Никита Культин

Delphi
в задачах и примерах

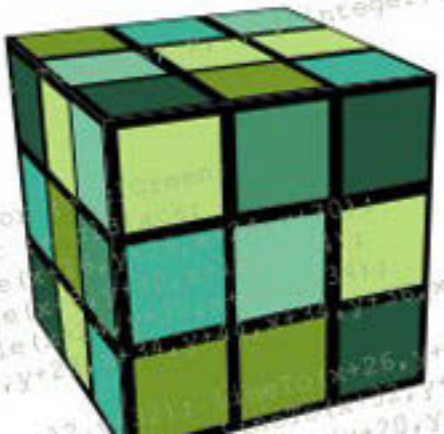
Готовые решения и тексты программ

Использование базовых компонентов

Работа с графикой, звуком и базами данных

Программирование игр

Справочник по компонентам и функциям



+ CD-ROM

Никита Культин

Delphi
в задачах и примерах

Санкт-Петербург
«БХВ-Петербург»
2003

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 Delphi в задачах и примерах. — СПб.: БХВ-Петербург, 2003. — 288 с.: ил.

ISBN 5-94157-353-7

Книга представляет собой сборник программ и задач для самостоятельного решения в среде разработки Delphi. Примеры различной степени сложности — от простейших до приложений работы с графикой, звуком и базами данных — демонстрируют возможности среды разработки Delphi, назначение основных компонентов. Книга также содержит краткий справочник наиболее часто используемых компонентов и функций. На прилагаемом компакт-диске находятся тексты программ.

Для начинающих программистов

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Анатолий Адаменко*
Зав. редакцией	*Григорий Добин*
Редактор	*Анатолий Хрипов*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 20.06.03.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 18.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-353-7

Содержание

Предисловие

В последнее время резко возрос интерес к программированию. Это связано с развитием и внедрением в повседневную жизнь информационно-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 идеологии систем быстрой разработки (RAD-систем, Rapid Application Development — среда быстрой разработки приложений) лежат технологии визуального проектирования и событийного объектно-ориентированного программирования, суть которых 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!

Среди RAD-систем особо выделяется среда Borland Delphi, которая позволяет создавать различные программы: от простейших однооконных приложений до программ управления распределенными базами данных. В качестве языка программирования в среде Borland Delphi используется язык Delphi (Delphi language), являющийся прямым потомком хорошо известного всем программистам языка Pascal.

Чтобы научиться программировать, надо программировать — писать программы, решать конкретные задачи. Для этого необходимо изучить язык программирования и среду разработки. Освоить язык программирования Delphi не очень сложно. Труднее

изучить среду программирования, точнее научиться использовать компоненты. И здесь хорошим подспорьем могут быть программы, которые демонстрируют назначение компонентов и особенности их применения.

В книге, которую вы держите в руках, собраны разнообразные примеры, которые не только демонстрируют возможности среды разработки Delphi, но и знакомят с принципами работы с графикой, звуком, базами данных. Следует обратить внимание, что большинство примеров не являются учебными в чистом смысле, это — вполне работоспособные программы.

Книга состоит из двух частей и приложений.

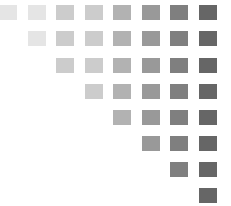
Первая часть содержит примеры и задачи для самостоятельного решения. Примеры представлены в виде краткого описания, сформулированного в форме задания для самостоятельного решения, диалоговых окон и хорошо документированных текстов программ. Для простых задач рассмотрены только функции обработки событий. Текст остальных программ приведен полностью.

Вторая часть книги — это краткий справочник по языку программирования Delphi. В нем можно найти описание свойств компонентов, использованных в приведенных примерах.

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

Если что-то не понятно, обратитесь к справочнику *(часть 2)*, к справочной системе Delphi или к литературе, например: **Культин Н. Б. Основы программирования в Delphi 7. — СПб.: БХВ-Петербург, 2003**. В ней, помимо описания языка программирования и среды разработки Delphi, компонентов, процессов создания и отладки программ, вы найдете ответы на многие вопросы, в том числе: как при помощи Microsoft Help Workshop сформировать файл справки или, используя Install-Shield Express, создать установочный CD-ROM.

Часть 1



Примеры и задачи

Базовые компоненты

В этом разделе приведены простые примеры и задачи, основное назначение которых — научить работать с базовыми компонентами.

Общие замечания

Приступая к решению задач этого раздела, необходимо вспомнить:

- Процесс создания программы в Delphi состоит из двух шагов: сначала нужно создать форму программы (диалоговое окно), затем — написать процедуры обработки *событий*. Форма *приложения* (так принято называть прикладные программы, работающие в Windows) создается путем добавления в форму *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 ввода/редактирования текста;

  Button — командную кнопку;

  CheckBox — независимую кнопку выбора;

  RadioButton — зависимую кнопку выбора;

  ListBox — список выбора;

  ComboBox — комбинированный список выбора.

- Вид компонента, его размер и поведение определяются значениями *свойств* (характеристик) компонента (описание свойств базовых компонентов можно найти в справочнике, во второй части книги).
- Основную работу в программе выполняют процедуры обработки *событий* (описание основных событий можно найти в справочнике, во второй части книги).
- Исходную информацию программа может получить из полей ввода/редактирования (компонент `Edit`), списка выбора (компонент `ListBox`) или комбинированного списка (компонент `ComboBox`). Для ввода значений логического типа можно использовать компоненты `CheckBox` и `RadioButton`.
- Результат программа может вывести в поле вывода текста (компонент `Label`) или в окно сообщения (функция `MessageDlg`).
- Для преобразования текста, например находящегося в поле ввода/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`.

1. Написать программу, которая пересчитывает скорость ветра из "метров в секунду" в "километров в час". Рекомендуемый вид формы приведен на рис. 1.1.

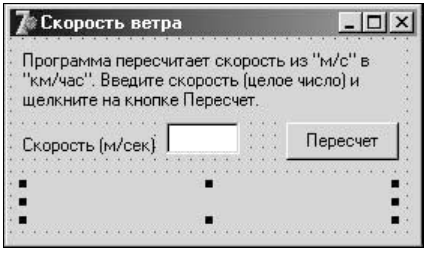


Рис. 1.1. Форма программы **Скорость ветра**

```
// щелчок на кнопке Пересчет
procedure TForm1.Button1Click(Sender: TObject);
var
    ms:  integer; // скорость м/с
    kmh: real;    // скорость км/час
begin
    ms  := StrToInt(Edit1.Text); // ввести исходные данные
    kmh := ms * 3.6;             // пересчитать
    // вывести результат
    Label3.Caption :=
       IntToStr(ms) + ' м/с — это ' +
       FloatToStr(kmh) + ' км/час'
end;
```

2. Написать программу, которая пересчитывает скорость ветра из "метров в секунду" в "километров в час". Рекомендуемый вид формы приведен на рис. 1.1. Программа должна быть спроектирована таким образом, чтобы пользователь мог ввести в поле **Скорость** только целое положительное число.

```
// щелчок на кнопке Пересчет
procedure TForm1.Button1Click(Sender: TObject);
var
    ms:  integer; // скорость м/с
    kmh: real;    // скорость км/час
begin
    // Если поле Edit1 пустое, то при выполнении функции
    // StrToInt возникает ошибка.
    // Проверим, ввел ли пользователь скорость в поле Edit1

    if Length(Edit1.Text) = 0 then
    begin
        ShowMessage('Надо ввести скорость');
        exit; // завершить обработку события
    end;

    ms  := StrToInt(Edit1.Text); // ввести исходные данные
    kmh := ms * 3.6;             // пересчитать
    // вывести результат
    Label3.Caption :=
       IntToStr(ms) + ' м/с — это ' + FloatToStr(kmh) + ' км/час'
end;
```

```
// нажатие клавиши в поле Edit1
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    // В поле Скорость (Edit1) можно ввести только
    // цифры. Процедура проверяет, является ли символ
    // допустимым. Если нет, то она заменяет
    // введенный символ нулевым. В результате символ
    // в поле редактирования не отображается.

    // Key — символ, соответствующий нажатой клавише
    if not ((Key >= '0') and (Key <='9') or (Key = #8))
        then Key := Chr(0);

end;
```

3. Написать программу, которая пересчитывает скорость ветра из "метров в секунду" в "километров в час". Рекомендуемый вид формы приведен на рис. 1.1. Программа должна быть спроектирована таким образом, чтобы пользователь мог ввести в поле **Скорость** только целое положительное число. Вычисление должно выполняться как в результате щелчка на кнопке **Пересчет**, так и при нажатии клавиши <Enter> после ввода последней цифры в поле **Скорость**.

```
// пересчитывает скорость из м/сек в км/час
procedure WindSpeed;
var
    ms:  integer; // скорость м/с
    kmh: real;    // скорость км/час
begin
    // Если поле Edit1 пустое, то при выполнении
    // функции StrToInt возникает ошибка.
    // Проверим, ввел ли пользователь скорость в поле Edit1

    if Length(Form1.Edit1.Text) = 0 then
    begin
        ShowMessage('Надо ввести скорость');
        exit; // завершить обработку события
    end;

    ms  := StrToInt(Form1.Edit1.Text); // ввести исходные данные
    kmh := ms * 3.6;                   // пересчитать
```

```
    // вывести результат
    Form1.Label3.Caption :=
       IntToStr(ms) + ' м/с — это ' + FloatToStr(kmh) + ' км/час'
end;

// щелчок на кнопке Пересчет
procedure TForm1.Button1Click(Sender: TObject);
begin
    WindSpeed; // пересчитать скорость
end;

// нажатие клавиши в поле Edit1
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    // Key — символ нажатой клавиши
    // #8  — клавиша <Backspace>
    // #13 — клавиша <Enter>

    case Key of
        '0'..'9',#8:;           // цифры и клавиша <Backspace>
        #13:         WindSpeed; // пересчитать скорость
         else  Key := Chr(0);   // остальные символы не отображать
    end;

end;
```

4. Написать программу, которая пересчитывает массу из фунтов в килограммы (1 фунт = 409,5 грамм). Рекомендуемый вид формы приведен на рис. 1.2. Программа должна быть спроектирована таким образом, чтобы кнопка **Пересчет** была доступна только в том случае, если пользователь ввел исходные данные.

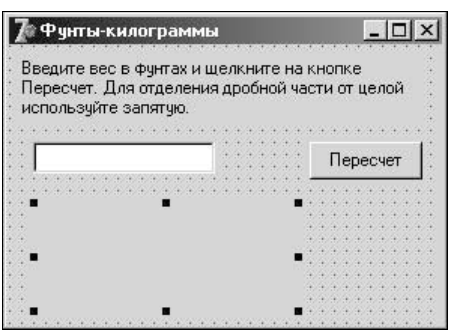


Рис. 1.2. Форма программы **Фунты-килограммы**

```
// щелчок на кнопке Пересчет
procedure TForm1.Button1Click(Sender: TObject);
var
   funt: real; // масса  в фунтах
   kg: real;   // масса  в килограммах
begin
   // 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 * 0.4095;
   Label2.Caption := FloatToStrF(funt,ffGeneral,5,2) +
                     ' ф — это ' +
                     FloatToStrF(kg,ffGeneral,5,2) + ' кг';

end;

// нажатие клавиши в поле Edit1
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    case Key of
        '0'..'9', #8:;  // цифры и клавиша <Backspace>

{ Обработку десятичного разделителя
  сделаем "интеллектуальной". Заменим точку и
  запятую на символ DecimalSeparator — символ,
  который используется при записи дробных чисел.
}
        '.',',':
             begin
                Key := DecimalSeparator;
                // Проверим, введен ли уже в поле
                // Edit десятичный разделитель
                if pos(DecimalSeparator,Edit1.Text) <> 0
                   then Key := Char(0);
             end;
         else Key := Char(0); // остальные символы запрещены
     end;
end;

// содержимое поля Edit1 изменилось
procedure TForm1.Edit1Change(Sender: TObject);
```

```
begin
        // проверим, есть ли в поле Edit1 исходные данные
    if Length(Edit1.Text) = 0
       then Button1.Enabled := False // кнопка Пересчет недоступна
       else Button1.Enabled := True; // кнопка Пересчет доступна
end;

// Событие onCreate происходит в момент создания формы,
// до того, как форма появится на экране
procedure TForm1.FormCreate(Sender: TObject);
begin
    { т. к. поле Edit1 пустое (пользователь
    еще не ввел исходные данные), то
    сделаем кнопку Пересчет недоступной }
    Button1.Enabled := False;
end;
```

5. Написать программу, которая пересчитывает массу из фунтов в килограммы (1 фунт = 409,5 грамм). Рекомендуемый вид формы приведен на рис. 1.2. Программа должна быть спроектирована таким образом, чтобы пользователь мог ввести в поле **Масса** только положительное число (целое или дробное).

```
// щелчок на кнопке Пересчет
procedure TForm1.Button1Click(Sender: TObject);
var
   funt: real; // масса  в фунтах
   kg: real;   // масса  в килограммах
begin
   if Length(Edit1.Text) = 0 then
   begin
        // в поле Edit1 нет исходной информации
        ShowMessage('Надо ввести массу.');
        exit;
   end;

   funt := StrToFloat(Edit1.Text);
   kg := funt * 0.4095;
   Label2.Caption := FloatToStrF(funt,ffGeneral,5,2) + ' ф — это ' +
                     FloatToStrF(kg,ffGeneral,5,2) + ' кг';

end;
```

```
// нажатие клавиши в поле Edit
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    case Key of
        '0'..'9', #8:;  // цифры и клавиша <Backspace>

{ Обработку десятичного разделителя
  сделаем "интеллектуальной". Заменим точку и
  запятую на символ DecimalSeparator — символ,
  который используется при записи дробных чисел.
  Этот символ задается в настройках Windows.
}
        '.',',':
            begin
               Key := DecimalSeparator;
               // проверим, введен ли уже в поле
               // Edit десятичный разделитель
               if pos(DecimalSeparator,Edit1.Text) <> 0
                  then Key := Char(0);
            end;
         else Key := Char(0); // остальные символы запрещены
    end;
end;
```

6. Написать программу, которая вычисляет скорость (км/час), с которой бегун пробежал дистанцию. Рекомендуемый вид формы приведен на рис. 1.3. Количество минут задается целым числом, секунд — дробным.

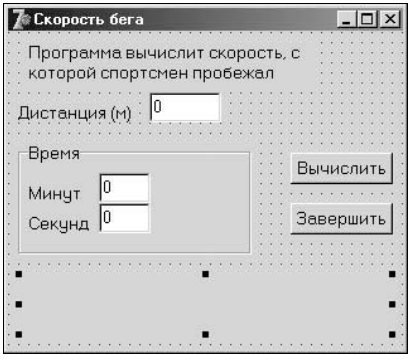


Рис. 1.3. Форма программы **Скорость бега**

```
// нажатие клавиши в поле Дистанция
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  // Key — символ, соответствующий нажатой клавише.
  // Если символ недопустимый, то процедура заменяет его
  // на символ с кодом 0. В результате этого символ в поле
  // редактирования не появляется и у пользователя создается
  // впечатление, что программа не реагирует на нажатие
  // некоторых клавиш.
  case Key of
    '0'..'9':              ; // цифры
    #8      :              ; // клавиша <Backspace>
    #13     : Edit2.SetFocus; // при нажатии <Enter>  курсор
                              // переводится в поле Время:минут

    // остальные символы запрещены
    else Key :=Chr(0);  // символ не отображать
  end;
end;


// нажатие клавиши в поле Время:минут
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
  case Key of
    '0'..'9':              ;
    #8      :              ;  // клавиша <Backspace>
    #13     : Edit3.SetFocus; // при нажатии <Enter> курсор
                              // переводится в поле Время:секунд

    // остальные символы запрещены
    else Key :=Chr(0);  // символ не отображать
  end;
end;

// нажатие клавиши в поле Время:секунд
procedure TForm1.Edit3KeyPress(Sender: TObject; var Key: Char);
begin
   case Key of
    '0'..'9':;
    ',','.' : // десятичный разделитель
             begin
                Key := DecimalSeparator;
```

```
            if Pos(DecimalSeparator,Edit3.Text) <> 0
               then Key := Char(0);
         end;
    #8  :                ;  // клавиша <Backspace>
    #13 : Button1.SetFocus; // при нажатии клавиши <Enter>
                            // активируется кнопка Вычислить

    // остальные символы — запрещены
    else Key :=Chr(0);  // символ не отображать
  end;
end;

// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
var
  dist : integer; // дистанция, метров
  min : integer;  // время, минуты
  sek : real;     // время, секунды

  v: real;        // скорость
begin
  // получить исходные данные из полей ввода
  dist := StrToInt(Edit1.Text);
  min  := StrToInt(Edit2.Text);
  sek  := StrToFloat(Edit3.Text);

  // дистанция и время не должны быть равны нулю
  if (dist = 0) or ((min = 0) and (sek = 0)) then
  begin
      ShowMessage('Надо задать дистанцию и время.');
      exit;
  end;

  // вычисление
  v := (dist/1000) / ((min*60 + sek)/3600);

  // вывод результата
  label5.Caption := 'Дистанция: '+ Edit1.Text + ' м' + #13 +
                    'Время: ' + IntToStr(min) + ' мин ' +
                    FloatToStrF(sek, ffGeneral,4,2) + ' сек ' +
                    #13 + 'Скорость: ' +
                    FloatToStrF(v,ffFixed,4,2) + ' км/час';

end;
```

```
// щелчок на кнопке Завершить
procedure TForm1.Button2Click(Sender: TObject);
begin
      Form1.Close; // закрыть главную форму — завершить работу
                   // программы
end;
```

7. Написать программу, которая вычисляет силу тока в электрической цепи. Рекомендуемый вид формы приведен на рис. 1.4. Программа должна быть спроектирована таким образом, чтобы кнопка **Вычислить** была доступна только в том случае, если пользователь ввел величину сопротивления.

Рис. 1.4. Форма программы **Сила тока**

8. Написать программу, которая вычисляет силу тока в электрической цепи. Цепь состоит из двух параллельно соединенных сопротивлений. Рекомендуемый вид формы приведен на рис. 1.5.

9. Написать программу, которая вычисляет стоимость покупки с учетом скидки. Скидка 1 % предоставляется, если сумма покупки больше 300 рублей, 2 % — если сумма больше 500 рублей, 3 % — если сумма больше 1 000. Информация о предоставленной скидке (процент и величина) должна быть выведена в диалоговом окне. Рекомендуемый вид формы программы приведен на рис. 1.6.

10. Напишите программу, которая вычисляет стоимость покупки. Пользователь должен вводить код товара и количество единиц. Программа должна формировать список товаров. Рекомен-

дуемый вид формы приведен на рис. 1.7. (Развитие. Список кодов и наименований и цен товаров вводить из файла.)

Рис. 1.5. Форма программы **Сила тока**

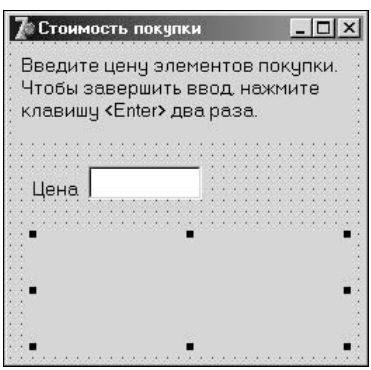


Рис. 1.6. Форма программы **Стоимость покупки**

Рис. 1.7. Форма программы **Супер маркет**

11. Напишите программу, которая вычисляет доход по вкладу. Программа должна обеспечивать расчет простых и сложных процентов. Простые проценты начисляются в конце срока вклада, сложные — ежемесячно и прибавляются к первоначальной

сумме вклада и в следующем месяце проценты начисляются на новую сумму. Рекомендуемый вид формы программы приведен на рис. 1.8.

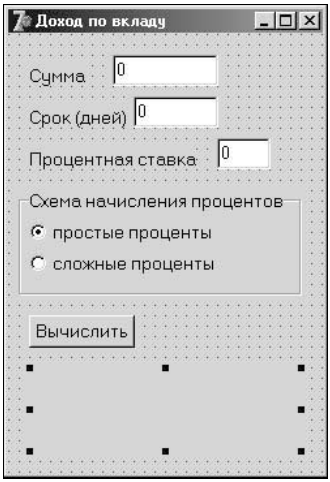


Рис. 1.8. Форма программы **Доход по вкладу**

```
// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
var
    sum : real;     // сумма вклада
    pr:   real;     // процентная ставка
    srok: integer; // срок вклада
    dohod: real;    // доход по вкладу

    buf: real;
    i: integer;
begin
    // получить исходные данные
    sum := StrToFloat(Edit1.Text);
    pr := StrToFloat(Edit2.Text);
    srok := StrToInt(Edit3.Text);

    if RadioButton1.Checked then
        // выбран переключатель Простые проценты
        dohod := sum * (pr/100) * (srok/360)
```

```
    else
        // т. к. в группе два переключателя, то если
        // не выбран RadioButton1, то выбран
        // RadioButton2 — Сложные проценты
        begin
            buf:= sum;
            for i:=1 to srok do
                buf:= buf + buf * (pr/100);
            // здесь buf — сумма в конце срока вклада
            dohod := buf — sum;
        end;

    sum := sum + dohod;
    Label4.Caption := 'Доход: ' + Float-
ToStrF(dohod,ffGeneral,9,2) + #13 +
              'Сумма в конце срока вклада: ' +
              FloatToStrF(sum,ffGeneral,9,2);
end;

// выбор переключателя Простые проценты
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
    Label2.Caption := 'Срок (дней)';
    Label4.Caption := '';

end;

// выбор переключателя Сложные проценты
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
    Label2.Caption := 'Срок (мес.)';
    Label4.Caption := '';
end;
```

12. Написать программу, которая вычисляет сопротивление электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Рекомендуемый вид формы приведен на рис. 1.9. Если величина сопротивления цепи превышает 1 000 Ом, то результат должен быть выведен в килоомах.

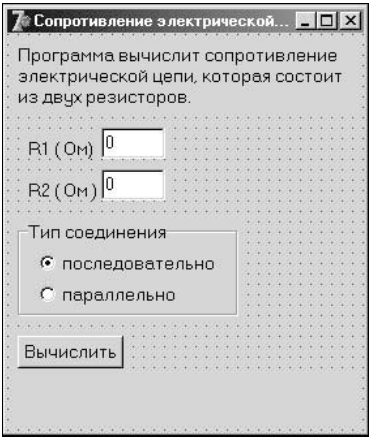


Рис. 1.9. Форма программы **Сопротивление электрической цепи**

```
// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
var
    r1,r2: real; // величины сопротивлений
    r: real;     // сопротивление цепи
begin
    // получить исходные данные
    r1 := StrToFloat(Edit1.Text);
    r2 := StrToFloat(Edit2.Text);

    if (r1 = 0) and (r2 = 0) then
    begin
        ShowMessage('Надо задать величину хотя бы одного
                     сопротивления');
        exit;
    end;

    // переключатели RadioButton1 и RadioButton2
    // зависимые, поэтому о типе соединения можно
    // судить по состоянию одного из этих
    // переключателей
    if RadioButton1.Checked
        then // выбран переключатель Последовательно
            r:= r1+r2
        else //  выбран переключатель Параллельно
            r:= (r1*r2)/(r1+r2);
```

```
    Label4.Caption := 'Сопротивление цепи: ';
    if r < 1000 then
       Label4.Caption := Label4.Caption +
                  FloatToStrF(r,ffGeneral,3,2) + ' Ом'
    else
        begin
            r:=r/1000;
            Label4.Caption := Form1.Label4.Caption +
                  FloatToStrF(r,ffGeneral,3,2) + ' кОм';
        end
end;

// щелчок на переключателе Последовательно
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
   // пользователь изменил тип соединения
   Label4.Caption := '';
end;

// щелчок на переключателе Параллельно
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
   // пользователь изменил тип соединения
   Label4.Caption := '';
end;
```

13. Написать программу, которая вычисляет силу тока в электрической цепи. Цепь состоит из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Рекомендуемый вид формы приведен на рис. 1.10.

14. Написать программу, которая, используя закон Ома, вычисляет силу тока, напряжение или сопротивление электрической цепи. Рекомендуемый вид формы приведен на рис. 1.11. Во время работы программы, в результате выбора переключателя **Ток**, **Напряжение** или **Сопротивление**, текст, поясняющий назначение полей ввода, должен меняться.

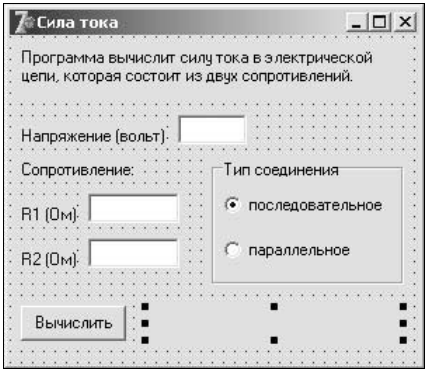


Рис. 1.10. Форма программы **Сила тока**

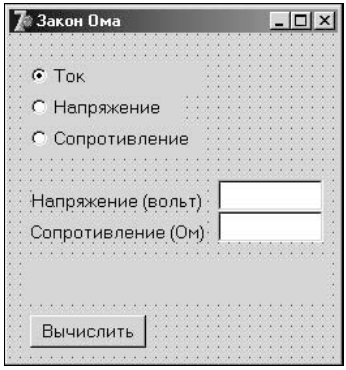


Рис. 1.11. Форма программы **Закон Ома**

```
// вычисление тока, напряжения или сопротивления
procedure Calculate;
var
  I,U,R:  real; // ток, напряжение, сопротивление
begin
  if Form1.RadioButton1.Checked then
  // ток
  begin
    U := StrToFloat(Form1.Edit1.Text);
    R := StrToFloat(Form1.Edit2.Text);
    if (R <> 0) then
    begin
      I := U/R;
      Form1.Label3.Caption := 'Ток: ' + Float-
ToStrF(I,ffFixed,4,2) + ' A';
    end
    else ShowMessage('Сопротивление не должно быть равно нулю.');
    exit;
  end;

  if Form1.RadioButton2.Checked then
  // напряжение
  begin
    I := StrToFloat(Form1.Edit1.Text);
    R := StrToFloat(Form1.Edit2.Text);
    U := I*R;
```

```
    Form1.Label3.Caption := 'Напряжение: ' +
                            FloatToStrF(U,ffFixed,4,2) + ' В';
    exit;
  end;

  if Form1.RadioButton3.Checked then
  // сопротивление
  begin
    U := StrToFloat(Form1.Edit1.Text);
    I := StrToFloat(Form1.Edit2.Text);
    if (I <> 0) then
    begin
      R := U/I;
      Form1.Label3.Caption := 'Сопротивление: ' + Float-
ToStrF(R,ffFixed,4,2) + ' Ом';
    end
    else ShowMessage('Ток не должен быть равен нулю.');
  end;

end;

// Выбор переключателя Ток
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
  Label1.Caption := 'Напряжение (вольт)';
  Label2.Caption := 'Сопротивление (Ом)';
  Label3.Caption := '';
end;

// Выбор переключателя Напряжение
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
  Label1.Caption := 'Ток (ампер)';
  Label2.Caption := 'Сопротивление (Ом)';
  Label3.Caption := '';
end;

// Выбор переключателя Сопротивление
procedure TForm1.RadioButton3Click(Sender: TObject);
begin
  Label1.Caption := 'Напряжение (вольт)';
  Label2.Caption := 'Ток (ампер)';
  Label3.Caption := '';
end;
```

```
// Нажатие клавиши в поле Edit1
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key:Char);
begin
  case Key of
    '0'..'9',#8: ; // цифры и клавиша <Backspase>
    #13: Edit2.SetFocus; // клавиша <Enter>
    '.',',':
      begin
        if Key = '.'
          then Key := ',';
        // не позволяет вводить знак запятой повторно
        if Pos(',',Edit1.Text) <> 0
          then Key := Chr(0);
      end;
    else Key := Chr(0);
  end;
end;

// Нажатие клавиши в поле Edit2
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key:Char);
begin
  case Key of
    '0'..'9',#8: ;
    #13: Calculate; // клавиша <Enter> — вычислить
    '.',',':
      begin
        if Key = '.'
          then Key := ',';
        // не позволяет вводить знак запятой повторно
        if Pos(',',Edit2.Text) <> 0
          then Key := Chr(0);
      end;
    else Key := Chr(0);
  end;
end;

// Щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
begin
  if (Edit1.Text <> '') and (Edit2.Text <> '')
    then Calculate // вычислить ток, напряжение или сопротивление
  else ShowMessage('Надо ввести исходные данные в оба поля');
end;
```

15. Написать программу, которая вычисляет стоимость поездки на автомобиле, например, на дачу. Рекомендуемый вид формы приведен на рис. 1.12.

Рис. 1.12. Форма программы **Поездка на дачу**

```
implementation

{$R *.dfm}

{ Процедура EditKeyPress обрабатывает нажатие клавиш
  в поле Расстояние, Цена и Потребление.
  Сначала надо обычным образом создать процедуру
  обработки события OnKeyPress для поля Edit1,
  затем назначить эту процедуру событию OnKeyPres
  полей Edit2 и Edit3. Кроме того, свойству Tag
  компонентов Edit1, Edit2 и Edit3 надо присвоить
  соответственно значения 1, 2 и 3. Свойство Tag
  используется в процедуре EditKeyPress
  для идентификации компонента. }

procedure TForm1.EditKeyPress(Sender: TObject; var Key: Char);
var
    Edit: TEdit;
begin
    Edit := Sender as TEdit;
    // в поле Edit можно ввести только дробное число
    case Key of
    '0'..'9':; // цифры
    #8:      ; // клавиша <Backspace>
```

```
    '.',',':  begin
                Key := DecimalSeparator;
                if Pos(DecimalSeparator,Edit.Text) <> 0
                    then Key := #0;
              end;
    #13: // клавиша <Enter>
        case Edit.Tag of
        1:  // клавиша нажата в поле Edit1
            Edit2.SetFocus; // фокус в поле Edit2
        2:  // клавиша нажата в поле Edit1
            Edit3.SetFocus; // фокус в поле Edit3
        3:  // клавиша нажата в поле Edit3
            Button1.SetFocus; // фокус на кнопку Button1
        end;
    end;
end;

// щелчок на кнопке Вычислить
procedure TForm1.Button1Click(Sender: TObject);
var
  rast : real; // расстояние
  cena : real; // цена
  potr : real; // потребление на 100 км
  summ : real; // сумма
  mes: string;
begin
  // здесь возможно исключение типа EConvertError
  // в случае, если пользователь оставит
  // одно из полей ввода незаполненным
  try
        rast := StrToFloat(Edit1.Text);
        cena := StrToFloat(Edit2.Text);
        potr := StrToFloat(Edit3.Text);
    except
    on EConvertError do
    begin
        ShowMessage('Даные надо ввести во все поля!');
        // попытаемся найти пустое поле
        if Length(Edit1.Text) = 0
            then Edit1.SetFocus
        else if Length(Edit2.Text) = 0
                then Edit2.SetFocus
```

```
                else Edit3.SetFocus;
          exit;
      end;
    end;

    summ := (rast / 100) * potr * cena;
    mes := 'Поездка на дачу';

    if  CheckBox1.Checked  then
    begin
        summ := summ * 2;
        mes := mes + ' и обратно';
    end;

    mes := mes + 'обойдется в ' + FloatToStrF(summ,ffGeneral,4,2)
               + ' руб.';
    Label4.Caption := mes;
end;

end.
```

16. Напишите программу-калькулятор, выполняющий сложение и вычитание. Рекомендуемый вид формы приведен на рис. 1.13.

К этой задаче даны два варианта решения. В первом варианте для каждой цифровой кнопки создана отдельная процедура обработки события `OnClick`. Во втором варианте событие `OnClick` всех цифровых кнопок обрабатывает одна процедура, что позволило значительно сократить текст программы.

Рис. 1.13. Форма программы **Калькулятор**