к свойству `Canvas` этого объекта.
- Цвет, стиль и толщину линий, вычерчиваемых методами `Line`, `Ellipse`, `Rectangle` и т. д., определяет свойство `Pen` объекта `Canvas`.
- Цвет закраски внутренних областей геометрических фигур, вычерчиваемых методами `Line`, `Ellipse`, `Rectangle` и т. д., определяет свойство `Brush` объекта `Canvas`.
- Характеристики шрифта текста, выводимого методом `TextOut` определяет свойство `Font` объекта `Canvas`.
- Основную работу по выводу графики на поверхность формы должна выполнять функция обработки события `OnPaint`.

Приветствие

Программа **Приветствие** в зависимости от времени суток выводит приветствие **Доброе утро**, **Добрый день**, **Добрый вечер** или

Доброй ночи (рис. 1.26). Демонстрирует вывод текста на поверхность формы. Следует обратить внимание на то, что вне зависимости от размера шрифта, который используется для вывода приветствия, текст всегда размещается в центре окна, даже в том случае, если пользователь во время работы программы изменит размер окна.

Рис. 1.26. Вне зависимости от размера формы и шрифта текст приветствия находится в центре окна

```
Graphics::TBitmap *bgp; // фоновый рисунок

AnsiString sMonth[] = {"","января","февраля","марта",
      "апреля","мая","июня",
      "июля","августа","сентября",
      "октября","ноября","декабря"};

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    bgp = new Graphics::TBitmap();
    try
    {
        bgp->LoadFromFile("sky.bmp");
    }
```

```
    catch (EFOpenError &e)
    {    }
}

// обработка события Paint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    int h;              // текущее время (часы)
    AnsiString mes;     // сообщение
    int wt,ht;          // размер (ширина и высота) области
                        // вывода текста
    int x,y;            // координаты области вывода текста

    h = HourOf( Now() );

    if ( h <= 4 ) mes = "Доброй ночи!";
    else if ( h < 12 ) mes = "Доброе утро!" ;
    else if ( h <= 16) mes = "Добрый день!" ;
    else mes = "Добрый вечер!";

    Form1->Font->Name = "Times New Roman";
    //Form1->Font->Color = clBlue;
    Form1->Canvas->Font->Size = 20;

    // определить размер области вывода текста
    wt = Canvas->TextWidth(mes);
    ht = Canvas->TextHeight(mes);

    x = (ClientWidth - wt) / 2;
    y = ClientHeight / 2 - ht;

    // фоновая картинка
    Canvas->Draw(0,0,bgp);

    // чтобы область вывода текста была прозрачной
```

```
    Canvas->Brush->Style = bsClear;

    Canvas->TextOutA(x,y,mes);

    y = y + ht; // координата нижней границы
                // области вывода текста

    // дата и день недели
    mes = FormatDateTime("Сегодня d", Now() );
    mes = mes + " " + sMonth[MonthOf( Now())] + ", " +
                FormatDateTime("dddd", Now() );

    /* размер шрифта вывода даты на 4 пункта меньше
       размера шрифта приветствия */
    Canvas->Font->Size -= 4;
    wt = Canvas->TextWidth(mes);
    ht = Canvas->TextHeight(mes);

    x = (ClientWidth - wt) / 2;
    y = y + 6;

    Canvas->TextOutA(x,y,mes);
}

// пользователь изменил размер формы
void __fastcall TForm1::FormResize(TObject *Sender)
{
    Form1->Refresh(); // перерисовать окно
}
```

Олимпийский флаг

Программа **Олимпийский флаг** (рис. 1.27) демонстрирует вывод графики на поверхность формы.

Рис. 1.27. Окно программы **Олимпийский флаг**

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    Canvas->Font->Name = "Tahoma";
    Canvas->Font->Size = 12;
}

void __fastcall TForm1::FormPaint(TObject *Sender)
{
    #define WB 140  // ширина полотнища
    #define HB 80     // высота полотнища
    #define D  36      // диаметр колец

    int x,y;
    AnsiString st = "Быстрее, выше, сильнее!";

    // вычислить координаты левого верхнего угла флага
    x = (ClientWidth -  WB) / 2;
    y = (ClientHeight - HB ) / 2 - Canvas->Font->Size;

    // полотнище
    Canvas->Brush->Color = (TColor) RGB(255,255,255);
    Canvas->FillRect( Rect(x,y,x+WB,y+HB) );
```

```
    int x1 = (ClientWidth - Canvas->TextWidth(st)) / 2;

/* Чтобы область вывода текста не была закрашена цветом фона,
   а также чтобы метод Ellipse рисовал окружность, а не круг,
   значение свойства Brush->Style должно быть равно bsClear */
    Canvas->Brush->Style = bsClear;

    // девиз
    Canvas->TextOutA(x1,y+HB+6,st);

    Canvas->Pen->Width = 2; // ширина колец - два пиксела

    // первый ряд колец
    // 3.2 *  D - ширина области, занимаемой кольцами 1-го
    // ряда
    x = x + (WB - 3.2 * D) / 2;
    y = y + (HB - 1.6 * D) / 2;

    Canvas->Pen->Color = (TColor) RGB(0,0,225);  // синий
    Canvas->Ellipse(x,y,x+D,y+D);

    x = x + 1.1* D;
    Canvas->Pen->Color = clBlack;        // черный
    Canvas->Ellipse(x,y,x+D,y+D);

    x = x + 1.1 * D;
    Canvas->Pen->Color = (TColor) RGB(255,0,0);  // красный
    Canvas->Ellipse(x,y,x+D,y+D);

    // второй ряд колец
    x = x - D * 0.55;
    y = y + 0.6 * D;

    Canvas->Pen->Color = (TColor) RGB(0,128,0);  // зеленый
    Canvas->Ellipse(x,y,x+D,y+D);
```

```
    x = x - 1.1 * D;
    Canvas->Pen->Color = (TColor) RGB(250,217,25);  // желтый
    Canvas->Ellipse(x,y,x+D,y+D);
}

// пользователь изменил размер окна
void __fastcall TForm1::FormResize(TObject *Sender)
{
    Form1->Refresh();
}
```

Диаграмма

В окне программы **Диаграмма** (рис. 1.28) отображается диаграмма, которая иллюстрирует изменение курса доллара. Программа спроектирована так, что, независимо от размера окна, диаграмма располагается в центре окна и занимает бóльшую его часть, даже после того, как пользователь изменит размер окна.

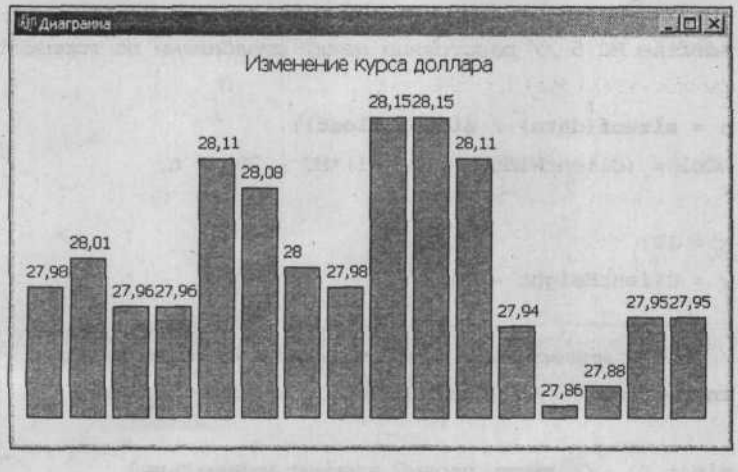


Рис. 1.28. Окно программы **Диаграмма**

```
float data[] = {27.98, 28.01, 27.96, 27.96, 28.11, 28.08,
                28.00, 27.98, 28.15, 28.15, 28.11, 27.94,
                27.86, 27.88, 27.95, 27.95};
```

```
AnsiString Title = "Изменение курса доллара";

void __fastcall TForm1::FormPaint(TObject *Sender)
{
    int x,y;

    // заголовок
    Canvas->Font->Name = "Tahoma";
    Canvas->Font->Size = 12;
    x = (ClientWidth - Canvas->TextWidth(Title)) /2;
    Canvas->Brush->Style = bsClear;
    Canvas->TextOutA(x,10,Title);

    // гистограмма
    int n;     // количество столбцов
    int wCol; // ширина столбца

    #define MC 5 // расстояние между столбиками по горизонтали

    n = sizeof(data) / sizeof(float);
    wCol = (ClientWidth - (n - 1)*MC - 20) / n;

    x = 10;
    y = ClientHeight - 20;

    // найти минимальное и максимальное значение данных
    int min,max;  // индекс миним. и максим. элемента

    min = 0;  // пусть первый элемент минимальный
    max = 0;  // пусть первый элемент максимальный
    for (int i = 1; i < n; i++)
    {
        if (data[i] < data[min]) min = i;
```

```
    if (data[i] > data[max]) max = i;
}

/* Если отклонение значений ряда от среднего значения
   незначительное, то диаграмма получается ненаглядной.
   В этом случае можно построить не абсолютные значения,
   а отклонения от минимального значения ряда. */

bool frmin = true; // true - отсчитывать от минимального
int h; // высота столбика
// максимальному значению соответствует
// столбик высотой ClientHeight - 90

Canvas->Font->Size -= 2;
for ( int i = 0; i < n; i++)
{
    if (! frmin)
        h = (ClientHeight - 90) * data[i]/data[max];
    else
        /* Отсчитывать от минимального значения.
           Минимальное значение отобразим столбиком
           высотой 10 пикселов */
        h = (ClientHeight - 90) * (data[i] -
            data[min])/(data[max] - data[min]) + 10;

    Canvas->Brush->Style = bsSolid;
    Canvas->Brush->Color = clLime;
    Canvas->Rectangle(x,y,x+wCol,y-h); // столбик

    // подпись данных
    AnsiString st;
    st = FloatToStrF(data[i],ffGeneral,5,2);
    Canvas->Brush->Style = bsClear; // область вывода
                                    // текста - прозрачная
```

```
        Canvas->TextOutA(x,y-h-20,st);

        x = x + wCol + MC;
    }
}

// изменился размер окна
void __fastcall TForm1::FormResize(TObject *Sender)
{
    Form1->Refresh(); // обновить содержимое окна
}
```

График

В окне программы **График** (рис. 1.29) отображается график изменения курса доллара. Следует обратить внимание, что на графике отображаются не значения, а отклонение от минимального значения ряда. Это делает график наглядным даже в том случае, если разброс значений (разница между минимальным и максимальным значениями) ряда незначительный.

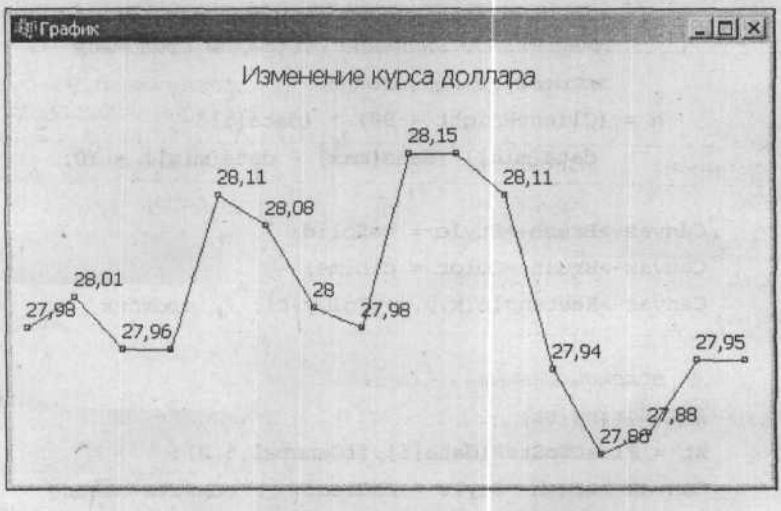


Рис. 1.29. Окно программы **График**

```
float data[] = {27.98, 28.01, 27.96, 27.96, 28.11, 28.08,
28.00, 27.98, 28.15,28.15, 28.11, 27.94, 27.86, 27.88, 27.95,
27.95};

AnsiString Title = "Изменение курса доллара";

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm1::FormPaint(TObject *Sender)
{
    int n;       // количество точек
    int x,y;     // координаты точки
    int dx;      // шаг по X

    int x0,y0;   // координаты левого нижнего угла области
                 // построения графика

    // заголовок
    Canvas->Font->Name = "Tahoma";
    Canvas->Font->Size = 12;
    x0 = (ClientWidth - Canvas->TextWidth(Title)) /2;
    Canvas->Brush->Style = bsClear;
    Canvas->TextOutA(x0,10,Title);

    n = sizeof(data) / sizeof(float);

    // найти минимальное и максимальное значение данных
    int min,max;  // индекс миним. и максим. элемента

    min = 0;  // пусть первый элемент минимальный
    max = 0;  // пусть первый элемент максимальный
    for (int i = 1; i < n; i++)
```

```
{
    if (data[i] < data[min]) min = i;
    if (data[i] > data[max]) max = i;
}

/* Если отклонение значений ряда от среднего значения
   незначительное, то диаграмма получается ненаглядной.
   В этом случае можно построить не абсолютные значения,
   а отклонения от минимального значения ряда. */

bool frmin = true; // true - отсчитывать от минимального
                   // значения

dx= (ClientWidth - 20) / (n-1);

Canvas->Font->Size -= 2;
Canvas->Pen->Color = clGreen;
Canvas->Pen->Width = 1;

x0 = 10;
y0 = ClientHeight - 10;

x = x0;
dx= (ClientWidth - 20) / (n-1);
for ( int i = 0; i < n; i++)
{
    /* максимальному значению соответствует
       точка с координатой ClientHeight - 90 */
    if (! frmin)
        y = y0 + (ClientHeight - 90) * data[i]/data[max];
    else
        // Отсчитывать от минимального значения
        y = y0-(ClientHeight - 90) * (data[i] -
                data[min])/(data[max] - data[min])-10;
```

```
        // поставить точку
        Canvas->Rectangle(x-2,y-2,x+2,y+2);

        if (i != 0)
            Canvas->LineTo(x,y);

        // ** подпись данных **
        /* т.к. метод TextOutA изменит положение точки
           привязки (точки, из которой рисует метод LineTo, то
           после вывода текста надо будет переместить
           указатель в точку (x,y) */
        if ( ( i == 0) || (data[i] != data[i-1]))
        {
            AnsiString st;
            st = FloatToStrF(data[i],ffGeneral,5,2);
            Canvas->Brush->Style = bsClear;  // область вывода
                                             // текста - прозрачная
            Canvas->TextOutA(x,y-20,st);
        }
        Canvas->MoveTo(x,y);
        x += dx;
    }
}

// изменился размер окна
void __fastcall TForm1::FormResize(TObject *Sender)
{
    Form1->Refresh(); // обновить содержимое окна
}
```

Круговая диаграмма

В окне программы **Круговая диаграмма** (рис. 1.30) отображается диаграмма, которая отображает долю каждой категории в общей сумме. Исходные данные могут быть представлены как в "готовом

виде" (доля каждой категории в общей сумме), так и без предварительной обработки. Во втором случае программа сама вычисляет долю каждой категории. Также выполняется сортировка исходных данных.

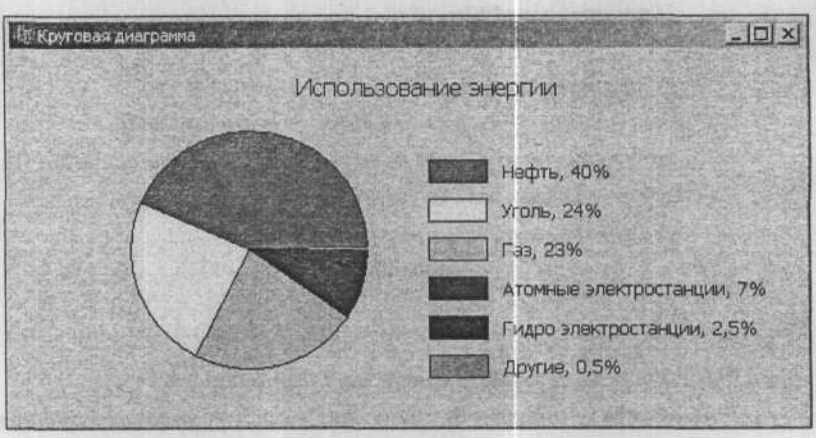


Рис. 1.30. Окно программы **Круговая диаграмма**

```
#include "Math.h" // для доступа к sin и cos

TForm1 *Form1;

#undef PEOPLE    // диаграмма "Население земли"

#define ENERGY   // диаграмма "Источники энергии"

#ifdef PEOPLE
#define HB 6
#define OBR
/* выполнить предварительную обработку исходных данных
(вычислить долю каждой категории в общей сумме) */

// *** исходные данные ***
AnsiString Title = "Население земли";
// значения по каждой категории
```

```
float data[HB] = {1.25e9,1e9,274e6,216e6,172e6,146e6};
// доля категории в общей сумме (процент)
float pr[HB];

// *** подписи легенды ***
AnsiString dTitle[HB] = {"Китай","Индия","США",
                         "Индонезия","Бразилия","Россия"};

// *** цвет для каждой категории ***
TColor cl[HB] = {clLime, clBlue, clMaroon,
                 clGreen, clYellow, clTeal};

#endif

#ifdef ENERGY
#define HB 6
#undef OBR
/* предварительную обработку исходных данных выполнять не
надо - сумма элементов массива data должна быть равна 100 */

// *** исходные данные ***
AnsiString Title = "Использование энергии";
float data[HB] = {0.5,2.5,7,23,24,40}; // значения по каждой
                                       // категории
float pr[HB];  // доля категории в общей сумме (процент)

// *** подписи легенды ***
AnsiString dTitle[HB] = {"Другие",
                         "Гидро электростанции",
                         "Атомные электростанции",
                         "Газ","Уголь","Нефть"};

// *** цвет для каждой категории ***
TColor cl[HB] = {clLime, clBlue, clPurple, clSkyBlue,
                 clYellow, clTeal};
```

```
#endif

#define R 80  // радиус диаграммы
#define D 160 // диаметр диаграммы

#define TORAD 0.0174532 // коэф. пересчета угла из градусов
                        // в радианы
/* для пересчета величины угла из градусов в радианы,
   величину в градусах надо умножить на Pi/180 */

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    int i,j;
    // сортировка исходных данных методом "пузырька"
    float bd;
    AnsiString bt;
    TColor bc;
    for (i = 0; i < HB; i ++)
        for (j = 0; j < HB-1; j++)
            if (data[j+1] < data[j])
            {
                // поменять местами i-ый и i+1-ый элементы
                bd = data[j];
                data[j] = data[j+1];
                data[j+1] = bd;

                bt = dTitle[j];
                dTitle[j] = dTitle[j+1];
                dTitle[j+1] = bt;

                bc = cl[j];
                cl[j] = cl[j+1];
```

```
                cl[j+1] = bc;
            }

    #ifdef OBR
    // обработка данных - вычисление доли
    // каждой категории в общей сумме
    float sum = 0;
    for (i = 0; i < HB; i++)
        sum += data[i];

    for (i = 0; i < HB; i++)
        pr[i] = ( data[i] / sum) * 100;
    #else
        // исходные данные представлены в виде процентов
        for ( i = 0; i < HB; i++)
            pr[i] = data[i];

    #endif
}

// процедура обработки события Paint рисует диаграмму
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    int x,y;
    int i;

    // *** заголовок ***
    Canvas->Font->Name = "Tahoma";
    Canvas->Font->Size = 12;
    x = (ClientWidth - Canvas->TextWidth(Title)) /2;
    Canvas->Brush->Style = bsClear;
    Canvas->TextOutA(x,15,Title);
```

```
    // *** круговая диаграмма ***

  // здесь x,y - координаты левого верхнего угла
    // прямоугольника, в который вписан круг, из которого
    // вырезается сектор
    x = (ClientWidth - D) /2 - R;
    y = 15 + Canvas->TextHeight(Title) + 20;

    int x0,y0; // центр сектора (круга)
    int x1,y1; // координата точки начала дуги
    int x2,y2; // координата точки конца дуги
    int a1,a2; // угол между осью OX и прямыми,
               // ограничивающими сектор

//    int n;      // количество категорий (секторов)
//    n = sizeof(data) / sizeof(float);

    x0 = x + R;
    y0 = y + R;

    a1 = 0; // первый сектор откладываем от оси OX

    //Canvas->Pen->Style = psClear;
    for (int i = 0; i < HB; i++ )
    {
        /* из-за ошибок округления возможна
           ситуация, когда между первым и последним
           секторами будет небольшой промежуток или
           последний сектор перекроет первый.
           Чтобы этого не было, зададим что граница
           последнего сектора совпадает с прямой OX */
        if (i != HB-1)
            a2 = ( a1 + 3.6 * pr[i]);
```

```
    else
        a2 = 359;

    // координата точки начала дуги
    x1 = x0 + R * cos (a2 * TORAD);
    y1 = y0 + R * sin (a2 * TORAD);

    // координата точки конца дуги
    x2 = x0 + R * cos (a1 * TORAD);
    y2 = y0 + R * sin (a1 * TORAD);

    if ( abs(a1-a2) <= 6 )
        Canvas->Pen->Style = psClear;
    else
        Canvas->Pen->Style = psSolid;

    Canvas->Brush->Color = cl[i];
    Canvas->Pie(x,y,x+D,y+D,x1,y1,x2,y2);

    a1 =a2; // следующий сектор рисуем от начала текущего
}

// легенда
Canvas->Font->Size -= 2;
int dy = Canvas->TextHeight("a");
x = x + D + 40;
y = y + 20;
for (i =HB-1; i >=0; i--)
{
    Canvas->Brush->Color = cl[i];
    Canvas->Rectangle(x,y,x+40,y+dy);

    Canvas->Brush->Style = bsClear;
    Canvas->TextOutA(x+50,y,dTitle[i]+
          ", " + FloatToStrF(pr[i],ffGeneral,2,2) + "%");
```

```
        y = y + dy +10;
    }
}

void __fastcall TForm1::FormResize(TObject *Sender)
{
    Form1->Refresh();
}
```

Просмотр иллюстраций

Программа **Просмотр иллюстраций** (рис. 1.31) позволяет просмотреть файлы формата JPEG, например — фотографии. Выбор рабочей папки (каталога) выполняется в стандартном окне **Выбор папки**. Иллюстрации можно просматривать по кадрам или в режиме слайд-шоу. Форма программы приведена на рис. 1.32, значения свойств компонента `Image1` — в табл. 1.7. Частоту смены кадров в режиме слайд-шоу определяет значение свойства `Interval` таймера.

Рис. 1.31. Программа **Просмотр иллюстраций** может отображать иллюстрации в режиме слайд-шоу

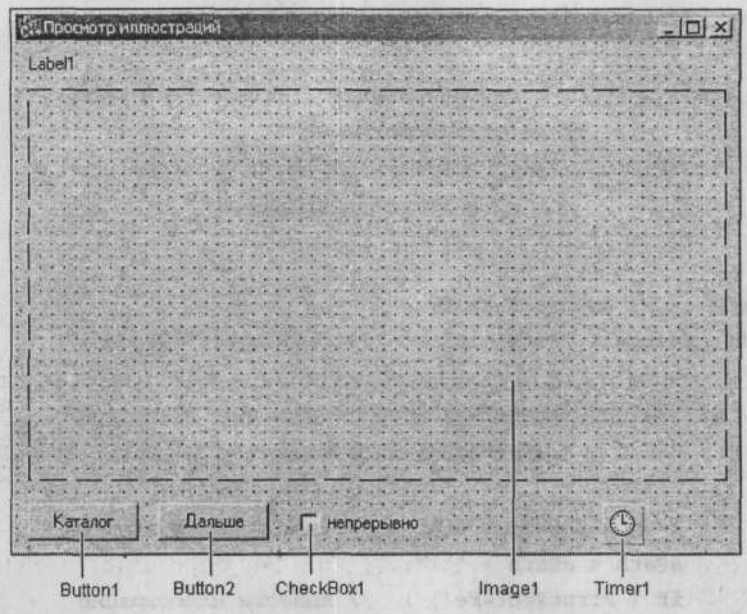


Рис. 1.32. Форма программы **Просмотр иллюстраций**

Таблица 1.7. *Значения свойств компонента* `Image1`

Свойство	Значение
`Autosize`	`false`
`Stretch`	`false`
`Proportional`	`true`

```
#include <FileCtrl.hpp>  // для доступа к SelectDirectory
#include <jpeg.hpp>      // обеспечивает работу с
                         // иллюстрациями в формате JPEG

AnsiString aPath;        // каталог, в котором находится
                         // иллюстрация
TSearchRec aSearchRec; // рез-т поиска файла
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    aPath = ""; // текущий каталог - каталог, из которого
                // запущена программа
    FirstPicture(); // показать картинку, которая есть в
                    // каталоге программы
}

// щелчок на кнопке Каталог
void __fastcall TForm1::Button1Click(TObject *Sender)
{
     if (SelectDirectory("Выберите каталог, в котором"
                " находятся иллюстрации", "", aPath) != 0 )
     {
        // пользователь выбрал каталог и щелкнул на OK
        aPath = aPath + "\\";
        if ( FirstPicture() )  // вывести иллюстрацию
            CheckBox1->Enabled = true;
        else
            Label1->Caption = "В каталоге " + aPath +
                              " нет jpg-иллюстраций." ;
     }
}

// вывести первую картинку и найти следующую
bool __fastcall TForm1::FirstPicture()
{
    Image1->Visible = false;    // скрыть компонент Image1
    Button2->Enabled = false;  // кнопка Дальше недоступна
    CheckBox1->Enabled = false;
    CheckBox1->Checked = false;
    Label1->Caption = "";
    if (FindFirst(aPath+ "*.jpg", faAnyFile, aSearchRec) == 0)
    {
        Image1->Picture->LoadFromFile(aPath+aSearchRec.Name);
```

```
        Image1->Visible = true;
        Label1->Caption = aPath + aSearchRec.Name;
        if ( FindNext(aSearchRec) == 0 )  // найти след. ил.
        {
            // иллюстрация есть
            Button2->Enabled = true; // теперь кнопка Дальше
                                     // доступна
            CheckBox1->Enabled = true;
            return true;
        }
    }
    return false;
}

// вывести текущую и найти следующую картинку
bool __fastcall TForm1::NextPicture()
{
    Image1->Picture->LoadFromFile(aPath+aSearchRec.Name);
    Label1->Caption = aPath + aSearchRec.Name;
    if ( FindNext(aSearchRec) != 0 )  // найти след.
                                      // иллюстрацию
    {
        // иллюстраций больше нет
        Button2->Enabled = false;   // теперь кнопка Дальше
                                    // недоступна
        CheckBox1->Enabled = false;
        return false;
    }
    return true;
}

// щелчок на кнопке Дальше
void __fastcall TForm1::Button2Click(TObject *Sender)
{
```

```
    NextPicture();
}

// щелчок на переключателе "непрерывно"
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
    if ( CheckBox1->Checked)
        Timer1->Enabled = true;  // слайд-шоу
    else
        Timer1->Enabled = false; // по кадрам
}

// Сигнал от таймера - вывести следующую иллюстрацию
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    if ( ! NextPicture())
        Timer1->Enabled = false;
}
```

Часы

В окне программы **Часы** (рис. 1.33) находится изображение идущих часов, которые показывают текущее время.

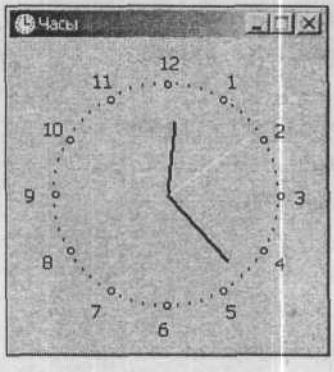


Рис. 1.33. В окне программы **Часы** отображается текущее время

```
#include "DateUtils.hpp" // для доступа к SecondOf, MinuteOf и
                         // HourOf
#include "math.h"        // для доступа к sin и cos

/* Для вычисления синуса и косинуса можно воспользоваться
   функцией SinCos, которая возвращает синус и косинус угла,
   заданного в радианах. Чтобы использовать эту функцию,
   в программу надо добавить директиву #include "Math.hpp" */

#define R 75     // радиус циферблата часов

int  x0, y0;          // центр циферблата
int ahr,amin,asec;    // положение стрелок (угол)

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    TDateTime t;

    // зададим размер формы
    // в соответствии с размером циферблата
    ClientHeight = (R + 30)*2;
    ClientWidth  = (R + 30)*2;
    x0  =  R + 30;
    y0  =  R + 30;

    t  =  Now();

    /* Определить положение стрелок.
     Угол между метками (цифрами) часов, например, цифрами 2 и
     3, - 30 градусов.
     Угол между метками минут - 6 градусов.
     Угол отсчитываем от 12-ти часов */

    ahr  =   90 - HourOf(t)*30-(MinuteOf(Today() ) / 12) *6;
```

```
    amin  =  90 - MinuteOf(t)*6;
    asec  =  90 - SecondOf( Today() )*6;

    Timer1->Interval =  1000; // период сигнала от таймера
                              // 1 сек
    Timer1->Enabled  =  true; // пуск таймера
}

// рисует вектор из точки (x0,y0) под углом a относительно
// оси X. Длинна вектора l
void __fastcall TForm1::Vector(int x0, int y0, int a, int l)
{
    // x0,y0 - начало вектора
    // a - угол между осью x и вектором
    // l - длина вектора
    #define TORAD 0.0174532  // коэффициент пересчета угла из
                             // градусов в радианы

  int x, y;        // координаты конца вектора

  Canvas->MoveTo(x0,y0);
  x  =  x0 + l * cos(a*TORAD);
  y  =  y0 - l * sin(a*TORAD);
  Canvas->LineTo(x,y);

}

// прорисовка циферблата
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    int x, y;  // координаты маркера на циферблате
    int a;     // угол между OX и прямой (x0,yo) (x,y)
    int h;     // метка часовой риски

    TBrushStyle bs; // стиль кисти
    TColor pc;      // цвет карандаша
```

```
int pw;          // ширина карандаша

bs  =  Canvas->Brush->Style;
pc  =  Canvas->Pen->Color;
pw  =  Canvas->Pen->Width;

Canvas->Brush->Style  =  bsClear;
Canvas->Pen->Width  =  1;
Canvas->Pen->Color =  clBlack;

a = 0; // метки ставим от 3-х часов, против часовой
       // стрелки
h = 3; // угол 0 градусов - это 3 часа

// циферблат
while ( a < 360 )
{
    x = x0 +  R * cos(a * TORAD);
    y = x0 -  R * sin(a * TORAD);
    Form1->Canvas->MoveTo(x,y);
    if ( (a % 30) == 0 )
    {
        Canvas->Ellipse(x-2,y-2,x+3,y+3);
        // цифры по большему радиусу
        x = x0 +  (R+15) * cos(a * TORAD);
        y = x0 -  (R+15) * sin(a * TORAD);
        Canvas->TextOut(x-5,y-7,IntToStr(h));
        h--;
        if ( h == 0 ) h = 12;
    }
    else
        Canvas->Ellipse(x-1,y-1,x+1,y+1);
    a = a + 6; // 1 минута - 6 градусов
}
// восстановить карандаш и кисть
```

```
    Canvas->Brush->Style = bs;
    Canvas->Pen->Width = pw;
    Canvas->Pen->Color = pc;

    DrawClock();
}

void __fastcall TForm1::DrawClock(void)
{
    TDateTime t;

  // шаг секундной и минутной стрелок 6 градусов,
  // часовой - 30.

  // стереть изображение стрелок
  Canvas->Pen->Color = clBtnFace;
  Canvas->Pen->Width = 3;
  // часовую
  Vector(x0,y0, ahr, R-20);
  // минутную
  Vector(x0,y0, amin, R-15);
  // секундную
  Vector(x0,y0, asec, R-7);

  t = Now();

  // новое положение стрелок
  ahr = 90 - HourOf(t)*30-(MinuteOf(t)% 12)*6;
  amin = 90 - MinuteOf(t)*6;
  asec = 90 - SecondOf(t)*6;

  // нарисовать стрелки
  // часовая стрелка
  Canvas->Pen->Width = 3;
  Canvas->Pen->Color = clBlack;
```

```
  Vector(x0,y0, ahr, R-20);

  // минутная стрелка
  Canvas->Pen->Width  =  2;
  Canvas->Pen->Color =  clBlack;
  Vector(x0,y0, amin, R-15);

  // секундная стрелка
  Canvas->Pen->Width  =  1;
  Canvas->Pen->Color =  clYellow;
  Vector(x0,y0, asec, R-7);
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    DrawClock();
}
```

Пинг-понг

Программа **Пинг-понг** демонстрирует, как можно сделать графику интерактивной. В окне программы два объекта: мячик и ракетка (рис. 1.34), которой при помощи клавиш перемещения курсора управляет игрок. На форме программы только один компонент — `Timer`.

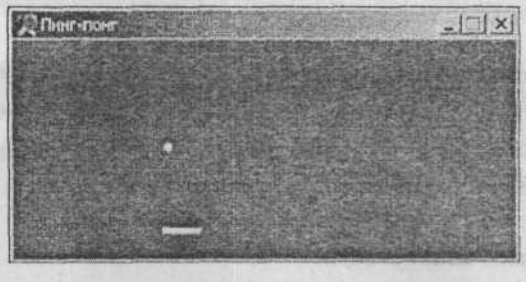


Рис. 1.34. Окно программы **Пинг-Понг**

```
/* Игра "Пинг-понг"
   Демонстрирует принцип программирования интерактивной
   графики */

int x, y;      // положение мячика
int dx, dy;    // приращение координат
int r;         // радиус мячика
TColor cBall; // цвет мячика
TColor cBack; // цвет поля
int wp, hp;    // размер поля (формы)

// это переменные для управления движением ракетки
int rd;         //  0 - ракетка не движется; 1 - движется
                // влево; 2 - движется вправо
int rx1, rx2;  // координаты X концов ракетки
int ry;        // координата Y концов ракетки
int rdx;       // шаг перемещения ракетки

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    r = 5;           // радиус мячика
    x = r; y = 50;  // начальное положение мячика
    dx = 1;
    dy = 1;

    cBall = (TColor)RGB(217, 217, 25); // цвет мячика
    cBack = (TColor)RGB(33, 94, 33);   // цвет поля

    Form1->Color = cBack;

    wp = Form1->ClientWidth;
    hp = Form1->ClientHeight;
```

```
    // управление ракеткой
    rd = 0;      // ракетка на месте (не движется)
    rx1 = 100;
    rx2 = 125;
    ry = Form1->ClientHeight - 20; // расстояние до нижней
                                   // границы формы 20 пикселей
    rdx = 2; // шаг движения ракетки

    // настройка таймера
    Timer1->Interval = 10;
    Timer1->Enabled = true;
}

void __fastcall TForm1::FormPaint(TObject *Sender)
{
    // нарисовать ракетку
    Form1->Canvas->Pen->Color = clRed;
    Form1->Canvas->Rectangle(rx1, ry, rx2, ry+1);
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    // стереть изображение мяча
    Form1->Canvas->Pen->Color = cBack;
    Form1->Canvas->Ellipse(x,y,x+r,y+r);

    // ** вычислить новое положение мяча **
    if ( dx > 0 )
    {
        // мяч движется вправо
        if ( x + dx + r > wp ) dx = - dx;
    }
```

```
else
    // мяч движется влево
    if ( x + dx - r < 0 ) dx = -dx;

if ( dy > 0 )
{
    // мяч движется вниз
    if ((x >= rx1) && (x <= rx2 -r) && (y == ry - r - 1) )
        // мячик попал в ракетку
        dy = - dy;
    else
        if ( y + dy + r > Form1->ClientHeight ) dy = -dy;
}
else
{
    // мяч движется вверх
    if ((x >= rx1) && (x <= rx2) && (y >= ry - r) && (y <=
    ry + r) )
    {
        // Мячик отскочил от нижней стенки и попал в
        // ракетку снизу.
        // Чтобы не было дырок в ракетке, перерисуем ее.
        Form1->Canvas->Pen->Color = clRed;
        Form1->Canvas->Rectangle(rx1, ry, rx2, ry+1);
    }
    if ( y + dy - r < 0 ) dy = -dy;
}
x += dx;
y += dy;

// нарисовать мяч в новой точке
Form1->Canvas->Pen->Color = cBall;
Form1->Canvas->Ellipse(x,y,x+r,y+r);
```

```
// *** ракетка ***
if ( rd != 0 )
{
    // игрок нажал и удерживает одну из клавиш
    // "стрелка вправо" или "стрелка влево"
    // (см. FormKeyDown )
    if ( rd == 1 )
    {
        // вправо
        if ( rx2 < wp )
        {
            // стереть часть слева
            Form1->Canvas->Pen->Color = cBack;
            Form1->Canvas->Rectangle(rx1, ry, rx1 + rdx,
                                     ry + 1);
            // дорисовать часть справа
            Form1->Canvas->Pen->Color = clRed;
            Form1->Canvas->Rectangle (rx2, ry, rx2 + rdx,
                                      ry + 1);
            rx1 += rdx;
            rx2 += rdx;
        }
    }
    else
       // влево
       if (rx1 > 1 )
       {    // стереть часть справа
            Form1->Canvas->Pen->Color = cBack;
            Form1->Canvas->Rectangle (rx2, ry, rx2 - rdx,
                                      ry+1);
            // дорисовать слева
            Form1->Canvas->Pen->Color = clRed;
            Form1->Canvas->Rectangle(rx1 - rdx, ry, rx1 +
                                     rdx, ry+1);
```

```
                rx1 -= rdx;
                rx2 -= rdx;
            }
    }

}

// нажата клавиша
void __fastcall TForm1::FormKeyDown(TObject *Sender,
                            WORD &Key, TShiftState Shift)
{
    if (rd != 0 )
        // пользователь удерживает клавишу, ракетка движется
        return;

    switch ( Key )
    {
        case VK_LEFT : // Шаг (стрелка) право
            rd = 2; break;
        case VK_RIGHT :  // Шаг (стрелка) влево
            rd = 1; break;
    }
}

// клавиша отпущена
void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
    rd = 0;
}
```

Полет в облаках

Программа **Полет в облаках** (рис. 1.35) демонстрирует принципы мультипликации (движение объекта на фоне картинки).

Рис. 1.35. Летящий в облаках самолет

Изображения фона и объекта (рис. 1.36) загружаются из файла. Очередной кадр формируется в памяти (на поверхности невидимого битового образа), а затем выводится на поверхность формы с некоторым смещением относительно предыдущего положения.

Рис. 1.36. Фоновый рисунок и объект

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    // загрузить фоновый рисунок из bmp-файла
    back = new Graphics::TBitmap();
    back->LoadFromFile("sky.bmp");

    // установить размер клиентской (рабочей) области формы
    // в соответствии с размером фонового рисунка
```

```
    ClientWidth = back->Width;
    ClientHeight = back->Height;

    // загрузить картинку
    sprite =  new Graphics::TBitmap();
    sprite->LoadFromFile("plane.bmp");
    sprite->Transparent = true;

    // сформировать кадр
    kadr = new Graphics::TBitmap();
    kadr->LoadFromFile("plane.bmp");

    // исходное положение самолета
    x=-40; // чтобы самолет "вылетал" из-за левой границы окна
    y=20;

    Timer1->Interval = 10;
    Timer1->Enabled = true;
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    TRect badRect; // положение и размер области фона,
                   // которую надо восстановить

    TRect  frameRect;

    badRect = Rect(x,y,x+sprite->Width,y+sprite->Height);
    frameRect = Rect(0,0, kadr->Width, kadr->Height);

    #ifdef ONCANVAS
```

```
// *** изображение формируем на поверхности формы ***

// стереть самолет (восстановить "испорченный" фон)
Canvas->CopyRect(badRect,back->Canvas,badRect);

// вычислим новые координаты спрайта
x +=2;
if (x > ClientWidth)
{
    // самолет улетел за правую границу формы
    // изменим высоту и скорость полета
    x = -20;
    y = random(ClientHeight - 30);  // высота полета
    // скорость полета определяется периодом возникновения
    // события OnTimer, который, в свою очередь, зависит
    // от значения свойства Interval
    Timer1->Interval = random(20) + 10; // скорость
                                        // "полета" от 10 до 29
}
Canvas->Draw(x,y,sprite);

#else
//  изображение формируем на рабочей поверхности,
// затем выводим на поверхность формы

// сформировать очередной кадр
// скопировать фрагмент фона
kadr->Canvas->CopyRect(frameRect, back->Canvas,badRect);
// нарисовать объект
kadr->Canvas->Draw(0,0,sprite);

// вывести кадр
```

```
    Form1->Canvas->Draw(x,y,kadr);

// вычислим новые координаты спрайта
    x += 1;
    if (x > ClientWidth)
    {
        // самолет улетел за правую границу формы
        //   изменим высоту и скорость полета
        x = -20;
        y = random(ClientHeight - 30);  // высота полета
        // скорость полета определяется периодом возникновения
        // события OnTimer, который, в свою очередь, зависит
        // от значения свойства Interval
        Timer1->Interval = random(20) + 10; // скорость
                                    // "полета" от 10 до 29
    }
    #endif
}

void __fastcall TForm1::FormPaint(TObject *Sender)
{
    // восстановить фоновый рисунок
    Canvas->Draw(0,0,back);     // фон
}
```

Баннер

Программа **Бегущая строка** (рис. 1.37) демонстрирует использование битового образа для вывода баннера в стиле бегущей строки. Битовый образ (бегущая строка) загружается из ресурса программы. Баннер "выплывает" из-за правой границы формы. В момент времени, когда баннер достигает центра окна, движение приостанавливается на несколько секунд, а затем — возобновляется.

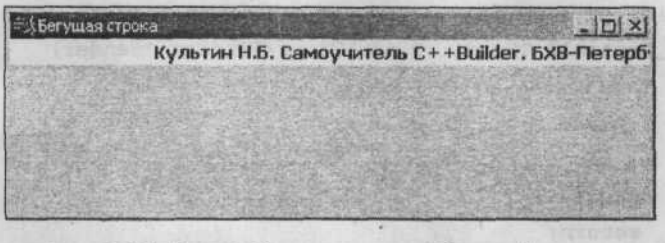


Рис. 1.37. Вывод баннера в стиле бегущей строки

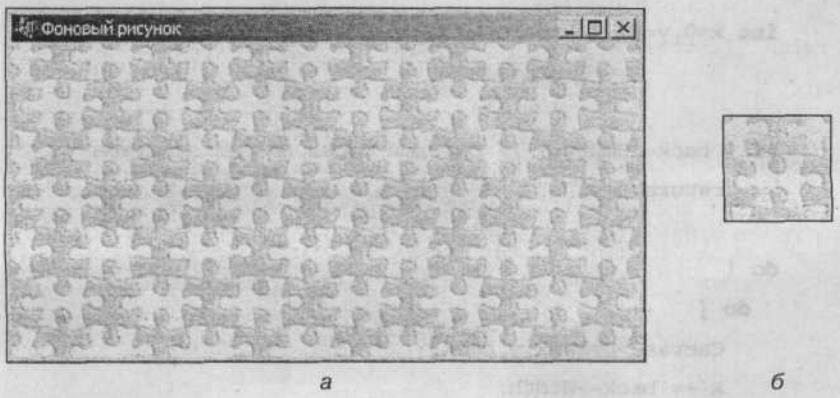
```
Graphics::TBitmap *banner;    // баннер - картинка загружается
                              // из ресурса

int x,y;      // координаты левого верхнего угла области вывода
              // баннера
int pause;    // время (количество тактов таймера) приостановки
              // движения баннера
int xp;       // координата приостановки движения баннера
TColor bc;    // цвет фона баннера

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    banner = new Graphics::TBitmap();
    // загрузить баннер из ресурса
    banner->LoadFromResourceID((int)HInstance,101);
    bc = banner->Canvas->Pixels[0][0]; // цвет фона баннера
    x = Form1->ClientWidth; // баннер "выплывает" из-за правой
                            // границы окна
    y = 0;
    // вычислить координату X точки приостановки движения
    // баннера
    xp = (Form1->ClientWidth - banner->Width) / 2;
    if (xp < 0 ) xp = 0;
}
```

```
// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    if ( pause > 0 )
    {
        pause--;
        return;
    }

    Form1->Canvas->Draw(x,y,banner);
    if ( --x == xp)
    {
        pause = 100; // остановить движение баннера на
                     // 100 тактов таймера
    }

    if ( x < - banner->Width)
        x = Form1->ClientWidth;

}

// обработка события Paint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    TColor b,p;

    b = Canvas->Brush->Color; // сохранить текущий цвет кисти
    p = Canvas->Pen->Color;   // сохранить текущий цвет
                              // карандаша

    // закрасить область вывода баннера цветом фона баннера
    Canvas->Brush->Color = bc;
    Canvas->Pen->Color = bc;
    Canvas->Rectangle(0,0,ClientWidth,banner->Height);
```

```
    Canvas->Brush->Color = b; // восстановить цвет кисти
    Canvas->Pen->Color = p;   // восстановить цвет карандаша
}

// пользователь изменил размер формы
void __fastcall TForm1::FormResize(TObject *Sender)
{
    // вычислить координату X точки приостановки движения
    // баннера
    xp = (Form1->ClientWidth - banner->Width) / 2;
    if (xp < 0 ) xp = 0;
}
```

Фоновый рисунок

Программа **Фоновый рисунок** демонстрирует, как можно получить фоновый рисунок путем многократного вывода битового образа на поверхность формы. Битовый образ загружается из файла, но может быть загружен и из ресурса. Окно программы и битовый образ, использованный для формирования фонового рисунка, приведены на рис. 1.38.



а *б*

Рис. 1.38. Фоновый рисунок формы (*а*) сформирован путем многократного вывода битового образа (*б*)

```
// начало работы программы
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    back = new Graphics::TBitmap(); // создать объект -
                                    // битовый образ

     // загрузить картинку
    try // в процессе загрузки картинки возможны ошибки
    {
        Form1->back->LoadFromFile("canvas.bmp");
    }
    catch (EFOpenError &e)
    {
        return;
    }
}

// формирует фоновый рисунок
void __fastcall TForm1::Background()
{
    int x=0,y=0; // координаты левого верхнего угла битового
                 // образа

    if ( back->Empty ) // битовый образ не был загружен
        return;

    do {
      do {
        Canvas->Draw(x,y,back);
        x += back->Width;
      }
      while (x < ClientWidth);
```

```
        x = 0;
        y += back->Height;
      }
    while (y < ClientHeight);
}

// обработка события Paint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    Background(); // обновить фоновый рисунок
}
```

Мультимедиа

В этом разделе собраны программы, которые демонстрируют работу со звуком, видео и анимацией.

Общие замечания

- Работу с анимацией и звуком обеспечивают компоненты `Animate` и `MediaPlayer`.
- Компонент `MediaPlayer` позволяет воспроизводить звук, анимацию и видео.
- Компонент `Animate` позволяет воспроизвести только простую, не сопровождаемую звуком анимацию.

WAV

Программа **Звуки Windows** (рис. 1.39) позволяет прослушать звуковые файлы (форматов WAV, RMI и MID), которые находятся в каталоге Windows\Media.

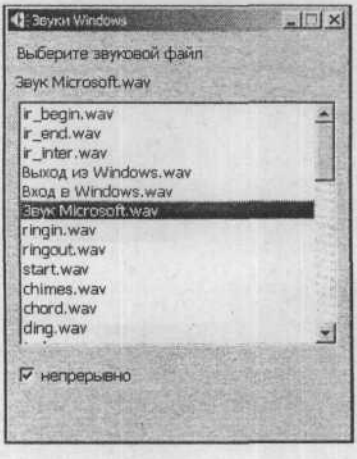


Рис. 1.39. Программа **Звуки Windows** позволяет прослушать звуковые файлы, которые находятся в каталоге Windows\Media

Список файлов формируется в начале работы программы. Если переключатель **непрерывно** выбран, то после воспроизведения текущего файла автоматически активизируется воспроизведение следующего. Форма программы приведена на рис. 1.40.

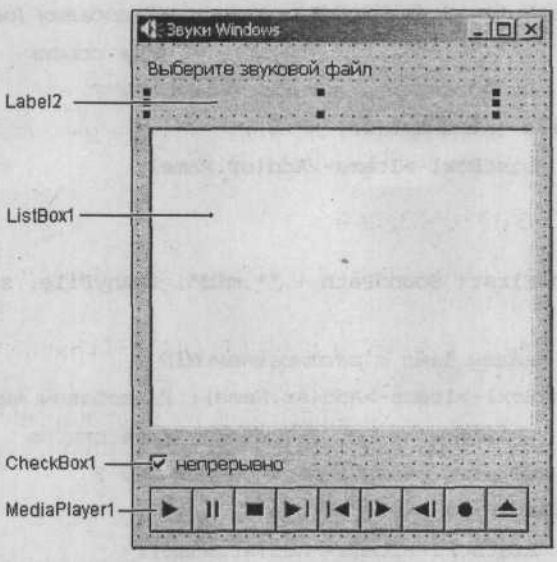


Рис. 1.40. Форма программы **Звуки Windows**

```
// начало работы программы
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    char *wd; // каталог Windows

    wd = (char*)AllocMem(MAX_PATH);
    GetWindowsDirectory(wd,MAX_PATH);
    SoundPath = wd;

    // звуковые файлы находятся в подкаталоге Media
    SoundPath = SoundPath + "\\Media\\";

    // сформируем список звуковых файлов
```

```
TSearchRec sr;
if (FindFirst( SoundPath + "*.wav", faAnyFile, sr) == 0 )
{
    // найден файл с расширением WAV
    ListBox1->Items->Add(sr.Name); // добавим имя файла
                                   // в список
    // еще есть файлы с расширением WAV ?
    while (FindNext(sr) == 0 )
        ListBox1->Items->Add(sr.Name);
}

if (FindFirst( SoundPath + "*.mid", faAnyFile, sr) == 0 )
{
    // найден файл с расширением MID
    ListBox1->Items->Add(sr.Name); // добавим имя файла
                                   // в список
    // еще есть файлы с расширением MID ?
    while (FindNext(sr) == 0 )
        ListBox1->Items->Add(sr.Name);
}

if (FindFirst( SoundPath + "*.rmi", faAnyFile, sr) == 0 )
{
    // найден файл с расширением RMI
    ListBox1->Items->Add(sr.Name); // добавим имя файла
                                   // в список
    // еще есть файлы с расширением RMI ?
    while (FindNext(sr) == 0 )
        ListBox1->Items->Add(sr.Name);
}

// воспроизвести первый файл
if ( ListBox1->Items->Count != 0)
```

```
    {
        Label2->Caption = ListBox1->Items->Strings[1];
        ListBox1->ItemIndex = 0;

        MediaPlayer1->FileName = SoundPath + ListBox1->
                                        Items->Strings[1];
        MediaPlayer1->Open();
        MediaPlayer1->Play();
    }
}

// щелчок на элементе списка
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
    if ( (CheckBox1->Checked) &&
         ( MediaPlayer1->Mode == mpPlaying ) )
        return;
    Label2->Caption = ListBox1->Items->
                        Strings[ListBox1->ItemIndex];
    MediaPlayer1->FileName = SoundPath + Label2->Caption;
    MediaPlayer1->Open();
    if ( ! CheckBox1->Checked)
          MediaPlayer1->Notify = false;
    MediaPlayer1->Play();
}

/* событие Notify возникает в момент завершения воспроизведения
звукового файла, если перед активизацией метода Play значение
свойства Notify равно true */
void __fastcall TForm1::MediaPlayer1Notify(TObject *Sender)
{
    if ( ListBox1->ItemIndex < ListBox1->Items->Count )
    {
        ListBox1->ItemIndex = ListBox1->ItemIndex + 1;
```

```
        Label2->Caption = ListBox1->Items->
                                Strings[ListBox1->ItemIndex];
        MediaPlayer1->FileName = SoundPath + Label2->Caption;
        MediaPlayer1->Open();
        if ( ! CheckBox1->Checked)
            MediaPlayer1->Notify = false;
        MediaPlayer1->Play();
    }
}
```

MP3 Player

Программа **MP3 Player** (рис. 1.41) позволяет прослушать музыкальные файлы формата MP3.

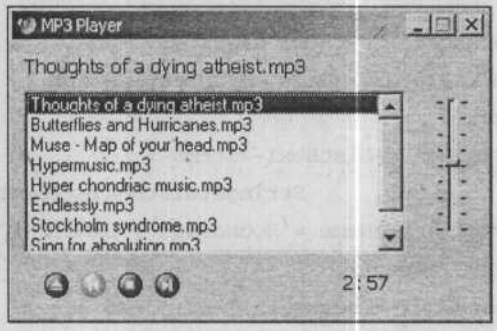


Рис. 1.41. В окне программы отображается время воспроизведения композиции, есть возможность регулировки громкости

Форма программы приведена на рис. 1.42. Следует обратить внимание, что на форме нет компонента `MediaPlayer1`, он создается в начале работы программы (делает это конструктор формы). Выбор папки (каталога), в котором находятся файлы формата MP3, осуществляется в стандартном диалоговом окне **Выбор папки**, которое становится доступным в результате щелчка на кнопке **Eject** (`SpeedButton1`). Следует обратить внимание на то, что битовые образы кнопок `SpeedButton2` и `SpeedButton4` содержат по два изображения кнопки — доступной и недоступной, а кнопок

SpeedButton1 и SpeedButton3 — одно (доступной). Тем не менее картинка на кнопке **Play/Stop** (SpeedButton3) изменяется — в нужный момент загружается из ресурса. Ресурс программы содержит три битовых образа: "доступная кнопка Play", "доступная кнопка Stop" и "недоступная кнопка Play". Идентификаторы этих битовых образов, соответственно, 101, 102 и 103. Значения свойств компонента TrackBar1 приведены в табл. 1.8.

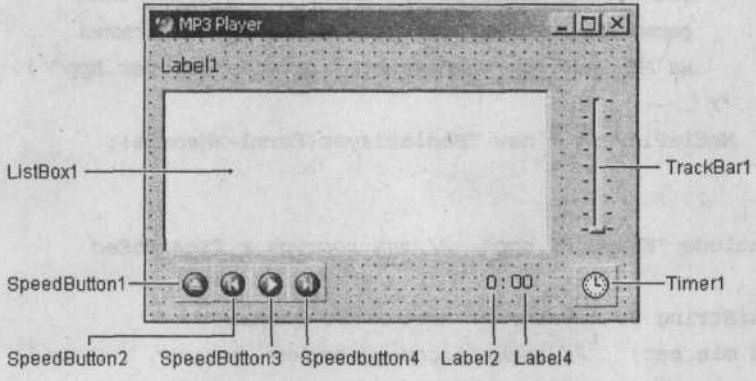


Рис. 1.42. Форма программы **MP3 Player**

Таблица 1.8. *Значения свойств компонента* `TrackBar1`

Свойство	Значение
Orientation	vrVertical
Max	0
Min	-65535
ThumbLength	10
TickMarks	tmBoth
Frequency	8192
ShowHint	true
Hint	Громкость

```
TForm1 *Form1;

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    /* Создать компонент MediaPlayer.
       Объявление MediaPlayer1 находится в объявлении
       формы (см. mp3main.h), там же находится ссылка
       на MPlayer.hpp (директива #include "MPlayer.hpp" ).
    */
    MediaPlayer1 = new TMediaPlayer(Form1->Handle);
}

#include "FileCtrl.hpp"  // для доступа к TSearchRec

AnsiString SoundPath; // путь к MP3-файлам
int min,sec;    // время воспроизведения (минуты, секунды)

int mode = 0;  // 0 - режим "Стоп"
               // 1 - режим "Воспроизведение"

/* Процедуре vaweOutSetVolume в качестве параметра
   передается двойное слово, старший байт которого
   определяет громкость левого канала, младший - правого.
   Определив таким образом тип TVolume, имеется возможность
   независимой регулировки громкости каждого канала.
   Максимальной громкости канала соответствует значение
   0xFFFF.
*/
union TVolume{
          unsigned long Volume;
          struct
            {
                Word Left;
```

```
                Word Right;
            };
    } volume;

// начало работы программы
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    PlayList("");

    // установить движок регулятора громкости в соответствии
    // с текущим уровнем, установленным в системе
    waveOutGetVolume(0,&volume.Volume);
    TrackBar1->Position = - volume.Left;

    ListBox1->Color = (TColor)RGB(56,176,222);
}

// формирует список MP3-файлов, находящихся в указанном
// каталоге
void __fastcall TForm1::PlayList(AnsiString path)
{
    /* структура SearchRec содержит информацию о файле,
       удовлетворяющем условию поиска */
    TSearchRec SearchRec;

    ListBox1->Clear();
    // сформировать список MP3-файлов
    if ( FindFirst(path + "*.mp3", faAnyFile, SearchRec)!= 0 )
    {
        // в выбранном каталоге нет MP3-файлов
        SpeedButton2->Enabled = false;
        SpeedButton3->Glyph->
                    LoadFromResourceID((int)HInstance,103);
        SpeedButton4->Enabled = false;
```

```
        Label1->Caption = "";
        return;
    }

    // в каталоге есть файл с расширением mp3
    // добавим имя этого файла в список
    ListBox1->Items->Add(SearchRec.Name);
    // пока в каталоге есть другие файлы с расширением wav
    while (FindNext(SearchRec) == 0)
            ListBox1->Items->Add(SearchRec.Name);

    ListBox1->ItemIndex  =  0;
    Label1->Caption = ListBox1->Items->
                        Strings [ListBox1->ItemIndex];
    SpeedButton2->Enabled = false;
    if (ListBox1->Count == 1)
        SpeedButton4->Enabled = false;
    else
        SpeedButton4->Enabled = true;
     SpeedButton3->Glyph->
                LoadFromResourceID((int)HInstance,101);
}

// активизировать воспроизведение MP3-файла, имя которого
// выделено в списке ListBox
void __fastcall TForm1::Play()
{
    Label1->Caption = ListBox1->Items->
                            Strings [ListBox1->ItemIndex];
    MediaPlayer1->FileName = SoundPath +
            ListBox1->Items->Strings [ListBox1->ItemIndex];

    MediaPlayer1->Open();
    MediaPlayer1->Play();
```

```
    min = 0;
    sec = 0;
    Timer1->Enabled = true;
}

// остановить воспроизведение
void __fastcall TForm1::Stop()
{
    MediaPlayer1->Stop();
    Timer1->Enabled = false;
    Label2->Caption = "0";
    Label4->Caption = "00";
}

// щелчок на кнопке Play/Stop
// ( картинки для кнопок лучше загружать из ресурса )
void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
    if ( mode == 1 )
    {
        SpeedButton3->Glyph->
                    LoadFromResourceID((int)HInstance, 101);
        SpeedButton3->Hint = "Воспроизведение";
        Stop();
        mode = 0;
    }
    else {
        SpeedButton3->Glyph->
                    LoadFromResourceID((int)HInstance, 102);
        SpeedButton3->Hint = "Стоп";
        Play();
        mode = 1;
    }
}
```

```
// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    // индикация времени воспроизведения
    if ( sec < 59 )
    {
        sec++;
        if ( sec < 10)
            Label4->Caption = "0" + IntToStr(sec);
        else
            Label4->Caption = IntToStr(sec);
    }
    else
    {
        sec = 0;
        min++;
        Label2->Caption = IntToStr(min);
        Label4->Caption = "00";
    }

    // завершено воспроизведение текущего файла?
    if ( MediaPlayer1->Position < MediaPlayer1->Length )
        // воспроизведение текущей композиции не завершено
        return;

    // воспроизведение текущей композиции завершено
    Stop();

    if ( ListBox1->ItemIndex < ListBox1->Count - 1 )
    {
        ListBox1->ItemIndex += 1;
        Play(); // активизировать воспроизведение следующего
                // MP3-файла
```

```
        if ( ListBox1->ItemIndex == ListBox1->Count - 1 )
            SpeedButton4->Enabled = false;
    }
    else {
        // закончено воспроизведение последнего MP3-файла
        SpeedButton3->Glyph->
                    LoadFromResourceID((int)HInstance, 101);
        SpeedButton3->Hint = "Воспроизведение";
        mode = 0;
    }
}

// щелчок на кнопке "К следующему файлу"
void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
{
    if ( mode == 1 )
        Stop();  // остановить воспроизведение текущей
                 // композиции
    ListBox1->ItemIndex += 1;
    Label1->Caption = ListBox1->
                    Items->Strings [ListBox1->ItemIndex];

    // проверить и, если надо, изменить состояние
    // кнопок перехода к следующему и предыдущему файлу
    if ( ListBox1->ItemIndex == ListBox1->Count - 1)
        SpeedButton4->Enabled = false;

    if ( ! SpeedButton2->Enabled  )
        SpeedButton2->Enabled = true;

    if ( mode == 1)
        Play(); // активизировать воспроизведение следующей
                // композиции
}
```

```
// щелчок на кнопке "К предыдущему файлу"
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
     if ( mode == 1 )
        Stop();  // остановить воспроизведение текущей
                 // композиции
     ListBox1->ItemIndex -= 1;
     Label1->Caption = ListBox1->Items->
                       Strings [ListBox1->ItemIndex];

     // проверить и, если надо, изменить состояние
     // кнопок перехода к следующему и предыдущему файлу
     if ( ! SpeedButton4->Enabled )
        SpeedButton4->Enabled = true;

     if ( ListBox1->ItemIndex == 0 )
        SpeedButton2->Enabled = false;

     if (mode == 1 )
        Play(); // активизировать воспроизведение предыдущей
                // композиции
}

#include "FileCtrl.hpp"

// щелчок на кнопке Eject - выбор каталога
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
    AnsiString dir;
    if ( SelectDirectory("Выберите каталог","",dir) )
    {
        if ( mode == 1) // режим "Воспроизведение"
        {
            Stop();
```

```
            SpeedButton3->Glyph->
                    LoadFromResourceID((int)HInstance, 101);
            SpeedButton3->Hint = "Воспроизведение";
            Stop();
            mode = 0;
        }
        SoundPath = dir + "\\";
        PlayList(SoundPath);
    }
}

// щелчок на имени файла (композиции)
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
    if ( MediaPlayer1->Mode == 2) // плеер в режиме
воспроизведения
    {
        Stop(); // остановить воспроизведение текущей
композиции
        Play(); // активизировать воспроизведение выбранной
композиции
    }

    // изменить, если надо, состояние кнопок
    // перехода к предыдущей и следующей композиции
    if (ListBox1->ItemIndex == 0 )
        SpeedButton2->Enabled = false;
    else
        SpeedButton2->Enabled = true;

    if (ListBox1->ItemIndex == ListBox1->Count -1  )
        SpeedButton4->Enabled = false;
    else
        SpeedButton4->Enabled = true;
}

#include "mmsystem.hpp"
```

```
// пользователь изменил положение регулятора громкости
void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
    volume.Left = - TrackBar1->Position;
    volume.Right = - TrackBar1->Position;
    waveOutSetVolume(0,volume.Volume);
}
```

Воспроизведение MIDI

Программа **Успеть за 60 секунд**, ее форма приведена на рис. 1.43, демонстрирует использование компонента `MediaPlayer` для воспроизведения звука в формате MIDI. Мелодия воспроизводится "по кругу", до тех пор, пока пользователь не угадает число или не истечет время, отведенное на решение задачи.

Рис. 1.43. Форма программы **Успеть за 60 секунд**

```
int pw;         // "секретное" число
int rem = 60; // остаток времени на выполнение задания

// начало работы программы
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    TSearchRec sr;

    if ( FindFirst("*.mid",faAnyFile, sr) == 0)
```

```
    {
        MediaPlayer1->FileName = sr.Name;
        MediaPlayer1->Open();
        MediaPlayer1->Play();
    }

    Randomize();
    pw = RandomRange(100,999); // "секретное" число
}

// проверяет, правильное ли число ввел игрок
void __fastcall TForm1::isRight(void)
{
    if ( StrToInt(Edit1->Text) == pw )
        {
            Timer1->Enabled = false;
            Button1->Enabled = false;
            Edit1->Enabled = false;
            MediaPlayer1->Stop();
            // аплодисменты!
            //PlaySound("Applause.wav",0, SND_ASYNC);
            ShowMessage("Поздравляю!\nВы угадали число за " +
                        IntToStr(60 - rem)+ " сек");
        }
}

// Нажатие клавиши в поле редактирования
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char
&Key)
{
    if ( ( Edit1->Text.Length() < 3) &&
         ((Key >= '0') && (Key <= '9')))
        return;

    if ( Key == VK_RETURN)
```

```
    {
        isRight(); // проверить, правильное ли число
                   // ввел пользователь
        return;
    }

    if ( Key == VK_BACK) return;

    // остальные символы запрещены
    Key = 0;
}

// Содержимое поля редактирования изменилось
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    // кнопка OK доступна только в том случае,
    // если в поле редактирования три символа (цифры)
    if ( Edit1->Text.Length() == 3)
        Button1->Enabled = true;
    else
        Button1->Enabled = false;
}

// щелчок на кнопке OK
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    isRight();  // проверить, правильное ли число ввел
                // пользователь
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    rem--;
    Label4->Caption = IntToStr(rem);
```

```
    if (rem == 0 )
    {
        // время, отведенное на решение задачи, истекло
        Timer1->Enabled = false;
        Edit1->Enabled = false;
        Button1->Enabled = false;

        MediaPlayer1->Stop();

        ShowMessage("К сожалению, Вы не справились с "
                "поставленной задачей\n"
                "\"Секретное\" число - " + IntToStr(pw) );
    }
}
// состояние плеера изменилось
void __fastcall TForm1::MediaPlayer1Notify(TObject *Sender)
{
    /* Если плеер воспроизводит файл и значение свойства
       Notify равно true (метод Play по умолчанию присваивает
       свойству Notify значение true), то в момент окончания
       воспроизведения возникает событие Notyfy */

    if (Timer1->Enabled)
        /* Длительность мелодии меньше времени, отведенного
           на решение задачи. Проиграть еще раз. */
        MediaPlayer1->Play();
}
// завершение работы программы
void __fastcall TForm1::FormClose(TObject *Sender,
TCloseAction &Action)
{
    Timer1->Enabled=false;
    MediaPlayer1->Stop();
    MediaPlayer1->Close();
}
```

Compact Disk Player (версия 1)

Программа **Compact Disk Player** позволяет прослушать компакт-диск. После запуска или после того, как в дисковод будет вставлен компакт-диск, в окне программы отображается количество треков диска и общее время звучания CD, в процессе воспроизведения — номер воспроизводимого трека, его длина (время) и время от начала воспроизведения (рис. 1.44).

Рис. 1.44. В окне программы отображается информация о воспроизводимом треке

Отличительной особенностью программы является то, что она контролирует наличие диска в дисководе и его тип. Форма программы приведена на рис. 1.45 (компонент `MediaPlayer` находится вне рабочей области формы).

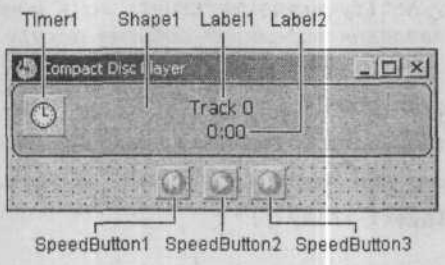


Рис. 1.45. Форма программы **Compact Disk Player** (версия 1)

```
// картинки для кнопки Play/Stop
Graphics::TBitmap *bmPlay;    // Play
Graphics::TBitmap *bmStop;    // Stop

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
```

```
{
    bmPlay   = new Graphics::TBitmap();
    bmStop   = new Graphics::TBitmap();

    // загрузить картинки для кнопки Play/Stop
    bmPlay->LoadFromResourceID((int)HInstance,101);
    bmStop->LoadFromResourceID((int)HInstance,102);
    // отобразить картинку
    SpeedButton2->Glyph->Assign(bmPlay);
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    MediaPlayer->Notify = true; // разрешить событие Notify
}

// эти макросы обеспечивают перевод интервала времени
// выраженного в миллисекундах, в минуты и секунды
#define MINUTE(ms) ((ms/1000)/60)
#define SECOND(ms) ((ms/1000)%60)

// выводит в поле Label1 информацию о текущем треке
void __fastcall TForm1::TrackInfo()
{
    int ms; // время звучания трека, мсек
    AnsiString st;

    Track  =  MCI_TMSF_TRACK(MediaPlayer->Position);

    MediaPlayer->TimeFormat = tfMilliseconds;
    ms = MediaPlayer->TrackLength[Track];
    MediaPlayer->TimeFormat = tfTMSF;

    st = IntToStr(SECOND(ms));
    if ( st.Length() == 1)
```

```
            st = "0" + st;

    st = "Трек "+ IntToStr(Track) +
         ". Длительность "+ IntToStr(MINUTE(ms)) + ":" + st;

    Label1->Caption = st;
}

// изменение состояния плеера
void __fastcall TForm1::MediaPlayerNotify(TObject *Sender)
{
  switch ( MediaPlayer->Mode )
  {
    case mpOpen: // пользователь открыл дисковод
    {
        SpeedButton1->Enabled  =  false;
        SpeedButton2->Glyph->Assign(bmPlay);
        SpeedButton2->Tag  =  0;
        SpeedButton3->Enabled  =  false;
        Label2->Caption  =  "00:00";

        /* по сигналу от таймера будем проверять
           состояние дисковода */
        Timer->Enabled  =  True;
    }
  }
  MediaPlayer->Notify  =  true;
}

// сигнал от таймера: вывести номер трека
// и время воспроизведения
void __fastcall TForm1::TimerTimer(TObject *Sender)
{
  int trk;          // трек
```

```
int min, sec;      // время
AnsiString st;

if ( MediaPlayer->Mode == mpPlaying ) // Воспроизведение
{
  // получить номер воспроизводимого трека и
  trk  =  MCI_TMSF_TRACK(MediaPlayer->Position);

  if ( trk != Track ) // произошла смена трека
  {
    TrackInfo();
    Track  =  trk;
    if ( Track > 1 )
        SpeedButton1->Enabled  =  true;  // доступна кнопка
                                         // "пред.трек"
    if ( Track == MediaPlayer->Tracks)
        SpeedButton3->Enabled  =  false; // кнопка
                                         // "след.трек"
                                         // недоступна
  }

  // вывод информации о воспроизводимом треке
  min  =  MCI_TMSF_MINUTE(MediaPlayer->Position);
  sec  =  MCI_TMSF_SECOND(MediaPlayer->Position);
  st.printf("%d:%.2d",min,sec);
  Label2->Caption = st;
  return;
}

/* Если дисковод открыт или в нем нет
 AudioCD, то Mode == mpOpen.
 Ждем диск, т.е. до тех пор пока не будет
 Mode == mpStopped + кол-во треков > 1 */
if ( (MediaPlayer->Mode == mpStopped) &&
     (MediaPlayer->Tracks > 1) )
```

```
  {
    // диск вставлен
    Timer->Enabled  =  false;
    SpeedButton2->Enabled = true;;
    SpeedButton2->Tag = 0;
    SpeedButton3->Enabled = true;
    MediaPlayer->Notify  =  true;

    // получить информацию о времени звучания CD
    MediaPlayer->TimeFormat = tfMilliseconds;

    int ms = MediaPlayer->Length;
    AnsiString st = "Audio CD. Время звучания: ";

    st = st +  IntToStr(MINUTE(ms));
    st = st + ":" + IntToStr(SECOND(ms));
    Label1->Caption  =  st;

    MediaPlayer->TimeFormat = tfTMSF;
    Label1->Visible  =  true;
    Track = 0;
    return;
  }

  // дисковод открыт или в дисководе не AudioCD
  if (( MediaPlayer->Mode == mpOpen )  ||
      (MediaPlayer->Mode == mpStopped) &&
      (MediaPlayer->Tracks == 1))
  {
    Label1->Caption  =  "Вставьте AudioCD";
    if ( Label1->Visible )
         Label1->Visible  =  false;
    else  Label1->Visible  =  true;
  }
}
```

```
// пользователь закрыл окно программы
void __fastcall TForm1::FormClose(TObject *Sender,
                                  TCloseAction &Action)
{
    MediaPlayer->Stop();
    MediaPlayer->Close();
}

// предыдущий трек
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
    MediaPlayer->Previous(); // в начало текущего трека
    MediaPlayer->Previous(); // в начало предыдущего трека
    if ( MCI_TMSF_TRACK(MediaPlayer->Position) == 1 )
      SpeedButton1->Enabled  =  false;
    if ( ! SpeedButton3->Enabled )
      SpeedButton3->Enabled  =  true;
    TrackInfo();
    Label2->Caption = "0:00";
}

// следующий трек
void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
  MediaPlayer->Next();
  // если перешли к последнему треку, то кнопку
  // Next сделать недоступной
  if ( MCI_TMSF_TRACK(MediaPlayer->Position) ==
                                    MediaPlayer->Tracks )
      SpeedButton3->Enabled  =  false;
  if (!SpeedButton1->Enabled ) SpeedButton1->Enabled = true;
  TrackInfo();
  Label2->Caption = "0:00";
}
```

```
// щелчок на кнопке Play/Stop
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
if ( SpeedButton2->Tag == 0 ) {
    // щелчок на кнопке Play
    MediaPlayer->Play();
    SpeedButton2->Glyph->Assign(bmStop);
    SpeedButton2->Hint = "Стоп";
    SpeedButton2->Tag  =  1;
    //SpeedButton3->Enabled  =  true; // доступна кнопка
                                      // "следующий трек"
    MediaPlayer->Notify  =  true;
    Timer->Enabled  = true;
    TrackInfo();
  }
  else {
    // щелчок на кнопке Stop
    SpeedButton2->Glyph->Assign(bmPlay);
    SpeedButton2->Hint = "Воспроизведение";
    SpeedButton2->Tag  =  0;
    MediaPlayer->Notify  =  true;
    MediaPlayer->Stop();
    Timer->Enabled  =  false;
  }
}
```

Compact Disk Player (версия 2)

Программа **Compact Disk Player**, форма и окно которой приведены, соответственно, на рис. 1.46 и рис. 1.47, позволяет прослушать компакт-диск. Как можно видеть, заголовок и граница окна во время работы программы не отображаются (рис. 1.47), тем не менее пользователь может переместить окно программы, установив указатель мыши на изображение индикатора (в поле компонента Shape1). Значения свойств формы приведены в табл. 1.9. Следует обратить внимание, что все компоненты

находятся на поверхности компонента Shape2, а не на поверхности формы. Процедуры обработки событий компонента MediaPlayer те же, что и в программе **Compact Disk Player** (версия 1). Свойству Cursor компонента Shape1 надо присвоить значение crDrag.

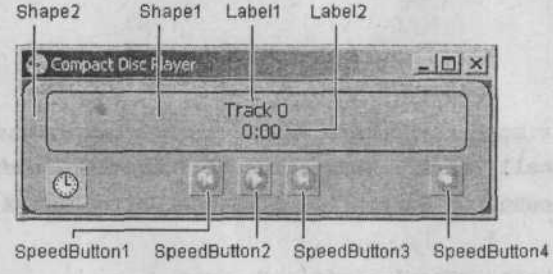


Рис. 1.46. Фома программы **Compact Disk Player** (версия 2)

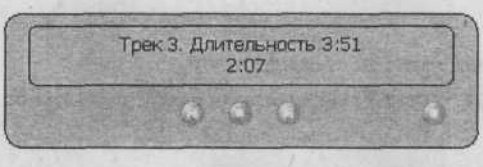


Рис. 1.47. Окно программы **Compact Disk Player** (версия 2)

Таблица 1.9. *Значения свойств формы*

Свойство	Значение
BorderStyle	bsNone
Color	clFuchsia
TransparentColorValue	clFuchsia
TransparentColor	true

```
int px,py; // точка, в которой нажата кнопка мыши

// нажатие кнопки мыши в поле компонента Shape1
void __fastcall TForm1::Shape1MouseDown(TObject *Sender,
```

```
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   // запомнить координаты точки, в которой
   // пользователь нажал кнопку мыши
   px = X;
   py = Y;
}

// нажатая в поле компонента Shape1 кнопка мыши отпущена
void __fastcall TForm1::Shape1MouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
   Form1->Left = Form1->Left + X - px;
   Form1->Top = Form1->Top + Y - py;
}

// щелчок на кнопке Off (Выключить)
void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
{
   // завершение работы программы
   MediaPlayer->Stop();
   Form1->Close();
}
```

Video Player

Программа **Video Player**, ее оно и форма приведены на рис. 1.48 и рис. 1.49 соответственно, позволяет просмотреть видеоролик форматов AVI или MPG. Выбор клипа осуществляется в стандартном окне **Открыть файл**, которое становится доступным в результате щелчка на кнопке **Eject** (`SpeedButton1`). Кнопка `SpeedButton2` используется как для активизации процесса воспроизведения, так и для его приостановки. Картинки для кнопки загружаются из ресурса программы. Следует обратить внимание на то, что программа определяет размер кадров видеоролика. Это

позволяет разместить экран в центре формы и, если размер кадров превышает размер рабочей области формы, выполнить масштабирование.

Рис. 1.48. Программа **Video Player** позволяет просмотреть видеоролик

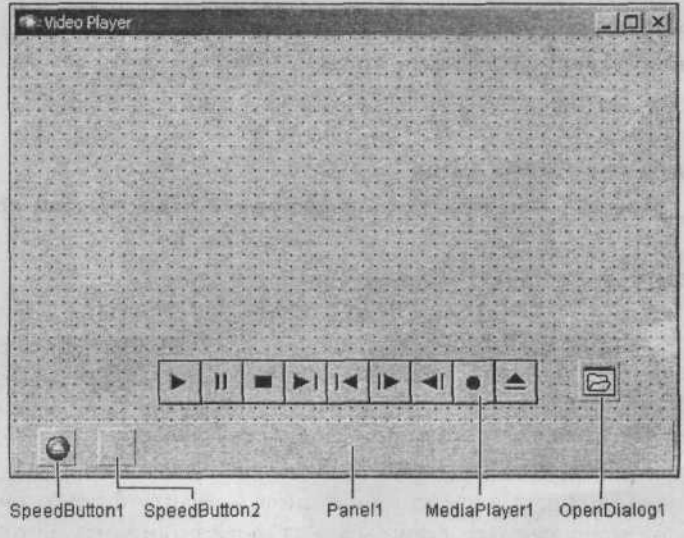


Рис. 1.49. Форма программы **Video Player**

```
// эти макросы обеспечивают перевод интервала времени,
// выраженного в миллисекундах, в минуты и секунды
#define MINUTE(ms) ((ms/1000)/60)
#define SECOND(ms) ((ms/1000)%60)

// картинки для кнопок
Graphics::TBitmap *bmPlay;   // Play
Graphics::TBitmap *bmPause;  // Pause

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    bmPlay   = new Graphics::TBitmap();
    bmPause   = new Graphics::TBitmap();

    // загрузить картинки для кнопки Play/Stop
    bmPlay->LoadFromResourceID((int)HInstance,101);
    bmPause->LoadFromResourceID((int)HInstance,102);
    // отобразить картинку
    SpeedButton2->Glyph->Assign(bmPlay);
    MediaPlayer1->Display = Form1;
}

// возвращает размер кадра
void __fastcall GetFrameSize(AnsiString f, int *w, int *h)
{
    if ( f.Pos(".avi") == 0 )
    {
        // пользователь выбрал mpg-файл
        *w = 352;
        *h = 240;
        return;
    }

    // *** Пользователь выбрал AVI-файл ***
```

```
    // В заголовке AVI-файла есть информация о размере кадра
    struct  {
        char     RIFF[4]; // строка RIFF
        long int nu_1[5]; // не используется
        char     AVIH[4]; // строка AVIH
        long int nu_2[9]; // не используется
        long int w;       // ширина кадра
        long int h;       // высота кадра
    } header;

    TFileStream *fs;   // поток для чтения заголовка файла

    /* операторы объявления потока и его создания
    можно объединить:
    TFileStream *fs = new   TFileStream(f,fmOpenRead); */

    fs = new TFileStream(f,fmOpenRead);   // открыть поток для
                                          // чтения
    fs->Read(&header, sizeof(header));    // прочитать
                                          // заголовок файла
    *w = header.w;
    *h = header.h;
    delete fs;
}

// щелчок на кнопке Eject (выбор видеоклипа)
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
    int fw, fh;   // размер кадра клипа
    int top,left; // левый верхний угол экрана
    int sw, sh;   // размер экрана (ширина, высота)

    int mw, mh;   // максимально возможный размер экрана
                  // (определяется текущим размером формы)
```

```
float kw, kh; // коэф-ты масштабирования кадра
              // по ширине и высоте
    float k;  // коэффициент масштабирования кадра

    OpenDialog1->Title = "Выбор клипа";
    OpenDialog1->InitialDir = "";
    OpenDialog1->Filter =
        "Все форматы|*.avi;*.mpg;*.mpeg|"
        "AVI|*.avi|MPG|*.mpg|MGEG|*.mpeg";

    if ( ! OpenDialog1->Execute() )
        return; // пользователь нажал кнопку Отмена

     /* При попытке открыть файл клипа, который уже открыт,
        возникает ошибка. */

    if ( MediaPlayer1->FileName == OpenDialog1->FileName )
        return;

    /* Пользователь выбрал клип. Зададим размер и положение
    "экрана", на котором будет выведен клип. Для этого надо
    знать размер кадров клипа. */

    // получить размер кадра
    GetFrameSize(OpenDialog1->FileName,&fw, &fh);

    // вычислим максимально возможный размер кадра
    mw = Form1->ClientWidth;
    mh = Form1->Panel1->Top-10;

    if ( fw < mw )
       kw = 1; // кадр по ширине меньше размера экрана
    else kw = (float) mw / fw;

    if ( fh < mh )
       kh = 1; // кадр по высоте меньше размера экрана
    else kh = (float) mh / fh;
```

```
// масштабирование должно быть пропорциональным
if ( kw < kh )
     k = kw;
else k = kh;

// здесь масштаб определен
sw = fw * k; // ширина экрана
sh = fh * k; // высота экрана

left = (Form1->ClientWidth - sw) / 2;
top = (Panel1->Top - sh) / 2;

MediaPlayer1->FileName = OpenDialog1->FileName;

MediaPlayer1->Open();
MediaPlayer1->DisplayRect = Rect(left,top,sw,sh);
/* если размер кадра выбранного клипа меньше размера
   кадра предыдущего клипа, то экран (область формы)
   надо очистить */
Form1->Canvas->FillRect(Rect(0,0,ClientWidth,Panel1->Top));

SpeedButton2->Enabled = true; // теперь кнопка Play
                              // доступна

// вывести информацию о времени воспроизведения
MediaPlayer1->TimeFormat = tfMilliseconds;
int ms = MediaPlayer1->Length;
AnsiString st = IntToStr(SECOND(ms));
if ( st.Length() == 1)
     st = "0" + st;
st = IntToStr(MINUTE(ms)) + ":" + st;
Label1->Caption = st;
Label3->Caption = "0:00";

// активизируем процесс воспроизведения
SpeedButton2->Glyph->Assign(bmPause);
```

```
    SpeedButton2->Hint = "Pause";
    SpeedButton2->Tag = 1;
    SpeedButton1->Enabled = False; // кнопка Eject недоступна
    MediaPlayer1->Play();
    Timer1->Enabled = true;
}

// щелчок на кнопке Play/Stop (воспроизведение/стоп)
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
    if (SpeedButton2->Tag == 0)
    {
        // нажата кнопка Play
        SpeedButton2->Glyph->Assign(bmPause);
        SpeedButton2->Hint = "Pause";
        SpeedButton2->Tag = 1;
        SpeedButton1->Enabled = False; // кнопка Eject
                                       // недоступна
        MediaPlayer1->Play();
        Timer1->Enabled = true;
    }
    else  // нажата кнопка Stop
    {
        MediaPlayer1->Stop();
        SpeedButton2->Glyph->Assign(bmPlay);
        SpeedButton2->Hint = "Play";
        SpeedButton2->Tag = 0;
        SpeedButton1->Enabled = True; // кнопка Eject доступна
        Timer1->Enabled = false;
    }
}

// сигнал от плеера
void __fastcall TForm1::MediaPlayer1Notify(TObject *Sender)
```

```
{
    if ( ( MediaPlayer1->Mode == mpStopped ) &&
         ( SpeedButton2->Tag == 1))
    {
        Timer1->Enabled = false;
        SpeedButton2->Glyph->Assign(bmPlay);
        SpeedButton2->Hint = "Play";
        SpeedButton2->Tag = 0;
        SpeedButton1->Enabled = True; // сделать доступной
                                      // кнопку Eject
    }
}

/* Процедура обработки события Pain обеспечивает
   отображение (перерисовку) первого кадра,
   при появлении окна, например, после того,
   как пользователь отодвинет другое окно, перекрывающее
   окон Video Player. */
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    if ( MediaPlayer1->Mode == mpStopped )
    {
        MediaPlayer1->Position = 1;
        MediaPlayer1->Position = 0;
    }
}

// завершение работы программы
void __fastcall TForm1::FormClose(TObject *Sender,
                                  TCloseAction &Action)
{
    MediaPlayer1->Close();
}

void __fastcall TForm1::Timer1Timer(TObject *Sender)
```

```
{
   // вывести информацию о времени воспроизведения
   // MediaPlayer1->TimeFormat = tfMilliseconds;
   int ms = MediaPlayer1->Position;
   AnsiString st = IntToStr(SECOND(ms));
   if ( st.Length() == 1)
        st = "0" + st;
   st = IntToStr(MINUTE(ms)) + ":" + st;
   Label3->Caption = st;
}
```

Анимация

Программа **Анимация**, ее форма и окно приведены на рис. 1.50, демонстрирует воспроизведение AVI-анимации при помощи компонента `Animate`. Анимация загружается из файла в начале работы программы. Процесс воспроизведения активизируется автоматически, в момент появления окна программы на экране. Следует обратить внимание, что компонент `Animate` обеспечивает воспроизведение только простой, не сопровождаемой звуком анимации.

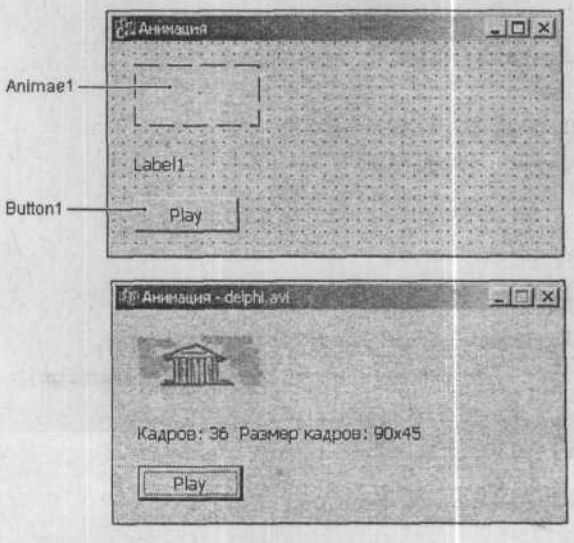
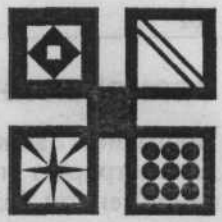


Рис. 1.50. Форма и окно программы

```
bool loaded = false; // анимация загружена

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    /* если файл анимации недоступен или анимация
       сопровождается звуком, возникает исключение */
    try
    {
        Animate1->FileName = "delphi.avi";
    }
    catch (Exception &e)
    {   }
    Form1->Caption = "Анимация - " + Animate1->FileName;
    loaded = true;
    Label1->Caption =
        "Кадров: " + IntToStr(Animate1->FrameCount) +
        "  Размер кадров: " + IntToStr(Animate1->Width) +
                    "x" + IntToStr(Animate1->Height);
}

// начало работы программы
void __fastcall TForm1::FormActivate(TObject *Sender)
{
    if ( loaded)
        // воспроизвести анимацию один раз с первого по
        // последний кадр
        Animate1->Play(1,Animate1->FrameCount,1);
}
```

```
// щелчок на кнопке Play
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if ( loaded)
        // воспроизвести анимацию один раз с первого по
        // последний кадр
        Animate1->Play(1,Animate1->FrameCount,1);
}
```



Базы данных

В состав C++Builder включены компоненты, поддерживающие различные технологии доступа к данным. В этом разделе приведены примеры использования BDE- и ADO-компонентов.

Общие замечания

- Компоненты BDE для доступа к данным используют процессор баз данных Borland Database Engine.
- Компоненты ADO для доступа к данным используют ActiveX-компоненты (библиотеки) Microsoft.
- Для того чтобы программа, которая для доступа к данным использует BDE-компоненты, могла работать с базой данных, на компьютере должен быть установлен процессор баз данных — Borland Database Engine (BDE). BDE устанавливается на компьютер программиста в процессе инсталляции C++Builder.
- База данных, для доступа к которой используются BDE компоненты, должна быть зарегистрирована в системе. Зарегистрировать базу данных, создать псевдоним (Alias) можно при помощи утилиты BDE Administrator.
- Создать базу данных (таблицу) и наполнить ее информацией можно при помощи утилиты Database Desktop или SQL Explorer. Перед тем как приступить к созданию таблицы данных надо создать псевдоним (Alias) базы данных.
- Для того чтобы перенести программу работы с базой данных на другой компьютер, надо создать установочный CD. Для решения этой задачи Borland рекомендует использовать утилиту InstallShield Express, которая поставляется вместе с C++Builder.

Записная книжка

Программа **Записная книжка**, ее форма приведена на рис. 1.51, демонстрирует использование компонентов BDE для работы с одноименной базой данных формата Paradox. База данных состоит из одной единственной таблицы adrbk.db (табл. 1.10). Программа работает с данными в режиме таблицы и позволяет просматривать, редактировать, добавлять и удалять записи, а также обеспечивает выборку (поиск) информации по содержимому поля **Name**. Для доступа к базе данных программа использует псевдоним adrbk. Создать псевдоним можно при помощи утилиты BDE Administrator.

Значения свойств компонентов `Table`, `DataSource` и `DBGrid` приведены в табл. 1.11, 1.12, 1.13 соответственно.

Таблица 1.10. *Поля таблицы adrbk базы данных Записная книжка (adrbk.db)*

Поле	Тип	Размер	Комментарий
Name	A (Alpha)	25	Имя, фамилия
Phone	A (Alpha)	20	Телефон
Cell	A (Alpha)	20	Сотовый телефон
Email	A (Alpha)	30	Адрес электронной почты

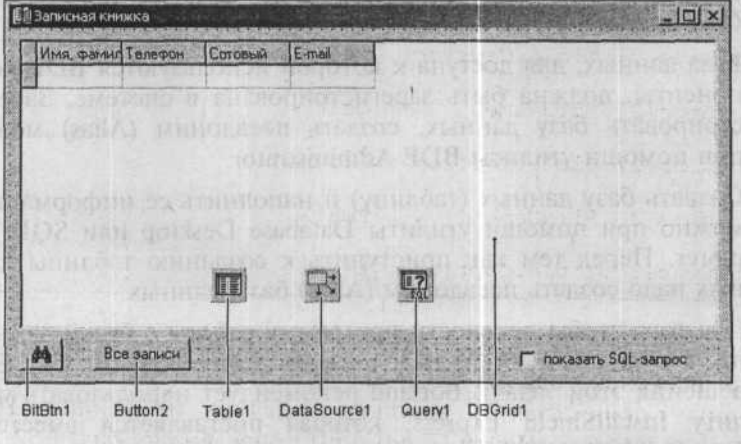


Рис. 1.51. Форма программы **Записная книжка**

Таблица 1.11. Значения свойств компонента `Table1`

Свойство	Значение	Комментарий
DatabaseName	adrbk	Псевдоним базы данных
TableName	adrbk.db	Файл, в котором находится таблица

Таблица 1.12. Значения свойств компонента `DataSource1`

Свойство	Значение
DataSet	Table1

Таблица 1.13. Значения свойств компонента `DBGrid1`

Свойство	Значение
DataSource	DataSource1
Columns[0].FildName	Name
Columns[0].Title.Caption	Имя, фамилия
Columns[1].FildName	Phone
Columns[1].Title.Caption	Телефон
Columns[2].FildName	Cell
Columns[2].Title.Caption	Сотовый
Columns[3].FildName	Email
Columns[3].Title.Caption	E-mail

Критерий запроса (имя, фамилия или фрагмент имени, например, несколько первых букв) вводится в окне **Найти** (FindForm) которое появляется в результате щелчка на кнопке **Поиск** (`BitBtn1`). Форма **Найти** (FindForm) приведена на рис. 1.52.

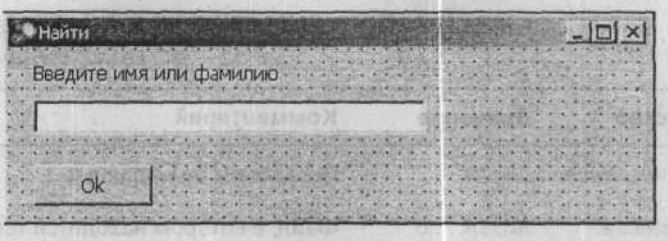


Рис. 1.52. Форма **Найти**

```
// *** Это модуль главной (стартовой) формы ***
#include "Find_.h" // эта директива вставлена вручную

// начало работы программы
void __fastcall TMainForm::FormShow(TObject *Sender)
{
  // если псевдоним adrbk не зарегистрирован, возникает ошибка
    try
    {
        Table1->Open();
    }
    catch (EDBEngineError &e)
    {
        ShowMessage("Ошибка доступа к базе данных: "
                    "не определен псевдоним adrbk\n" +
                    e.Message );

        Button2->Enabled = false;
        BitBtn1->Enabled = false;
        CheckBox1->Enabled = false;
    }
}

// щелчок на кнопке поиска информации
void __fastcall TMainForm::BitBtn1Click(TObject *Sender)
```

```
{
        FindForm->Tag = 0;
        FindForm->ShowModal();  // отобразить окно Запрос
        if ( FindForm->Tag )
        {
            // пользователь закрыл окно поиска
            // щелчком на кнопке ОК, то есть он ввел
            // фамилию или имя
            Query1->SQL->Text =
                  "SELECT * FROM adrbk WHERE Name LIKE \042%" +
                           FindForm->Edit1->Text  +"%\042";
                // \042 - это восьмеричный код двойной кавычки
            if ( CheckBox1->Checked )
                   ShowMessage (Query1->SQL->Text);

            Query1->Open(); // открыть (выполнить) запрос
             if ( Query1->RecordCount != 0)
                 DataSource1->DataSet = Query1;
             else
             {
                 ShowMessage ("В базе данных нет запрашиваемой"
                     " информации: " +  FindForm->Edit1->Text);
                    DataSource1->DataSet = Table1;
             }
        }
}
// щелчок на кнопке Все записи
void __fastcall TMainForm::Button2Click(TObject *Sender)
{
    // источник данных - таблица
    DataSource1->DataSet = Table1;
}

// завершение работы программы
```

```
void __fastcall TMainForm::FormClose(TObject *Sender,
                                     TCloseAction &Action)
{
    Table1->Close();
}


//  *** Это модуль формы Найти ***
// окно Найти стало доступным
void __fastcall TFindForm::FormShow(TObject *Sender)
{
    Edit1->SetFocus(); // установить курсор в поле
                       // редактирования
}

// Щелчок на кнопке OK (пользователь ввел критерий запроса)
void __fastcall TFindForm::Button1Click(TObject *Sender)
{
    Tag = 1; // пользователь щелкнул на кнопке OK
    Close();
}

// нажата клавиша
void __fastcall TFindForm::Edit1KeyPress(TObject *Sender,
                                         char &Key)
{
    if ( Key == 13) Button1->SetFocus(); // переместить фокус
                                         // на кнопку OK
}
```

Магазин

Программа **Магазин** (рис. 1.53) работает с одноименной базой данных и демонстрирует использование компонентов BDE.

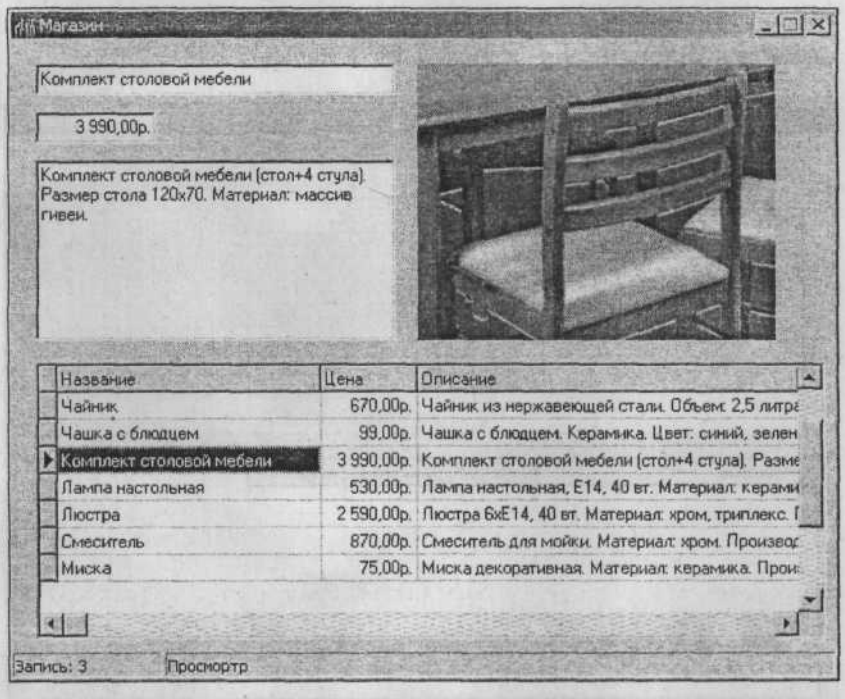


Рис. 1.53. Программа **Магазин**

База данных "Магазин" состоит из одной таблицы stock.db и содержит информацию о товарах (табл. 1.14).

Таблица. 1.14. *Поля таблицы stock (stock.db)*

Поле	Тип	Размер	Комментарий
Title	A (Alpha)	50	Название товара
Price	$ (Money)	—	Цена
Memo	A (Alpha)	100	Описание товара
Image	A (Alpha)	30	Файл иллюстрации (в формате BMP)

Для доступа к базе данных используется псевдоним stock. Создать псевдоним можно при помощи утилиты BDE Administrator.

Форма программы **Магазин** приведена рис. 1.54, значения свойств компонентов — в табл. 1.15, 1.16, 1.17, 1.18.

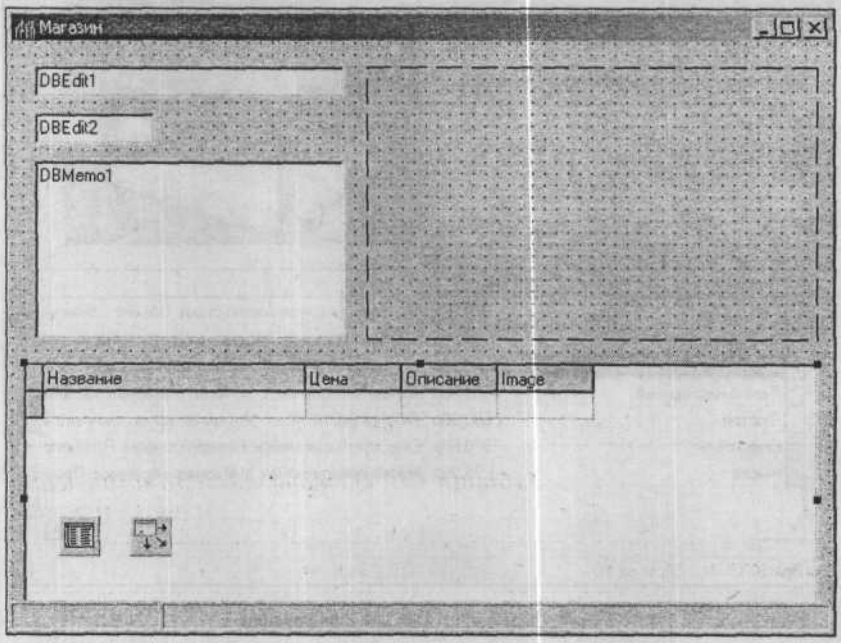


Рис. 1.54. Форма программы работы с базой данных **Магазин**

Таблица 1.15. *Значения свойств компонента* `Table1`

Свойство	Значение	Комментарий
`DatabaseName`	`stock`	Псевдоним базы данных
`TableName`	`stock.db`	Файл, в котором находится таблица

Таблица 1.16. *Значения свойств компонента* `DataSource1`

Свойство	Значение
`DataSet`	`Table1`

Таблица 1.17. *Значения свойств компонента* `DBGrid1`

Свойство	Значение
DataSource	DataSource1
Columns[0].FildName	Title
Columns[0].Title.Caption	Название
Columns[1].FildName	Price
Columns[1].Title.Caption	Цена
Columns[2].FildName	Memo
Columns[2].Title.Caption	Описание
Columns[3].FildName	Image
Columns[3].Title.Caption	Image

Таблица 1.18. *Значения свойств компонентов* `DBEdit` *и* `DBMemo`

Свойство	Значение
DBEdit1.DataSource	DataSource1
DBEdit1.DataField	Title
DBEdit2.DataSource	DataSource1
DBEdit2.DataField	Price
DBMemo1.DataSource	DataSource1
DBMemo1.DataField	Memo

```
// начало работы программы
void __fastcall TForm1::FormShow(TObject *Sender)
{
    try
    {
        Table1->Open(); // открыть базу данных
    }
```

```
    catch ( EDBEngineError &e)
    {
        ShowMessage("Для доступа к базе данных надо создать "
                    "псевдоним stock");
    }
}

// изменилось состояние набора данных
void __fastcall TForm1::DataSource1StateChange(
                                              TObject *Sender)
{
    if ( DataSource1->State == dsBrowse)
        StatusBar1->Panels->Items[1]->Text = "Просмортр";
    else
        StatusBar1->Panels->Items[1]->Text = "Редактирование";
}

// событие AfterScroll возникает после перехода к другой
// записи (смены текущей записи)
void __fastcall TForm1::Table1AfterScroll(TDataSet *DataSet)
{
    AnsiString Picture;
    if ( Table1->RecNo != -1)
    {
        StatusBar1->Panels->Items[0]->Text =
                    "Запись: " + IntToStr(  Table1->RecNo );
        /* Доступ к значению поля текущей записи можно
           получить через свойство FieldValue. Если поле Image
           пустое, то при попытке чтения из него данных
           возникает ошибка.*/
        try {
            Picture =
                Table1->Database->Directory +
                        DataSet->FieldValues["Image"];
        }
```

```
        catch (EVariantTypeCastError &e) {
            Image1->Visible = false;
            return;
        }
        ShowPhoto(Picture);
    }
    else
    {
        StatusBar1->Panels->Items[0]->Text = "";
        StatusBar1->Panels->Items[1]->Text = "Новая запись";
        Image1->Visible = false;
    }
}

// отображает картинку в поле компонента Image1
void __fastcall TForm1::ShowPhoto(AnsiString Picture)
{
    try
    {
        Image1->Picture->LoadFromFile(Picture);
    }
    catch ( EFOpenError &e)
    {
        // ничего не делаем, просто не отображаем картинку
        Image1->Visible = false;
        return;
    }
    Image1->Visible = true;
}

// завершение работы программы
void __fastcall TForm1::FormClose(TObject *Sender,
                                  TCloseAction &Action)
```

```
{
    if (Table1->State == dsEdit )
        // таблица в режиме редактирования
        Table1->Post(); // сохранить внесенные изменения
}
```

Ежедневник

Программа **Ежедневник** демонстрирует использование компонентов ADO для доступа к базе данных формата Microsoft Access.

База данных содержит информацию о запланированных мероприятиях (дата, задача). Программа позволяет вносить в базу данных изменения (добавлять, удалять и редактировать записи), а также обеспечивает выбор информации по запросу — выводит список мероприятий, запланированных "на сегодня", "на завтра" и "на эту неделю". При запуске программа автоматически выводит список мероприятий, запланированных "на сегодня" или, если программа запущена в пятницу, субботу или воскресенье, "на сегодня и ближайшие дни" (рис. 1.55).

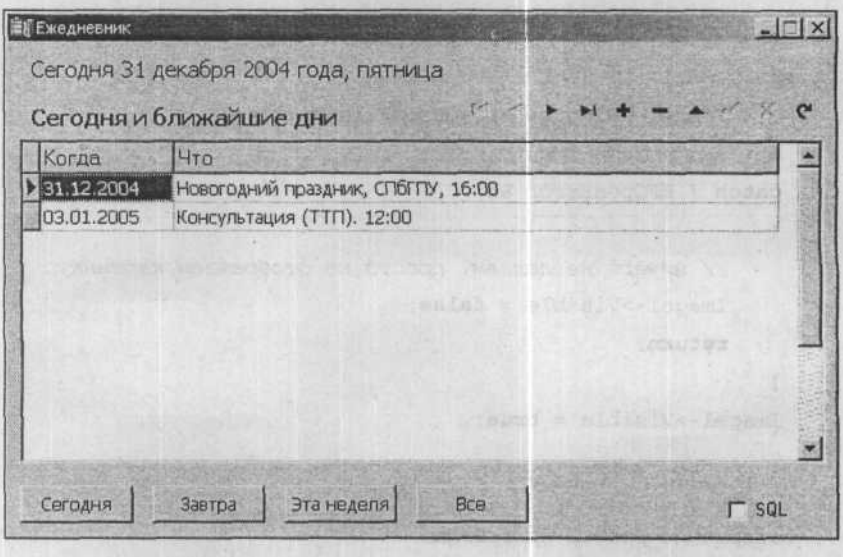


Рис. 1.55. При запуске программа выводит список дел, запланированных на ближайшие дни

База данных "Ежедневник" (Planner.mdb) состоит из одной-единственной таблицы schedule (табл. 1.19). Форма программы с базой данных приведена на рис. 1.56, значения свойств компонентов — в табл. 1.20, 1.21, 1.22, 1.23.

База данных должна быть зарегистрирована в системе как источник данных ODBC:

- драйвер — Microsoft Access Driver (*.mdb);
- имя источника данных — DPlanner;
- описание — Ежедневник;
- база данных — Planner.mdb.

Таблица. 1.19. *Поля таблицы schedule базы данных "Ежедневник" (Planner.mdb)*

Поле	Тип	Комментарий
aDate	ДАТА/ВРЕМЯ	Дата
aTask	Строковый, 50 символов	Запланированное мероприятие (задача)

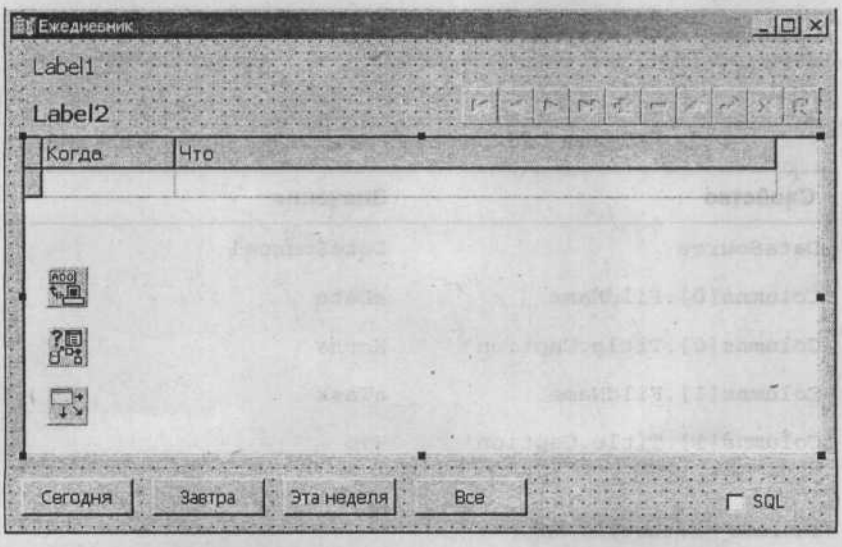


Рис. 1.56. Форма программы **Ежедневник**

Таблица 1.20. *Значения свойств компонента* `ADOConnection1`

Свойство	Значение
ConnectionString	Provider=MSDASQL.1; Persist Security Info=False; Data Source=DPlanner или DSN=DPlanner

Таблица 1.21. *Значения свойств компонента* `ADODataSet1`

Свойство	Значение
Connection	ADOConnection1
CommandText	SELECT * FROM schedule ORDER BY aDate

Таблица 1.22. *Значения свойств компонента* `DataSource1`

Свойство	Значение
DataSet	ADODataSet1

Таблица 1.23. *Значения свойств компонента* `DataGrid1`

Свойство	Значение
DataSource	DataSource1
Columns[0].FildName	aDate
Columns[0].Title.Caption	Когда
Columns[1].FildName	aTask
Columns[1].Title.Caption	Что

```
#include <DateUtils.hpp>
#include <ComObj.hpp>  // для доступа к EOleException
```

```
AnsiString stDay[7] = {"воскресенье","понедельник","вторник",
                       "среда","четверг","пятница","суббота"};

AnsiString stMonth[12] = {"января","февраля","марта",
                          "апреля","мая","июня","июля",
                          "августа","сентября","октября",
                          "ноября","декабря"};

void __fastcall TForm1::FormShow(TObject *Sender)
{
    TDateTime Today, // сегодня
              NextDay; // следующий день (не обязательно завтра)

    Word  Year, Month, Day; // год, месяц, день

    Today = Now ();

    DecodeDate(Today, Year, Month, Day);

    Label1->Caption = "Сегодня " + IntToStr(Day) + " " +
                      stMonth[Month-1] + " " +
                      IntToStr(Year) + " года, " +
                      stDay[DayOfWeek(Today) -1];

    Label2->Caption = "Сегодня и ближайшие дни";

    /* вычислим следующий день, если сегодня пятница, то,
    чтобы не забыть, что запланировано на понедельник,
    считаем, что следующий день - понедельник */
    switch ( DayOfWeek(Today) ) {
      case 6  : NextDay = Today + 3; break; // сегодня пятница
      case 7  : NextDay = Today + 2; break; // сегодня суббота
      default : NextDay = Today + 1; break;
    }
```

```
    ADODataSet1->CommandText =
    "SELECT * FROM schedule WHERE aDate BETWEEN DateValue('" +
     FormatDateTime("dd/mm/yyyy",Today) +
     "') AND DateValue('" +
     FormatDateTime("dd/mm/yyyy",NextDay) +
     "') ORDER BY aDate";

    // если надо, отобразить SQL-команду
    if ( CheckBox1->Checked) ShowSQL();

    // если БД не зарегистрирована как источник данных ODBC,
    // возникает исключение EOleException

    try
    {
      // открыть набор данных (выполнить
      // SQL-команду ADODataSet1->CommandText
      ADODataSet1->Open();
    }

    catch ( EOleException &e)
    // чтобы тип EOleException был доступен, в программу
    // надо поместить директиву #include <ComObj.hpp>
    {

       ShowMessage(
"Ошибка обращения к БД. База данных Planner.mdb должна"
"быть зарегистрирована\nв системе как источник данных ODBC "
                   "под именем dplaner"
                  );

       Button1->Enabled = false;
       Button2->Enabled = false;
       Button3->Enabled = false;
```

```
        Button4->Enabled = false;
        return;
    }

    if ( ! ADODataSet1->RecordCount )
        ShowMessage("На сегодня и ближайшие дни ни каких дел "
                    "не запланировано.");
}

// Щелчок на кнопке Сегодня
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    AnsiString today = FormatDateTime("dd/mm/yyyy",Now());

    Form1->Label2->Caption = "Сегодня";

    ADODataSet1->Close(); // закрыть набор данных

    // изменить критерий запроса
    ADODataSet1->CommandText =
        "SELECT * FROM schedule WHERE aDate =
                            DateValue('" + today +"')";

    if ( CheckBox1->Checked) ShowSQL(); // отобразить запрос

    ADODataSet1->Open(); // открыть набор данных с новым
                         // запросом
}

// щелчок на кнопке Завтра
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    AnsiString tomorrow = FormatDateTime("dd/mm/yyyy",
                                         Now() +1 );

    Label2->Caption = "Завтра";
```

```
    ADODataSet1->Close();

    // изменить критерий запроса
    ADODataSet1->CommandText =
       "SELECT * FROM schedule WHERE aDate = DateValue('" +
       tomorrow + "')";

    if ( CheckBox1->Checked) ShowSQL();

    ADODataSet1->Open();  // выполнить запрос

    if ( ! ADODataSet1->RecordCount )
    {
        ShowMessage("На завтра никаких дел не"
                    "запланировано!");
    }
}

// щелчок на кнопке На этой неделе
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  // "эта неделя" - от текущего дня до конца недели
  //(до воскресенья)
  TDateTime Present, eWeek;

  Label2->Caption = "На этой неделе";

  Present= Now(); // Now - возвращает текущую дату

  eWeek = EndOfAWeek(YearOf(Present),WeekOf(Present));
  /*
    для доступа к StartOfWeek, EndOfAWeek, YearOf и WeekOf
    надо подключить DateUtils.hpp (см. директивы #include )
  */
```

```
  ADODataSet1->Close();

  ADODataSet1->CommandText =
  "SELECT * FROM schedule WHERE aDate BETWEEN DateValue('" +
  FormatDateTime("dd/mm/yyyy",Present)+ "') AND DateValue('" +
  FormatDateTime("dd/mm/yyyy",eWeek)+"') ORDER BY aDate";

  if ( CheckBox1->Checked) ShowSQL();

  ADODataSet1->Open();

  if ( ! ADODataSet1->RecordCount )
        ShowMessage("На эту неделю никаких дел "
                    "не запланировано.");
}
// Щелчок на кнопке Все
void __fastcall TForm1::Button4Click(TObject *Sender)
{
    ADODataSet1->Close();

    ADODataSet1->CommandText =
                "SELECT * FROM schedule ORDER BY aDate";
    if ( CheckBox1->Checked) ShowSQL();
    ADODataSet1->Open();
    Label2->Caption = "Все, что намечено сделать";
}
// отображает SQL-команду
void __fastcall TForm1::ShowSQL(void)
{
    ShowMessage ( ADODataSet1->CommandText );
}
```

Игры и другие полезные программы

Сапер

Игра **Сапер**, окно которой приведено на рис. 1.57, — аналог одноименной игры, хорошо знакомой всем пользователем Windows.

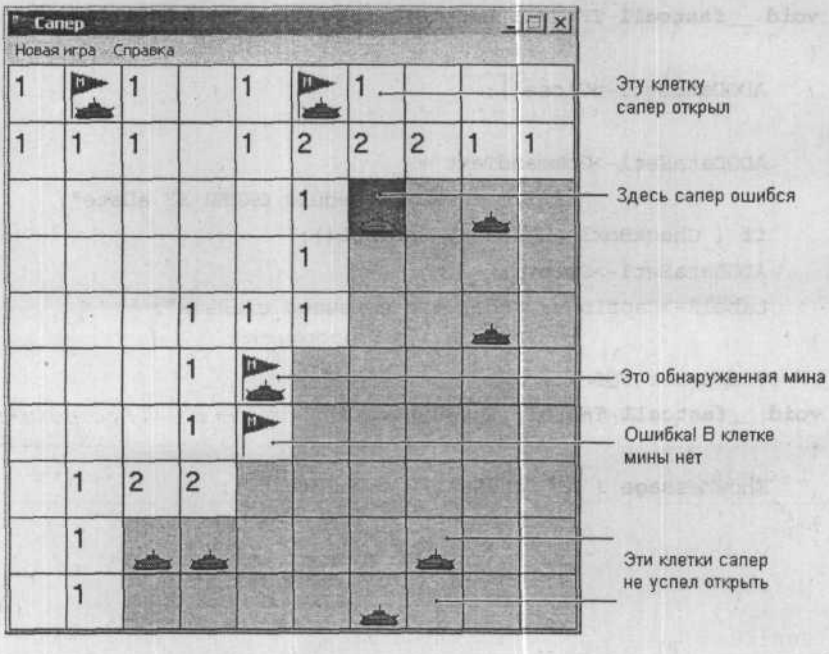


Рис. 1.57. Окно программы **Сапер**

Программа демонстрирует работу с графикой, массивами, вывод справочной информации и показывает, как можно запустить программу доступа в Internet. Главная форма и форма **О программе** приведены на рис. 1.58, 1.59 соответственно. Окно **О программе** появляется на экране в результате выбора в меню **Справка** команды **О программе**. Значения свойств формы **О программе** приведены в табл. 1.24.

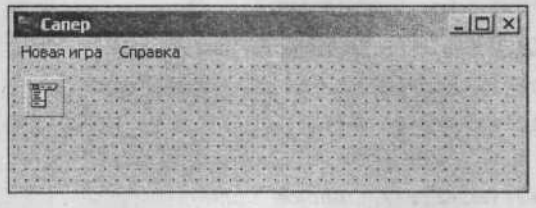


Рис. 1.58. Главная форма программы **Сапер**

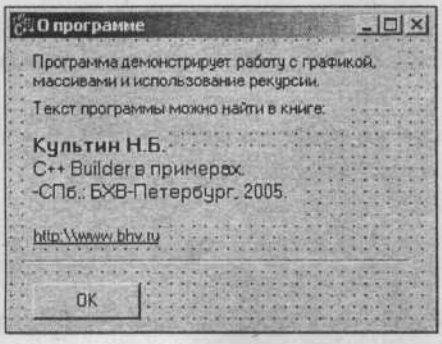


Рис. 1.59. Форма **О программе**

Таблица 1.24. *Значения свойств формы* ***О программе***

Свойство	Значение
Name	About
BorderStyle	bsToolWindow
Position	poOwnerFormCenter

```
// *** модуль главной формы ***
#include <stdlib.h>  // для доступа к
                     // генератору случайных чисел
#include <time.h>
#include "SaperAbout.h"

TMain *Main; // главное окно

#define MR 10 // кол-во клеток по вертикали
#define MC 10 // кол-во клеток по горизонтали
#define NM 10 // кол-во мин

int Pole[MR+2][MC+2]; // минное поле
                      // 0..8 - количество мин в соседних клетках
                      // 9 - в клетке мина
                      // 100..109 - клетка открыта
                      // 200..209 - в клетку поставлен флаг

int nMin;   // кол-во найденных мин
int nFlag;  // кол-во поставленных флагов

int status = 0; // 0 - начало игры; 1 - игра; 2 - результат

// смещение игрового поля относительно левого верхнего угла
// поверхности формы
#define LEFT 0   // по X
#define TOP  1   // по Y

#define W    40  // ширина клетки поля
#define H    40  // высота клетки поля

// новая игра - "разбрасывает" мины
void __fastcall NewGame();
// открывает текущую и соседние пустые клетки
```

```
void __fastcall Open(int row, int col);

// нажатие кнопки мыши на игровом поле
void __fastcall TMain::FormMouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int x, int y)
{
    if ( status == 2 ) return;
    if ( status == 0) status = 1;

    x -= LEFT;
    y -= TOP;
    if (x > 0 && y > 0)
    {
      // преобразуем координаты мыши в индексы
      // клетки поля
      int row = y/H + 1;
      int col = x/W + 1;
      if (Button == mbLeft)
      {
        if ( Pole[row][col] == 9 )
        {
          Pole[row][col] +=100;
          status = 2;  // игра закончена
          ShowPole(status);
        }
        else if  ( Pole[row][col] < 9 )
            {
              Open(row,col);
              ShowPole(status);
            }
      }
      else if (Button == mbRight)
          {
            nFlag++;
```

```
                if ( Pole[row][col] == 9 )
                   nMin++;
                Pole[row][col] += 200;  // поставили флаг
                if (nMin == NM && nFlag == NM)
                {
                  status = 2;  // игра закончена
                  ShowPole(status);
                }
                else Kletka(row, col, status);
              }
    }
}

// Функция обработки события OnCreate обычно используется
// для инициализации глобальных переменных
void __fastcall TMain::FormCreate(TObject *Sender)
{
   // В неотображаемые эл-ты массива, которые соответствуют
   // клеткам по границе игрового поля, запишем число -3.
   // Это значение используется функцией Open для завершения
   // рекурсивного процесса открытия соседних пустых клеток.
   for ( int row=0; row <= MR+1; row++)
        for ( int col=0; col <= MC+1; col++)
       Pole[row][col] = -3;

   NewGame();  // "разбросать" мины
   ClientWidth = W*MC;
   ClientHeight = H*MR+TOP+1;
}

// Вывод поля как результат обработки события Paint
// позволяет проводить любые манипуляции с формой во время
// работы программы
void __fastcall TMain::FormPaint(TObject *Sender)
```

```
{
        ShowPole(status);
}

// Показывает поле
void __fastcall TMain::ShowPole( int status)
{
    for ( int row = 1; row <= MR; row++ )
         for ( int col = 1; col <= MC; col++ )
              Kletka(row, col, status);
}

// рисует на экране клетку
void __fastcall TMain::Kletka(int row, int col, int status)
{
     int x = LEFT + (col-1)* W;
     int y = TOP + (row-1)* H;

     if (status == 0) // начало игры
     {
        // клетка - серый квадрат
        Canvas->Brush->Color = clBtnFace;
        Canvas->Rectangle(x-1,y-1,x+W,y+H);
        return;
     }

    // во время (status = 1) и в конце (status = 2) игры
    if ( Pole[row][col] < 100 )
    {
         // клетка не открыта
         Canvas->Brush->Color = clBtnFace;  // не открытые -
                                            // серые
         Canvas->Rectangle(x-1,y-1,x+W,y+H);
         if (status == 2 && Pole[row][col] == 9)
             Mina( x, y); // игра закончена, показать мину
```

```
        return;
    }

    // клетка открыта
    Canvas->Brush->Color = clWhite;       // открытые – белые
    Canvas->Rectangle(x-1,y-1,x+W,y+H);
    if  ( Pole[row][col] == 100 ) // клетка открыта, но она
                                  // пустая
        return;

    if ( Pole[row][col] >= 101 && Pole[row][col] <= 108 )
    {
            Canvas->Font->Size = 11;
            Canvas->Font->Color = clBlue;
            Canvas->TextOutA(x+3, y+2,
                        IntToStr(Pole[row][col] -100 ));
            return;
    }

    if ( Pole[row][col] >= 200 )
        Flag(x, y);

    if (Pole[row][col] == 109 ) // на этой мине подорвались!
    {
        Canvas->Brush->Color = clRed;
        Canvas->Rectangle(x-1,y-1,x+W,y+H);
    }
    if (( Pole[row][col] % 10 == 9)  &&  (status == 2))
            Mina( x, y);
}

// рекурсивная функция открывает текущую и все соседние
// клетки, в которых нет мин
void __fastcall Open(int row, int col)
```

```
{
    if (Pole[row][col] == 0)
    {
        Pole[row][col] = 100;
        // открываем клетки слева, справа, снизу и сверху
        Open(row,col-1);
        Open(row-1,col);
        Open(row,col+1);
        Open(row+1,col);
        // открываем примыкающие диагонально
        Open(row-1,col-1);
        Open(row-1,col+1);
        Open(row+1,col-1);
        Open(row+1,col+1);
    }
    else
        // -3 это граница игрового поля
        if (Pole[row][col] < 100 && Pole[row][col] != -3)
            Pole[row][col] += 100;
}

// новая игра - генерирует новое поле
void __fastcall NewGame()
{
   // Очистим эл-ты массива, соответствующие отображаемым
   // клеткам, а в неотображаемые (по границе игрового поля)
   // запишем число -3. Уникальное значение клеток границы
   // используется функцией Open для завершения рекурсивного
   // процесса открытия соседних пустых клеток.
   int row,col;
   for (row=0; row <= MR+1; row++)
       for (col=0; col <= MC+1; col++)
           Pole[row][col] = -3;
   for (row=1; row <= MR; row++)
```

```
        for (col=1; col <= MC; col++)
            Pole[row][col] = 0;

// расставим мины
time_t t;     // используется ГСЧ
srand((unsigned) time(&t)); // инициализация ГСЧ
int n = 0; // кол-во мин
do
{
    row = rand() % MR +1;
    col = rand() % MC +1;
    if ( Pole[row][col] != 9)
    {
        Pole[row][col] = 9;
    n++;
    }
}
while ( n < 10);

// вычисление кол-ва мин в соседних клетках
int k;
for ( row = 1; row <= MR; row++)
    for ( col = 1; col <= MC; col++)
    if ( Pole[row][col] != 9 ) {
     k =0;
     if ( Pole[row-1][col-1] == 9 ) k++;
     if ( Pole[row-1][col]   == 9 ) k++;
     if ( Pole[row-1][col+1] == 9 ) k++;
     if ( Pole[row][col-1]   == 9 ) k++;
     if ( Pole[row][col+1]   == 9 ) k++;
     if ( Pole[row+1][col-1] == 9 ) k++;
     if ( Pole[row+1][col]   == 9 ) k++;
     if ( Pole[row+1][col+1] == 9 ) k++;
     Pole[row][col] = k;
    }
```

```
    status = 0; // начало игры
    nMin = 0;   // нет обнаруженных мин
    nFlag = 0;  // нет флагов
}

// рисует мину
void __fastcall TMain::Mina(int x, int y)
{
    Canvas->Brush->Color = clGreen;
    Canvas->Pen->Color = clBlack;
    Canvas->Rectangle(x+16,y+26,x+24,y+30);

    // корпус
    Canvas->Rectangle(x+8,y+30,x+32,y+34);
    Canvas->Pie(x+6,y+28,x+34,y+44,x+34,y+36,x+6,y+36);

    // полоса на корпусе
    Canvas->MoveTo(x+12,y+32); Canvas->LineTo(x+28,y+32);

    // основание
    Canvas->MoveTo(x+8,y+36);  Canvas->LineTo(x+32,y+36);

    // вертикальный "ус"
    Canvas->MoveTo(x+20,y+22); Canvas->LineTo(x+20,y+26);

    // боковые "усы"
    Canvas->MoveTo(x+8, y+30); Canvas->LineTo(x+6,y+28);
    Canvas->MoveTo(x+32,y+30); Canvas->LineTo(x+34,y+28);
}

// Рисует флаг
void __fastcall TMain::Flag( int x, int y)
{
    TPoint p[4]; // координаты флажка и нижней точки древка

    // точки флажка
```

```
    p[0].x=x+4;   p[0].y=y+4;
    p[1].x=x+30;  p[1].y=y+12;
    p[2].x=x+4;   p[2].y=y+20;

    // установим цвет кисти и карандаша
    Canvas->Brush->Color = clRed;
    Canvas->Pen->Color = clRed; // чтобы контур флажка был
                                // красный
    Canvas->Polygon(p, 2);  // флажок

    // древко
    Canvas->Pen->Color = clBlack;
    Canvas->MoveTo(p[0].x, p[0].y);
    Canvas->LineTo(x+4,y+36);

    TPoint m[5];      // буква M

    m[0].x=x+8; m[0].y=y+14;
    m[1].x=x+8; m[1].y=y+8;
    m[2].x=x+10; m[2].y=y+10;
    m[3].x=x+12; m[3].y=y+8;
    m[4].x=x+12; m[4].y=y+14;
    Canvas->Pen->Color = clWhite;
    Canvas->Polyline(m,4);
    Canvas->Pen->Color = clBlack;
}

// команда главного меню Новая игра
void __fastcall TMain::N1Click(TObject *Sender)
{
        NewGame();
        ShowPole(status);
}
```

```
// выбор в меню "?" команды О программе
void __fastcall TMain::N4Click(TObject *Sender)
{
    About->ShowModal();
}

// выбор в меню "?" команды Справка
void __fastcall TMain::N3Click(TObject *Sender)
{
    /* Отображение справочной информации
       обеспечивает утилита hh.exe, входящая
       в состав Windows. Ключ mapid задает
       отображаемый раздел справочной информации. */
    WinExec("hh.exe -mapid 1 saper.chm", SW_RESTORE);
}

// *** модуль формы О программе***
// Выбор URL-адреса (щелчок в поле компонента Label5)
void __fastcall TAbout::Label5Click(TObject *Sender)
{
  /* наиболее просто передать в функцию ShellExecute
     строку-константу (URL-адрес) так, как показано ниже
    ShellExecute(AboutForm->Handle,
                 "open",
                 "http:\\\\www.bhv.ru",
                 NULL,NULL)
    Лучше URL-адрес брать из поля метки.
    В функцию ShellExute надо передать указатель (char*) на
    null terminated строку, но свойство Caption -
    это AnsiString.
    Преобразование Ansi строки в (char*) строку выполняет
    метод c_str()
*/
```

```
    // открыть файл, имя которого находится в поле Label5
    ShellExecute(About->Handle,"open",Label5->Caption.c_str(),
                 NULL,NULL,SW_RESTORE);
}

// щелчок на кнопке OK
void __fastcall TAbout::Button1Click(TObject *Sender)
{
    ModalResult = mrOk;
}
```

Игра 15

Всем известна игра "15". Вот ее правила. В прямоугольной коробочке находятся 15 фишек, на которых написаны числа от 1 до 15. Размер коробочки — 4×4, таким образом в коробочке есть одна пустая ячейка. В начале игры фишки перемешаны (рис. 1.60). Задача игрока состоит в том, чтобы, не вынимая фишки из коробочки, выстроить фишки в правильном порядке (рис. 1.61).

Рис. 1.60. В начале игры фишки перемешаны

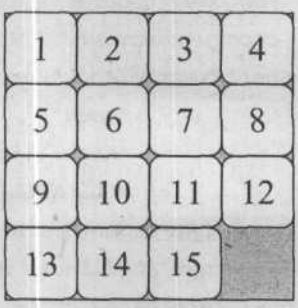


Рис. 1.61. Правильный порядок фишек

Программа **Игра 15** реализует описанную игру. Форма и окно программы приведены на рис. 1.62.

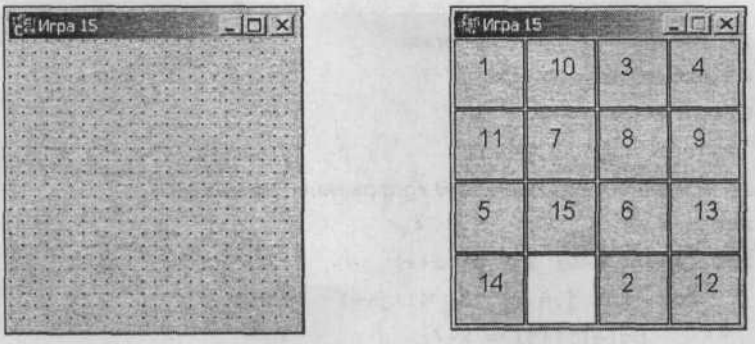


Рис. 1.62. Форма и окно программы **Игра 15**

```
#include "math.hpp" // для доступа к Randomize и RandomRange

// размер клеток 48x48
#define WC 48
#define HC 48

byte pole[4][4]; // игровое поле
byte ex,ey;      // координаты пустой клетки

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    Form1->Font->Size = 12;
}

// начало работы программы
void __fastcall TForm1::FormShow(TObject *Sender)
{
    NewGame();
}

// новая игра
void __fastcall TForm1::NewGame()
```

```
{
    // установить размер формы
    ClientWidth = WC *\4;
    ClientHeight = HC * 4;

    // исходное (правильное) положение фишек
    int k = 1;
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            pole[i][j] = k++;

    Mixer();      // перемешать фишки
    ShowPole();   // отобразить игровое поле
}

// перемешивает фишки
void __fastcall TForm1::Mixer()
{
    int x1,y1; // пустая клетка
    int x2,y2; // эту переместить в пустую
    int d;     // направление относительно пустой

    Randomize();

    x1 = 3; y1 = 3; // см. описание массива stp
    for ( int i = 0; i < 150; i++)  // кол-во перестановок
    {
        do {
            x2 = x1;
            y2 = y1;
            /* выберем фишку, примыкающую к пустой клетке,
               которую переместим в пустую клетку */
            d = RandomRange(1,5);
            switch ( d ) {
```

```
                case 1: x2--; break;
                case 2: x2++; break;
                case 3: y2--; break;
                case 4: y2++; break;
            }
        } while ((x2 < 0) || (x2 >= 4) || (y2 < 0) || (y2 >=
4));

        /* здесь определили фишку, которую
           надо переместить в пустую клетку */
        pole[y1][x1] = pole[y2][x2];
        pole[y2][x2] = 16;
        x1 = x2;
        y1 = y2;
    };
    // запомним координаты пустой клетки
    ex = x1;
    ey = y1;
}

// отображает на поверхности формы игровое поле
void __fastcall TForm1::ShowPole()
{
    int x,y; // координаты левого верхнего угла клетки

    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
        {
            x = j * HC;
            y = i * WC;
            if (pole[i][j] != 16 ) {
                Canvas->Brush->Color = clBtnFace;
                Canvas->Rectangle(x,y, x+WC-1, y+HC-1);
```

```
                Canvas->TextOutA(x+15,y+10,
                                 IntToStr(pole[i][j]));
            }
            else {
                Canvas->Brush->Color = clBtnHighlight;
                Canvas->Rectangle(x,y, x+WC-1, y+HC-1);
            }
        }
}

bool Finish(); // проверяет, правильно ли размещены фишки

// щелчок в клетке
void __fastcall TForm1::FormMouseDown(TObject *Sender,
        TMouseButton Button, TShiftState Shift, int X,
      int Y)
{
    int cx,cy; // координаты клетки

    cx = X / WC;
    cy = Y / HC;

    // переместить выбранную клетку в соседнюю свободную
    if  ( ( abs(cx - ex) == 1  &&  cy-ey == 0  ) ||
          ( abs(cy - ey) == 1  &&  cx-ex == 0  ) )
    {
        // переместить фишку из (cx,cy) в (ex,ey)
        pole[ey][ex] = pole[cy][cx];
        pole[cy][cx] = 16;
        ex = cx;
        ey = cy;
        ShowPole();     // отрисовать поле
        if ( Finish () )
        {
            ShowPole();
```

```
            int r = MessageDlg ("Цель достигнута!\Еще раз?",
                    mtInformation, TMsgDlgButtons() << mbYes
                                                    << mbNo, 0);
            if ( r == mrNo )
                Form1->Close(); // завершить работу программы
            else
            {
                NewGame();
                ShowPole();
            }
        }
    }
}

/* проверяет, расположены ли
  клетки (фрагменты картинки) в нужном порядке */
bool Finish()
{
    bool result;
    int row, col;
    int k = 1;

    result  =  true; // пусть фишки в нужном порядке
    for (row = 0; row < 4; row++)
    {
        for (col = 0; col < 4; col++)
            if ( pole[row][col] == k )
                k++;
            else {
                result = false;
                break;
            }
        if ( ! result ) break;
    }
```

```
    return (result);
};

// обработка события Paint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    ShowPole();
}
```

Игра "Собери картинку" (Puzzle)

Игра **Собери картинку** — аналог игры "15", но в отличие от последней, на фишках нарисованы не цифры, а фрагменты картинки (рис. 1.63). Задача игрока — расположить фишки в правильном порядке. Игра заканчивается, когда картинка будет собрана.

Рис. 1.63. Окно программы **Собери картинку**

Программа загружает картинку из файла формата BMP, имя которого указано в командной строке запуска программы, или из файла, который находится в каталоге программы. Если в каталоге программы несколько файлов с расширением bmp, то будет

загружен первый по порядку файл. При повторной активизации игры в результате выбора в строке меню команды **Новая игра** загружается следующий файл с расширением bmp.

Форма программы **Собери картинку** приведена на рис. 1.64.

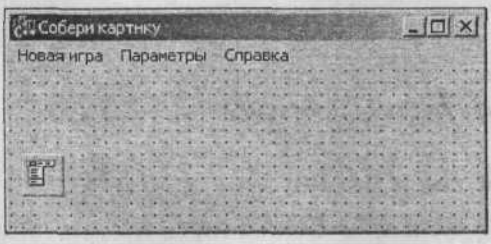


Рис. 1.64. Форма программы **Собери картинку**

```
#include "math.hpp" // для доступа к Randomize и RandomRange

// размер поля WxH
#define W 4
#define H 4

int wc, hc; // размер клетки

byte pole[H][W]; // игровое поле
byte ex,ey;      // координаты пустой клетки

bool GameOver;

AnsiString fn; // имя bmp-файла (картинка)
TSearchRec SearchRec; // результат поиска файла

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    pic = new Graphics::TBitmap();
}
```

```
// начало работы программы
void __fastcall TForm1::FormShow(TObject *Sender)
{
    NewGame();
}

// новая игра
void __fastcall TForm1::NewGame()
{
    static int Tag = 0;
       // загрузить файл иллюстрации
    if (( ParamCount() == 0 ) && (Tag == 0 )) Tag = 1;

    switch ( Tag )
    {
        case 0 :  // имя файла - из командной строки
              fn = ParamStr(1);
              Tag = 1;
              break;

        case 1:  // выбрать первый по порядку bmp-файл
              {
                FindFirst("*.bmp",faAnyFile,SearchRec);
                fn = SearchRec.Name;
                Tag = 2;
              }
              break;

        case 2: // выбрать следующий bmp-файл
              {
                 if ( FindNext(SearchRec) != 0)
                     FindFirst("*.bmp",faAnyFile,SearchRec);
                 fn = SearchRec.Name;
              }
```

```
            break;
    }

    // загрузить иллюстрацию
    try {
        pic->LoadFromFile(fn);
    }
    catch (EFOpenError &e)
    {
        MessageDlg("Ошибка доступа к файлу иллюстрации",
          mtWarning, TMsgDlgButtons()<< mbOK<<mbHelp,0);
        return;
    }

    // определить размер клетки
    wc = pic->Width / W;
    hc = pic->Height / H;

    // установить размер формы
    ClientWidth = wc * W;
    ClientHeight = hc * H;

    // исходное (правильное) положение фишек
    int k = 1;
    for (int i = 0; i < H; i++)
        for (int j = 0; j < W; j++)
            pole[i][j] = k++;

    GameOver = false;
    Mixer();      // перемешать фишки
    ShowPole();   // отобразить игровое поле
}
```

```
// перемешивает фишки
void __fastcall TForm1::Mixer()
{
    int x1,y1; // пустая клетка
    int x2,y2; // эту переместить в пустую
    int d;     // направление относительно пустой

    Randomize();

    x1 = 3; y1 = 3; // см. описание массива stp
    for ( int i = 0; i < 150; i++)  // кол-во перестановок
    {
        do {
            x2 = x1;
            y2 = y1;
            // выберем фишку, примыкающую к пустой клетке,
            // которую переместим в пустую клетку
            d = RandomRange(1,5);
            switch ( d ) {
                case 1: x2--; break;
                case 2: x2++; break;
                case 3: y2--; break;
                case 4: y2++; break;
            }
        } while ((x2 < 0) || (x2 >= W) ||
                 (y2 < 0) || (y2 >= H));

        /* здесь определили фишку, которую
           надо переместить в пустую клетку */

        pole[y1][x1]  =  pole[y2][x2];
        pole[y2][x2]  =  16;
```

```
        x1 = x2;
        y1 = y2;
    };
    // запомним координаты пустой клетки
    ex =  x1;
    ey =  y1;
}

// отображает на поверхности формы игровое поле
void __fastcall TForm1::ShowPole()
{
    TRect src, dst; // фрагмент картинки и область ее
                    // ее отображения на поверхности формы
    int sx,sy;

    for (int i = 0; i < H; i++)
        for (int j = 0; j < W; j++)
        {
            // Преобразуем номер фрагмента картинки в
            // координаты левого
            // верхнего угла области-источника
            sx = ((pole[i][j]-1) % W) * wc;
            sy = ((pole[i][j]-1) / H) * hc;

            src = Bounds(sx,sy,wc,hc);
            dst = Bounds(j*wc,i*hc,wc,hc);
            if ((  pole[i][j] != 16 ) || GameOver )
                // фрагмент картинки
                Canvas->CopyRect(dst,pic->Canvas,src);
            else
            {
                // пустая клетка
                Canvas->Brush->Style = bsSolid;
```

```
                Canvas->Brush->Color = clBtnFace;
                Canvas->Rectangle(dst);
            }
        }
    if ( N6->Checked )
    {
        // вывести номер фишки
        Canvas->Brush->Style = bsClear;
        for (int i = 0; i < H; i++)
        for (int j = 0; j < W; j++)
            Canvas->TextOutA(wc*j,hc*i, IntToStr(pole[i][j]));
    }
}

// щелчок в клетке
void __fastcall TForm1::FormMouseDown(TObject *Sender,
        TMouseButton Button, TShiftState Shift, int X, int Y)
{
    int cx,cy; // координаты клетки

    cx = X / wc;
    cy = Y / hc;

    Move(cx,cy); // переместить выбранную клетку в соседнюю
                 // свободную
}

bool Finish(); // проверяет, правильно ли размещены фишки

// перемещает фишку из клетки, в которой сделан щелчок
// в свободную клетку
void __fastcall TForm1::Move(int cx, int cy)
```

```
{
    if  ( ( abs(cx - ex) == 1  &&  cy-ey == 0  ) ||
          ( abs(cy - ey) == 1  &&  cx-ex == 0  ) )
    {
        // переместить фишку из (cx,cy) в (ex,ey)
        pole[ey][ex] = pole[cy][cx];
        pole[cy][cx] = 16;
        ex = cx;
        ey = cy;
        // отрисовать поле
        ShowPole();
        if ( Finish () )
        {
            GameOver = true;
            ShowPole();
            int r = MessageDlg ("Цель достигнута! "
                        "Еще раз (другая картинка)?",
                  mtInformation,
                  TMsgDlgButtons() << mbYes << mbNo, 0);
            if ( r == mrNo )
                Form1->Close(); // завершить работу программы
            else
            {
                NewGame();
                ShowPole();
            }
        }
    }
}

// проверяет, расположены ли клетки (фрагменты картинки) в
// нужном порядке
bool Finish()
```

```
{
    bool result;
    int row, col;
    int k = 1;

    result  =  true; // пусть фишки в нужном порядке
    for (row = 0; row < H; row++)
    {
        for (col = 0; col < W; col++)
            if ( pole[row][col] == k )
                k++;
            else {
                result = false;
                break;
            }
        if ( ! result ) break;
    }
     return (result);
};

// обработка события Paint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    ShowPole();
}

// выбор в строке меню команды Новая игра
void __fastcall TForm1::N1Click(TObject *Sender)
{
    NewGame();
}

// выбор в меню Справка команды Справка
```

```
void __fastcall TForm1::N3Click(TObject *Sender)
{
    WinExec("hh.exe puzzle.chm", SW_RESTORE);
}

// выбор в меню Справка команды О программе
void __fastcall TForm1::N4Click(TObject *Sender)
{
    WinExec("hh.exe -mapid 3 puzzle.chm", SW_RESTORE);

}

// команда Параметры/Номер фишки
void __fastcall TForm1::N6Click(TObject *Sender)
{
    N6->Checked = ! N6->Checked;
    ShowPole();
}
```

Игра "Парные картинки"

Игра **Парные картинки** развивает внимание. Вот ее правила. Игровое поле разделено на клетки, за каждой из которых скрыта картинка. Картинки парные, т. е. на игровом поле есть две клетки, в которых находятся одинаковые картинки. В начале игры все клетки "закрыты". Щелчок левой кнопкой мыши "открывает" клетку, в клетке появляется картинка. Теперь надо найти клетку, в которой находится такая же картинка, как и в открытой клетке. Щелчок по другой клетке открывает вторую картинку (рис. 1.65). Если картинки в открытых клетках одинаковые, то эти клетки "исчезают". Если разные — то клетки остаются открытыми. Следующий щелчок закрывает открытые клетки и открывает следующую. Следует обратить внимание, что две открытые клетки закрываются даже в том случае, если открытая картинка такая же, как и одна из двух открытых. Игра заканчивается, когда игрок откроет — "найдет" все пары картинок.

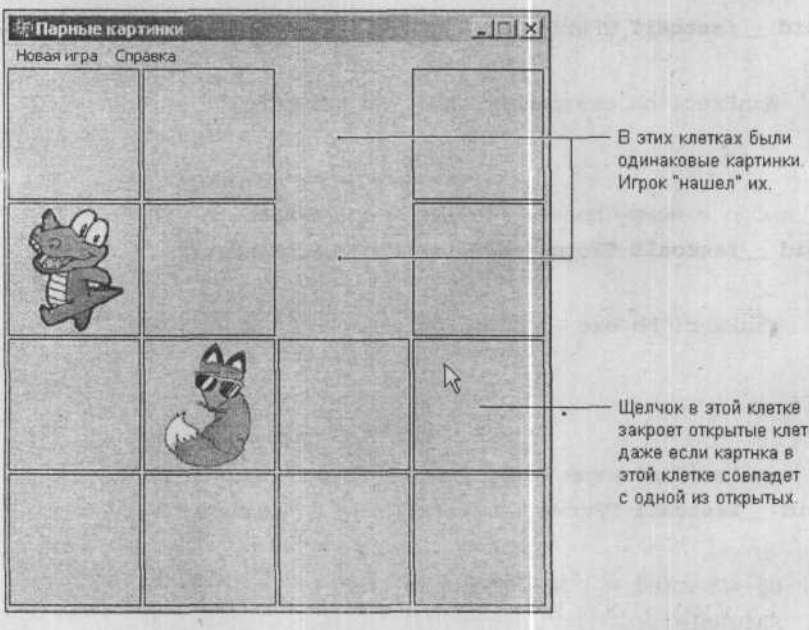


Рис. 1.65. Игровое поле программы **Парные картинки**

Программа, реализующая игру **Парные картинки**, демонстрирует работу с графикой. В приведенной реализации игры все картинки квадратные и находятся в одном файле (рис. 1.66). Это позволило сделать программу "интеллектуальной" — размер игрового поля (количество клеток по горизонтали и вертикали) определяется количеством картинок в файле: зная высоту и ширину картинки в файле, программа вычисляет размер и количество картинок и устанавливает соответствующий размер игрового поля.

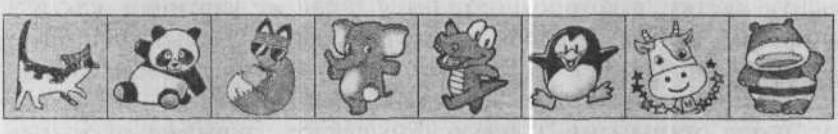


Рис. 1.66. Все картинки находятся в одном файле

Форма программы **Парные картинки** приведена на рис. 1.67. Таймер используется для организации задержки исчезновения открытых клеток, в которых находятся одинаковые картинки.

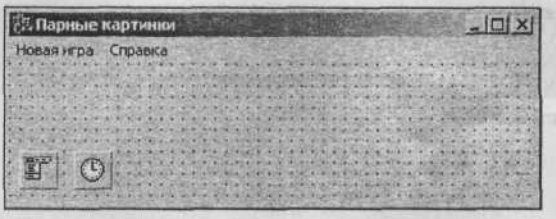


Рис. 1.67. Форма программы **Парные картинки**

```
#include <Math.hpp>  // для доступа к генератору случайных
чисел

#define  MAX_SIZE  32 // максимальное кол-во пар картинок
#define  MAX_H  8     // максимальный размер поля - 8x8
#define  MAX_W  8

// #define DEBUG    // режим отладки

int  Pole[MAX_W][MAX_H]; // поле
     /* Pole[i][j] < 100 - код картинки, клетка закрыта;
        Pole[i][j] >= 100 но < 200 - клетка открыта
                                    (игрок видит картинку);
        Pole[i][j] >= 200 - игрок нашел пару для этой картинки
*/

Graphics::TBitmap  *Pictures;     // картинки

int np;  // количество пар картинок
int nf;  // кол-во открытых (найденных) пар картинок
int no;  // количество открытых в данный момент клеток

TPoint open1;  // координаты 1-ой открытой клетки
TPoint open2;  // координаты 2-ой открытой клетки

int W,H;     // Кол-во клеток по горизонтали и вертикали.
             // Произведение W и H должно быть кратно 2-м

int WK, HK;  // размер клетки (картинки)
```

```
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  int np; // кол-во картинок в файле pictures

  Pictures  =  new Graphics::TBitmap();

  try {
    Pictures->LoadFromFile("pictures.bmp");
  }
  catch (EFOpenError &e)
  {
   ShowMessage("Ошибка доступа к файлу картинок.");
   return;
  }

  /* В файле pictures находятся все картинки. Предполагается,
     что каждая картинка квадратная.

     Считаем, что в файле 8,10,12,14 или 15 картинок:

    Картинок   Клеток    Поле
    ---------------------------------------
       8         16       4x4
       10        20       4x5
       12        24       4x6
       14        28       4x7
       15        30       5x6
 */
```

```
  HK  =  Pictures->Height-1; // высота картинки
  WK  =  HK;                 // ширина картинки

  np =  Pictures->Width / WK;
  if ( np < 15)
    H  =  4;
  else H  = 5;

  W  =  (np*2)/H;

  // установить размера поля (формы)
  ClientHeight  =  H * HK;
  ClientWidth  =  W * WK;

  // настройка таймера
  Timer1->Enabled  =  False;
  Timer1->Interval  =  200;

  NewGame();
}

// новая игра
void __fastcall TForm1::NewGame()
{
  /* В каждую ячейку Pole надо записать номер картинки.
     Так как для каждой картинки должна быть пара, то
     число i должно быть в двух ячейках Pole */

  int r;        // случайное число

  int buf[MAX_SIZE];
  /* в buf[i] записываем, сколько раз число i
    записали в массив Pole */

  int i, j; // индексы массивов
```

```
  // запишем в массив Pole случайные числа
  // от 0 до np, где n - кол-во картинок
  // каждое число должно быть записано два раза

  for (i = 0; i < np; i++)
      buf[i] = 0;

  Randomize(); // инициализация ГСЧ
  for (i = 0; i < H; i++)
    for (j = 0; j < W; j++) {
        do {
          r = RandomRange(0,np);
        } while ( buf[r] == 2 );
        Pole[i][j] = r;   // код картинки
        buf[r]++;
    }
  // здесь поле сгенерировано
   nf = 0;
};

// отрисовывает поле
void __fastcall TForm1::ShowPole()
{
   int row, col;

   for ( row = 0; row < H; row++)
      for (col = 0; col < W; col++)
          Kletka(row,col);
}

// рисует клетку поля
void __fastcall TForm1::Kletka(int col,int row)
```

```
{
  int x, y;        // левый верхний угол клетки (координаты)
  TRect src, dst; // источник и получатель битового образа

  // преобразуем координаты клетки
  // в координаты на поверхности формы
  x  =  (col)*WK;
  y  =  (row)*HK;

  if ( Pole[col][row] >= 200 ) {
        /* Для этой клетки найдена пара.
           Клетку надо убрать с поля */


     // установить цвет границы и закраски области
     Canvas->Brush->Color =  clBtnFace;
     Canvas->Pen->Color =  clBtnFace;
     Canvas->Rectangle(x, y, x+WK-2, y+HK-2);
     return;
  };

  if ( (Pole[col][row] >= 100) && (Pole[col][row] < 200) ) {
      // клетка открыта - вывести картинку

     // Pole[col,row] == номер картинки + 100,
     // где 100 - признак того, что клетка открыта
     // определим положение картинки в Pictures
     src  =  Bounds((Pole[col][row]-100 )*WK,0,WK,HK);

     // координаты картинки (клетки) на форме
     dst  =  Bounds(x,y,HK-2,WK-2);

     // вывести картинку в клетку
```

```
      Form1->Canvas->CopyRect(dst,Pictures->Canvas,src);

      // нарисовать контур клетки
      Canvas->Pen->Color =  clBlack;
      Canvas->Brush->Style  =  bsClear;
      Canvas->Rectangle(x,y,x+WK-2,y+HK-2);
      return;
  };

  if ( (Pole[col][row] >= 0) && (Pole[col][row] < 100) )
    // клетка закрыта, рисуем только контур
    {
      Canvas->Brush->Color = clBtnFace;
      Canvas->Pen->Color = clBlack;
      Canvas->Rectangle(x,y,x+WK-2,y+HK-2);
    }

#ifdef DEBUG
  // подсказка - номер картинки
  Canvas->Font->Color = clBlack;
  Canvas->TextOut(x+15,y+15, IntToStr(Pole[col][row]));
#endif
};

// нажатие кнопки мыши
void __fastcall TForm1::FormMouseDown(TObject *Sender,
      TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
  int col;   // номер клетки по горизонтали
  int row;   //номер клетки по вертикали

  col = X/WK;
  row = Y/HK;
```

```
if ( Pole[col][row] >= 200 )
   // щелчок на месте уже открытой картинки
  return;

if ( no == 0 ) // открытых клеток нет
{
  no  =  1;
  open1.x  =  col;
  open1.y  =  row;

  // клетка помечается как открытая
  Pole[open1.x][open1.y]  +=  100;
  Kletka(open1.x,open1.y);
  return;
};

if ( no == 1 )
{
  // открыта одна клетка, надо открыть вторую
  open2.x  =  col;
  open2.y  =  row;

  // если открыта одна клетка и щелчок сделан
  // в этой клетке, то ничего не происходит
  if ( (open1.x == open2.x) && (open1.y == open2.y) )
      return;

  else
  {
    no  =  2; // теперь открыты две клетки
    Pole[open2.x][open2.y]  += 100;
    Kletka(open2.x,open2.y); // отрисуем вторую клетку
```

```
    // проверим, открытые картинки одинаковые?
    if ( Pole[open1.x][open1.y] == Pole[open2.x][open2.y] )
      // открыты две одинаковые картинки
      {
        nf++;
        Form1->Timer1->Enabled  =  True; // запустить таймер
        // процедур обработки события OnTimer
        // "сотрет" две одинаковые картинки
      };
    return;
  }
};

if ( no == 2 )
{
  // открыты 2 клетки с разными картинками
  // закроем их и откроем новую, в которой
  // сделан щелчок

  // закрыть открытые клетки
  Pole[open1.x][open1.y]  -= 100;
  Pole[open2.x][open2.y]  -= 100;
  Kletka(open1.x,open1.y);
  Kletka(open2.x,open2.y);

  // запись в open1 номера текущей клетки
  open1.x  =  col;
  open1.y  =  row;
  no  =  1;   // счетчик открытых клеток

  // открыть текущую клетку
  Pole[open1.x][open1.y]  +=  100;
```

```
    Kletka(open1.x,open1.y);
  };
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
  /* Сейчас отображаются две одинаковые картинки.
     Пометим их как найденные и уберем с экрана. */
  Pole[open1.x][open1.y]  +=  100;
  Pole[open2.x][open2.y]  +=  100;
  no  =  0; // количество открытых клеток

  // Отрисовать клетки
  Kletka(open2.x,open2.y);
  Kletka(open1.x,open1.y);

  // остановить таймер
  Form1->Timer1->Enabled  =  false;

  if ( nf == W*H/2 )
  {
   // открыты все пары
    Canvas->Font->Name  =  "Tahoma";
    Canvas->Font->Size  =  16;
    Canvas->Font->Color = clBlue;
    Canvas->Font->Color =  clBlack;
    Canvas->TextOut(100,160,"Game Over!");
    Canvas->Font->Size  =  10;
    Canvas->TextOut(120,210,"(c) Культин Н.Б., 2005");
  };
}
```

```
// обработка события Paint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    ShowPole();
}

// команда Новая программа
void __fastcall TForm1::N1Click(TObject *Sender)
{
    NewGame();
    ShowPole();
}

// Команда Справка/Справка
void __fastcall TForm1::N3Click(TObject *Sender)
{
    WinExec("hh.exe dpic.chm", SW_RESTORE);
}

// Команда Справка/О программе
void __fastcall TForm1::N4Click(TObject *Sender)
{
    WinExec("hh.exe -mapid 3 dpic.chm", SW_RESTORE);
}
```

Экзаменатор

Программа **Экзаменатор** (рис. 1.68) позволяет автоматизировать процесс тестирования. В оке программы отображается тест — последовательность вопросов, на которые испытуемый должен ответить путем выбора правильного ответа. В рассматриваемой программе вопросы загружаются из файла (пример файла теста приведен на рис. 1.69). Имя файла теста передается программе при ее запуске — указывается в качестве параметра команды запуска.

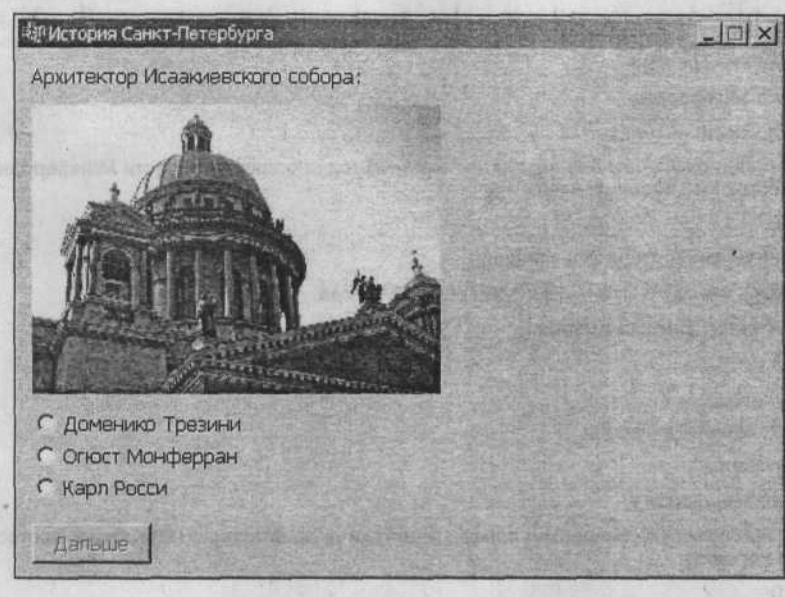


Рис. 1.68. Окно программы **Экзаменатор**. Пользователь должен выбрать правильный ответ

```
История Санкт-Петербурга

Сейчас Вам будут предложены вопросы о знаменитых памятниках и архитектурных сооружениях Санкт-Петербурга. Вы должны из предложенных нескольких вариантов ответа выбрать правильный.

7

Вы прекрасно знаете историю Санкт-Петербурга!

6

Вы много знаете о Санкт-Петербурге, но на некоторые вопросы ответили не верно.

5

Вы не достаточно хорошо знаете историю Санкт-Петербурга.

4

Вы, вероятно, только начали знакомиться с историей Санкт-Петербурга?

Архитектор Исаакиевского собора:

3 2 1
```

Рис. 1.69. Пример файла теста *(начало)*

isaak.jpg

Доменико Трезини

Огюст Монферран

Карл Росси

Александровская колона воздвигнута в 1836 году по проекту Огюста Монферрана как памятник, посвященный:

2 1 0

деяниям императора Александра I.

подвигу народа в Отечественной войне 1812 года.

Архитектор Зимнего дворца

3 2 1

herm.jpg

Бартоломео Растрелли

Карл Росси

Огюст Монферран

Михайловский (Инженерный) замок - жемчужина архитектуры Петербурга построен по проекту

3 1 0

Воронихина Андрея Никифоровича

Старова Ивана Егоровича

Баженова Василия Ивановича

Остров, на котором находится Ботанический сад, основанный императором Петром I, называется:

3 3 0

Заячий

Медицинский

Аптекарский

Невский проспект получил свое название

3 2 0

по имени реки, на которой стоит Санкт-Петербург.

по имени близко расположенного монастыря, Александро-Невской лавры.

в память о знаменитом полководце - Александре Невском.

Скульптура знаменитого памятника Петру I выполнена

2 1 0

Фальконе

Клодтом

Рис. 1.69. Пример файла теста *(окончание)*

Первые два абзаца файла — это название теста и общая информация. Далее следует раздел оценок, в котором указываются количество балов, необходимое для достижения уровня, и оценка (всего четыре уровня). За разделом оценок следуют вопросы. Каждый вопрос представляет собой последовательность абзацев. Первый абзац — вопрос, второй — последовательность цифр (количество альтернативных ответов, номер правильного ответа и признак наличия иллюстрации). Следующие несколько абзацев — это варианты ответа.

Имя файла теста передается программе тестирования при ее запуске — указывается в качестве параметра команды запуска.

Форма программы **Экзаменатор** приведена на рис. 1.70. Следует обратить внимание, что кнопки выбора ответа создаются динамически, во время работы программы, и поэтому на форме их нет.

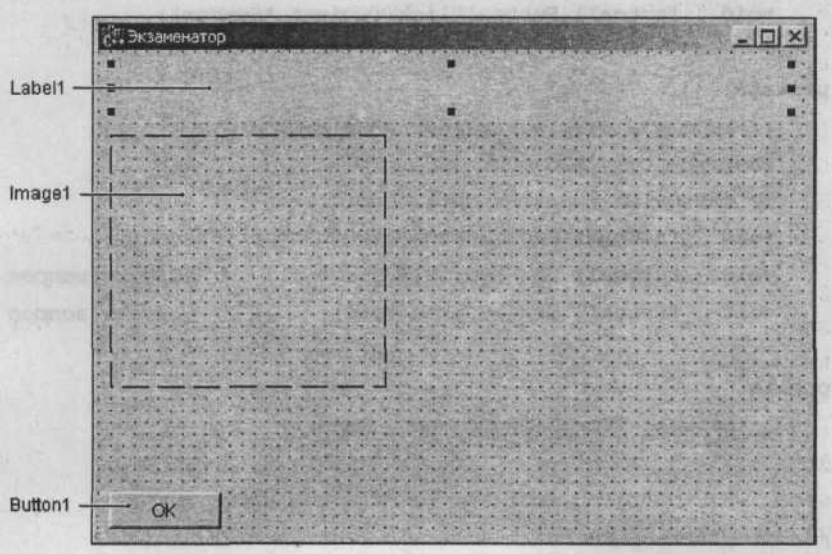


Рис. 1.70. Форма программы **Экзаменатор**

```
// *** заголовочный (h) файл формы ***
// вопрос
struct TVopros {
    AnsiString Vopr;    // вопрос
```

```
    AnsiString Img;     // иллюстрация (имя BMP-файла)
    AnsiString Otv[4]; // варианты ответа
    int        nOtv;    // кол-во вариантов ответа
    int        rOtv;    // номер правильного ответа
};

class TForm1 : public TForm
{
__published:
    TLabel *Label1;    // информационное сообщение, вопрос
    TImage *Image1;    // иллюстрация к вопросу
    TButton *Button1; // кнопка OK / Дальше
    void __fastcall FormActivate(TObject *Sender);
    void __fastcall Button1Click(TObject *Sender);

private:
    // варианты ответа - радиокнопки выбора
    TRadioButton *RadioButton[4];
    // щелчок на кнопке выбора ответа
    void __fastcall RadioButtonClick(TObject *Sender);
    void __fastcall ShowVopros(TVopros v);  // выводит вопрос
    void __fastcall EraseVopros(void);      // удаляет вопрос

public:
    __fastcall TForm1(TComponent* Owner);
};

// *** модуль формы ***
#include <stdio.h>      // для доступа к функции sscanf
#include <jpeg.hpp>     // обеспечивает работу с
                        // jpg-иллюстрациями

#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TForm1  *Form1;    // форма

int f;             // дескриптор файла теста
                   // имя файла теста берем из командной строки

int level[4];      // кол-во правильных ответов, необходимое
                   // для достижения уровня
AnsiString mes[4];// сообщение о достижении уровня

TVopros Vopros;    // вопрос
int otv;           // номер выбранного ответа

int right = 0;     // кол-во правильных ответов

// функции, обеспечивающие чтение вопроса из файла теста
int GetInt(int f);                          // читает целое
int GetString(int f, AnsiString *st);  // читает строку

// конструктор
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    int i;
    int left = 10;

    // создадим радиокнопки для выбора
    // правильного ответа, но сделаем их невидимыми
    for (i = 0; i < 4; i++ )
    {
        // создадим радиокнопку
        RadioButton[i] = new TRadioButton(Form1);
        // установим значения свойств
        RadioButton[i]->Parent = Form1;
        RadioButton[i]->Left = left;
        RadioButton[i]->Width = Form1->
```

```
                    ClientWidth - left - 20;
        RadioButton[i]->Visible = false;
        RadioButton[i]->Checked = false;

        // зададим функцию обработки события Click
        RadioButton[i]->OnClick = RadioButtonClick;
    }
}

void __fastcall TForm1::FormActivate(TObject *Sender)
{
    AnsiString st;

    // имя файла теста должно быть указано в командной строке
    int n = ParamCount();
    if ( n < 1 )
    {
        Label1->Font->Style = TFontStyles()<< fsBold;
        Label1->Caption = "В командной строке запуска"
                "программы надо задать имя файла теста";
        Button1->Tag = 2;
        return;
    }

    // открыть файл теста
    f = FileOpen(ParamStr(1), fmOpenRead);
    if ( f == -1 )
    {
        Label1->Font->Style = TFontStyles()<< fsBold;
        Label1->Caption = "Ошибка доступа к файлу теста" +
                          ParamStr(1);
        Button1->Tag = 2;
        return;
    }
```

```
    //  вывести информацию о тесте
    GetString(f, &st); // прочитать название теста
    Form1->Caption = st;

    GetString(f, &st); // прочитать вводную информацию
    Label1->Width = Form1->ClientWidth - Label1->Left -20;
    Label1->Caption = st;
    Label1->AutoSize = true;

    // прочитать информацию об уровнях оценки
    for (int i=0; i<4; i++)
    {
        level[i] = GetInt(f);
        GetString(f, &mes[i]);
    }
}

// читает из файла очередной вопрос
bool GetVopros(TVopros *v)
{
    AnsiString st;
    int p;   // если p=1, то к вопросу есть иллюстрация

    if ( GetString(f, &(v->Vopr)) != 0 )
    {
        /* прочитать кол-во вариантов ответа, номер
           правильного ответа и  признак наличия
           иллюстрации */
        v->nOtv = GetInt(f);
        v->rOtv = GetInt(f);
        p = GetInt(f);

        if (p) // к вопросу есть иллюстрация
            GetString(f,&(v->Img) );
```

```
        else v->Img = "";

        // читаем варианты ответа
        for (int i = 0; i < v->nOtv; i++)
        {
            GetString(f,&(v->Otv[i]));
        }
        return true;
    }
    else return false;
}

// выводит вопрос
void __fastcall TForm1::ShowVopros(TVopros v)
{
    int top;
    int i;

    // вопрос
    Label1->Width = ClientWidth - Label1->Left -20;
    Label1->Caption = v.Vopr;
    Label1->AutoSize = true;

    if (v.Img != "") // к вопросу есть иллюстрация
    {
       /* определим высоту области, которую можно
          использовать для вывода иллюстрации */
       int RegHeight =  Button1->Top
                - (Label1->Top + Label1->Height +10)
                - (RadioButton[1]->Height + 10) * v.nOtv;

        Image1->Top = Label1->Top + Label1->Height + 10;
        // загрузим картинку и определим ее размер
```

```
        Image1->Visible = false;
        Image1->AutoSize = true;
        Image1->Picture->LoadFromFile(v.Img);
        if (Image1->Height > RegHeight) // картинка не
                                        // помещается
        {
            Image1->AutoSize = false;
            Image1->Height = RegHeight;
            Image1->Proportional = true;
        }
        Image1->Visible = true;
        // положение полей отсчитываем от иллюстрации
         top = Image1->Top + Image1->Height + 10;
    }
    else // положение полей отсчитываем от вопроса
         top = Label1->Top + Label1->Height + 10;

    // варианты ответа
    for (i = 0; i < v.nOtv; i++)
    {
        RadioButton[i]->Top = top;
        RadioButton[i]->Caption = v.Otv[i];
        RadioButton[i]->Visible = true;
        RadioButton[i]->Checked = false;
        top += 20;
    }
}

// щелчок на радиокнопоке выбора ответа
void __fastcall TForm1::RadioButtonClick(TObject *Sender)
{
    int i =0;
    while (  ! RadioButton[i]->Checked )
        i++;
```

```
    otv = i+1;
    // ответ выбран, сделаем доступной кнопку Дальше
    Button1->Enabled = true;
}

// удаляет вопрос с экрана
void __fastcall TForm1::EraseVopros(void)
{
    Image1->Visible = false; // скрыть поле вывода иллюстрации
    // скрыть поля выбора ответа
    for (int i = 0; i <4; i++)
    {
        RadioButton[i]->Visible = false;
        RadioButton[i]->Checked = false;
    }
    // сделать недоступной кнопку Дальше
    Button1->Enabled = false;
}

// щелчок на кнопке ОК/Дальше/ОК
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    bool ok; // рез-т чтения из файла очередного вопроса

    switch (Button1->Tag) {
        case 0: // щелчок на кнопке ОК в начале работы
                // программы
                // прочитать и вывести первый вопрос
                GetVopros(&Vopros);
                ShowVopros(Vopros);

                Button1->Caption = "Дальше";
```

```
        Button1->Enabled = false;
        Button1->Tag = 1;
        break;

case 1: // щелчок на кнопке Дальше
        if (otv == Vopros.rOtv) // выбран правильный
                                // ответ
            right++;
        EraseVopros();
        ok = GetVopros(&Vopros);
        if ( ok )
            ShowVopros(Vopros);
        else
        // вопросов больше нет
        {
            FileClose(f);
            // вывести результат
            AnsiString st; // сообщение
            int i; // номер достигнутого уровня

            Form1->Caption = "Результат тестирования";
            st.printf(
                 "Правильных ответов: %i\n",right
                    );

            // определим оценку
            i = 0; // предположим, что испытуемый
                   // ответил на все
                   // вопросы
            while (( right < level[i]) && (i < 3))
                i++;

            st = st + mes[i];
```

```
                Label1->Caption = st;

                Button1->Caption = "OK";
                Button1->Enabled = true;
                Button1->Tag = 2;
            }
            break;

        case 2: // щелчок на OK в конце работы программы
            Form1->Close(); // завершить работу программы
    }
}

// Функция GetString читает строку из файла
// значение функции - кол-во прочитанных символов
int GetString(int f, AnsiString *st)
{
    unsigned char buf[300]; // строка (буфер)
    unsigned char *p = buf;  // указатель на строку

    int n;  // кол-во прочитанных байт (значение ф-и FileRead)
    int len = 0; // длина строки

    n = FileRead(f, p, 1);
    while ( n != 0 )
    {
        if ( *p == '\r')
        {
            n = FileRead(f, p, 1); // прочитать '\n'
            break;
        }
        len++;
```

```
        p++;
        n = FileRead(f, p, 1);
    }
    *p = '\0';
    if ( len !=0) st->printf("%s", buf);
    return len;
}

// читает из файла целое число
int GetInt(int f)
{
    char buf[20];   // строка (буфер)
    char *p = buf;  // указатель на строку

    int n;  // кол-во прочитанных байт (значение ф-и FileRead)
    int a;  // число, прочитанное из файла

    n = FileRead(f, p, 1) ;
    while ( (*p >= '0') && (*p <= '9') && (n > 0) )
    {
        p++;
        n = FileRead(f, p, 1) ;
    }

    if ( *p == '\r')
        n = FileRead(f, p, 1); // прочитать '\n'

    *p = '\0';

    // преобразуем строку из буфера в целое
    sscanf(buf,"%i", &a);
    return a;
}
```

Экзаменатор-2

Программа **Экзаменатор-2** (рис. 1.71) позволяет автоматизировать процесс тестирования. Имя файла теста передается программе при ее запуске — указывается в качестве параметра команды запуска (рис. 1.72).

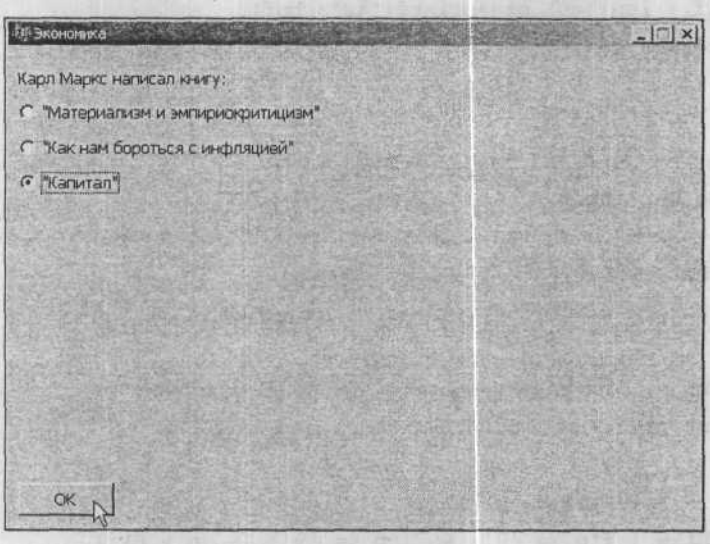


Рис. 1.71. Окно программы **Экзаменатор-2**

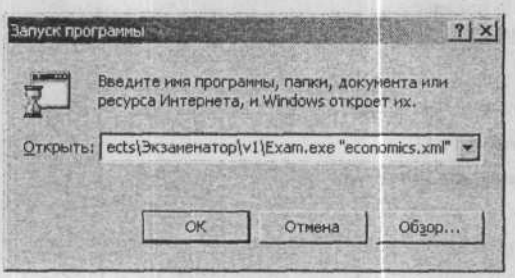


Рис. 1.72. Файл теста надо указать в команде запуска программы

Тест представляет собой XML-документ определенной структуры (рис. 1.73). Узлы `head` и `description` содержат название и общую

информацию о тесте. Узлы `q` — это вопросы. Значение атрибута `text` узла `q` представляет собой вопрос, атрибута `right` — номер правильного ответа. Каждый узел `a` — это вариант ответа. Узлы `level` содержат информацию об уровнях оценки результата тестирования: атрибут `score` определяет количество правильных ответов, необходимое для достижения уровня, атрибут `text` — оценку. Создать файл теста можно, например, при помощи Блокнота. В качестве примера на рис. 1.73 приведен тест "Экономика".

```
<?xml version="1.0" encoding="Windows-1251"?>
<test>
    <head>Экономика</head>
    <description>Сейчас Вам будут предложены вопросы из разных разделов экономики. Вы должны из предложенных нескольких вариантов ответов выбрать правильный.</description>
    <qw>
    <q text="Карл Маркс написал книгу:" src ="marks.jpg" right="3">
      <a>"Материализм и эмпириокритицизм"</a>
      <a>"Как нам бороться с инфляцией"</a>
      <a>"Капитал"</a>
    </q>

    <q text="Когда впервые появились бартерные сделки?" src ="" right="1">
      <a>при первобытнообщинном строе</a>
      <a>в период общественного разделения труда</a>
      <a>в наше время</a>
    </q>

    <q text="«ноу-хау» обозначает:" src ="" right="3">
      <a>секрет</a>
      <a>новое предприятие</a>
      <a>новая идея (знаю, как)</a>
    </q>
    <q text="Ставка дисконтирования позволяет:" src ="" right="1">
      <a>привести стоимость денег в будущем к текущему моменту</a>
      <a>расчитать скидку по кредиту</a>
      <a>учесть инфляцию</a>
```

Рис. 1.73. Пример файла теста *(начало)*

```
    </q>
                                        </qw>
  <levels>
    <level score="4" text = "Оценка - ОТЛИЧНО."/>
    <level score="3" text = " Оценка - ХОРОШО."/>
    <level score="2" text = "Оценка - УДОВЛЕТВОРИТЕЛЬНО."/>
    <level score="0" text = "Оценка - ПЛОХО!"/>
  </levels>
</test>
```

Рис. 1.73. Пример файла теста *(окончание)*

Форма программы **Экзаменатор-2** приведена на рис. 1.74. Компонент `XMLDocument` обеспечивает чтение из XML-файла: вопросов, альтернативных ответов и другой информации. Ниже приведен пример файла теста. Вопрос, а также информация о тесте и результат тестирования, отображаются в поле компонента `Label1`. Компоненты `RadioButton1`, `RadioButton2` и `RadioButton3` используются для отображения вариантов ответа. Невидимый во время работы программы компонент `RadioButton4` используется для сброса переключателей выбора ответа перед выводом очередного вопроса.

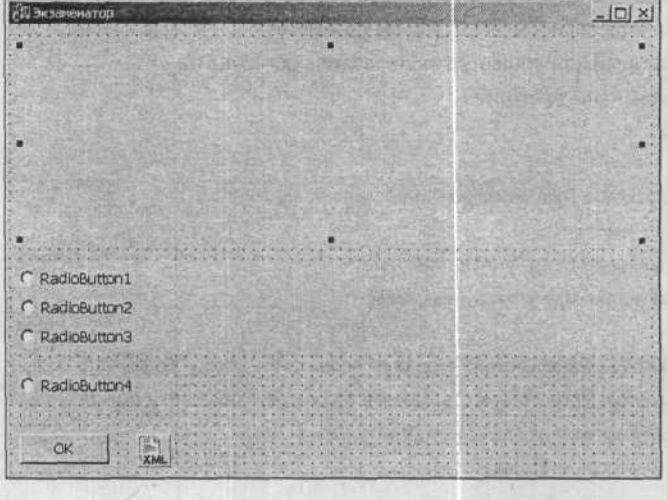


Рис. 1.74. Форма программы **Экзаменатор-2**

```
int nQuery = 0; // всего вопроов
int nRight;     // правильных ответов

int Right;      // правильный ответ
int Sel;        // ответ, выбранный испытуемым

static int mode = 0; // 0 - начало работы программы
                     // (вывести первый вопрос)
                     // 1 - процесс тестирования
                     // 2 - тестирование завершено

// начало работы
void __fastcall TForm1::FormActivate(TObject *Sender)
{
    if (ParamCount() == 0) {
        Label1->Caption = "В командной строке надо указать "
                          "имя файла теста";
        mode = 2;
        return;
    }

    XMLDocument1->FileName = ParamStr(1);
    try
    {
        // открыть XML-документ
        XMLDocument1->Active = True;
    }
    catch (EDOMParseError &e)
    {
        Label1->AutoSize = True;
        Label1->Caption = "Ошибка доступа к файлу теста " +
                          ParamStr(1) +
                          "\nMessage: " + e.Message;
        mode = 2;
```

```
        return;
    }
    Form1->Info(); // вывести информацию о тесте
}

// считывает и выводит вопрос с указанным номером
int __fastcall TForm1::Qery(int i)
{

    // привести форму в исходное состояние
    RadioButton1->Visible = False;
    RadioButton2->Visible = False;
    RadioButton3->Visible = False;
    RadioButton4->Checked = True;
    Button3->Enabled = False;

    // настроить интерфейс на работу с узлом qw
     _di_IXMLNode qw = XMLDocument1->
                         DocumentElement->ChildNodes->
                           Nodes[WideString("qw")];
     if ( i > qw->ChildNodes->Count -1 )
     {
        return -1;
     };

     nQuery++; // количество вопросов

     /* Узел q - это вопрос.
       Параметр text узла q - это текст вопроса
       дети узла q - это альтернативные ответы */
     _di_IXMLNode q = qw->ChildNodes->Nodes[i];

     /* Атрибут Text узла qw - это текст вопроса,
        атрибут right - номер правильного ответа */
```

```
    // вопрос
    Label1->AutoSize = false;
    Label1->Width = ClientWidth -20;
    Label1->Height = 150;
    Label1->Caption = q->GetAttribute(WideString("text"));
    Label1->AutoSize = true;

    Right = StrToInt( q->GetAttribute(WideString("right")));

    // узел "q" состоит из нескольких узлов "a"
    //(альтернативных ответов)
    _di_IXMLNode a;
    int j = 0; // номер узла "a"
    while ( j < q->ChildNodes->Count )
    {
       a = q->ChildNodes->Nodes[j];
       switch ( j ) {
           case 0 : RadioButton1->Caption = a->Text;
                    RadioButton1->Top = Label1->Top +
                                        Label1->Height + 10;
                    RadioButton1->Visible = True; break;
           case 1 : RadioButton2->Caption = a->Text;
                    RadioButton2->Top = RadioButton1->Top +
                                RadioButton1->Height + 10;
                    RadioButton2->Visible = True; break;
           case 2 : RadioButton3->Caption = a->Text;
                    RadioButton3->Top = RadioButton2->Top +
                                RadioButton2->Height + 10;
                    RadioButton3->Visible = True; break;
       }
       j++;
    }
    return 0;
}
```

```
// информация о тесте
void __fastcall TForm1::Info()
{
   Form1->Caption = XMLDocument1->DocumentElement->
                ChildNodes->Nodes[WideString("head")]->Text;
   Label1->Caption = XMLDocument1->DocumentElement->
          ChildNodes->Nodes[WideString("description")]->Text;
}

// щелчок на кнопке OK
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    static int i = 0;     // номер вопроса
    int r;                // результат вывода вопроса:
                          // -1 - вопросов больше нет
    switch ( mode) {
        case 0:
                r = Qery(i++);
                mode = 1;
                break;

        case 1:
                // проверим, правильный ли ответ выбрал
                // испытуемый
                if ( Sel == Right) nRight++;
                // вывести следующий вопрос
                r = Qery(i++);
                if ( r == -1 ) // больше вопросов нет
                {
                    // результат тестирования
                    Result();
                    Button3->Enabled = true;
                    mode = 2;
                }
```

```
                break;

        case 2: Form1->Close(); // завершить работу программы
      }
}

// вывести результат тестирования
void __fastcall TForm1::Result()
{
    int i = 0;
    int score;
    _di_IXMLNode ls; // интерфейс доступа к узлу levels
    _di_IXMLNode l;  // интерфейс доступа к узлу level

    // считываем последовательно узлы level и сравниваем
    // значение параметра score с количеством правильных
    // ответов
    ls = XMLDocument1->DocumentElement->
                    ChildNodes->Nodes[WideString("levels")];
    while ( i < ls->ChildNodes->Count)
    {
        l = ls->ChildNodes->Nodes[i];
        score = StrToInt( l->
                    GetAttribute(WideString("score")));
        if ( nRight >= score ) break;
        i++;
    }
    AnsiString mes;
    mes.printf("Экзамен закончен\n"
               "Всего вопросов: %i\n"
               "Правильных ответов: %i\n",
               nQuery,nRight);
```

```
    mes = mes + l->GetAttribute(WideString("text"));
    Label1->Width = Form1->ClientWidth - 20;
    Label1->Caption = mes;
}

// пользователь выбрал первый ответ
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
    Sel = 1;
    Button3->Enabled = True;
}

// пользователь выбрал второй ответ
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
    Sel = 2;
    Button3->Enabled = True;
}

// пользователь выбрал третий ответ
void __fastcall TForm1::RadioButton3Click(TObject *Sender)
{
    Sel = 3;
    Button3->Enabled = True;
}
```

Примечание

Следует обратить внимание, что операционную систему Windows можно настроить так, что программа тестирования будет запускаться автоматически в результате двойного щелчка на имени файла теста. Чтобы это сделать, измените расширение файла теста, например, на etr и сделайте двойной щелчок на имени файла теста. Затем, в окне **Выбор программы**, сделайте щелчок на кнопке **Другая**, откройте папку, в которой находится ваша программа тестирования, и выберите исполняемый файл.

Календарь

Программа **Календарь** выводит изображение календаря на текущий месяц. Имеется возможность задать праздничные дни. Демонстрирует вывод графики на поверхность формы, работу с функциями манипулирования датами. Форма и окно программы приведены на рис. 1.75. Как видно, заголовок окна во время работы программы не отображается (значение свойства `BorderStyle` равно `bsNone`), однако пользователь все-таки может переместить окно, "захватив" мышью сам календарь. Непосредственное перемещение календаря (окна программы) выполняет функция обработки события `MouseUp`, которое возникает в момент отпускания кнопки мыши.

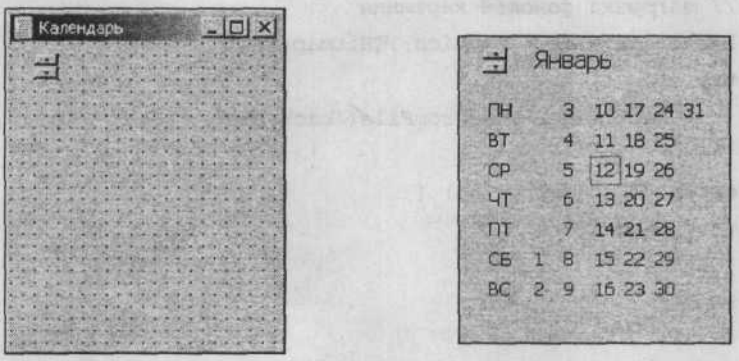


Рис. 1.75. Форма и окно программы **Календарь**

```
#define BACKGROUND
#undef BACKIMAGE

AnsiString stMonth[12] = {"Январь","Февраль","Март","Апрель",
                          "Май","Июнь","Июль","Август",
                          "Сентябрь","Октябрь","Ноябрь",
                          "Декабрь"};

AnsiString stDay[7] = {"ПН","ВТ","СР", "ЧТ","ПТ","СБ","ВС"};

// праздничные дни в формате dd.mm
```

```
AnsiString holiday =
     "01.01;02.01;07.01;23.02;08.03;01.05;09.05;07.11;12.12;";

Word aYear, aMonth, aDay; // год, месяц, день

__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
   DecodeDate(Now(),aYear,aMonth,aDay);
   UpDown1->Position = aMonth;
   #ifdef BACKIMAGE
   // загрузка фоновой картинки
   backimage = new Graphics::TBitmap();
   try {
        backimage->LoadFromFile("back.bmp");
   }
   catch (EFOpenError &e) {
        return;
   }
   #endif
}

// возвращает строковое представление двузначного числа
// с ведущем нулем: 01, 02 и т.д.
AnsiString __fastcall IntToStr_(int i)
{
  AnsiString s;
  if ( IntToStr(i).Length() == 1) s = "0"+IntToStr(i);
  else s = IntToStr(i);
  return s;
}

// обработка события Paint - вывести календарь
void __fastcall TForm1::FormPaint(TObject *Sender)
```

```
{
    Word aDayOfWeek; // день недели
    AnsiString st;   // дата в формате dd.mm

    int x, y,
        dx, dy, // шаг столбцов и строк цифр
        x0,y0;  // левый верхний угол области вывода календаря
    int i;

    x0 = 50; y0 = 40;
    dx = 20; dy = 20;

    Caption = "Календарь " + IntToStr(aYear);

    #ifdef BACKGROUND
      Canvas->Brush->Color = clBackground;
    #endif

    #ifdef BACKIMAGE
      Canvas->Draw(0,0,backimage);
    #else
      Canvas->Pen->Color = Canvas->Brush->Color;
      Canvas->Rectangle(0,0,ClientWidth,ClientHeight);
    #endif
    // вывести календарь на текущий месяц
    Canvas->Brush->Style = bsClear;
    Canvas->Font->Size = 12;
    Canvas->Font->Color = clBlack;
    Canvas->TextOutA(x0,y0-35, stMonth[aMonth-1]);

    Canvas->Font->Size = 10;

    // первая колонка - название дней недели
    x = x0 - 30;
```

```
y = y0;
for ( i = 0; i < 7; i++)
{
    if ( i < 5 )
        Canvas->Font->Color = clBlack;
    else
        Canvas->Font->Color = clRed;

    Canvas->TextOutA(x,y, stDay[i]);
    y += dy;
}

/* определим день недели, с которого
  начинается месяц */
aDayOfWeek = DayOfTheWeek( EncodeDate(aYear,aMonth,1));

x = x0;
y = y0 + dy * (aDayOfWeek-1);

for ( i = 1; i <= DaysInAMonth(aYear,aMonth); i++ )
{
    // проверим, не является ли день праздничным
    st = IntToStr_(i) + "." + IntToStr_(aMonth);
    if ( holiday.Pos(st) !=0 )
        Canvas->Font->Color = clRed; // праздничный
    else  // обычный
        if ( aDayOfWeek < 6 )
            Canvas->Font->Color = clBlack;
        else  // суббота или воскресенье
            Canvas->Font->Color = clRed;

    Canvas->TextOutA(x,y, IntToStr(i));

    if ( i == aDay) {
```

```
            // выделить сегодняшнее число
            // рамка
            Canvas->Brush->Style = bsClear;
            Canvas->Pen->Color = clGray;
            Canvas->Rectangle(x-3,y-1,x+dx-2,y+dy-1);
         }

        if ( aDayOfWeek != 7 ) {
            y += dy;
            aDayOfWeek ++;
        }
        else {
            aDayOfWeek =1;
            x += dx;
            y = y0;
        }
    }
}
// щелчок на кнопке компонента UpDown
void __fastcall TForm1::UpDown1Click(TObject *Sender,
                                     TUDBtnType Button)
{
     switch (UpDown1->Position) {
        case 0 : UpDown1->Position = 12; aYear--; break;
        case 13: UpDown1->Position = 1;  aYear++;; break;
     };
     aMonth = UpDown1->Position;
     Paint(); // обновить календарь
}

// отпущена кнопка мыши
void __fastcall TForm1::FormMouseUp(TObject *Sender,
TMouseButton Button,
      TShiftState Shift, int X, int Y)
```

```
{
    // переместить окно в ту точку экрана,
    // в которой находится указатель мыши
    Form1->Left = Form1->Left + X;
    Form1->Top = Form1->Top + Y;
}

// нажатие клавиши
void __fastcall TForm1::FormKeyDown(TObject *Sender,
                          WORD &Key, TShiftState Shift)
{
   if ( Key == 27) // клавиша <Esc>
        Form1->Close(); // завершить работу программы
}
```

Будильник

Программа **Будильник** (рис. 1.76) выводит сообщение в установленное пользователем время.

Рис. 1.76. Окно программы **Будильник**

После того как пользователь задаст сообщение, время и сделает щелчок на кнопке **OK**, окно программы исчезает с экрана (сворачивается). В установленное время окно **Будильник** появляется на экране. Отличительной особенностью программы

является то, что значок, обозначающий работающую программу (когда окно свернуто), отображается не в панели задач, а в системной области панели задач (System Tray) (рис. 1.77). При позиционировании указателя мыши на значок программы отображается время, на которое установлен будильник (рис. 1.78), а в результате нажатия правой кнопки появляется контекстное меню, команды которого позволяют завершить работу программы или развернуть ее окно.

Рис. 1.77. Значок работающей программы **Будильник** отображается в системной области панели задач

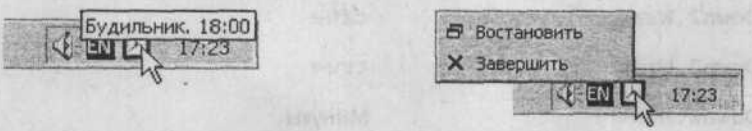


Рис. 1.78. Во всплывающей подсказке отображается время, на которое установлен будильник, а в контекстном меню — команды управления программой

Форма программы **Будильник** приведена на рис. 1.79, значения свойств компонентов — в табл. 1.25.

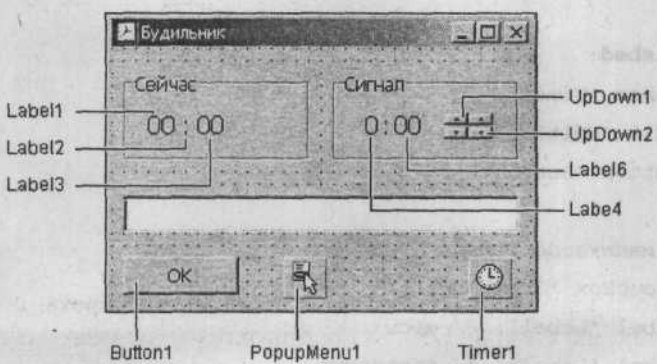


Рис. 1.79. Форма программы **Будильник**

Таблица 1.25. Значения свойств компонентов

Свойство	Значение
UpDown1.Min	0
UpDown1.Max	23
UpDown1.Wrap	true
UpDown1.Hint	Часы
UpDown1.ShowHint	true
UpDown2.Min	0
UpDown2.Max	59
UpDown2.Wrap	true
UpDown2.Wrap	true
UpDown2.Hint	Минуты

```
// *** заголовочный модуль (AlarmForm.h) ***
#define WM_MYTRAYNOTIFY (WM_USER + 123)

class TForm1 : public TForm
{
__published:
    TTimer *Timer1;
    TEdit *Edit1;
    TButton *Button1;

    // индикатор текущего времени
    TGroupBox *GroupBox2;
    TLabel *Label1; // часы
    TLabel *Label2; // двоеточие
    TLabel *Label3; // минуты
```

```
    // индикатор времени сигнала
    TGroupBox *GroupBox1;
    TLabel *Label4; // часы
    TLabel *Label5; // двоеточие
    TLabel *Label6; // минуты

    // кнопки установки времени будильника
    TUpDown *UpDown1; // часы
    TUpDown *UpDown2; // минуты

    TPopupMenu *PopupMenu1; // контекстное меню
    TMenuItem *N1; // команда Восстановить
    TMenuItem *N2; // команда Закрыть

    void __fastcall Button1Click(TObject *Sender);
    void __fastcall Timer1Timer(TObject *Sender);
    void __fastcall UpDown1Click(TObject *Sender,
                                 TUDBtnType Button);
    void __fastcall UpDown2Click(TObject *Sender,
                                 TUDBtnType Button);
    void __fastcall N1Click(TObject *Sender);
    void __fastcall N2Click(TObject *Sender);

    // *** определение этих функций вставлено сюда вручную ***

    // создать и поместить значок на System Tray
    void __fastcall CreateTrayIcon(int n, AnsiString Tip);
    // удалить значок из System Tray
    void __fastcall DeleteTrayIcon(int n);

protected:
    // процедура обработки сообщения WM_MYTRAYNOTIFY,
    // которое генерирует значок, находящийся на System Tray
    void __fastcall MYTRAYNOTIFY(TMessage &Message);
```

```
    BEGIN_MESSAGE_MAP
        MESSAGE_HANDLER(WM_MYTRAYNOTIFY, TMessage,
                        MYTRAYNOTIFY)
    END_MESSAGE_MAP(TControl)

private:

public:

    __fastcall TForm1(TComponent* Owner);
};

// *** модуль формы AlarmForm.cpp ***
#include "DateUtils.hpp"
#include "ShellAPI.hpp"  // для доступа к Shell_NottifyIcon
#include "mmsystem.hpp"  // для доступа к PlaySound

int cHour, cMinute;      // время на индикаторе
int alrHour, alrMinute; // время сигнала

// преобразует целое в строку с ведущим нулем
AnsiString __fastcall mm(int m)
{
    if (m <= 9)
        return "0" + IntToStr(m);
    else
        return IntToStr(m);
}

// констуктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    // отобразить текущее время
```

```
    cHour = HourOf( Now() );
    Label1->Caption = IntToStr(cHour);
    cMinute = MinuteOf( Now() );
    Label3->Caption = mm(cMinute);
}

// добавить значок на System Tray
void __fastcall TForm1::CreateTrayIcon(int n, AnsiString Tip)
{
    TNotifyIconData nidata;

    /* заполнить структуру nidat, поля которой
       определяют значок на System Tray */
   nidata.cbSize = sizeof(TNotifyIconData);
   nidata.hWnd = Form1->Handle; // окно приложения, которое
                                // представляет значок
   nidata.uID = n; // номер значка (одно приложение может
                   // разместить на панели
                   // несколько значков
   nidata.uFlags =  NIF_ICON + NIF_MESSAGE + NIF_TIP;

   /* при позиционировании указателя мыши на
      на значке, генерируется определенное
      программистом событие WM_MYTRAYNOTIFY
     ( см. AlarmMainForm.h ) */
   nidata.uCallbackMessage = WM_MYTRAYNOTIFY;

   // значок
   nidata.hIcon = Application->Icon->Handle;

   // посказка (всплывающий текст)
   StrPCopy(nidata.szTip,Tip);

   Shell_NotifyIcon(NIM_ADD, &nidata);  // добавить значок
}
```

```
// удалить картинку с System Tray
void __fastcall TForm1::DeleteTrayIcon(int n)
{
    TNotifyIconData nidata;

    nidata.cbSize = sizeof(TNotifyIconData);
    nidata.hWnd = Form1->Handle;
    nidata.uID = n; // номер значка, который надо убрать
                    // (одно приложение может разместить на
                    // панели несколько значков)

    Shell_NotifyIcon(NIM_DELETE, &nidata);
}

// щелчок на кнопке OK
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    AnsiString st;
    alrHour = UpDown1->Position;
    alrMinute = UpDown2->Position;

    if ( ( alrHour == cHour ) && (alrMinute <= cMinute) ||
         ( alrHour < cHour )  )
    {
        AnsiString st;
        int r;
        st.printf(
            "Сейчас %i:%i\nБудильник установлен на %i:%i",
            cHour,cMinute,alrHour,alrMinute
                );
        r = MessageDlg(st, mtWarning, TMsgDlgButtons()
                    << mbOK <<mbCancel, 0);
        if (r == mrCancel) return;
    }
```

```
    st = "Будильник. " + IntToStr(alrHour) + ":" +
         mm(alrMinute);
    CreateTrayIcon(1,st);
    Form1->Hide();
}

// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    if ( Form1->Visible )
    {
        // окно программы на экране
        if ( HourOf( Now() ) != cHour) {
            cHour = HourOf( Now() );
            Label1->Caption = IntToStr(cHour);
        }
        if ( MinuteOf( Now() ) != cMinute) {
            cMinute = MinuteOf( Now() );
            Label3->Caption = mm(cMinute);
        }
        // показать/скрыть двоеточие
        Label2->Visible = ! Label2->Visible;
    }
    else
    {
        TDateTime t = Now();
        if ( (alrHour == HourOf(t) ) &&
             (alrMinute == MinuteOf(t)) )
        {
            PlaySound("notify.wav", 0, SND_ASYNC);
            DeleteTrayIcon(1); // убрать значок с System Tray
            Form1->Show();
        }
    }
}
```

```
// щелчок на кнопке компонента UpDown1 (часы)
void __fastcall TForm1::UpDown1Click(TObject *Sender,
                                     TUDBtnType Button)
{
    Label4->Caption = IntToStr(UpDown1->Position);
}

// щелчок на кнопке компонента UpDown2 (минуты)
void __fastcall TForm1::UpDown2Click(TObject *Sender,
                                     TUDBtnType Button)
{
    Label6->Caption = mm(UpDown2->Position);
}

// обработка определенного пользователем сообщения
// WM_MYTRAYNOTIFY
void __fastcall TForm1::MYTRAYNOTIFY(TMessage &Message)
{
    TPoint p;
    if ( Message.LParam == WM_RBUTTONDOWN )
    {
        GetCursorPos(&p);
        SetForegroundWindow(Application->MainForm->Handle);
        Form1->PopupMenu1->Popup(p.x,p.y);
    }
}

// команда Восстановить
void __fastcall TForm1::N1Click(TObject *Sender)
{
    Timer1->Enabled = false;
    Form1->Show();
    DeleteTrayIcon(1); // убрать значок с System Tray
}
```

```
// команда контекстного меню Закрыть
// (завершение работы программы)
void __fastcall TForm1::N2Click(TObject *Sender)
{
    Form1->DeleteTrayIcon(1); // убрать значок с System Tray
    Form1->Close();
}
```

Очистка диска

Программа **Очистка диска** удаляет ненужные, созданные в процессе компиляции проектов C++Builder, файлы obj, tds, и резервные копии (~bpr, ~dfm, ~h, ~cpp) из указанного пользователем каталога и всех его подкаталогов. Для выбора каталога (папки) используется стандартное окно **Обзор папок** (рис. 1.80). Форма программы приведена на рис. 1.81.

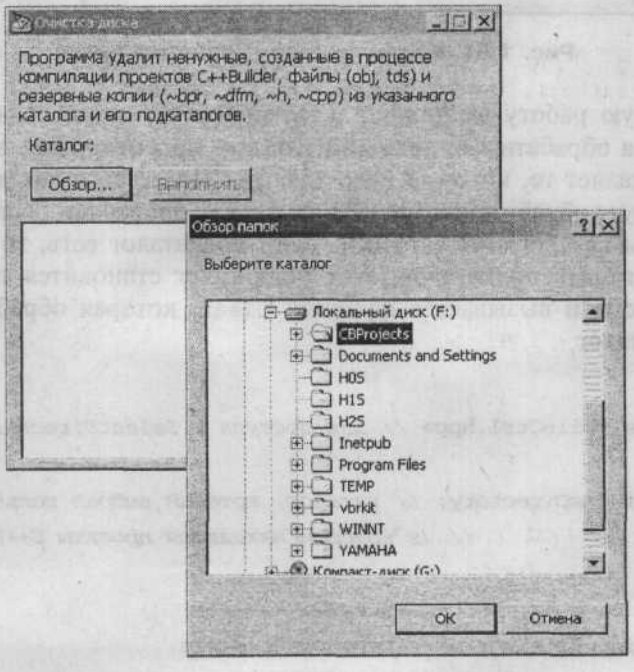


Рис. 1.80. Окно программы **Очистка диска**. Для выбора каталога используется окно **Обзор папок**

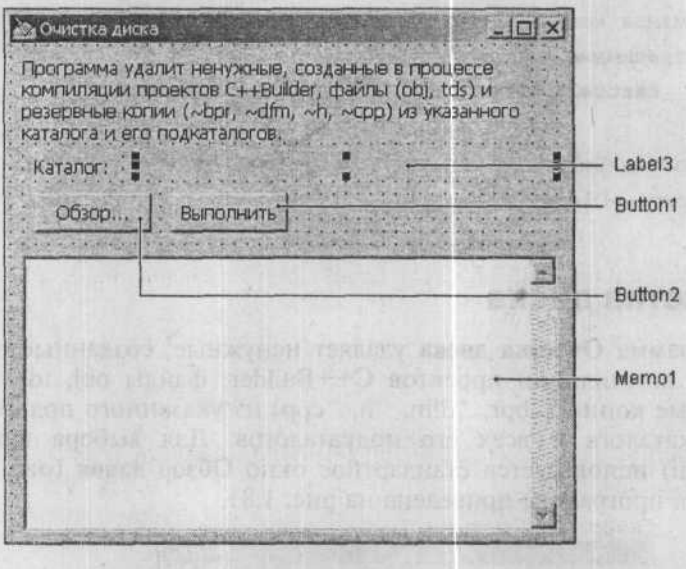


Рис. 1.81. Форма программы **Очистка диска**

Основную работу выполняет рекурсивная функция `Clear`. Сначала она обрабатывает текущий каталог: просматривает все файлы и удаляет те, которые надо удалить. После того как все файлы будут обработаны, функция `Clear` проверяет, есть ли в текущем каталоге подкаталоги. Если подкаталог есть, то выполняется вход в подкаталог (этот подкаталог становится текущим каталогом) и вызывается функция `Clear`, которая обрабатывает этот каталог.

```
#include <FileCtrl.hpp> // для доступа к SelectDirectory

AnsiString aDirectory; // каталог, который выбрал пользователь
                       // (в котором находятся проекты C++Builder)

AnsiString cDir;       // текущий каталог
AnsiString FileExt;    // расширение файла

int n = 0;             // количество удаленных файлов
```

```
// Щелчок на кнопке Обзор (выбор каталога)
void __fastcall TMainForm::Button1Click(TObject *Sender)
{
    if ( SelectDirectory("Выберите каталог","", aDirectory))
    {
        // диалог Выбор файла завершен щелчком на OK
        Label3->Caption = aDirectory;
        Button2->Enabled = true; // теперь кнопка Выполнить
                                 // доступна
    };
}

// удаляет ненужные файлы из текущего каталога и его
// подкаталогов
void __fastcall Clear(void)
{
   TSearchRec SearchRec; // информация о файле или каталоге

   cDir = GetCurrentDir()+"\\";

   if ( FindFirst("*.*", faArchive,SearchRec) == 0)
      do {
          // проверим расширение файла
          int p = SearchRec.Name.Pos(".");
          FileExt = SearchRec.Name.SubString(p+1,MAX_PATH);
          if ( ( FileExt[1] == '~') || ( FileExt == "obj" )
                                    || ( FileExt == "tds" )
)
          {
              MainForm->Memo1->
                        Lines->Add(cDir+SearchRec.Name);
              DeleteFile(SearchRec.Name);
              n++;
          }
      }
```

```
        while ( FindNext(SearchRec) == 0);

        // обработка подкаталогов текущего каталога
        if ( FindFirst("*", faDirectory, SearchRec) == 0)
           do
              if ((SearchRec.Attr & faDirectory) ==
                                          SearchRec.Attr )
              {
                    // каталоги ".." и "." тоже каталоги,
                    // но в них входить не надо!!!
                    if (( SearchRec.Name != "."  ) &&
                        ( SearchRec.Name != ".." ))
                    {
                        // войти в подкаталог
                        ChDir(SearchRec.Name);
                        // очистить каталог
                        Clear();
                        // выйти из каталога
                        ChDir("..");
                    };
               }
           while ( FindNext(SearchRec) == 0 );
}

// щелчок на кнопке Выполнить
void __fastcall TMainForm::Button2Click(TObject *Sender)
{
    Memo1->Clear();          // очистить поле Memo1
    ChDir(aDirectory);       // войти в каталог, который выбрал
                             // пользователь

    Clear();                 // очистить текущий каталог и его
                             // подкаталоги
```

```
    Memo1->Lines->Add("");
    if (n)
        Memo1->Lines->Add("Удалено файлов: " + IntToStr(n));
     else
        Memo1->Lines->Add("В указанном каталоге нет файлов, "
                          "которые надо удалить.");
}
```

Печать

Программа **Счет**, ее окно приведено на рис. 1.82, позволяет распечатать (вывести на принтер) счет.

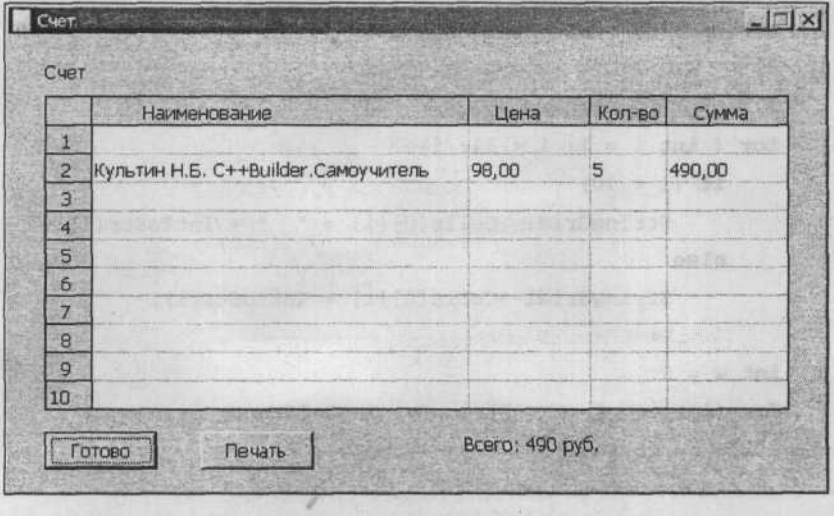


Рис. 1.82. В результате щелчка на кнопке **Печать** счет будет распечатан

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    // *** настроить таблицу ***
    StringGrid1->Options
            << goEditing // разрешить редактировать
```

```
            << goTabs;    // <Tab> - переход к следующей ячейке
    // заголовки столбцов
    StringGrid1->Cells[0][0] = "";
    StringGrid1->Cells[1][0] = " Наименование";
    StringGrid1->Cells[2][0] = " Цена";
    StringGrid1->Cells[3][0] = " Кол-во";
    StringGrid1->Cells[4][0] = " Сумма";

    // ширина столбцов
    StringGrid1->ColWidths[0] = 30;
    StringGrid1->ColWidths[1] = 250;
    StringGrid1->ColWidths[2] = 80;
    StringGrid1->ColWidths[3] = 50;
    StringGrid1->ColWidths[4] = 80;

    // заполнить первый столбец
    for ( int i = 1; i < 11; i++)
        if (i < 10)
            StringGrid1->Cells[0][i] = "  " + IntToStr(i);
        else
            StringGrid1->Cells[0][i] = IntToStr(i);

    int w = 0;
    for (int i = 0; i < StringGrid1->ColCount; i++)
        w += StringGrid1->ColWidths[i];

    // установить размер StringGrid в соответствии с размером
    // столбцов и количеством строк
    StringGrid1->Width = w + StringGrid1->ColCount + 1 ;
    StringGrid1->Height =
        StringGrid1->DefaultRowHeight * StringGrid1->
                        RowCount + StringGrid1->RowCount + 1;
}
```

```
#include "Printers.hpp"
// щелчок на кнопке Печать
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    TPrinter *Prn; // принтер

    #define LEFT_MARGIN 2 // отступ слева 2 см
    #define TOP_MARGIN  2 // отступ сверху 2 см

    float dpiX, dpiY; // разрешение принтера по X и Y
    float kx, ky;     // коэф. пересчета координат экрана
                      // в координаты принтера по X и Y
    // таблица
    int p[5];  // позиции колонок
    int x1,y1,x2,y2; // границы таблицы

    int px, py; // указатель точки вывода
    int i, j;

    /* Разрешение экрана и принтера разное,
       поэтому чтобы добиться соответствия
       размеров изображения на экране и принтере,
       координаты точек экрана надо преобразовать
       в координаты принтера, умножить на коэф.,
       значение которого зависит от разрешения принтера.
       Например, если разрешение принтера 300 dpi,
       то значение коэффициента равно 3.125, т. к.
       разрешение экрана - 96 dpi */

    Prn = Printer();

     /* ф-я GetDeviceCaps позволяет получить характеристики
       устройства. LOGPIXELSX - кол-во пикселов на дюйм по X
*/
```

```
dpiX  =  GetDeviceCaps(Prn->Handle,LOGPIXELSX);
dpiY  =  GetDeviceCaps(Prn->Handle,LOGPIXELSY);
kx  =  dpiX / Screen->PixelsPerInch;
ky  =  dpiY / Screen->PixelsPerInch;

px  =  LEFT_MARGIN / 2.54 * dpiX;
py  =  TOP_MARGIN  / 2.54 * dpiY;

// вычислим "принтерные" координаты колонок таблицы
p[0]  =  px;
for (i = 1; i < 5; i++ )
    p[i]  =  p[i-1] + StringGrid1->ColWidths[i-1]* kx + i;

Prn->BeginDoc(); // открыть печать

// заголовок таблицы
Prn->Canvas->Font->Name  =  Label1->Font->Name;
Prn->Canvas->Font->Size  =  Label1->Font->Size;
Prn->Canvas->TextOut(px,py,Label1->Caption);

// таблица - содержимое StringGrid1
py  =  py + Label1->Font->Size * 2 * ky;

x1  =  px; y1  =  py; // левый верхний угол таблицы

Prn->Canvas->Font->Name  =  StringGrid1->Font->Name;
Prn->Canvas->Font->Size  =  StringGrid1->Font->Size;

x2  =  p[4] + StringGrid1->ColWidths[4]* kx;
y2  =  py + StringGrid1->RowCount *
                    StringGrid1->RowHeights[1] * ky;

for ( j = 0; j < StringGrid1->RowCount; j++)
```

```
    {
      // строки таблицы
      for (i = 0; i < StringGrid1->ColCount; i++)
      {
        Prn->Canvas->TextOut(p[i],py,
                             StringGrid1->Cells[i][j]);
        // гор->линия
        Prn->Canvas->MoveTo(p[0],py);
        Prn->Canvas->LineTo(x2,py);
      }
      py = py+ StringGrid1->RowHeights[j]* ky;
    }

    // вертикальные линии
    for ( i = 0; i < StringGrid1->ColCount; i++ )
    {
      Prn->Canvas->MoveTo(p[i],y1);
      Prn->Canvas->LineTo(p[i],y2);
    }
    // правая граница
    Prn->Canvas->MoveTo(x2,y1);
    Prn->Canvas->LineTo(x2,y2);

    // нижняя граница
    Prn->Canvas->MoveTo(x1,y2);
    Prn->Canvas->LineTo(x2,y2);

    py = y2 + 0.5 / 2.54 * dpiY; // здесь 1 - это 1 см.
    Prn->Canvas->TextOut(p[3],py,Label2->Caption);

    Prn->EndDoc();  // закрыть печать
}

// щелчок на кнопке Готово
```

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float summ;

    summ = 0;
    for ( int i = 1; i < 11; i++)
    {
        // если ячейка Сумма пустая, то при выполнении
        // функции StrToFloat возникает ошибка (исключение)
        try
        {
            summ += StrToFloat(StringGrid1->Cells[4][i]);
        }
        catch (Exception &e)
        {   }
    }
    Label2->Caption = "Всего: " + FloatToStr(summ) + " руб.";
}
```

Задачи для самостоятельного решения

В этом разделе приведены задачи, решить которые предлагается читателю самостоятельно.

Скидка

Напишите программу вычисления стоимости покупки с учетом скидки. Скидка предоставляется, если сумма превышает 1000 руб., а также в выходные дни. Рекомендуемый вид формы приведен на рис. 1.83. В результате щелчка на кнопке **Скидка** в поле компонента `Label` должно появляться сообщение, информирующее о предоставлении скидки, и итоговая сумма с учетом скидки. Информацию о том, является ли день выходным, программа должна получать на основе анализа текущей даты.

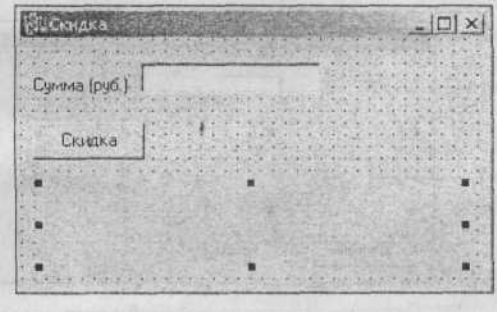


Рис. 1.83. Форма программы **Скидка**

Доход по вкладу

Напишите программу вычисления дохода по вкладу в банке. Доход вычисляется по формуле: Д = С * (СР / 360) * (СТ / 100), где: С — сумма вклада; СР — срок вклада (количество дней); СТ — процентная ставка (годовых). Рекомендуемый вид формы приведен на рис. 1.84.

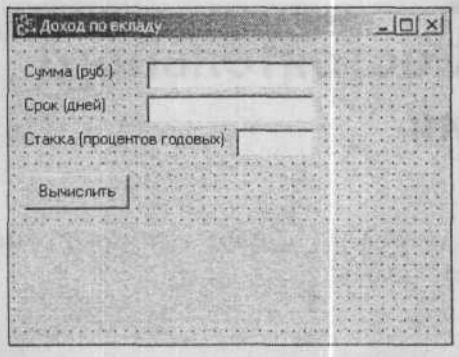


Рис. 1.84. Форма программы **Доход по вкладу**

Таблица умножения

Напишите программу, при помощи которой можно проконтролировать знание таблицы умножения. Программа должна предложить испытуемому 10 примеров и по окончании процесса тестирования выставить оценку. Рекомендуемый вид формы приведен на рис. 1.85. Компонент `Label1` используется для вывода примера, `Label2` — для вывода сообщения об ошибке и результатов тестирования.

Рис. 1.85. Форма программы проверки знания таблицы умножения

Поездка на автомобиле

Напишите программу, при помощи которой можно вычислить стоимость поездки на автомобиле. Рекомендуемый вид формы приведен на рис. 1.86.

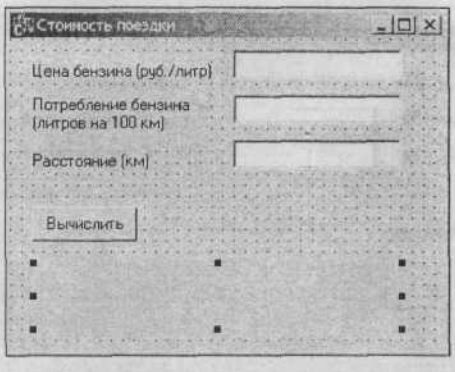


Рис. 1.86. Форма программы **Стоимость поездки**

Стоимость разговора

Напишите программу вычисления стоимости исходящего звонка с сотового телефона. Рекомендуемый вид формы приведен на рис. 1.87.

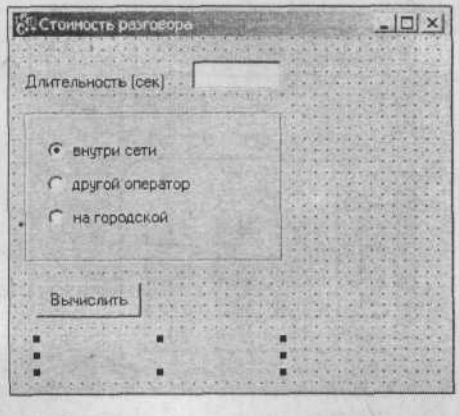


Рис. 1.87. Форма программы **Стоимость разговора**

Стеклопакет

Напишите программу, при помощи которой можно вычислить стоимость окна (стеклопакета). Рекомендуемый вид формы приведен на рис. 1.88.

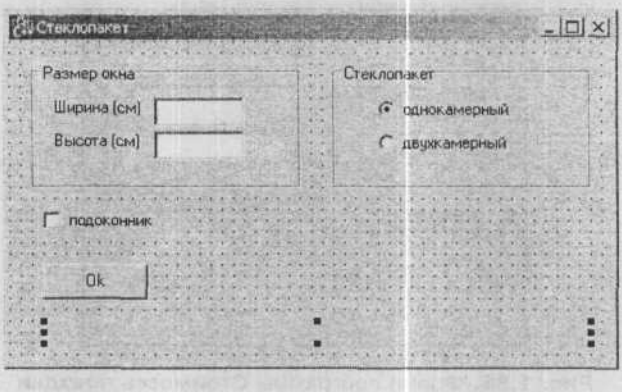


Рис. 1.88. Форма программы **Стеклопакет**

Калькулятор

Усовершенствуйте программу **Калькулятор** так, чтобы можно было выполнять операции умножения и деления. Рекомендуемая форма приведена на рис. 1.89.

Рис. 1.89. Форма программы **Калькулятор**

Электроэнергия

Напишите программу, которая сохраняет в файле electr.txt показания счетчика расхода электроэнергии (один раз в месяц). Рекомендуемый вид формы приведен на рис. 1.90.

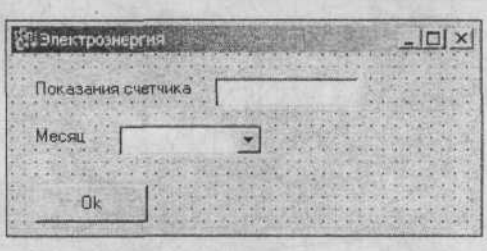


Рис. 1.90. Форма программы **Электроэнергия**

Напишите программу, в окне которой (в поле компонента Memo) отображается содержимое файла electr.txt.

Добрый день

Измените программу **Добрый день** таким образом, чтобы в зависимости от времени суток менялся не только текст приветствия, но и фоновый рисунок.

Часы

Напишите программу **Часы**, в окне которой отображается текущее время. Рекомендуемый вид формы приведен на рис. 1.91. Двоеточие на индикаторе должно мигать.

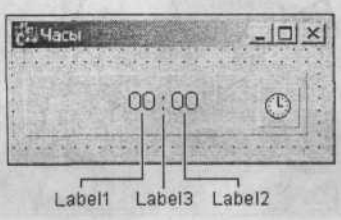


Рис. 1.91. Форма программы **Часы**

Узоры

Напишите программу, в окне которой формируется узор из окружностей произвольного диаметра и цвета (рис. 1.92).

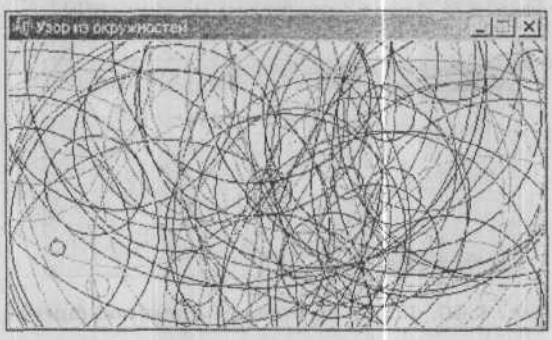


Рис. 1.92. Узор из окружностей

Напишите программу, в окне которой формируется узор из прямоугольников произвольного размера и цвета.

Курс доллара

Напишите программу, в окне которой на фоне картинки отображается график изменения курса доллара (рис. 1.93). Данные должны загружаться из файла.

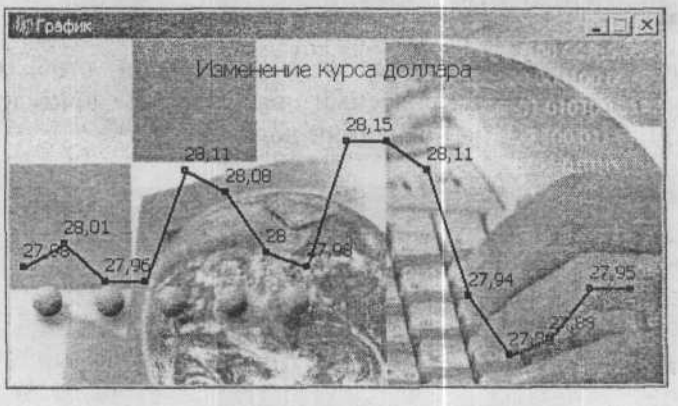


Рис. 1.93. График на фоне картинки

Диаграмма

Напишите программу, в окне которой, в виде диаграммы, отображается динамика изменения, например, цены мониторов (рис. 1.94).

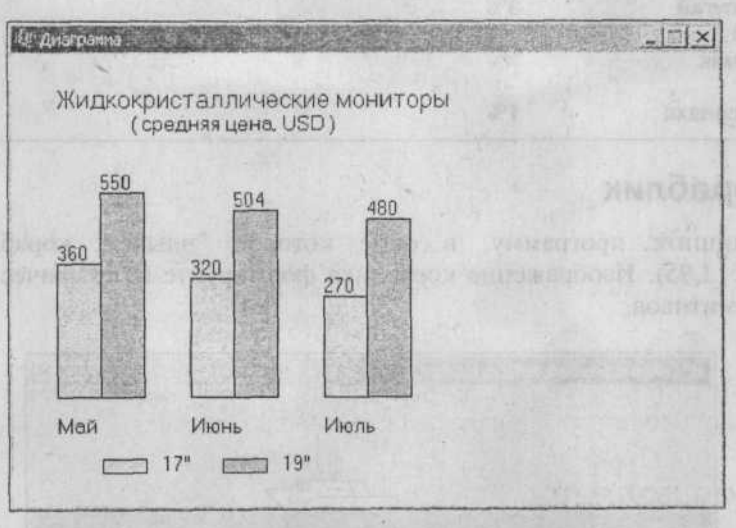


Рис. 1.94. Диаграмма

Домашние животные

Напишите программу, в окне которой формируется круговая диаграмма, иллюстрирующая результат опроса — ответа на вопрос "Есть ли у вас домашние животные? Какие?" Данные для построения диаграммы приведены в табл. 1.26.

Таблица 1.26. *Результат опроса*

Ответ	Доля (процент от общего количества ответов)
Нет	40%
Кошка	30%
Собака	22%

Таблица1.26 (окончание)

Ответ	Доля (процент от общего количества ответов)
Рыбки	3%
Попугай	3%
Хомяк	1%
Черепаха	1%

Кораблик

Напишите программу, в окне которой "плывет" кораблик (рис. 1.95). Изображение кораблика формируйте из графических примитивов.

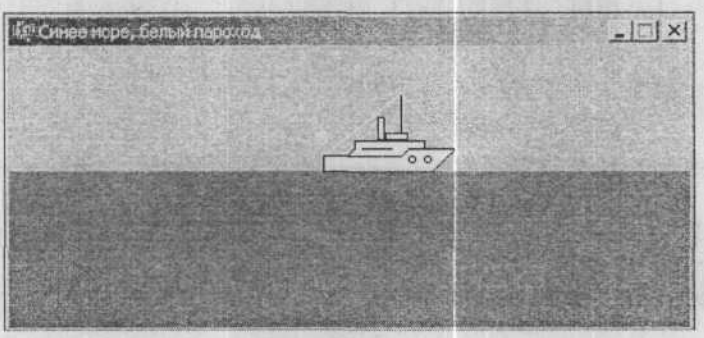


Рис. 1.95. Кораблик

Сапер

Внесите такие изменения в программу **Сапер**, чтобы изображения клеток (пустая клетка; флажок; мина; мина, помеченная флажком) загружались из файла.

Тест памяти (на внимательность)

Напишите программу, используя которую можно оценить способность игрока (испытуемого) запоминать числа. Программа

должна выводить числа, а испытуемый — вводить эти числа с клавиатуры. Время, в течение которого игрок будет видеть число, ограничьте, например, одной секундой. По окончании теста программа должна вывести результат: количество показанных чисел и количество чисел, которые испытуемый запомнил правильно. Форма программы приведена на рис. 1.96.

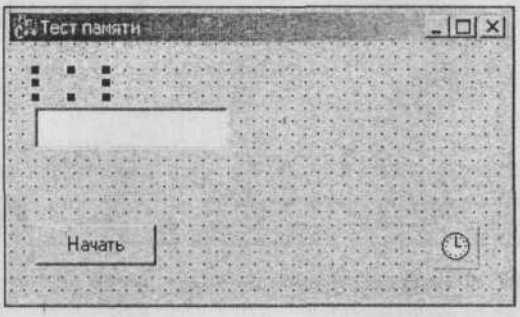


Рис. 1.96. Форма программы **Тест памяти**

Экзаменатор

Усовершенствуйте программу **Экзаменатор** так, чтобы она запрашивала имя тестируемого и сохраняла результат тестирования в файле. Для ввода имени тестируемого используйте стандартное окно ввода, которое выводит функция `InputBox`.

База данных "Расходы"

Напишите программу работы с базой данных **Расходы**. В базе данных должна фиксироваться сумма, дата и то, на что потрачены деньги (по категориям, например: еда, транспорт, образование, развлечения, прочее). Программа должна обеспечивать статистическую обработку — выводить сумму затрат за период. Базу данных в формате Paradox (таблицу rash.db) можно создать при помощи утилиты Database Desktop. Рекомендуемый вид формы программы работы с базой данных приведен на рис. 1.97.

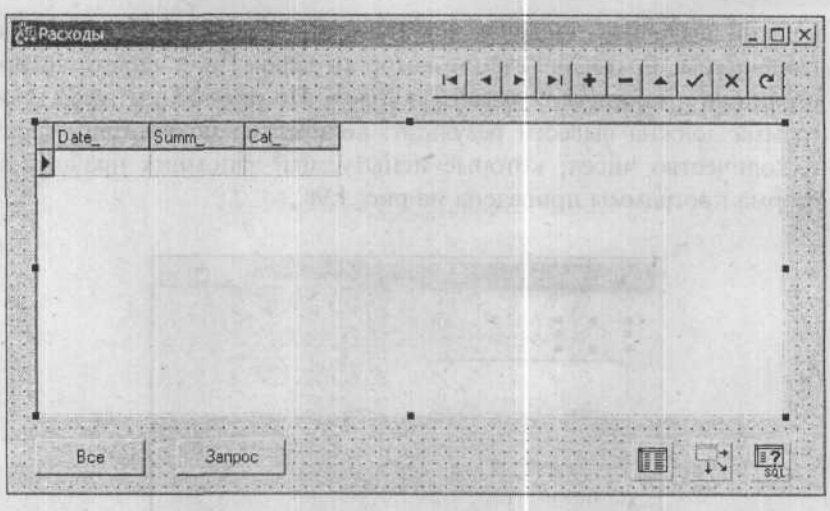
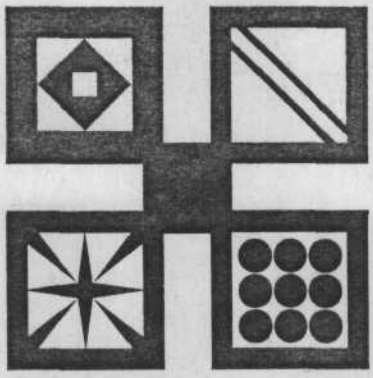
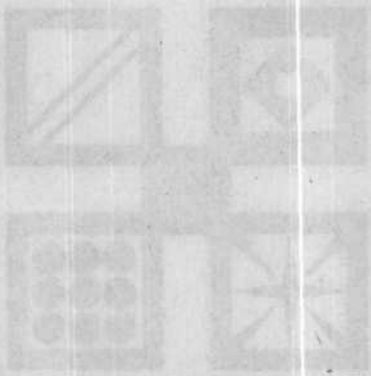


Рис. 1.97. Форма программы работы с базой данных **Расходы**

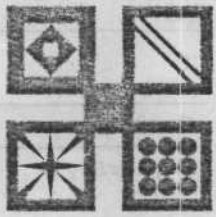


ЧАСТЬ 2

Borland C++ Builder — краткий справочник



Вторая часть книги представляет собой краткий справочник по компонентам и функциям Borland C++ Builder.



Форма

Форма (объект типа `TForm`) является основой программы. Свойства формы (табл. 2.1) определяют вид окна программы.

***Таблица 2.1.** Свойства формы (объекта `TForm`)*

Свойство	Описание
`Name`	Имя формы. В программе имя формы используется для управления формой и доступа к компонентам формы
`Caption`	Текст заголовка
`Top`	Расстояние от верхней границы формы до верхней границы экрана
`Left`	Расстояние от левой границы формы до левой границы экрана
`Width`	Ширина формы
`Height`	Высота формы
`ClientWidth`	Ширина рабочей (клиентской) области формы, т. е. без учета ширины левой и правой границ
`ClientHeight`	Высота рабочей (клиентской) области формы, т. е. без учета высоты заголовка и ширины нижней границы формы

Таблица 2.1 (окончание)

Свойство	Описание
BorderStyle	Вид границы. Граница может быть обычной (bsSizeable), тонкой (bsSingle) или отсутствовать (bsNone). Если у окна обычная граница, то во время работы программы пользователь может при помощи мыши изменить размер окна. Изменить размер окна с тонкой границей нельзя. Если граница отсутствует, то на экран во время работы программы будет выведено окно без заголовка. Положение и размер такого окна во время работы программы изменить нельзя
BorderIcons	Кнопки управления окном. Значение свойства определяет, какие кнопки управления окном будут доступны пользователю во время работы программы. Значение свойства задается путем присвоения значений уточняющим свойствам biSystemMenu, biMinimize, biMaximize и biHelp. Свойство biSystemMenu определяет доступность кнопки системного меню, biMinimize — кнопки **Свернуть**, biMaximize — кнопки **Развернуть**, biHelp — кнопки вывода справочной информации
Icon	Значок в заголовке окна
Color	Цвет фона. Цвет можно задать, указав название цвета или привязку к текущей цветовой схеме операционной системы. Во втором случае цвет определяется текущей цветовой схемой, выбранным компонентом привязки и меняется при изменении цветовой схемы операционной системы
Font	Шрифт, используемый "по умолчанию" компонентами, находящимися на поверхности формы. Изменение свойства Font формы приводит к автоматическому изменению свойства Font компонента, располагающегося на поверхности формы. То есть компоненты наследуют свойство Font от формы (имеется возможность запретить наследование)
Canvas	Поверхность, на которую можно вывести графику

Компоненты

В этом разделе приведено краткое описание базовых компонентов C++ Builder. Подробное описание этих и других компонентов можно найти в справочной системе.

Label

Компонент Label (рис. 2.1) предназначен для вывода текста на поверхность формы. Свойства компонента (табл. 2.2) определяют вид и расположение текста.

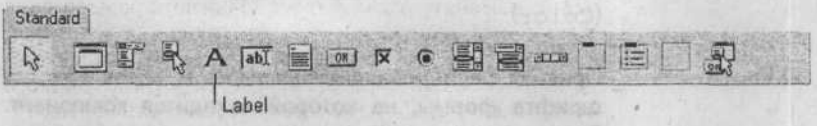


Рис. 2.1. Компонент Label — поле вывода текста

Таблица 2.2. *Свойства компонента* Label *(поле вывода текста)*

Свойство	Описание
Name	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
Caption	Отображаемый текст
Left	Расстояние от левой границы поля вывода до левой границы формы
Top	Расстояние от верхней границы поля вывода до верхней границы формы
Height	Высота поля вывода
Width	Ширина поля вывода
AutoSize	Признак того, что размер поля определяется его содержимым
WordWrap	Признак того, что слова, которые не помещаются в текущей строке, автоматически переносятся на следующую строку (значение свойства AutoSize должно быть false)
Alignment	Задает способ выравнивания текста внутри поля. Текст может быть выравнен по левому краю (taLeftJustify), по центру (taCenter) или по правому краю (taRightJustify)

Таблица 2.2 (окончание)

Свойство	Описание
Font	Шрифт, используемый для отображения текста. Уточняющие свойства определяют шрифт (Name), размер (Size), стиль (Style) и цвет символов (Color)
ParentFont	Признак наследования компонентом характеристик шрифта формы, на которой находится компонент. Если значение свойства равно true, то текст выводится шрифтом, установленным для формы
Color	Цвет фона области вывода текста
Transparent	Управляет отображением фона области вывода текста. Значение true делает область вывода текста прозрачной (область вывода не закрашивается цветом, заданным свойством Color)
Visible	Позволяет скрыть текст (false) или сделать его видимым (true)

Edit

Компонент Edit (рис. 2.2) представляет собой поле ввода/редактирования строки символов. Свойства компонента приведены в табл. 2.3.

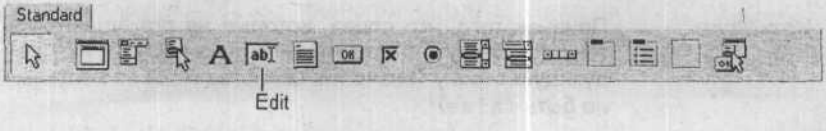


Рис. 2.2. Компонент Edit — поле ввода/редактирования строки символов

Таблица 2.3. *Свойства компонента* `Edit` *(поле ввода/редактирования)*

Свойство	Описание
`Name`	Имя компонента. Используется в программе для доступа к компоненту и его свойствам, в частности — для доступа к тексту, введенному в поле редактирования
`Text`	Текст, находящийся в поле ввода и редактирования
`Left`	Расстояние от левой границы компонента до левой границы формы
`Top`	Расстояние от верхней границы компонента до верхней границы формы
`Height`	Высота поля
`Width`	Ширина поля
`Font`	Шрифт, используемый для отображения вводимого текста
`ParentFont`	Признак наследования компонентом характеристик шрифта формы, на которой находится компонент. Если значение свойства равно `true`, то при изменении свойства `Font` формы автоматически меняется значение свойства `Font` компонента
`Enabled`	Используется для ограничения возможности изменить текст в поле редактирования. Если значение свойства равно `false`, то текст в поле редактирования изменить нельзя
`Visible`	Позволяет скрыть компонент (`false`) или сделать его видимым (`true`)

Button

Компонент `Button` (рис. 2.3) представляет собой командную кнопку. Свойства компонента приведены в табл. 2.4.

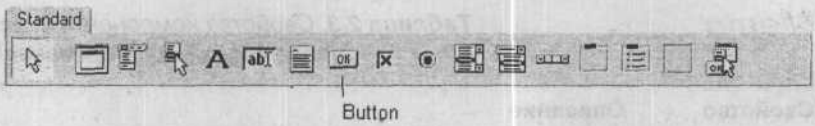


Рис. 2.3. Компонент `Button` — командная кнопка

Таблица 2.4. *Свойства компонента* `Button` *(командная кнопка)*

Свойство	Описание
`Name`	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
`Caption`	Текст на кнопке
`Left`	Расстояние от левой границы кнопки до левой границы формы
`Top`	Расстояние от верхней границы кнопки до верхней границы формы
`Height`	Высота кнопки
`Width`	Ширина кнопки
`Enabled`	Признак доступности кнопки. Если значение свойства равно `true`, то кнопка доступна. Если значение свойства равно `false`, то кнопка не доступна, например, в результате щелчка на кнопке событие `Click` не возникает
`Visible`	Позволяет скрыть кнопку (`false`) или сделать ее видимой (`true`)
`Hint`	Подсказка — текст, который появляется рядом с указателем мыши при позиционировании указателя на командной кнопке (для того чтобы текст появился, значение свойства `ShowHint` должно быть `true`)
`ShowHint`	Разрешает (`true`) или запрещает (`false`) отображение подсказки при позиционировании указателя на кнопке

Memo

Компонент Memo (рис. 2.4) представляет собой элемент редактирования текста, который может состоять из нескольких строк. Свойства компонента приведены в табл. 2.5.

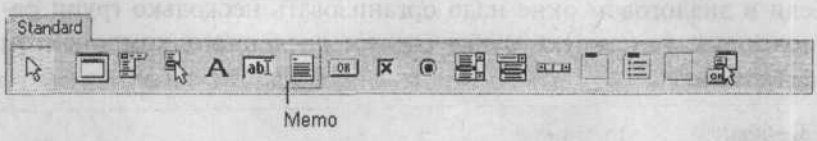


Рис. 2.4. Компонент Memo

Таблица 2.5. *Свойства компонента* Memo

Свойство	Описание
Name	Имя компонента. Используется в для доступа к свойствам компонента
Text	Текст, находящийся в поле Memo. Рассматривается как единое целое
Lines	Массив строк, соответствующий содержимому поля. Доступ к строке осуществляется по номеру. Строки нумеруются с нуля
Left	Расстояние от левой границы поля до левой границы формы
Top	Расстояние от верхней границы поля до верхней границы формы
Height	Высота поля
Width	Ширина поля
Font	Шрифт, используемый для отображения вводимого текста
ParentFont	Признак наследования свойств шрифта родительской формы

RadioButton

Компонент RadioButton (рис. 2.5) представляет зависимую кнопку, состояние которой определяется состоянием других кнопок группы. Свойства компонента приведены в табл. 2.6.

Если в диалоговом окне надо организовать несколько групп радиокнопок, то каждую группу следует представить компонентом RadioGroup.

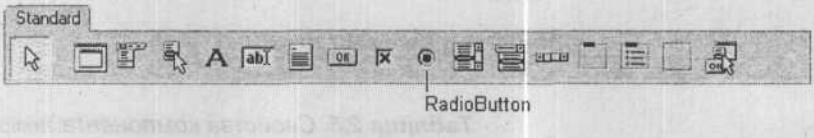


Рис. 2.5. Компонент RadioButton

Таблица 2.6. *Свойства компонента* RadioButton

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Caption	Текст, который находится справа от кнопки
Checked	Состояние, внешний вид кнопки: если кнопка выбрана, то Checked = true, если кнопка не выбрана, то Checked = false
Left	Расстояние от левой границы флажка до левой границы формы
Top	Расстояние от верхней границы флажка до верхней границы формы
Height	Высота поля вывода поясняющего текста
Width	Ширина поля вывода поясняющего текста
Font	Шрифт, используемый для отображения поясняющего текста
ParentFont	Признак наследования характеристик шрифта родительской формы

CheckBox

Компонент CheckBox (рис. 2.6) представляет собой независимую кнопку (переключатель). Свойства компонента приведены в табл. 2.7.

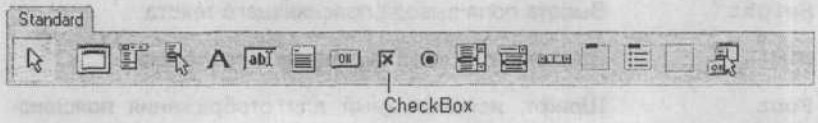


Рис. 2.6. Компонент CheckBox

Таблица 2.7. *Свойства компонента* CheckBox

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Caption	Текст, который находится справа от флажка
Checked	Состояние, внешний вид флажка: если флажок установлен (в квадратике есть "галочка"), то значение Checked равно true; если флажок сброшен (нет "галочки"), то значение Checked равно false
State	Состояние флажка. В отличие от свойства Checked, позволяет различать установленное, сброшенное и промежуточное состояния. Состояние флажка определяет одна из констант: cbChecked (установлен); cbGrayed (серый, неопределенное состояние); cbUnChecked (сброшен)
AllowGrayed	Свойство определяет, может ли флажок быть в промежуточном состоянии: если значение AllowGrayed равно false, то флажок может быть только установленным или сброшенным; если значение AllowGrayed равно true, то допустимо промежуточное состояние
Left	Расстояние от левой границы флажка до левой границы формы

Таблица 2.7 (окончание)

Свойство	Описание
Top	Расстояние от верхней границы флажка до верхней границы формы
Height	Высота поля вывода поясняющего текста
Width	Ширина поля вывода поясняющего текста
Font	Шрифт, используемый для отображения поясняющего текста
ParentFont	Признак наследования характеристик шрифта родительской формы

ListBox

Компонент ListBox (рис. 2.7) представляет собой список, в котором можно выбрать нужный элемент. Свойства компонента приведены в табл. 2.8.

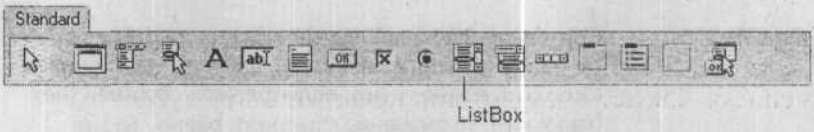


Рис. 2.7. Компонент ListBox

Таблица 2.8. Свойства компонента ListBox

Свойство	Описание
Name	Имя компонента. В программе используется для доступа к компоненту и его свойствам
Items->Strings	Элементы списка — массив строк (нумеруются с нуля)
Count	Количество элементов списка
Sorted	Признак необходимости автоматической сортировки (true) списка после добавления очередного элемента.

Таблица 2.8 (окончание)

Свойство	Описание
ItemIndex	Номер выбранного элемента (элементы списка нумеруются с нуля). Если в списке ни один из элементов не выбран, то значение свойства равно минус один
Left	Расстояние от левой границы списка до левой границы формы
Top	Расстояние от верхней границы списка до верхней границы формы
Height	Высота поля списка
Width	Ширина поля списка
Font	Шрифт, используемый для отображения элементов списка
ParentFont	Признак наследования свойств шрифта родительской формы

ComboBox

Компонент ComboBox (рис. 2.8) дает возможность ввести данные в поле редактирования путем набора на клавиатуре или выбором из списка. Свойства компонента приведены в табл. 2.9.

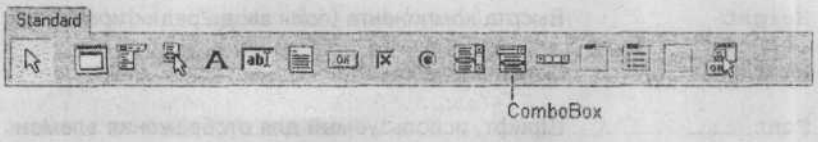


Рис. 2.8. Компонент ComboBox

Таблица 2.9. Свойства компонента ComboBox

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Text	Текст, находящийся в поле ввода/редактирования

Таблица 2.9 (окончание)

Свойство	Описание
`Items->Strings`	Элементы списка – массив строк (нумеруются с нуля)
`Count`	Количество элементов списка
`ItemIndex`	Номер элемента, выбранного в списке. Если ни один из элементов списка не был выбран, то значение свойства равно минус 2
`Sorted`	Признак необходимости автоматической сортировки (`true`) списка после добавления очередного элемента
`DropDownCount`	Количество отображаемых элементов в раскрытом списке. Если количество элементов списка больше чем `DropDownCount`, то появляется вертикальная полоса прокрутки
`Left`	Расстояние от левой границы компонента до левой границы формы
`Top`	Расстояние от верхней границы компонента до верхней границы формы
`Height`	Высота компонента (поля ввода/редактирования)
`Width`	Ширина компонента
`Font`	Шрифт, используемый для отображения элементов списка
`ParentFont`	Признак наследования свойств шрифта родительской формы

StringGrid

Компонент `StringGrid` (рис. 2.9) представляет собой таблицу, ячейки которой содержат строки символов. Свойства компонента приведены в табл. 2.10.

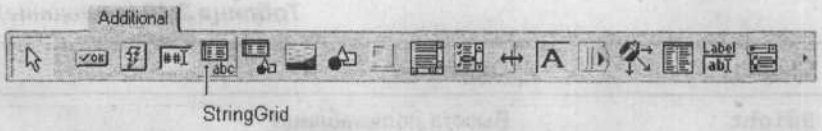


Рис. 2.9. Компонент `StringGrid`

Таблица 2.10. *Свойства компонента* `StringGrid`

Свойство	Описание
`Name`	Имя компонента. Используется в программе для доступа к компоненту и его свойствам
`ColCount`	Количество колонок таблицы
`RowCount`	Количество строк таблицы
`DefaultColWidth`	Ширина колонок таблицы
`DefaultRowHeight`	Высота строк таблицы
`FixedCols`	Количество зафиксированных слева колонок таблицы. Зафиксированные колонки выделяются цветом и при горизонтальной прокрутке таблицы остаются на месте
`FixedRows`	Количество зафиксированных сверху строк таблицы. Зафиксированные строки выделяются цветом и при вертикальной прокрутке таблицы остаются на месте
`Cells`	Соответствующий таблице двумерный массив. Ячейке таблицы, находящейся на пересечении столбца с номером `col` и строки с номером `row`, соответствует элемент `cells[col][row]`
`GridLineWidth`	Ширина линий, ограничивающих ячейки таблицы
`Left`	Расстояние от левой границы поля таблицы до левой границы формы
`Top`	Расстояние от верхней границы поля таблицы до верхней границы формы

Таблица 2.10 (окончание)

Свойство	Описание
Height	Высота поля таблицы
Width	Ширина поля таблицы
Options.goEditing	Признак допустимости редактирования содержимого ячеек таблицы. true — редактирование разрешено, false — запрещено
Options.goTab	Разрешает (true) или запрещает (false) использование клавиши <Tab> для перемещения курсора в следующую ячейку таблицы
Options.goAlwaysShowEditor	Признак нахождения компонента в режиме редактирования. Если значение свойства false, то для того, чтобы в ячейке появился курсор, надо начать набирать текст, нажать клавишу <F2> или сделать щелчок мышью
Font	Шрифт, используемый для отображения содержимого ячеек таблицы
ParentFont	Признак наследования характеристик шрифта формы

Image

Компонент Image (рис. 2.10) обеспечивает вывод на поверхность формы иллюстраций, представленных в формате BMP (чтобы компонент можно было использовать для отображения иллюстраций в формате JPG, надо подключить модуль JPEG — включить в текст программы директиву #include <jpeg.hpp>). Свойства компонента Image приведены в табл. 2.11.

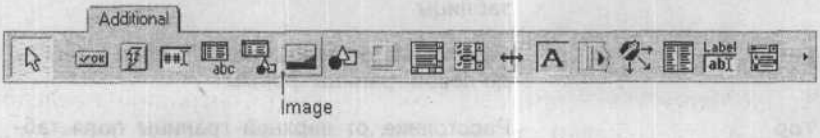


Рис. 2.10. Компонент Image

Таблица 2.11. *Свойства компонента* `Image`

Свойство	Описание
`Picture`	Иллюстрация, которая отображается в поле компонента
`Width, Height`	Размер компонента. Если размер компонента меньше размера иллюстрации и значение свойств `AutoSize`, `Strech` и `Proportional` равно `false`, то отображается часть иллюстрации
`Proportional`	Признак автоматического масштабирования картинки без искажения. Чтобы масштабирование было выполнено, значение свойства `AutoSize` должно быть `false`
`Strech`	Признак автоматического масштабирования (сжатия или растяжения) иллюстрации в соответствии с реальным размером компонента. Если размер компонента не пропорционален размеру иллюстрации, то иллюстрация будет искажена
`AutoSize`	Признак автоматического изменения размера компонента в соответствии с реальным размером иллюстрации
`Center`	Признак определяет расположение картинки в поле компонента по горизонтали, если ширина картинки меньше ширины поля компонента. Если значение свойства равно `false`, то картинка прижата к правой границе компонента, если `true` — то картинка располагается по центру
`Visible`	Отображается ли компонент, и, соответственно, иллюстрация, на поверхности формы
`Canvas`	Поверхность, на которую можно вывести графику

Timer

Компонент `Timer` (рис. 2.11) обеспечивает генерацию последовательности событий `OnTimer`. Свойства компонента приведены в табл. 2.12.

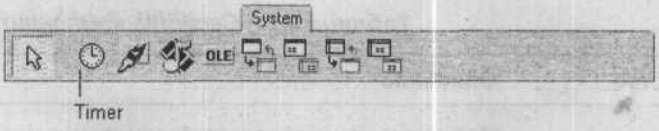


Рис. 2.11. Компонент `Timer`

Таблица 2.12. *Свойства компонента* `Timer`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к компоненту
`Interval`	Период генерации события `OnTimer`. Задается в миллисекундах
`Enabled`	Разрешение работы. Разрешает (значение `true`) или запрещает (значение `false`) генерацию события `OnTimer`

SpeedButton

Компонент `SpeedButton` (рис. 2.12) представляет собой кнопку, на поверхности которой находится картинка. Свойства компонента приведены в табл. 2.13.

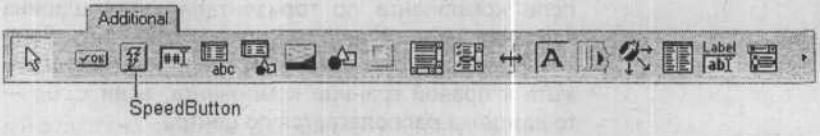


Рис. 2.12. Компонент `SpeedButton`

Таблица 2.13. *Свойства компонента* `SpeedButton`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к компоненту и его свойствам
`Glyph`	Битовый образ, в котором находятся картинки для каждого из состояний кнопки. В битовом образе может быть до четырех изображений кнопки (рис. 2.13)

Таблица 2.13 (окончание)

Свойство	Описание
NumGlyphs	Количество картинок в битовом образе Glyph
Flat	Свойство Flat определяет вид кнопки (наличие границы). Если значение свойства равно true, то граница кнопки появляется только при позиционировании указателя мыши на кнопке
GroupIndex	Идентификатор группы кнопок. Кнопки, имеющие одинаковый идентификатор группы, работают подобно радиокнопкам: нажатие одной из кнопок группы вызывает срабатывание других кнопок этой группы. Чтобы кнопку можно было зафиксировать, значение свойства GroupIndex не должно быть равно нулю
Down	Идентификатор состояния кнопки. Изменить значение свойства можно, если значение свойства GroupIndex не равно 0
AllowAllUp	Свойство определяет возможность отжать кнопку. Если кнопка нажата и значение свойства равно true, то кнопку можно отжать
Left	Расстояние от левой границы кнопки до левой границы формы
Top	Расстояние от верхней границы кнопки до верхней границы формы
Height	Высота кнопки
Width	Ширина кнопки
Enabled	Признак доступности кнопки. Если значение свойства равно true, то кнопка доступна. Если значение свойства равно false, то кнопка не доступна
Visible	Позволяет скрыть кнопку (false) или сделать ее видимой (true)
Hint	Подсказка — текст, который появляется рядом с указателем мыши при позиционировании указателя на командной кнопке (для того чтобы текст появился, надо, чтобы значение свойства ShowHint было true)
ShowHint	Разрешает (true) или запрещает (false) отображение подсказки при позиционировании указателя на кнопке

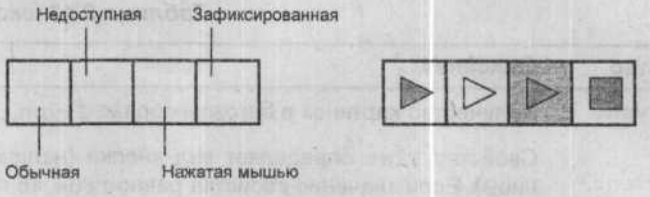


Рис. 2.13. Структура и пример битового образа `Glyph`: картинки, соответствующие состоянию кнопки Play/Stop

UpDown

Компонент `UpDown` (рис. 2.14) представляет собой две кнопки, используя которые можно изменить значение внутренней переменной-счетчика на определенную величину. Увеличение или уменьшение значения происходит при каждом щелчке на одной из кнопок. Свойства компонента приведены в табл. 2.14.

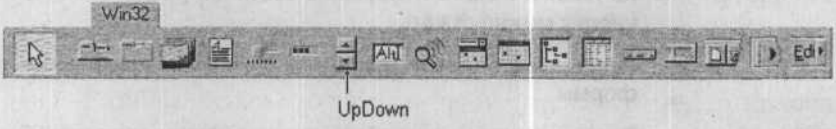


Рис. 2.14. Компонент `UpDown`

Таблица 2.14. Свойства компонента `UpDown`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к компоненту и его свойствам
`Position`	Счетчик. Значение свойства изменяется в результате щелчка на кнопке `Up` (увеличивается) или `Down` (уменьшается). Диапазон изменения определяют свойства `Min` и `Max`, величину изменения — свойство `Increment`
`Min`	Нижняя граница диапазона изменения свойства `Position`

Таблица 2.14 (окончание)

Свойство	Описание
Max	Верхняя граница диапазона изменения свойства Position
Increment	Величина, на которую изменяется значение свойства Position в результате щелчка на одной из кнопок компонента
Associate	Определяет компонент (Edit — поле ввода/редактирования), используемый в качестве индикатора значения свойства Position. Если значение свойства задано, то при изменении содержимого поля редактирования автоматически меняется значение свойства Position
Orientation	Задает ориентацию кнопок компонента. Кнопки могут быть ориентированы вертикально (udVertical) или горизонтально (udHorizontal)

ProgressBar

Компонент ProgressBar (рис. 2.15) представляет собой индикатор, который обычно используется для наглядного представления протекания процесса, например, обработки (копирования) файлов, загрузки информации и т. п. Свойства компонента ProgressBar приведены в табл. 2.15.

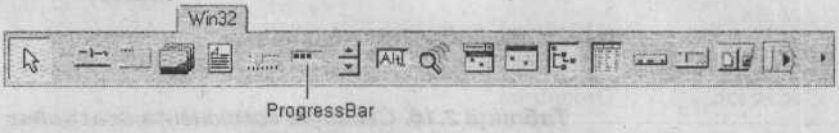


Рис. 2.15. Компонент ProgressBar

Таблица 2.15. Свойства компонента ProgressBar

Свойство	Описание
Position	Значение, которое отображается в поле компонента в виде последовательности прямоугольников (сегментов) или полосы. Количество прямоугольников (длина полосы) пропорционально значению свойства Position

Таблица 2.15 (окончание)

Свойство	Описание
Min	Минимально допустимое значение свойства Position
Max	Максимально допустимое значение свойства Position
Step	Приращение (шаг) изменения значения свойства Position при использовании для изменения значения свойства Value метода StepIt
Smooth	Определяет вид индикатора. Индикатор может быть разделен на сегменты (false) или представлять собой полосу

StatusBar

Компонент StatusBar (рис. 2.16) представляет собой область (полосу) вывода служебной информации, которая находится в нижней части окна программы (иногда полосу вывода служебной информации называют панелью или строкой состояния). Обычно панель вывода служебной информации разделена на области. Свойства компонента StatusBar приведены в табл. 2.16.

Рис. 2.16. Компонент StatusBar

Таблица 2.16. Свойства компонента StatusBar

Свойство	Описание
Panels	Коллекция объектов типа TStatusPanel (табл. 2.17), каждый из которых представляет собой отдельную область панели StatusBar
SimplePanel	Признак, определяющий вид панели состояния. Если значение свойства равно true, то панель состояния не разделяется на области, а текст, отображаемый в строке состояния, определяет свойство SimpleText

Таблица 2.16 (окончание)

Свойство	Описание
SimpleText	Текст, который отображается в панели состояния, если панель не разделена на области (значение свойства SimplePanel равно true). Если панель разделена на области, то текст, находящийся в области, определяет свойство Text соответствующего элемента коллекции Panels

Таблица 2.17. Свойства обьекта TStatusPanel

Свойство	Описание
Text	Текст, отображаемый в области
Width	Ширина области. Ширина самой правой области устанавливается так, что правая граница области совпадает с правой границей панели (компонента StatusBar)

Animate

Компонент Animate (рис. 2.17) позволяет воспроизводить простую, не сопровождаемую звуком анимацию, кадры которой находятся в файле формата AVI. Свойства компонента приведены в табл. 2.18.

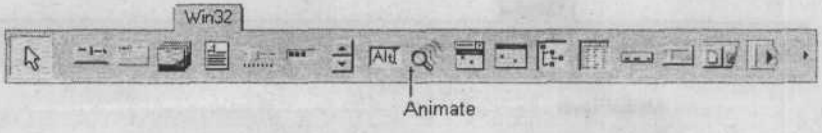


Рис. 2.17. Компонент Animate

Таблица 2.18. Свойства компонента Animate

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента и управления его поведением

Таблица 2.18 (окончание)

Свойство	Описание
FileName	Имя файла формата AVI, в котором находится анимация, отображаемая при помощи компонента
StartFrame	Номер кадра, с которого начинается отображение анимации
StopFrame	Номер кадра, на котором заканчивается отображение анимации
Activate	Признак активизации процесса отображения кадров анимации
Color	Цвет фона компонента (цвет "экрана"), на котором воспроизводится анимация
Transparent	Режим использования "прозрачного" цвета при отображении анимации
Repetitions	Количество повторов отображения анимации

MediaPlayer

Компонент MediaPlayer (рис. 2.18) позволяет воспроизвести видеоролик, звук и сопровождаемую звуком анимацию. Свойства компонента приведены в табл. 2.19.

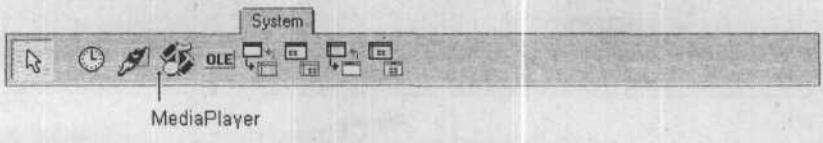


Рис. 2.18. Компонент MediaPlayer

Таблица 2.19. Свойства компонента MediaPlayer

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента и управления работой плеера

Таблица 2.19 (окончание)

Свойство	Описание
DeviceType	Тип устройства. Определяет конкретное устройство, которое представляет собой компонент MediaPlayer. Тип устройства задается именованной константой: dtAutoSelect — тип устройства определяется автоматически, dtVaweAudio — проигрыватель звука, dtAVIVideo — видеопроигрыватель, dtCDAudio — CD-проигрыватель
FileName	Имя файла, в котором находится воспроизводимый звуковой фрагмент или видеоролик
AutoOpen	Признак автоматического открытия сразу после запуска программы, файла видеоролика или звукового фрагмента
Display	Определяет компонент, на поверхности которого воспроизводится видеоролик (обычно в качестве экрана для отображения видео используют компонент Panel)
VisibleButtons	Составное свойство. Определяет видимые кнопки компонента. Позволяет сделать невидимыми некоторые кнопки

Table

Компонент Table (рис. 2.19) представляет собой таблицу базы данных. Свойства компонента приведены в табл. 2.20.

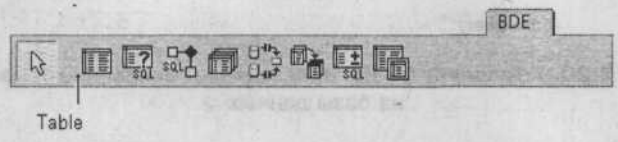


Рис. 2.19. Компонент Table — таблица базы данных

Таблица 2.20. Свойства компонента Table

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента

Таблица 2.20 (окончание)

Свойство	Описание
DatabaseName	Имя базы данных, частью которой является таблица (файл данных), для доступа к которой используется компонент. В качестве значения свойства следует использовать псевдоним базы данных
TableName	Имя файла данных (таблицы данных), для доступа к которому используется компонент
TableType	Тип таблицы. Таблица может быть набором данных в формате Paradox (ttParadox), dBase (ttDBase), FoxPro (ttFoxPro) или представлять собой форматированный текстовый файл (ttASCII)
Active	Признак того, что таблица активна (файл данных открыт). В результате присваивания свойству значения true происходит открытие файла таблицы

Query

Компонент Query (рис. 2.20) представляет часть базы данных — записи, удовлетворяющие критерию SQL-запроса к таблице. Свойства компонента приведены в табл. 2.21.

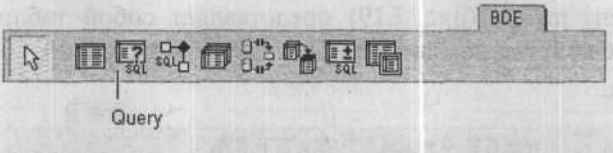


Рис. 2.20. Компонент Query обеспечивает выбор информации из базы данных

Таблица 2.21. Свойства компонента Query

Свойство	Описание
Name	Имя компонента. Используется компонентом DataSource для связи результата выполнения запроса (набора записей) с компонентом, обеспечивающим просмотр записей, например, DBGrid

Таблица 2.21 (окончание)

Свойство	Описание
SQL	Записанный на языке SQL-запрос к базе данных (к таблице)
Active	При присвоении свойству значения true активизирует выполнение запроса
RecordCount	Количество записей в базе данных, удовлетворяющих критерию запроса

DataSource

Компонент DataSource (рис. 2.21) обеспечивает связь между данными, представленными компонентом Table или Query, и компонентами отображения данных (DBEdit, DBMemo или DBGrid). Свойства компонента приведены в табл. 2.22.

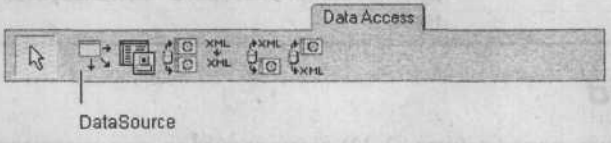


Рис. 2.21. Компонент DataSource обеспечивает связь между данными и компонентом просмотра/редактирования

Таблица 2.22. Свойства компонента DataSource

Свойство	Описание
Name	Имя компонента. Используется компонентом отображения данных для доступа к компоненту и, следовательно, к данным, связь с которыми обеспечивает компонент
DataSet	Компонент, представляющий собой входные данные (Table или Query)

DBEdit, DBMemo, DBText

Компоненты DBEdit и DBMemo (рис. 2.22) обеспечивают просмотр и редактирование полей записи базы данных, компонент DBText — только просмотр. Свойства компонентов приведены в табл. 2.23.

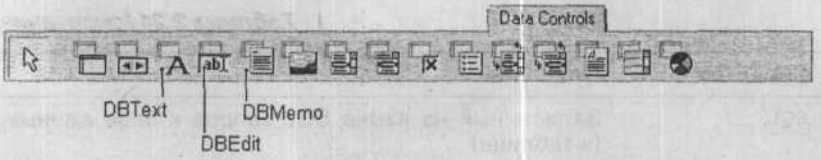


Рис. 2.22. Компоненты просмотра и редактирования полей БД

Таблица 2.23. *Свойства компонентов* `DBText`, `DBEdit` *и* `DBMemo`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к свойствам компонента
`DataSource`	Компонент-источник данных
`DataField`	Поле базы данных, для отображения или редактирования которого используется компонент

DBGrid

Компонент `DBGrid` (рис. 2.23) используется для просмотра и редактирования базы данных в режиме таблицы. Свойства компонента приведены в табл. 2.24.

Рис. 2.23. Компонент `DBGrid` обеспечивает работу с базой данных в режиме таблицы

Таблица 2.24. *Свойства компонента* `DBGrid`

Свойство	Описание
`Name`	Имя компонента
`DataSource`	Источник отображаемых в таблице данных (компонент `DataSource`)

Таблица 2.24 (окончание)

Свойство	Описание
Columns	Свойство Columns представляет собой массив объектов типа TColumn, каждый из которых определяет колонку таблицы и отображаемую в ней информацию (табл. 2.25)
Options.dgTitles	Разрешает вывод строки заголовка столбцов
Options.dgIndicator	Разрешает вывод колонки индикатора. Во время работы с базой данных текущая запись помечается в колонке индикатора треугольником, новая запись — звездочкой, редактируемая — специальным значком
Options.dgColumnResize	Разрешает менять во время работы программы ширину колонок таблицы
Options.dgColLines	Разрешает выводить линии, разделяющие колонки таблицы
Options.dgRowLines	Разрешает выводить линии, разделяющие строки таблицы

***Таблица 2.25.** Свойства объекта TColumn*

Свойство	Описание
FieldName	Поле записи, содержимое которого выводится в колонке
Width	Ширина колонки в пикселах
Font	Шрифт, используемый для вывода текста в ячейках колонки
Color	Цвет фона колонки
Alignment	Способ выравнивания текста в ячейках колонки. Текст может быть выровнен по левому краю (taLeftJustify), по центру (taCenter) или по правому краю (taRightJustify)

Таблица 2.25 (окончание)

Свойство	Описание
`Title.Caption`	Заголовок колонки. Значением по умолчанию является имя поля записи
`Title.Alignment`	Способ выравнивания заголовка колонки. Заголовок может быть выровнен по левому краю (`taLeftJustify`), по центру (`taCenter`) или по правому краю (`taRightJustify`)
`Title.Color`	Цвет фона заголовка колонки
`Title.Font`	Шрифт заголовка колонки

DBNavigator

Компонент `DBNavigator` (рис. 2.24) обеспечивает перемещение указателя текущей записи, активизацию режима редактирования, добавление и удаление записей.

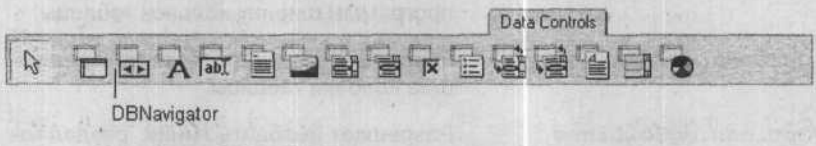


Рис. 2.24. Значок компонента `DBNavigator`

Компонент представляет собой совокупность командных кнопок (рис. 2.25, табл. 2.26). Свойства компонента приведены в табл. 2.27.

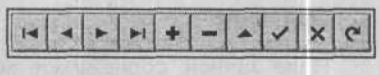


Рис. 2.25. Компонент `DBNavigator`

Таблица 2.26. Кнопки компонента `DBNavigator`

	Кнопка	Обозначение	Действие
	К первой	`nbFirst`	Указатель текущей записи перемещается к первой записи файла данных
	К предыдущей	`nbPrior`	Указатель текущей записи перемещается к предыдущей записи файла данных

Таблица 2.26 (окончание)

	Кнопка	Обозначение	Действие
	К следующей	nbNext	Указатель текущей записи перемещается к следующей записи файла данных
	К последней	nbLast	Указатель текущей записи перемещается к последней записи файла данных
	Добавить	nbInsert	В файл данных добавляется новая запись
	Удалить	nbDelete	Удаляется текущая запись файла данных
	Редактирование	nbEdit	Устанавливает режим редактирования текущей записи
	Сохранить	nbPost	Изменения, внесенные в текущую запись, записываются в файл данных
	Отменить	Cancel	Отменяет внесенные в текущую запись изменения
	Обновить	nbRefresh	Записывает внесенные изменения в файл

***Таблица 2.27.** Свойства компонента `DBNavigator`*

Свойство	Определяет
`Name`	Имя компонента. Используется для доступа к свойствам компонента
`DataSource`	Имя компонента, являющегося источником данных. В качестве источника данных может выступать база данных (компонент `Database`), таблица (компонент `Table`) или результат выполнения запроса (компонент `Query`)
`VisibleButtons`	Видимые командные кнопки

Графика

Canvas

Canvas — это поверхность (формы или компонента Image), на которой соответствующие методы (табл. 2.28) могут вычерчивать графические примитивы. Вид графических элементов определяют свойства поверхности, на которой эти элементы вычерчиваются (табл. 2.29).

Таблица 2.28. *Методы объекта* Canvas

Метод	Описание
TextOut(x,y,s)	Выводит строку s от точки с координатами (x, y). Шрифт определяет свойство Font поверхности, на которую выводится тест, цвет закраски области вывода текста — свойство Brush этой же поверхности
Draw(x,y,b)	Выводит от точки с координатами (x, y) битовый образ b. Если значение свойства Transparent поверхности, на которую выполняется вывод равно true, то точки, цвет которых совпадает с цветом левой нижней точки битового образа, не отображаются
LineTo(x,y)	Вычерчивает линию из текущей точки в точку с указанными координатами. Вид линии определяет свойство Pen
MoveTo(x,y)	Перемещает указатель текущей точки в точку с указанными координатами
PolyLine(pl)	Вычерчивает ломаную линию. Координаты точек перегиба задает параметр pl — массив структур типа TPoint. Если первый и последний элементы массива одинаковые, то будет вычерчен замкнутый контур. Вид линии определяет свойство Pen

Таблица 2.28 *(продолжение)*

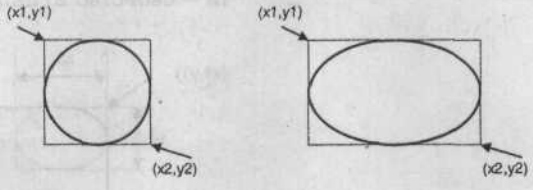
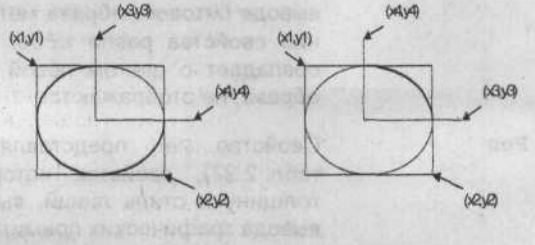
Метод	Описание
`Polygon(pl)`	Вычерчивает и закрашивает многоугольник. Координаты углов задает параметр `pl` — массив структур типа `TPoint`. Первый и последний элементы массива должны быть одинаковые. Вид границы определяет свойство `Pen`, цвет и стиль закраски внутренней области — свойство `Brush`
`Ellipse (x1,y,x2,y2)`	Вычерчивает эллипс, окружность или круг. Параметры `x1`, `y1`, `x2` и `y2` задают размер прямоугольника, в который вписывается эллипс. Вид линии определяет свойство `Pen` 
`Arc(x1,y1,x2,y2, x3,y3,x4,y4)`	Вычерчивает дугу. Параметры `x1`, `y1`, `x2`, `y2` определяют эллипс, из которого вырезается дуга, параметры `x2`, `y2`, `x3`, и `y4` — координаты концов дуги. Дуга вычерчивается против часовой стрелки от точки (`x3`, `y3`) к точке (`x4`, `y4`). Вид линии (границы) определяет свойство `Pen`, цвет и способ закраски внутренней области — свойство `Brush` 

Таблица 2.28 (окончание)

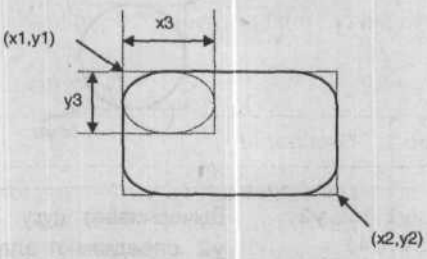
Метод	Описание
`Rectangle(x1,y,x2,y2)`	Вычерчивает прямоугольник. Параметры `x1`, `y1`, `x2` и `y2` задают координаты левого верхнего и правого нижнего углов. Вид линии определяет свойство `Pen`, цвет и способ закраски внутренней области — свойство `Brush`
`RoundRec (x1,y1,x2,y2,x3,y3)`	Вычерчивает прямоугольник со скругленными углами. Параметры `x1`, `y1`, `x2` и `y2` задают координаты левого верхнего и правого нижнего углов, `x3` и `y3` — радиус закругления. Вид линии определяет свойство `Pen`, цвет и способ закраски внутренней области — свойство `Brush` 

***Таблица 2.29.** Свойства объекта `Canvas`*

Свойство	Описание
`Transparent`	Признак использования "прозрачного" цвета при выводе битового образа методом `Draw`. Если значение свойства равно `true`, то точки, цвет которых совпадает с цветом левой нижней точки битового образа, не отображаются
`Pen`	Свойство `Pen` представляет собой объект (см. табл. 2.27), свойства которого определяют цвет толщину и стиль линий, вычерчиваемых методами вывода графических примитивов

Таблица 2.29 (окончание)

Свойство	Описание
Brush	Свойство Brush представляет собой объект (см. табл. 2.28), свойства которого определяют цвет и стиль закраски областей, вычерчиваемых методами вывода графических примитивов
Font	Свойство Font представляет собой объект, уточняющие свойства которого определяют шрифт (название, размер, цвет, способ оформления), используемый для вывода на поверхность холста текста

Pen

Объект Pen является свойством объекта Canvas. Свойства объекта Pen (табл. 2.30) определяют цвет, стиль и толщину линий, вычерчиваемых методами вывода графических примитивов.

Таблица 2.30. Свойства объекта Pen

Свойство	Описание
Color	Цвет линии (clBlack — черный; clMaroon — каштановый; clGreen — зеленый; clOlive — оливковый; clNavy — темно-синий; clPurple — розовый; clTeal — зелено-голубой; clGray — серый; clSilver — серебристый; clRed — красный; clLime — салатный; clBlue — синий; clFuchsia — ярко-розовый; clAqua — бирюзовый; clWhite — белый)
Style	Стиль (вид) линии. Линия может быть: psSolid — сплошная; psDash — пунктирная (длинные штрихи); psDot — пунктирная (короткие штрихи); psDashDot — пунктирная (чередование длинного и короткого штрихов); psDashDotDot — пунктирная (чередование одного длинного и двух коротких штрихов); psClear — не отображается (используется, если не надо изображать границу, например, прямоугольника)
Width	Толщина линии задается в пикселах. Толщина пунктирной линии не может быть больше 2

Brush

Объект Brush является свойством объекта Canvas. Свойства объекта Brush (табл. 2.31) определяют цвет, стиль закраски внутренних областей контуров, вычерчиваемых методами вывода графических примитивов.

Таблица 2.31. Свойства объекта TBrush (кисть)

Свойство	Описание
Color	Цвет закрашивания замкнутой области
Style	Стиль (тип) заполнения области (bsSolid — сплошная заливка; bsClear — область не закрашивается; bsHorizontal — горизонтальная штриховка; bsVertical — вертикальная штриховка; bsFDiagonal — диагональная штриховка с наклоном линий вперед; bsBDiagonal — диагональная штриховка с наклоном линий назад; bsCross — горизонтально-вертикальная штриховка, в клетку; bsDiagCross — диагональная штриховка в клетку)

Функции

В этом разделе приведено краткое описание наиболее часто используемых функций (табл. 2.32—2.35). Подробное описание можно найти в справочной системе.

Функции ввода и вывода

Таблица 2.32. Функции ввода и вывода

Функция	Описание
InputBox(*Заголовок*, *Подсказка*, *Значение*)	В результате выполнения функции на экране появляется диалоговое окно, в поле которого пользователь может ввести строку символов. Значением функции является введенная строка. Параметр *Значение* задает значение функции "по умолчанию", т. е. строку, которая будет в поле редактирования в момент появления окна

Таблица 2.32 (окончание)

Функция	Описание
`ShowMessage(s)`	Процедура `ShowMessage` выводит окно, в котором находятся сообщение `s` и командная кнопка **OK**
`MessageDlg(s,t,b,h)`	Выводит на экран диалоговое окно с сообщением `s` и возвращает код кнопки, щелчком на которой пользователь закрыл окно. Параметр `t` определяет тип окна: `mtWarning` — внимание; `mtError` — ошибка; `mtInformation` — информация; `mtConfirmation` — запрос; `mtCustom` — пользовательское (без значка). Параметр `b` (множество — заключенный в квадратные скобки список констант) задает командные кнопки диалогового окна (`mbYes`, `mbNo`, `mbOK`, `mbCancel`, `mbHelp`, `mbAbort`, `mbRetry`, `mbIgnore` и `mbAll`). Параметр `h` задает раздел справочной системы программы, который появится в результате нажатия кнопки **Help** или клавиши <F1>. Если справочная система не используется, значение параметра должно быть `0`. Значением функции может быть одна из констант: `mrAbort`, `mrYes`, `mrOk`, `mrRetry`, `mrNo`, `mrCancel`, `mrIgnore` или `mrAll`, обозначающая соответствующую командную кнопку

Математические функции

***Таблица 2.33.** Математические функции*

Функция	Значение
`abs(n)`	Абсолютное значение `n`
`sqrt(n)`	Квадратный корень из `n`
`exp(n)`	Экспонента `n`

Таблица 2.33 (окончание)

Функция	Значение
`rardom(n)`	Случайное целое число в диапазоне от 0 до `n-1` (перед первым обращением к функции необходимо вызвать функцию `randomize()`, которая выполнит инициализацию программного генератора случайных чисел)
`sin(α)`	Синус выраженного в радианах угла α
`cos(α)`	Косинус выраженного в радианах угла α
`tan(α)`	Тангенс выраженного в радианах угла α
`asin(n)`, `acos(n)`, `atan(n)`	Угол (в радианах), синус, косинус и тангенс которого равен `n`

Следует обратить внимание, для того чтобы в программе были доступны приведенные функции, в ее текст надо включить директиву `#include <math.h>`.

Величина угла тригонометрических функций должна быть выражена в радианах. Для преобразования величины угла из градусов в радианы используется формула `(a*3.1415256)/180`, где `a` — величина угла в градусах, `3.1415926` — число "ПИ". Вместо константы `3.1415926` можно использовать стандартную именованную константу `M_PI`. Константа `M_PI` определена в файле math.h.

Функции преобразования

Таблица 2.34. Функции преобразования

Функция	Значение/описание
`IntToStr(k)`	Строка, являющаяся изображением целого `k`
`FloatToStr(n)`	Строка, являющаяся изображением вещественного `n`

Таблица 2.34 (окончание)

Функция	Значение/описание
`FloatToStrF(n,f,k,m)`	Строка, являющаяся изображением вещественного `n`. При вызове функции указывают: `f` — формат; `k` — точность; `m` — количество цифр после десятичной точки.
	Формат определяет способ изображения числа: `ffGeneral` — универсальный; `ffExponent` — научный; `ffFixed` — с фиксированной точкой; `ffNumber` — с разделителями групп разрядов; `ffCurrency` — финансовый. Точность — нужное общее количество цифр: 7 или меньше для значения типа `Single`, 15 или меньше для значения типа `Double` и 18 или меньше для значения типа `Extended`
`StrToInt(s)`	Целое, изображением которого является строка `s`
`StrToFloat(s)`	Вещественное, изображением которого является строка `s`

Функции манипулирования датами и временем

Большинству функций манипулирования датами в качестве параметра передается переменная типа `TDateTime`, которая хранит информацию о дате и времени.

Для того чтобы в программе были доступны функции `DayOf`, `WeekOf`, `MonthOf` и другие, в ее текст надо включить директиву `#include <DateUtils.hpp>`.

Таблица 2.35. Функции манипулирования датами и временем

Функция	Значение
`Now()`	Системная дата и время — значение типа `TDateTime`
`DateToStr(dt)`	Строка символов, изображающая дату в формате dd.mm.yyyy

Таблица 2.35 (окончание)

Функция	Значение
`TimeToStr(dt)`	Строка символов, изображающая время в формате hh:mm:ss
`DayOf(dt)`	День (номер дня в месяце), соответствующий дате, указанной в качестве параметра функции
`MonthOf(dt)`	Номер месяца, соответствующий дате, указанной в качестве параметра функции
`WeekOf(dt)`	Номер недели, соответствующий дате, указанной в качестве параметра функции
`YearOf(dt)`	Год, соответствующий указанной дате
`DayOfWeek(dt)`	Номер дня недели, соответствующий указанной дате: 1 — воскресенье, 2 — понедельник, 3 — вторник и т. д.
`StartOfWeek(w)`	Дата первого дня указанной недели
`HourOf(dt)`	Количество часов
`MinuteOf(dt)`	Количество минут
`SecondOf(dt)`	Количество секунд
`DecodeDate(dt,y,m,d)`	Возвращает год, месяц и день, представленные отдельными числами
`DecodeTime(dt,h,m,s,ms)`	Возвращает время (часы, минуты, секунды и миллисекунды), представленное отдельными числами
`FormatDateTime(s,dt)`	Строка символов, представляющая собой дату или время. Способ представления задает строка формата `s`, например, строка `dd/mm/yyyy` задает, что значением функции является дата, а строка `hh:mm` — время

События

Таблица 2.36. События

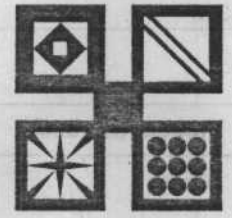
Событие	Происходит
`OnClick`	При щелчке кнопкой мыши
`OnDblClick`	При двойном щелчке кнопкой мыши
`OnMouseDown`	При нажатии кнопки мыши
`OnMouseUp`	При отпускании кнопки мыши
`OnMouseMove`	При перемещении мыши
`OnKeyPress`	При нажатии клавиши клавиатуры
`OnKeyDown`	При нажатии клавиши клавиатуры. События `OnKeyDown` и `OnKeyPress` — это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие `OnKeyUp`)
`OnKeyUp`	При отпускании нажатой клавиши клавиатуры
`OnCreate`	При создании объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий
`OnPaint`	При появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном и в других случаях. Событие сообщает о необходимости обновить (перерисовать) окно
`OnEnter`	При получении элементом управления фокуса
`OnExit`	При потере элементом управления фокуса

Исключения

Исключение — это ошибка, которая происходит во время работы программы.

Таблица 2.37. *Типичные исключения*

Тип исключения	Возникает
`EConvertError`	При выполнении преобразования, если преобразуемая величина не может быть приведена к требуемому виду. Наиболее часто возникает при преобразовании строки символов в число
`EDivByZero`	Целочисленное деление на ноль. При выполнении операции целочисленного деления, если делитель равен нулю
`EZeroDivide`	Деление на ноль. При выполнении операции деления над дробными операндами, если делитель равен нулю
`EFOpenError`	При обращении к файлу, например, при попытке загрузить файл иллюстрации при помощи метода `LoadFromFile`. Наиболее частой причиной является отсутствие требуемого файла или, в случае использования сменного диска, отсутствие диска в накопителе
`EInOutError`	При обращении к файлу, например, при попытке открыть для чтения (инструкция `reset`) несуществующий файл
`EDBEngineError`	При выполнении операций с базой данных, например, при попытке выполнить SQL-запрос к несуществующей таблице



Приложение

Описание CD-ROM

Прилагаемый к книге CD-ROM содержит проекты C++Builder, приведенные в книге в качестве примеров. Каждый проект (табл. П1) находится в отдельном каталоге. Помимо файлов проекта в каждом каталоге есть выполняемый файл, что позволяет запустить программу из Windows.

Большинство программ могут быть запущены непосредственно с CD-ROM (при условии, что на компьютере установлен Borland C++Builder Version 6). Некоторые программы, например, программы работы с базами данных, требуют дополнительной настройки системы (создания псевдонима или регистрации источника данных ODBC).

Для активной работы, чтобы иметь возможность вносить изменения в программы, скопируйте каталоги проектов на жесткий диск компьютера.

Таблица П1. *Содержимое CD-ROM*

Проект (каталог)	Описание
CD Player	Полнофункциональный проигрыватель CD-дисков. Контролирует наличие диска в дисководе и его тип. Демонстрирует использование компонента `MediaPlayer`
CDP	Полнофункциональный проигрыватель CD-дисков. Контролирует наличие диска в дисководе и его тип. Демонстрирует использование компонента `MediaPlayer`, а также отображение окна программы без границ и заголовка

Таблица П1 (продолжение)

Проект (каталог)	Описание
MEdit	Простой редактор текста. Демонстрирует использование компонентов `RichEdit`, `MainMenu`, `ToolBar`, `SpeedButton`, `OpenDialog`, `SaveDialog`, работу с меню, выполнение операций чтения и записи текста в файл
MIDI	Программа **Успеть за 60 секунд** демонстрирует использование компонента `MediaPlayer` для воспроизведения MIDI-файла. Мелодия воспроизводится "по кругу", до тех пор, пока пользователь не угадает число или не истечет время, отведенное на решение задачи
MP3 Player	MP3-плеер с регулятором громкости. Демонстрирует работу с компонентом MediaPlayer. Картинки для кнопки Play/Stop загружаются из ресурса
Spending	Программа **Расходы** обеспечивает работу с базой данных, которая представляет собой текстовый файл (tabl.grd). Для редактирования и просмотра данных используется компонент `StringGrid`. Внимание! Каталог проекта называется "Spending" а не "Расходы", так как при использовании в имени папки буквы "Ы" компилятор не может выполнить компиляцию
VideoPlayer	Программа **VideoPlayer** позволяет просмотреть видеоролик формата AVI или MPG
Анимация	Программа **Анимация** демонстрирует воспроизведение AVI-анимации при помощи компонента `Animate`. Анимация загружается из файла в начале работы программы
Бегущая строка	В этой программе баннер (бегущая строка) загружается из ресурса. Баннер "выплывает" из-за правой границы формы. В момент времени, когда баннер достигает центра окна, движение приостанавливается на несколько секунд, а затем возобновляется

Таблица П1 (продолжение)

Проект (каталог)	Описание
Будильник	Программа **Будильник**. Показывает, как поместить на System Tray значок программы, обеспечить вывод подсказки и контекстного меню значка
График	Программа **График** демонстрирует вывод графики (методы `LineTo`, `TextWidth`, `TextOutA`) на поверхность формы — выводит график изменения курса доллара
Доступ в Internet	Программа **Доступ в Internet** показывает, как запустить Internet Explorer или другой браузер для доступа к веб-странице
Ежедневник	Программа работы с базой данных "Ежедневник". Демонстрирует использование компонентов `ADOConnection`, `ADODataSet`, `DataSource`, `Table` и `DBNavigator`. База данных "Ежедневник" (Planer.mdb) должна быть зарегистрирована в системе как источник данных ODBC под именем dplanner
Записная книжка	Программа работы с базой данных "Записная книжка". Демонстрирует использование BDE-компонентов `Table` и `Qery`, а также компонентов `DBGrid` и `DataSource`. Для доступа к файлу таблицы (adrbk.db) программа использует псевдоним adrbk (Type: Standard, Default Driver: Paradox). Создать псевдоним можно при помощи BDEAdministrator
Звуки Windows	Программа **Звуки Windows** позволяет прослушать звуковые файлы, которые находятся в каталоге Windows\Media. Демонстрирует использование компонента `MediaPlayer`
КРДиаграмма	Программа **Круговая диаграмма** демонстрирует вывод графики (методы `Pie`, `Rectangle`, `TextOutA`) на поверхность формы — выводит круговую диаграмму
Календарь	Программа **Календарь** выводит изображение календаря на текущий месяц. Выходные и праздничные дни выделяются цветом, текущая дата — рамкой. Имеется возможность задать праздничные дни. Демонстрирует вывод графики на поверхность формы, работу с функциями манипулирования датами

Таблица П1 (продолжение)

Проект (каталог)	Описание
Калькулятор	Простейший калькулятор. Событие `Click` каждой кнопки обрабатывает отдельная функция
Калькулятор_2	Программа **Калькулятор-2** демонстрирует создание компонентов во время работы программы
Кафе	Программа **Кафе** демонстрирует использование компонента `CheckBox`
Конвертор	Программа **Конвертор** пересчитывает цену из долларов в рубли. Демонстрирует использование компонентов `TextBox` и `Label`. Программа спроектирована таким образом, что пользователь может ввести в поля редактирования только числа
Любимый напиток	Программа **Любимый напиток** демонстрирует использование компонента `ComboBox`
Магазин	Программа работы с базой данных "Магазин". Демонстрирует использование компонентов `TAble`, `DataSource`, `DBGrid`, `DBEdit`, `DBMemo`. Формат базы данных — Paradox. Для доступа к базе данных необходимо, при помощи BDE Administrator, создать псевдоним stock (Type: Standard; Default Driver: Paradox)
ОСАГО	Программа **ОСАГО** позволяет рассчитать размер страхового взноса по договору обязательного страхования гражданской ответственности. Демонстрирует использование компонента `ComboBox`, обработку одной функцией событий от нескольких компонентов
Очистка диска	Программа **Очистка диска** удаляет ненужные, созданные в процессе компиляции проектов C++Builder, файлы (obj, tds) и резервные копии (~bpr, ~dfm, ~h, ~cpp) из указанного пользователем каталога и всех его подкаталогов. Для выбора каталога используется стандартное окно **Обзор папок**
Парные картинки	Игра **Парные картинки**. Демонстрирует работу с графикой, отображение справочной информации. Картинки загружаются из файла pictures.bmp

Таблица П1 (продолжение)

Проект (каталог)	Описание
Печать	Программа **Счет** демонстрирует вывод на принтер и позволяет подготовить и распечатать счет
Пинг-понг	Программа **Пинг-понг** демонстрирует, как можно сделать графику интерактивной
Погода	Программа **Погода** (проект Meteo.bpr) демонстрирует операцию записи в файл — добавляет в файл meteo.txt информацию о температуре воздуха. Если файла данных в текущем каталоге нет, то программа создает его. Программа **Среднемесячная температура** (проект MeteoInfo.bpr) демонстрирует чтение данных из текстового файла (meteo.txt)
Полет в облаках	Мультипликация, элементы которой загружаются из bmp-файла. Очередной кадр формируется в памяти, а затем выводится на поверхность формы, что предотвращает мерцание изображения (стирает объект и рисует его на новом месте — как одна операция вывода)
Приветствие	Программа **Приветствие** демонстрирует вывод текста на поверхность формы. Вне зависимости от размера формы текст выводится в ее центре
Просмотр иллюстраций	Программа **Просмотр иллюстраций** позволяет просмотреть jpg-иллюстрации. Демонстрирует использование компонентов `ListBox`, `OpenDialog`, `Image`
Просмотр иллюстраций_2	Программа **Просмотр иллюстраций** позволяет просмотреть файлы формата JPEG, например, фотографии. Выбор папки выполняется в стандартном окне **Выбор папки**. Иллюстрации можно просматривать по кадрам или в режиме слайд-шоу
Сапер	Игра **Сапер**. Демонстрирует работу с графикой, вывод дочернего окна и отображение справочной информации
Секундомер	Программа **Секундомер** демонстрирует использование компонента `Timer`

Таблица П1 (продолжение)

Проект (каталог)	Описание
Сила тока	Программа **Сила тока** демонстрирует использование компонентов `TextBox` и `Label`, а также обработку исключения `EZeroDivide` (деление на ноль)
Собери картинку	Игра **Собери картинку** — игра "15" с графическим интерфейсом (вместо цифр — фрагменты картинки). Картинка загружается из bmp-файла, имя которого указано в командной строке запуска программы, или из файла, который находится в том же каталоге, что и файл программы
Сопротивление	Программа **Сопротивление** вычисляет сопротивление электрической цепи, состоящей из двух резисторов, которые могут быть соединены последовательно или параллельно. Демонстрирует использование компонента `RadioButton`
Справочная информация	Программа **Конвертор** демонстрирует различные способы отображения справочной информации. В подкаталоге hlp находятся файлы, необходимые для сосздания файла справки
Справочная информация_2	Программа демонстрирует различные методы отображения справочной информации. В подкаталоге\Chm находятся файлы, необходимые для создания файла справочной системы
Флаг	Программа **Олимпийский флаг** демонстрирует вывод графики на поверхность формы
Фоновый рисунок	Программа **Фоновый рисунок** демонстрирует, как можно получить фоновый рисунок путем многократного вывода битового образа на поверхность формы. Битовый образ загружается из файла Puzzle.bmp
Фунт	Программа **Фунты-килограммы** позволяет пересчитать вес из фунтов в килограммы. Кнопка **Пересчет** доступна только в том случае, если пользователь ввел исходные данные

Таблица П1 (окончание)

Проект (каталог)	Описание
Ходики	Часы с часовой, минутной и секундной стрелками. Показывают текущее время. Замечание. Если каталог проекта назвать "Часы", то возникает ошибка времени компиляции. Компилятору не нравится слово "Часы", точнее буква "ы"
Экзаменатор	Программа **Экзаменатор**. Вопросы считываются из txt-файла. Пример файла теста — см. peterburg.txt. Имя файла теста надо указать в командной строке запуска программы. Команду запуска надо набрать в окне **Запуск программы**, которое становится доступным в результате выбора команды **Пуск | Выполнить**
Экзаменатор_2	Универсальная программа тестирования **Экзаменатор**. Демонстрирует использование компонента XMLDocument (файл теста — XML-документ). Имя файла теста передается в программу через параметр командной строки. Для облегчения процесса запуска программы можно создать bat-файл (см. economics.bat)
Электроэнергия	Программа **Электроэнергия** показывает, как одна функция может обрабатывать события разных (но однотипных) компонентов

Предметный указатель

Ц

Ч

Э

не ищи информацию — подпишись!

что объединяет более 80 000 it-специалистов ведущих компаний? они подписаны на каталог softline®-direct!

Как регулярно получать свежую информацию из мира информационных технологий? Как добиться максимальной эффективности инвестиций в IT? Как выбрать правильную стратегию *в развитии IT-инфраструктуры, как выбрать правильное с технологической точки зрения* решение той или иной проблемы? Эти вопросы постоянно возникают как перед руководителями бизнеса, так и перед руководителями IT-подразделений. Где можно получить объективные ответы на все эти вопросы и узнать, как IT используются другими компаниями? Каталог SoftLine®-direct — источник такой информации.

Информация, которая необходима. Правильно построенная IT-инфраструктура компании — залог успешного и конкурентноспособного бизнеса. В каталоге представлена детальная информация о последних новинках в мире программного обеспечения, приведены обзорные статьи, примеры успешного внедрения и разнообразные типовые решения. Эта информация поможет Вам сделать правильный выбор.

Выберите каталог, который Вам нужен. В настоящее время выпускается несколько версий каталогов SoftLine®-direct и ряд спецвыпусков (Microsoft, Linux, Novell, Apple). Основной каталог SoftLine®-direct представлен в трех редакциях:

SoftLine®-direct Standard Edition
SoftLine®-direct Professional Edition
SoftLine®-direct Enterprise Edition

Подпишись на каталог сегодня!

Для бесплатной подписки на каталог SoftLine®-Direct позвоните по тел.: +7(095)232-00-23 или посетите сайт www.softline.ru

softline®

программное обеспечение — лицензирование, обучение, консалтинг

+7(095)232-00-23 www.softline.ru