

М.Д. Дибиров

Программируем на DELPHI

Методическое пособие для учителей



2010

Дибиров М.Д.

Программируем на Delphi. Методическое пособие для учителей / М.Д. Дибиров. – 2010. – 39 с.: ил.

В брошюре приведены практические рекомендации по разработке приложений с использованием среды программирования Delphi. Рассмотренные примеры проектов затрагивают основы использования основных объектов среды Delphi, задачи рассматривают такие конструкции, как линейные, ветвления, циклические, массивы. Пособие ориентировано на учителей информатики и может быть использовано на занятиях по программированию для начинающих.

ОГЛАВЛЕНИЕ

Введение	4
Линейные алгоритмы.....	5
Задача 1.....	5
Задача 2.....	6
Условный оператор.....	9
Задача 3.....	9
Задача 4.....	11
Операторы цикла	15
Задача 5.....	15
Задача 6.....	18
Задача 7.....	22
Массивы	25
Задача 8.....	25
Литература.....	38

ВВЕДЕНИЕ

Программирование – наиболее традиционная сфера деятельности при организации профильно-ориентированных курсов информатики. Если еще недавно вершина такого курса – изучение одного из языков программирования (Паскаль, Бейсик), то сегодня набирает оборот разработка несложных учебных приложений с использованием языков объектного программирования – C++, Delphi, Visual Basic.

Ознакомление с основами объектно-ориентированного программирования (ООП) представляется вполне возможным и полезным. В ходе изучения данного курса решаются такие задачи, как:

- Освоение методологии ООП;
- Овладение техникой ООП на одном из языков;
- Введение в проблематику, расширение общего кругозора (общеобразовательный компонент).

Выбор языка программирования играет немаловажную роль. Delphi – среда визуального программирования, созданная на базе Паскаля. По мнению многих специалистов [7], это один из самых мощных инструментов разработки программных продуктов любой сложности и направленности. Для начинающего использовать Delphi, желательно иметь первоначальные сведения по программированию на языке Паскаль.

Изучение языка Паскаль – последовательное освоение алфавита, конструкций, синтаксиса и т.д. Изучая среду Delphi, приходится отойти от классической схемы освоения языка и исходить от того, какие элементы непосредственно языка необходимы в данный момент. В процессе разработки приложения на Delphi больше внимания приходится уделить внешней стороне: используемые компоненты, дополнительные инструменты, размеры объектов, их расположение на форме, состояния свойств объектов и.д.

В данном пособии представлены 8 несложных с практической стороны задач, однако, на примере их решений приводятся описание и приемы использования визуальных компонентов среды Delphi. В качестве поддержки пособия имеются электронные материалы к задачам, которые могут быть скачаны на сайте автора по адресу <http://dibirov.narod.ru/>

ЛИНЕЙНЫЕ АЛГОРИТМЫ

Задача 1. [4, 1.43, с. 15] Даны два числа. Найти сумму, разность, произведение, а также частное от деления первого числа на второе.

Решение:

На примере реализации алгоритма решения этой простейшей задачи познакомимся со средой Delphi и некоторыми основными объектами.

Создадим новый проект (Файл►Новый►Приложение). Определим некоторые свойства объекта **Форма**: Caption='Задача 1', Width=500, Height=300. Разместим на форме метки (label1 – содержит текст условия задачи, label2, label3, label4 – текст «Число 1», «Число 2», «Ответ»), текстовые поля (edit1, edit2 – для ввода данных, edit3 – для вывода результата выполнения арифметической операции), кнопки (button1, button2, button3, button4 – предназначены для выполнения арифметических операций, button5 – для очистки текстовых полей ввода и вывода данных edit, button6 – выход из программы) (рис. 1).

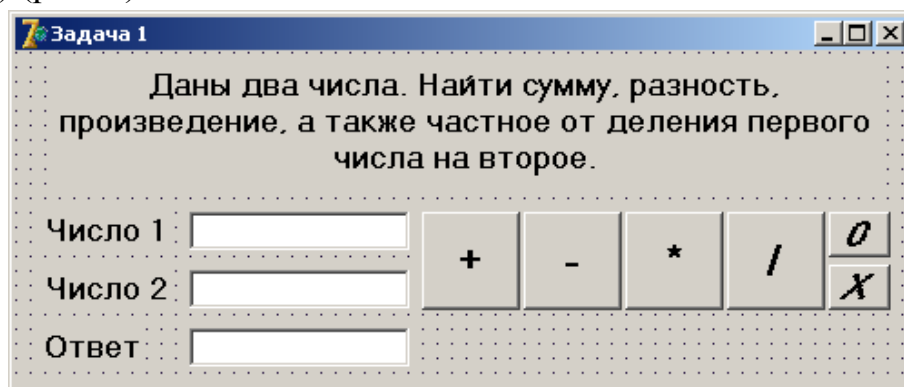


Рисунок 1

Начнем с самого простого – выход из программы. Двойным щелчком мыши по объекту button6 попадаем в редактор кода, где сможем отредактировать код подпрограммы события OnClick кнопки:

```
procedure TForm1.Button6Click(Sender: TObject);  
begin  
    form1.Close  
end;
```

Кнопка button5 предназначена для очистки данных в текстовых полях edit1, edit2 и edit3. За содержимое этих объектов отвечает свойство text, которое и должно измениться (точнее стать пустым):

```
procedure TForm1.Button5Click(Sender: TObject);  
begin  
    edit1.Text:=""; edit2.Text:=""; edit3.Text:="";  
end;
```

Для кнопок `button1`, `button2`, `button3`, `button4` содержимое кода будет достаточно схожим. Возможно, может показаться, что дальнейший код достаточно примитивен. Однако отметим также, что свойство `text` объекта `edit` является представителем строкового типа `string`. Арифметическая операция может быть выполнена только над представителем числового типа. Очевидна необходимость в использовании подпрограмм преобразования типов данных.

Таблица 1

Функция	Описание
<code>IntToStr</code>	Преобразует целочисленный аргумент в строковое выражение
<code>StrToInt</code>	Преобразует строковый аргумент в целочисленное выражение
<code>FloatToStr</code>	Преобразует вещественный аргумент в строковое выражение
<code>StrToFloat</code>	Преобразует строковый аргумент в вещественное выражение

Для наглядности и простоты восприятия кода введем дополнительные локальные переменные. Для кнопки `button1` (операция сложения) код подпрограммы события `OnClick` будет следующий:

```
procedure TForm1.Button1Click(Sender: TObject);
var
    a, b, c : extended;
begin
    a:= strtofloat(edit1.Text);
    b:= strtofloat(edit2.Text);
    c:=a+b;
    edit3.Text := floattostr(c);
end;
```

Приведя аналогичные коды подпрограмм для обработки события `OnClick` для объектов `button2`, `button3`, `button4`, можно считать задачу решенной. Приведенное решение пока имеет достаточно недочетов: не установлено ограничение на допустимый набор символов для ввода в поля `edit1` и `edit2`; нет контроля нулевого значения второго числа при выполнении операции деления; нет предупреждения перед выполнением арифметических действий при пустых текстовых полях; нет запрета ввода данных в поле `edit3`, хотя оно предназначено только для вывода данных. Эти и многие другие возможные недочеты программы могут быть исправлены при использовании дополнительных языковых средств, которые будут рассмотрены далее.

Задача 2. [9, 41, с. 55] Составить программу для вычисления пути, пройденного лодкой, если ее скорость в стоячей воде v [км/ч], скорость течения

реки v_1 [км/ч], время движения по озеру t_1 [ч], а против течения t_2 [ч].

Решение:

Начнем с подготовки формы и размещения на ней необходимых для решения задачи компонентов (рис. 2).

Задача 2

Составить программу для вычисления пути, пройденного лодкой, если ее скорость в стоячей воде v [км/ч], скорость течения реки v_1 [км/ч], время движения по озеру t_1 [ч], а против течения t_2 [ч].

v (км/ч) t_1 (ч) = t_2 (ч) =

v_1 (км/ч)

Ответ:

Рисунок 2

На форме размещены 9 меток – они предназначены для отображения статической информации. Три из них будут использованы для отображения значений исходных данных и конечного результата. Обратим внимание на примененный к тексту меток (свойство Caption) шрифт. Изменение параметров шрифта объекта **Метка** доступно через свойство **Font**. Данное свойство через инспектор объектов открывает разработчику стандартное диалоговое окно **Шрифт**, через которое можно произвести необходимые изменения.

Обратим внимание на кнопки «Вычислить» и «Close» – помимо приведенного текста на них можно увидеть соответствующие изображения. В отличие от объекта Кнопка (Button) со страницы Standard, использованного на форме при решении задачи 1, на форме с рис. 2 использованы кнопки BitBtn со страницы Additional. По умолчанию на кнопке нет никакого изображения; для выбора рисунка на кнопке необходимо переопределить свойство Kind – значение выбирается из выпадающего списка (bkClose, bkAbort, bkOK и т.д.).

Текстовые поля (edit) отображают информацию о скоростях v и v_1 . Однако, в данном случае переопределим свойства этих полей так, чтобы в них нельзя было вводить данные – данные будут только отображаться. Такого эффекта можно добиться через свойство Enabled объектов Edit. Значение по умолчанию True заменим на False.

Компоненты ScrollBar – полосы прокрутки – будут использованы для увеличения/уменьшения значений скоростей в текстовых полях. Они компактны, по умолчанию они горизонтальные; повернем их на 90° – необходимо поменять значение свойства Kind с sbHorizontal на sbVertical.

Следующий шаг в работе с объектом ScrollBar: установить новые значения свойств Min и Max. Пусть $\text{Min} = -100$, $\text{Max} = 0$.

Еще один новый компонент, использованный на нашей форме – TrackBar. В решении нашей задачи имеет определенное сходство с объектом ScrollBar – он будет определять выбор значения для скоростей t_1 и t_2 . Определим минимальное значение $\text{Min} = 0$ и максимальное $\text{Max} = 100$.

Приступим к написанию кода.

Для изменения значений скоростей – свойства Text объектов edit, – используем значения свойства Position компонентов ScrollBar:

```
procedure TForm1.ScrollBar1Change(Sender: TObject);  
begin  
    edit1.Text := inttostr(-scrollbar1.position)  
end;
```

Для изменения значений времени – свойства Caption объектов Label, – используем значения свойства Position компонентов TrackBar:

```
procedure TForm1.TrackBar1Change(Sender: TObject);  
begin  
    label6.Caption := inttostr(trackbar1.Position)  
end;
```

Аналогичны коды подпрограмм для компонентов ScrollBar2 и TrackBar2.

✎ Мы не будем разбирать приведенные коды подпрограмм обработки событий OnChange компонентов ScrollBar и TrackBar – оставим это для самостоятельного рассмотрения.

Еще один компонент, о котором пока не говорили, но не заметить его на форме невозможно – Картинка (Image) – позволяет вставить в форму графическое изображение. Само изображение было создано в редакторе Paint и сохранено в формате jpg. Для установления связи компонента с изображением предназначено свойство Picture. Среда Delphi позволяет использовать такие форматы графических изображений, как jpg, bmp, ico, emf, wmf. Еще один немаловажный момент – задать свойству Proportional значение True, – в противном случае изображение будет вставлено в своем оригинальном размере, а не поместившаяся часть отсечена.

Выполнение описанных действий не дает решение задачи. Процедура решения задачи в определении формулы, вычисление которой суть код подпрограммы события OnClick кнопки «Выполнить». Остается только решить задачу и вывести формулу:

$$s = vt_1 + |v_1 - v|t_2.$$

Для простоты восприятия кода введем дополнительные локальные переменные в подпрограмму обработки события OnClick.

```
procedure TForm1.BitBtn2Click(Sender: TObject);  
    var  
        s, v, v1, t1, t2 : integer;  
begin  
    v:=strtoint(edit1.Text);  
    v1:=strtoint(edit2.Text);  
    t1:=strtoint(label6.Caption);  
    t2:=strtoint(label7.Caption);  
    s:=v*t1+abs(v1-v)*t2;  
    label9.Caption := inttostr(s)+' км.';  
end;
```

УСЛОВНЫЙ ОПЕРАТОР

Задача 3. [9, 173 (ч), с. 93] Составить программу вычисления функции:

$$f(x) = \begin{cases} x^2 + 3x + 9, & \text{если } x \leq 3, \\ \frac{\sin x}{x^2 - 9}, & \text{если } x > 3 \end{cases}.$$

Решение:

Спроектируем форму для решения задачи:

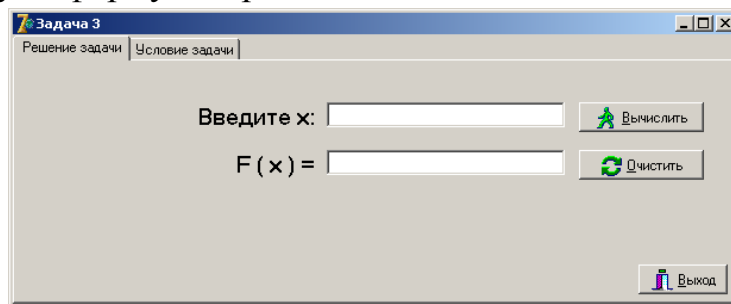


Рисунок 3

По внешнему виду она достаточно проста, содержит уже знакомые компоненты, но имеются и новые – это компонент PageControl (набор страниц) со страницы Win32 панели инструментов. На этот компонент добавлены две вкладки TabSheet – «Решение задачи» и «Условие задачи». Соответствующие названия вкладкам были присвоены изменением свойства Caption компонентов. На вкладке «Условие задачи» размещен только один объект Image, который отображает условие данной задачи – графического файла в формате jpg. На закладке «Решение задачи» две Метки (Label), два текстовых поля (Edit) и три

Кнопки (BitBtn). Код кнопок «Очистить» и «Выход» уже знаком по предыдущим задачам.

Разберем свойства полей Edit. Поле « $F(x)=$ » предназначено для отображения результата вычисления функции и предполагается, что он не допускает ввода данных. Для этого устанавливаем значение False свойства Enabled. Текстовое поле «Введите x :» предназначено для передачи программе числового значения параметра x функции $F(x)$, т.е. оно должно допускать ввод символов, которые образуют числовые значения – цифры, знак «-» (минус) и символ «.» (запятая). Ограничить набор вводимых в поле символов позволит событие OnKeyPress объекта Edit:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    if not(key in ['0'..'9','.', '-', #8]) then key:=#0;
end;
```


В данной подпрограмме реализован оператор условия if; в условии проверяется, не принадлежит набранный символ key множеству символов, состоящего из цифр, запятой, знака минус и кода клавиши Tab (#8).

Приступим к обработке события OnClick кнопки «Вычислить». Синтаксис оператора условия:

If <условие> then <оператор 1> else <оператор 2>;

В нашей подпрограмме оператор условия будет использован два раза: первый проверяет, не пусто ли текстовое поле, содержащее значение параметра x ; второе условие – проверка параметра x в функции $F(x)$.

```
procedure TForm1.BitBtn3Click(Sender: TObject);
var
    f, x : extended;
begin
    if edit1.Text <> '' then
    begin
        x:= strtofloat(edit1.Text);
        if x<=3 then f:=sqr(x)+3*x+9
            else f:=sin(x)/(sqr(x)-9);
        edit2.Text:=floattostrF(f,ffFixed,18,6);
    end;
end;
```

 В задаче 1 мы упоминали о некоторых недочетах приведенного решения. В качестве упражнения предлагаем доработать решение в задаче 1 с учетом возможностей условного оператора.

Задача 4. [4, 4.48, с. 39] Даны вещественные числа a, b, c ($a \neq 0$). Решить уравнение $ax^2 + bx + c = 0$.

Решение:

Приведем решение поставленной математической задачи в виде блок-схемы:

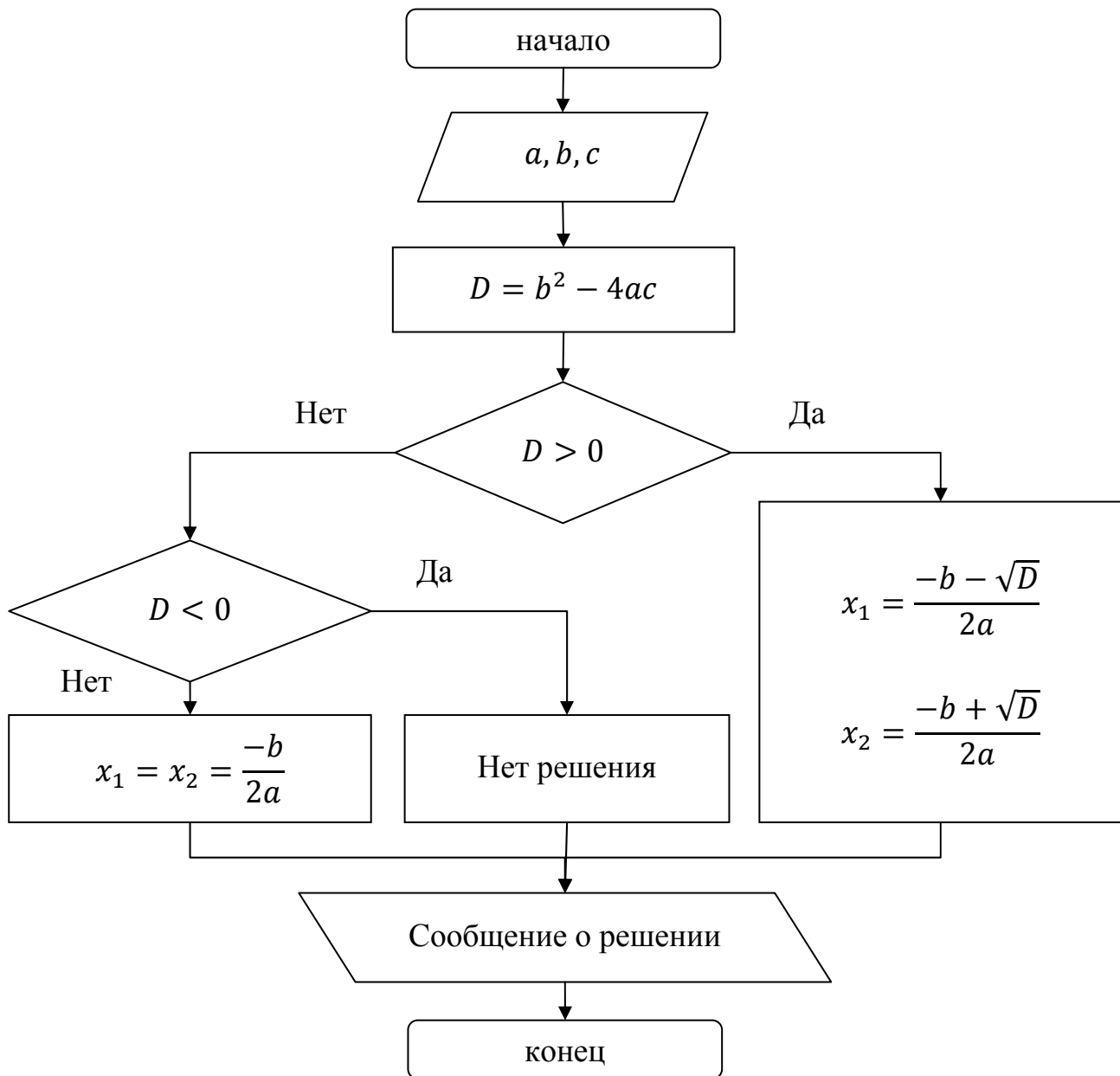


Рисунок 4

Созданная программа должна вычислить значение дискриминанта D , проверить выполнение одного из трех условий, накладываемых на дискриминант и вывести соответствующее сообщение на экран: «Существует два решения» и значения решений x_1 и x_2 для $D > 0$ или «Существует единственное решение» и его значение для $D = 0$ или «Нет решения» для $D < 0$.

Спроектирует форму проекта:

Рисунок 5

Многие из компонентов, использованных при построении формы, нам уже знакомы: контейнеры (GroupBox), кнопки (BitBtn), текстовые поля (edit), метки (label).

В решении данной задачи опустим некоторые коды обработки событий, которые оставим читателю в качестве самостоятельной работы. К таковым относятся обработки событий OnClick для кнопок «Очистить» и «Выход», коды событий OnKeyPress текстовых полей в группе объектов «Дано:». Также необходимо изменить значение свойства Enabled текстового поля edit в группе объектов «Дискриминант:».

Форма проекта обладает свойством BorderStyle, которое по умолчанию имеет значение bsSizeable. Это позволяет разработчику установить свои желаемые линейные размеры форме проекта. Однако, запущенное приложение – стандартное приложение операционной системы Windows и обладает стандартными элементами окна, такими как системный значок и кнопки навигации. Поменяем стандартный значок приложений, созданных в Delphi, на новый. Для создания новой иконки среда разработки Delphi предоставляет утилиту Image Editor 3.0 (Меню Инструменты►Image Editor). В редакторе создаем новый документ типа ico, создаем изображение новой иконки и сохраняем его. Заменить стандартную иконку на новую позволит свойство Icon формы. Также хотелось бы порекомендовать установить новые опции проекта (Меню Проект►Опции), на закладке Приложение можно задать название проекта и установить иконку приложения. Теперь разберемся с кнопками навигации, точнее с одной из них. По умолчанию компоненты на форме имеют абсолютные линейные размеры и значения, определяющие их положение на форме. Если воспользоваться кнопкой навигации Развернуть, то полученное окно приложения замостит весь экран, а компоненты загнаны в левый верхний угол окна. Логичнее установить параметры формы приложения так, чтобы она

не меняла размер окна. Для этого установим для свойства `biMaximize` группы свойств `BorderIcons` значение `false`.

В группе объектов «Решение:» имеются три метки. На стадии разработки и при запуске приложения они не заметны – свойство `Caption` пусто. Предположим, что первая из этих меток предназначена для вывода одного из сообщений: «Существует 2 корня уравнения!», «Уравнение не имеет корней!» и «Уравнение имеет 1 кратный корень!». Вторая и третья метки отображать значения решений уравнения, если они существуют, или останутся пустыми, если таковых нет.

Приведем код обработки события `OnClick` для кнопки «Вычислить»:

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  with GroupBox1 do
    with GroupBox1 do
      with GroupBox1 do
        if (edit1.Text<>'') and (edit2.Text<>'') and (edit3.Text<>'') then
          begin
            a:=strtofloat(edit1.Text);
            b:=strtofloat(edit2.Text);
            c:=strtofloat(edit3.Text);
            d:=sqr(b)-4*a*c;
            edit4.Text:=floattoStrF(d,ffFixed,18,4);
            if d>0 then begin
              x1:=(-b-sqrt(d))/2/a;
              x2:=(-b+sqrt(d))/2/a;
              label7.Caption:='x1 = '+ floattostrF(x1,ffFixed,18,4);
              label8.Caption:='x2 = '+ floattostrF(x2,ffFixed,18,4);
              st:='Существует 2 корня уравнения! ';
            end
            else if d<0 then begin
              st:='Уравнение не имеет корней! ';
            end
            else begin
              x1:=(-b)/2/a;
              label7.Caption:='x = '+ floattostrF(x1,ffFixed,18,4);
              st:='Уравнение имеет 1 кратный корень! ';
            end;
            label6.Caption:=st;
          end;
        end;
      end;
    end;
  end;
```

Заметим, что в приведенном листинге использованы переменные *a*, *b*, *c*, *d*, *x1*, *x2*, *st*, которые предварительно не были описаны. На самом деле, указанные переменные были описаны как глобальные в разделе *interface* модуля:

```
var  
    Form1: TForm1;  
    a, b, c, D, x1, x2 : extended;  
    st : string;
```

В построенной нами форме осталась без внимания кнопка «Условие». Реализуем ее обработчик события *OnClick* таким образом, чтобы он вызывал дополнительную форму, на которой будет размещена необходимая информация.

Для вызова дополнительной формы необходимо ее предварительно создать. По умолчанию наше приложение состоит из одной формы (объект *Form*). На ней мы размещали все остальные компоненты и именно с ней связан модуль *Unit1*, в котором прописывались все предыдущие коды подпрограмм.

Для вставки новой формы воспользуемся пунктами главного меню среды Delphi (Меню Файл ► Новый ► Другое...) – откроется диалоговое окно Новые элементы. Данное окно с достаточно большим числом закладок предоставляет большой выбор. Упростим процедуру разработки, выбрав на закладке *Forms* форму *About box*. В нашем проекте приложения появится новая форма. Это повлекло также появление нового модуля *Unit2*. Новая форма уже имеет некоторые компоненты. Некоторые из них в рамках нашего проекта останутся невостребованными, поэтому их можно просто удалить. В результате получим:

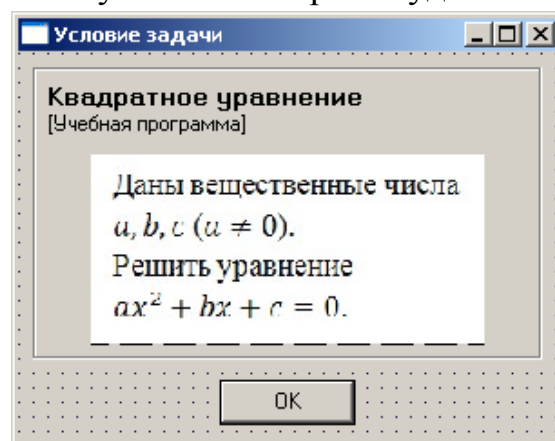


Рисунок 6

На форме *About box* от исходного остались кнопка (*Button*), компонент панель (*Panel*), на панели две метки (*label*) и графическое изображение (*Image*). В первую очередь следует поменять исходное изображение на изображение с условием задачи; будет не лишним поменять свойства *Caption* самой формы и меток (рис. 6). Еще одна особенность данной формы – значение *bsDialog* свойства *BorderStyle*. Изменение этого свойства изменит внешний вид формы –

исчезнет значок приложения в левом верхнем углу формы и из кнопок навигации в правом верхнем углу останется только кнопка «Заккрыть». Другие действия, в частности переопределения кода обработки события кнопки «ОК», не потребуются.

Приведем код обработки события OnClick для кнопки «Условие» на главной форме:

```
procedure TForm1.BitBtn3Click(Sender: TObject);  
begin  
    AboutBox.ShowModal;  
end;
```

Всего одна команда – вызов метода ShowModal формы AboutBox. Метод позволяет отобразить модальное окно. Когда форма отображается в виде модального окна, то приложение не может продолжать работу до тех пор, пока это окно не будет закрыто.

Если в данной редакции осуществить запуск программы, то появится информационное сообщение об ошибке (рис. 7).

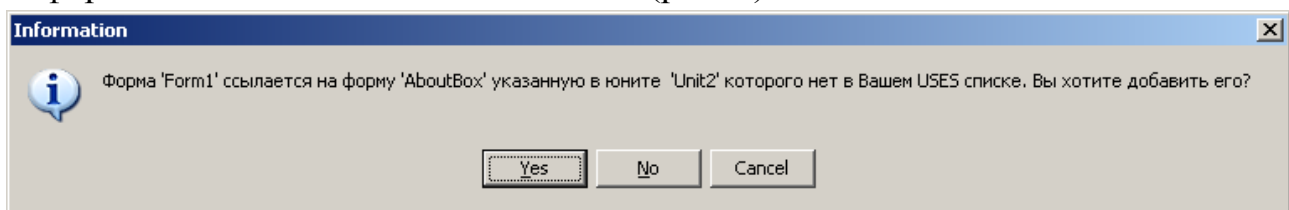


Рисунок 7

Согласимся с предложением, нажав на кнопку «Yes»; в коде модуля появится строка:

```
uses Unit2;
```

Задача решена.

ОПЕРАТОРЫ ЦИКЛА

Задача 5. [4, 6.14, с. 67] Дано число a ($1 < a \leq 1,5$). Найти такое наименьшее n , что в последовательности чисел $1 + \frac{1}{2}, 1 + \frac{1}{3}, \dots, 1 + \frac{1}{n}$ последнее число будет меньше a .

Решение:

Приступим к разработке формы приложения. Обратимся к условию задачи и отметим необходимость ввода числового значения в диапазоне от 1 до 1,5. В отличие от предыдущей задачи, в решении которой были использованы текстовые поля (edit), здесь будет использован компонент TrackBar. Значение a зависит от позиции ползунка. Это значение будет отображаться на метке (на рис. 7 – Label3). Переопределим значение свойства Orientation компонента TrackBar на trVertical, и пусть Min= 100000, Max= 149999 – это позволит в

дальнейшем выбрать значение параметра a в требуемом условии задачи диапазоне и с точностью до 5 знаков после запятой.

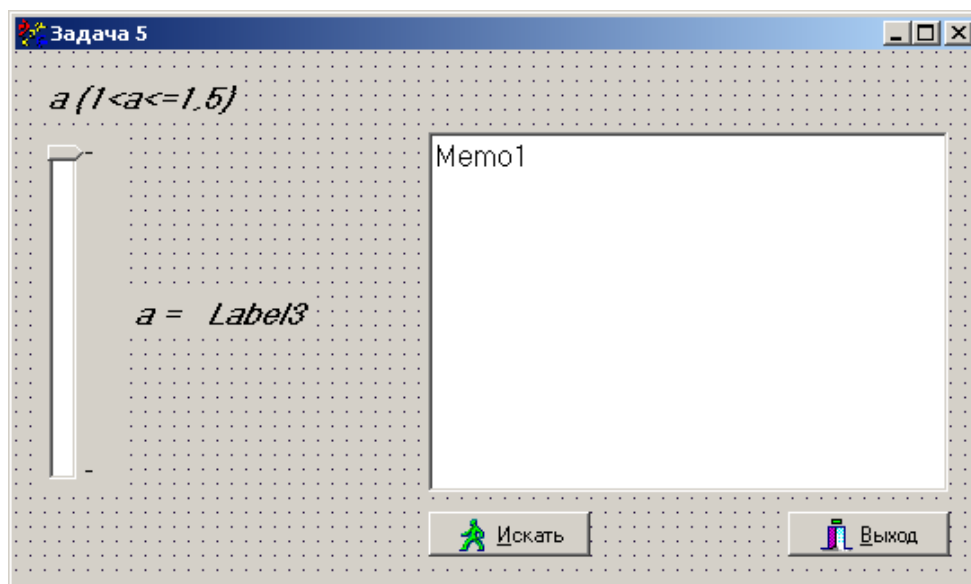


Рисунок 8

Движение ползунка от верхнего края до нижнего (отметим, что верхняя крайняя позиция соответствует значению свойства Min, а нижняя – значению Max) должно отображаться, как уже упоминалось выше, как значение свойства Caption метки Label3:

```
procedure TForm1.TrackBar1Change(Sender: TObject);
begin
    label3.Caption := floattostrF((250000-trackbar1.Position)/100000,
                                ffFixed, 18, 5);
    a:=strtofloat(label3.Caption);
end;
```

Центральное место на нашей форме занимает компонент Мемо – предназначен для ввода, редактирования и/или отображения достаточно длинного многострочного текста. Этот компонент существенно отличается уже от знакомого нам текстового поля (edit). Одно из основных отличий в том, что текст хранится в свойстве Lines, который представляет собой пронумерованный набор строк (нумерация начинается с нуля).

Поле Мемо позволяет вводить и редактировать текст, который в нем содержится, следовательно, нам необходимо запретить прямой доступ к содержимому. Для этого установим значение false свойству Enabled.

Вернемся к разработке формы проекта. Читатель может на свое усмотрение поменять иконку приложения, определить доступные кнопки навигации окна, задать новые линейные размеры и т.д. Но хотелось бы рассмотреть доступные события формы.

Событие OnCreate – возбуждается при создании формы. Обычно в этой процедуре, прописывают код программы, с помощью которой определяют создание нужных динамических объектов, выполняется настройка свойств формы или объектов, которые на ней находятся, задаются предварительные значения переменным, и т.д. Создать соответствующий обработчик события можно через Инспектор Объекта, на закладке События. Двойным щелчком левой кнопки мыши напротив события OnCreate будет автоматически создано тело подпрограммы выбранного события. Приведем код для нашей формы:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    trackBar1.Position:=trackBar1.Min;
    label3.Caption:=floattostrF((250000-trackbar1.Position)/100000,
                                ffFixed, 18, 5);
    memo1.Lines.Text:="";
end;
```

Первая строка тела подпрограммы определяет позицию ползунка компонента trackBar и устанавливает его равным свойству Min. Вторая строка переопределяет значение свойства Caption метки Label3 – оно станет равным числовому значению текущей позиции ползунка, а именно, значению 1,5. Третья строка очищает многострочное текстовое поле Мемо.

Осталась нерассмотренной только кнопка «Искать». Именно в событии OnClick этой кнопки пропишем код программы, который даст решение поставленной задачи. В качестве ответа, созданная программа отобразит в многострочном текстовом поле Мемо все значения искомого параметра n начиная с $n = 2$, отобразит соответствующее текущему n значение выражения $1 + \frac{1}{n}$, а в качестве последней строку: «Ответ: n =число». Приведем код программы:

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    memo1.Lines.Text:="";
    k:=0;
    x:=0;
    repeat
        k:=k+1; x:=1+1/(k+1);
        memo1.Lines.Add(' n = '+inttostr(k+1)+' : p ['+inttostr(k)+']= '+
                        floattostrF(x,ffFixed,18,5));
    until x<a;
    memo1.Lines.Add('Ответ : – n = '+inttostr(k+1));
end;
```

Использованные в подпрограмме переменные были описаны предварительно как глобальные:

```
var
    Form1: TForm1;
    k, i : integer;
    x, a : extended;
```

Вывод результирующих данных осуществляется в компоненте Memo. Набор строк Lines компонента Memo обладает методом Add, который позволяет добавить в поле новую строку.

В реализации алгоритма решения задачи использована конструкция repeat ... until ... – оператор цикла с последующей проверкой условия:

```
repeat <тело_цикла> until <условие>;
(повторять <тело_цикла> [до тех пор] пока [не выполнено условие]);
```

Здесь repeat и until – зарезервированные слова; <тело_цикла> – произвольная последовательность операторов Delphi; <условие> – выражение логического типа. Тот момент, что проверка условия выполняется после исполнения тела цикла, обеспечивает исполнение последнего хотя бы один раз. Еще одна отличительная особенность оператор цикла с последующей проверкой условия – пара слов repeat и until образуют своеобразные операторные скобки, наподобие пары begin и end.

Задача 6. [9, 456-а), с. 170] Вычислить значения определенных интегралов с точностью ε :

$$z = \int_1^4 \frac{\ln^2 x}{x} dx - \text{методом трапеций.}$$

Решение:

Очевидно, что форма будущего приложения будет содержать предельно малое число компонентов: даны пределы интегрирования a, b , есть подынтегральная функция, необходимо найти значение z (см. рис. 9). Вычисление значения интеграла будет произведено приближенным численным методом. Найденное значение, вообще говоря, не совпадает с точным, поэтому присутствует величина ε – допустимая погрешность искомого решения.

С математической точки зрения, поиск значения интеграла методом трапеций – сложная, математическая теория. Пользователю программы, которая появится как решение данной задачи, может быть не интересно, как она (программа) ищет значение интеграла. А возможно, кто-то заинтересуется, какой алгоритм реализован для поиска искомого значения. Можно предложить

ему воспользоваться литературой, а можно и встроить в программу справочную службу, которая появится при нажатии на кнопку F1.

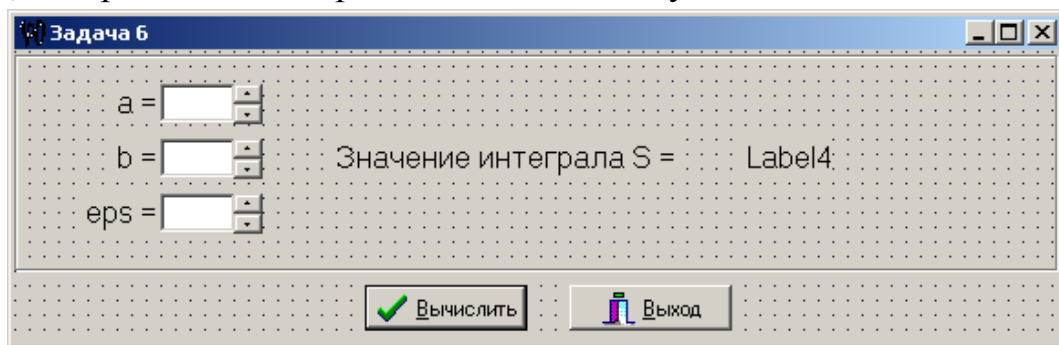


Рисунок 9

При разработке формы для решения данной задачи использованы метки (label), текстовые поля (edit), кнопки (bitbtn), а также компонент счетчик (UpDown), который предназначен для пошагового регулирования числовой величины.

Компонент счетчик (UpDown) связан с текстовыми полями (edit) и будет использован для изменения пределов интегрирования. Определим минимальные и максимальные значения компонентов UpDown 1 и 10 соответственно. Основное событие для данного компонента OnChanging возникает при любом изменении регулируемой величины. Приведем код для данного события:

```
procedure TForm1.UpDown1Changing(Sender: TObject;
    var AllowChange: Boolean);
begin
    edit1.Text:=inttostr(UpDown1.Position);
end;
```

Несколько иначе будет выглядеть этот метод для счетчика, который связан с текстовым полем, определяющим значение параметра ε :

```
procedure TForm1.UpDown3Changing(Sender: TObject;
    var AllowChange: Boolean);
begin
    edit3.Text:=FloattostrF(UpDown3.Position/100000, ffFixed, 18, 5);
end;
```

Воспользуемся уже знакомым нам методом OnCreate для задания начальных значений текстовых полей:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    edit1.Text:=inttostr(updown1.Position);
    edit2.Text:=inttostr(updown2.Position);
    UpDown3.Position:=UpDown3.Min;
```

```
edit3.Text:=FloatToStrF(UpDown3.Position/100000, ffFixed,18,5);
end;
```

Мы уже упомянули о справочной службе приложения. Справочная служба будет содержать теоретическую основу применения метода трапеций к вычислению интегралов.

Основным элементом справочной системы являются HLP-файлы, в которых находится справочная информация. В простейшем случае справочная система программы может представлять собой один единственный HLP-файл. В нашей программе это файл help.hlp:

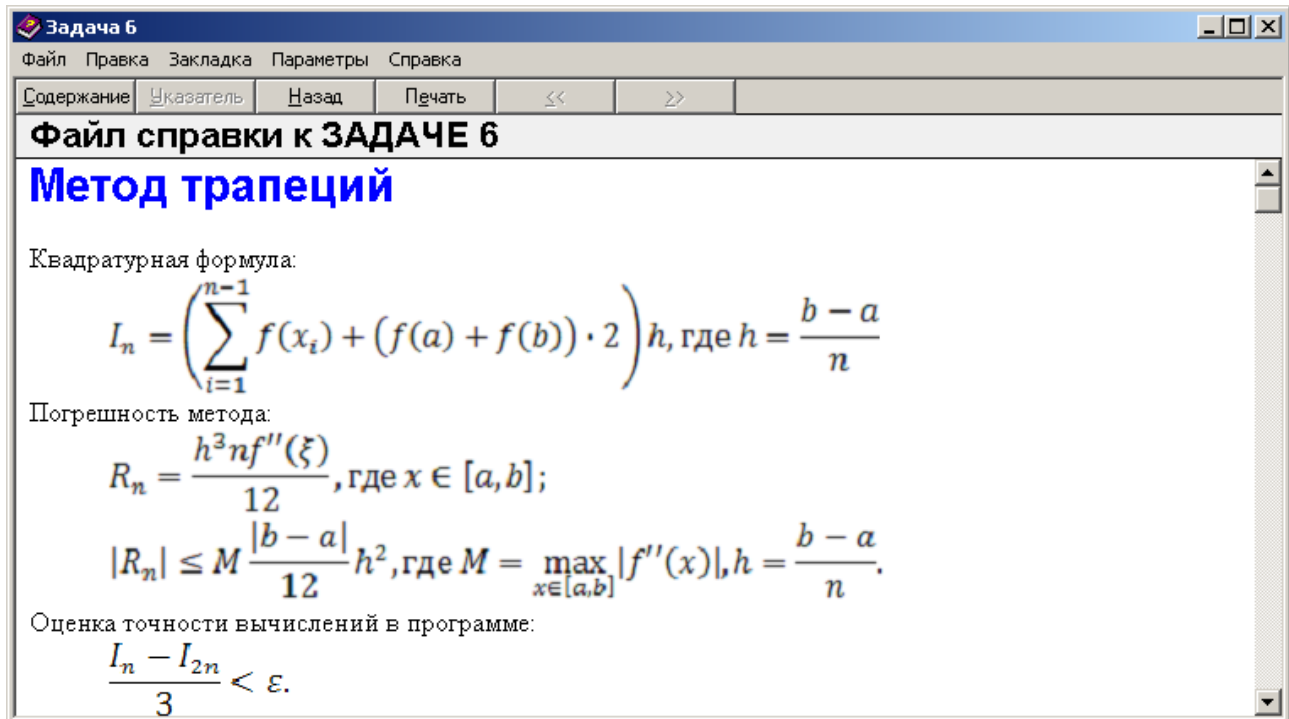


Рисунок 10

Для создания справочной службы в рамках решения данной задачи было использовано приложение EC Software Help & Manual 5, demo-версию которой можно скачать на официальном сайте проекта <http://www.helpandmanual.com/>. Это современное программное решение может быть использовано также для создания файлов с расширением CHM, HTML, PDF, HXS, RTF, и EXE – электронных книг.

Существующий файл help.hlp подключается к проекту через диалоговое окно Опции проекта (меню Проект ► Опции... ►) на закладке Приложение:

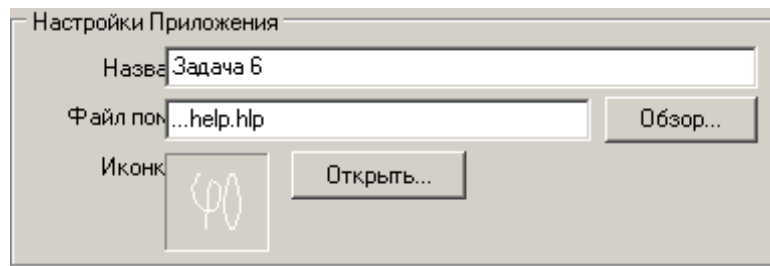


Рисунок 11

Для того чтобы во время работы программы пользователь, нажав клавишу F1, мог получить справочную информацию, надо чтобы свойство HelpContext формы содержало числовой идентификатор нужного раздела (в нашем случае это значение 1).

Ну и последний штрих – осталось просто решить задачу, прописав алгоритм вычисления значения интеграла с помощью заданного метода – это код метода обработки события OnClick кнопки «Вычислить»:

```
procedure TForm1.BitBtn2Click(Sender: TObject);
var
    n, i : integer;
    x, d, dx, z1, z2, z3 : extended;
begin
    a:= strtoint(edit1.Text);
    b:= strtoint(edit2.Text);
    eps:=strtofloat(edit3.Text);
    n:=2; z1:=0;
    z3:=(sqr(ln(a))/a+sqr(ln(b))/b)/2;
    repeat
        z2:=z3;
        dx:=(b-a)/n; x:=a;
        for i:=1 to n-1 do
            begin
                x:=x+dx;
                z2:=z2+sqr(ln(x))/x
            end;
        z2:=z2*dx;
        d:=abs(z2-z1);
        z1:=z2;
        n:=n*2;
    until d<eps;
    label4.Caption:=floattostrF(z2, ffFixed, 18,5);
end;
```

В приведенном коде использована инструкция `for` – оператора цикла, который применяется, когда число повторений заранее известно:

```
for <параметр>:=<начальное_значение> to <конечное_значение> do
begin
    {тело цикла}
end;
```

Здесь `<параметр>` – имя переменной, определяющей число повторений; `<начальное_значение>` – выражение, определяющее начальное значение переменной параметра цикла; `<конечное_значение>` – выражение, определяющее конечное значение переменной параметра цикла. Если тело цикла состоит из одного оператора, то операторные скобки `begin... end` могут быть опущены.

Задача 7. [9, 421-х), с. 165] Составить программу для вычисления значений функции $F(x)$ на отрезке $[a, b]$ с шагом h . Результат представить в виде таблицы, первый столбец которой – значения аргумента, второй – соответствующие значения функции:

$$F(x) = \sin^2 x - \cos 2x.$$

Решение:

В условии задачи требуется построить таблицу данных. Это подводит нас к использованию на форме компонента `StringGrid` – текстовая таблица, предназначена для создания таблиц, в ячейках которых располагаются произвольные текстовые строки.

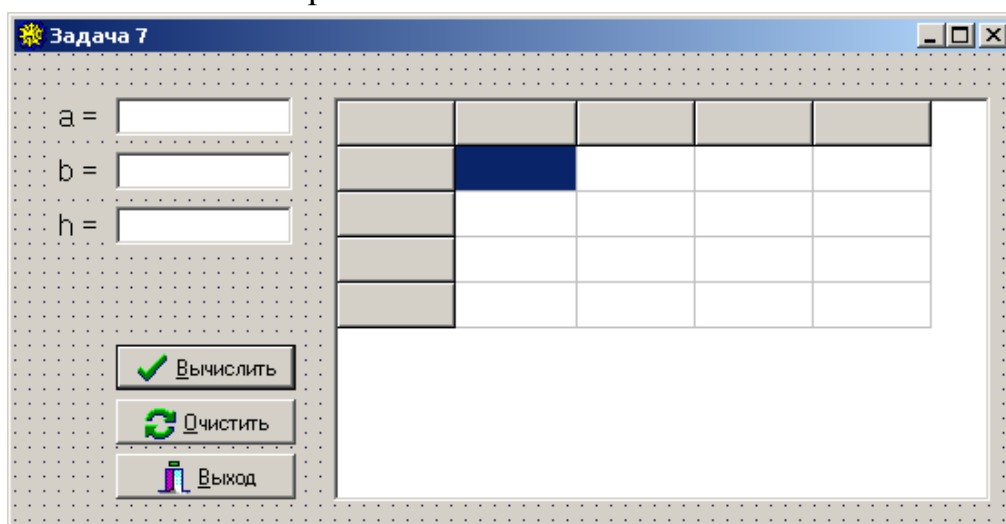


Рисунок 12

Изучим подробнее данный компонент. Текстовая таблица (`StringGrid`) делится на две части – фиксированную и рабочую. Фиксированная часть служит для показа заголовков строк и столбцов. Рабочая часть – основная часть таблицы. Она содержит произвольное число строк и столбцов, которые могут

быть изменены программно. Число строк таблицы определяется свойством RowCount, а число столбцов – свойством ColCount. Нумерация начинается с нуля. Важную роль играет свойство Cells, которая позволяет «добраться» до каждой ячейки таблицы. Например:

Cells [2, 1] := 'Второй столбец и первая строка рабочей части таблицы';

Работая над формой, начнем с обработки события OnCreate:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    stringgrid1.ColCount:=2;           // 1
    stringgrid1.RowCount:=2;          // 2
    stringgrid1.ColWidths[0]:=100;    // 3
    stringgrid1.ColWidths[1]:=250;    // 4
    stringgrid1.Cells[0,0]:='x';      // 5
    stringgrid1.Cells[1,0]:='F(x)';   // 6
    stringgrid1.Cells[0,1]:='';       // 7
    stringgrid1.Cells[1,1]:='';       // 8
    edit1.Text:='0';                  // 9
    edit2.Text:='1';                  // 10
    edit3.Text:='0,1';                // 11
    bitbtn1.Enabled:=true;            // 12
end;
```

Рассмотрим приведенный код: первая строка меняет исходное значение по умолчанию 5 свойства ColCount компонента StringGrid на 2, т.е. теперь в таблице только 2 колонки; вторая строка кода устанавливает значение 2 для свойства RowCount – число строк; строки 3 и 4 – свойство ColWidths[i] определяет ширину колонки с номером i; строки с 5 по 8 задают значения ячейкам таблицы; строки с 9 по 11 задают начальные значения текстовым полям edit. Строка 12 пока никак не вписывается в череду приведенных выше команд. Ее необходимость связана с наличием кнопки «Очистить».

Если рассмотреть обособленно назначение кнопки «Очистить», то можно заметить сходство ее функции с приведенным кодом обработки события OnCreate формы. В данном случае нет необходимости писать повторно код, который уже имеется в наличии в виде отдельной подпрограммы. На вкладке События Инспектора объекта напротив события OnClick кнопки «Очистить» в выпадающем списке выбираем метод FormCreate. Подобный ход делает программу более лаконичной и придает ей гибкость.

Этот же прием можно применить и в отношении текстовых полей – записать метод обработки события OnKeyPress для первого и связать его с остальными.

Приведем решение задачи – код метода обработки события OnClick кнопки «Вычислить»:

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var
    i:integer;
begin
    a:=strtofloat(edit1.text);
    b:=strtofloat(edit2.text);
    h:=strtofloat(edit3.text);
    x:=a; i:=1;
    while x<=b do
        begin
            f:=sqr(sin(x))-cos(2*x);
            stringgrid1.RowCount:=stringgrid1.RowCount+1;
            stringgrid1.Cells[0,i]:=floattostrf(x,ffFixed,10,4);
            stringgrid1.Cells[1,i]:=floattostrf(f,ffFixed,10,4);
            x:=x+h; i:=i+1;
        end;
        bitbtn1.Enabled:=false;
    end;
```

Интерес в данном коде представляет последняя строка, которая переопределяет значение свойства Enabled на false для кнопки «Вычислить». Данное изменение делает кнопку «Вычислить» неактивной. Именно это обстоятельство определило наличие строки 12 в приведенном ранее коде обработки события OnCreate формы, которое возвращало кнопке «Вычислить» исходное значение true свойства Enabled.

Возвращаясь к общей теме циклов, отметим, что в программе использована инструкция while. Инструкция while используется в том случае, когда некоторую последовательность действий необходимо выполнить несколько раз, причем число повторений заранее неизвестно и зависит от некоего условия:

```
while <условие> do
begin
    {тело цикла}
end;
```

Здесь <условие> – выражение логического типа. Если оно истинно, то выполняется {тело цикла}, иначе оператор завершает работу. Обычно, {тело цикла} влияет на <условие>. В противном случае, цикл может выполняться бесконечно.

МАССИВЫ

Задача 8. [9, 834, с. 279] На метеопосту за период наблюдения в n дней была получена таблица величины атмосферного давления. Записать программу, которая:

- а) Определяет максимальный и минимальный по величине элементы массива и номера соответствующих им дней в массиве;
- б) Определяет среднее значение давления за время наблюдения;
- в) Подсчитывает количество дней, в которые давление превышало заданную величину d ;
- г) Сортирует массив по возрастанию;
- д) Выводит исходный массив и результат вычислений.

Для отладки программы исходный массив сформировать с помощью генератора случайных чисел. Значение давления брать равным $760 \pm a$, где $a = [0, \dots, 20]$.

Решение:

Обработка данных в массиве представляет особый интерес с точки зрения программирования. Массив – структурированный тип данных, все элементы массива одного типа, доступ к элементам массива осуществляется по его индексу. Использование массива намного упрощает процедуру обработки большого набора данных.

Данная задача имеет вполне реальное практическое применение. Не

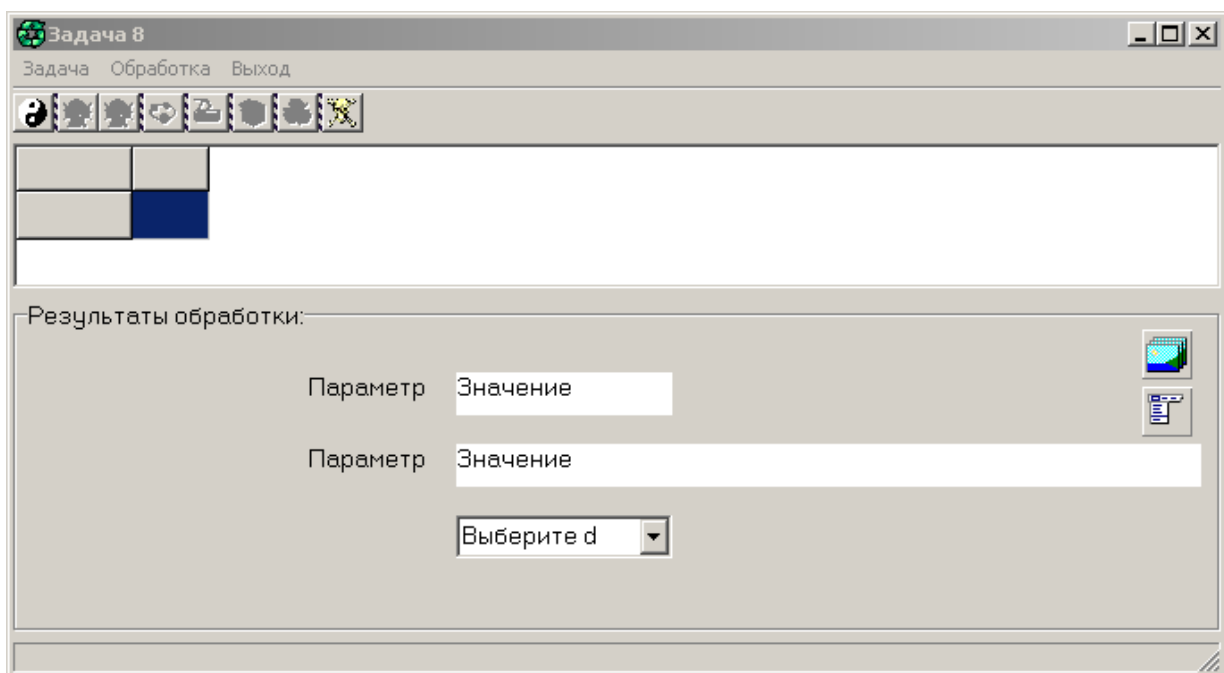


Рисунок 13

рассматривая вопрос ввода данных, можно привести решение задачи как реализацию полноценного программного продукта, который имеет полноценное меню, панель инструментов и т.д.

Построим форму проекта (рис. 13).

П.1. Меню приложения.

Главное меню приложения – компонент MainMenu на вкладке Standard – помещается в любое место на форме. После установки компонента на форму создаются пункты меню. Щелкнем по нему дважды левой кнопкой мыши и откроем окно конструктора меню (рис. 14).

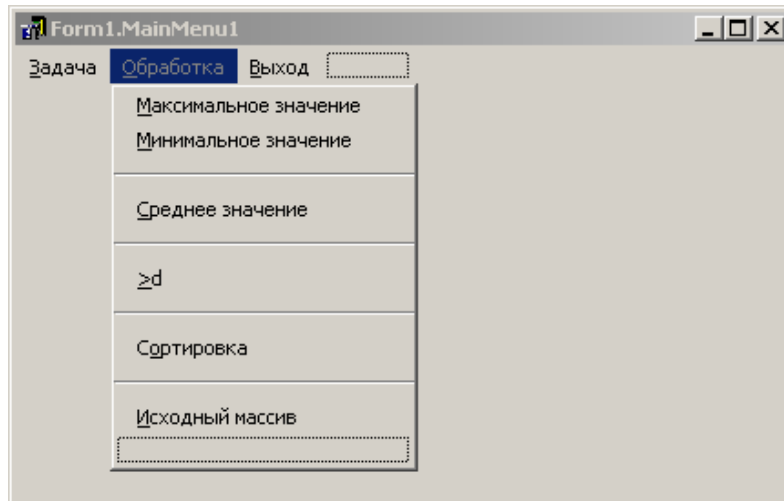


Рисунок 14

Создание пунктов меню не вызывает проблем. В окне конструктора меню выбирается соответствующий пункт, а в окне Инспектора объектов в свойстве Caption задаем его название. В меню можно выделить группы команд горизонтальными линиями – для этого в качестве значения свойства Caption задается значение «-» (знак «минус»).

Для нашей задачи создадим меню из трех основных пунктов: «Задача», «Обработка», «Выход». В меню «Задача» только один подпункт – «Загрузить данные». В меню «Обработка» создадим пункты «Максимальное значение», «Минимальное значение», «Среднее значение», «>d», «Сортировка», «Исходный массив». Пункт меню «Выход» оставим без подпунктов.

Меню, его подпункты также являются объектами и обладают свойствами и событиями. Для того чтобы пункты меню выполняли некое действие определяется событие OnClick.

П.2. Панель инструментов и компонент ImageList.

Панель инструментов предоставляет быстрый и удобный доступ к командам приложения через набор кнопок, который располагается в верхней части формы под строкой меню. Обычно, панель инструментов дублирует команды пунктов меню.



Рисунок 15

Вставим на форму компонент `ToolBar` с вкладки `Win32`. Он автоматически поместится в верхней части формы. Теперь добавим на панель несколько кнопок – 8 – по числу исполняемых команд меню. Каждая кнопка – компонент `ToolButton`. Но не стоит искать его на вкладках среды Delphi – их там нет. Для вставки кнопки нажимаем правую кнопку мыши по панели и в контекстном меню опцию Новая кнопка (`New Button`). Определенные кнопки можно группировать, для этого между кнопками вставляются Разделители (опция `New Separator`).

Новые кнопки на панели инструментов все одинаковые. Их принято различать по рисункам. Однако кнопки `ToolButton` не имеют свойства, способного хранить изображение или информацию о нем; а компонент `ToolBar` позволяет извлечь нужное изображение из контейнера `ImageList` и поместить это изображение на кнопку.

Добавим изображения кнопкам на форме решения нашей задачи.

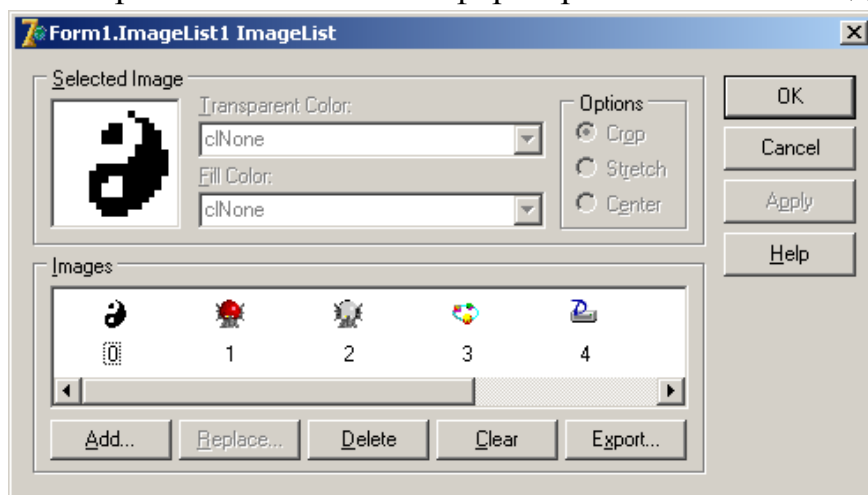


Рисунок 16

Вставим на форму компонент `ImageList` (хранилище изображений). Этот компонент представляет собой контейнер для хранения множества рисунков одинакового размера. Предварительно следует подготовить набор файлов с иконками (формат `.ico`) – либо взять несколько готовых файлов, либо нарисовать самостоятельно в инструменте `Image Editor` (Меню Инструменты ► `Image Editor`).

Двойным щелчком мыши по компоненту `ImageList` активируется диалоговое окно управления хранилищем изображений (рис. 16). Заполним контейнер щелкая на кнопке «`Add...`» и добавляя последовательно восемь иконок в соответствии с кнопками на панели инструментов. Нажав кнопку `OK`, примем сделанные изменения.

На форме выделим компонент ToolBar и перейдем в окно Инспектор объектов. Свойство Images компонента выделяется на общем фоне цветом; раскроем список в правой колонке свойства и выберем пункт ImageList1, т.е. укажем на источник изображений. Если все было сделано правильно, то кнопки на панели инструментов должны получить свои изображения (рис. 15).

Но наличие изображения на кнопке еще не гарантирует то обстоятельство, что пользователь сразу разберется в ее предназначении. Хорошая практика, когда «наехав» указателем мыши на ту или иную кнопку, появляется всплывающая подсказка, которая объяснит ее назначение.

Подсказка, которая появляется при наведении указателя мыши на объект – это текст, который содержится в его свойстве Hint. Выделяя каждую кнопку поочередно, выберем свойство Hint и внесем в поле справа соответствующее значение: «Загрузить данные», «Максимальное значение» и т.д. Помимо наличия самой подсказки, необходимо «разрешить» этой подсказке появляться. За это отвечает свойство ShowHint, которое по умолчанию имеет значение false. Установим значение true для этого свойства.

Вернувшись к условию задачи, обратим внимание на то, что реализация решения, в котором пункты а)-д) распределены по разным пунктам меню и кнопкам панели инструментов, независящим друг от друга, может возникнуть ситуация, когда будет запущена команда на обработку данных, которых в программе еще нет. Это может спровоцировать исключительную ситуацию и сбой в программе.

По этой причине, переопределим значение свойства Enabled пункта меню «Обработка» и кнопок панели инструментов на значение false.

П.3. Строка состояния.

Строка состояния обычно располагается в нижней части формы. Предназначена для отображения различной информации, которая отражает текущее состояние приложения, статус определенных процессов. Например, в приложении MS Office Word 2007 в строке статуса отражается информация о номере текущей страницы, общее число страниц, число слов в документе, текущая раскладка клавиатуры, режимы просмотра документа, масштаб.

Для вставки строки состояния на форму выбираем на вкладке Win32 компонент StatusBar. После вставки он автоматически располагается в нижней части формы, занимая ее по всей длине.

Компонент может иметь несколько секций, которые могут быть добавлены программно с помощью метода Add свойства Panels. Для изменения содержимого строки состояния необходимо обратиться к конкретной панели. Нумерация панелей начинается с 0. Например, фрагмент кода:

```
statusbar1.Panels.Items[0].Text:='Сортировка данных по возрастанию';
```

обращается к свойству Text панели (Panels) с индексом 0 (Items[0]) строки состояния StatusBar1.

В решении нашей задачи строку состояния будем использовать как дополнительный источник справки – при наведении указателя мыши на кнопки панели инструментов на строке состояния будут отображаться справочная информация о ее назначении. В данном случае необходимо воспользоваться событием OnMouseMove, который генерируется при перемещении мыши над объектом. Выберем первую кнопку на панели инструментов (она дублирует подпункт меню «Загрузить данные» пункта меню «Задача»); в окне Инспектора объектов перейдем на вкладку События; щелкнем дважды по левой кнопке мыши в поле напротив события OnMouseMove. В окне Редактора кода появится подпрограмма обработки данного события, где добавим всего одну строчку:

```
procedure TForm1.ToolButton1MouseMove(Sender: TObject; Shift: TShiftState;  
    X, Y: Integer);  
begin  
    statusbar1.Panels.Items[0].Text:='Загрузить данные';  
end;
```

✚ Приведенный код интуитивно понятен. Кнопок, как уже упоминалось ранее, на панели инструментов 8. Предоставляем читателю возможность самостоятельно добавить еще 7 соответствующих подпрограмм для остальных кнопок.

Последняя деталь – на строке состояния останется сообщение, сгенерированное обработчиком события OnMouseMove кнопки, над которой в последний раз был проведен указатель мыши. Для того, чтобы панель строки состояния очищалась, если никакая из кнопок не находится под указателем мыши необходимо описать еще один обработчик события OnMouseMove, но на этот раз для формы:

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState;  
    X, Y: Integer);  
begin  
    statusbar1.Panels.Items[0].Text:="";  
end;
```

П.4. «Результаты обработки» – компонент GroupBox.

На форме поместим компонент Панель группирования GroupBox – прямоугольная рамка и текст в разрыве рамки. На панель разместим две метки (label), два текстовых поля (edit) и выпадающий список (ComboBox). Эти компоненты необходимы для отображения результатов выполнения пунктов а)-д) задачи. При этом определимся с тем моментом, что некоторые из них будут

невидимыми пользователю, т.к. не будут отображать никакой информации. А сама панель группирования будет невидимой в момент запуска программы и до загрузки в программу данных.

П.5. Форма приложения.

Настройка формы приложения будет состоять из определения исходных значений ее свойств и задания некоторых событий.

Как и ранее, зададим свойству `biMaximize` группы свойств `BorderIcons` значение `false`. Это сделает недоступной кнопку «Распахнуть» в группе кнопок навигации окна приложения.

Однако, это не сделает невозможным изменение размера формы простым перетаскиванием краев формы. Рассмотрим и прокомментируем код события `OnCreate` формы:

```
procedure TForm1.FormCreate(Sender: TObject);
var
    i:integer;
begin
    constraints.MaxHeight:=height;           //1
    constraints.MinHeight:=height;           //2
    constraints.MaxWidth:=Width;              //3
    constraints.MinWidth:=Width;              //4
    stringgrid1.Cells[0,0]:='День:.';        //5
    stringgrid1.Cells[0,1]:='Давление:.';     //6
    randomize; n:=31;                         //7
    groupbox1.Visible:=false;                 //8
    label1.Visible:=false; edit1.Visible:=false; //9
    label2.Visible:=false; edit2.Visible:=false; //10
    combobox1.Visible:=false;                 //11
    statusbar1.Panels.Add;                     //12
    for i:=0 to 40 do                          //13
        combobox1.items.Add(inttostr(740+i))  //14
    end;
```

Строки 1-4 программно определяют значения параметров свойства `constraints` формы: `MaxHeight` – максимальная высота формы, `MinHeight` – минимальная высота формы, `MaxWidth` – максимальная ширина формы, `MinWidth` – минимальная ширина формы. Им присваиваем значения линейных размеров, которые установились в процессе разработки формы.

Строки 5 и 6 задают значения заголовкам первой и второй строк таблицы (компонент `StringGrid`).

Строка 7 инициализирует генератор псевдослучайных чисел (процедура `randomize`) и задает значение 31 переменной `n`.

Строка 8 переопределяет исходное значение свойства `Visible` панели группирования на `false`, что делает данный компонент невидимым. Аналогично действуют строки 9-11 в отношении меток, текстовых полей и выпадающего списка.

Строка 12 добавляет единственную панель на строку состояния.

Строки 13-14 задают список значений от 740 до 780 выпадающего списка.

П.6. Пункт меню «Задача».

Пункт меню «Задача» содержит только один подпункт – «Загрузить данные». Данный подпункт предназначен для задания значений таблицы величин атмосферного давления – массива целых чисел, который в программе определен переменной `mas`:

```
var  
    mas : tMas;
```

где `tMas` определяется пользовательским типом:

```
type  
    tMas = array[1..50] of integer;
```

Каждый пункт также является объектом и обладает своими свойствами и событиями. К данному пункту можно получить доступ через окно конструктора меню (рис. 14). Нас интересует только событие `OnClick` – определяет действие, которое будет выполнено при щелчке левой кнопкой мыши по пункту меню.

```
procedure TForm1.N3Click(Sender: TObject);  
var  
    i:integer;  
begin  
    for i:=1 to n do  
        begin  
            stringgrid1.ColWidths[i]:=35;  
            stringgrid1.Cells[i,0]:=inttostr(i);  
            mas[i]:=740+random(41);  
            stringgrid1.Cells[i,1]:=inttostr(mas[i]);  
            stringgrid1.ColCount:=stringgrid1.ColCount+1;  
        end;  
    n1.Enabled:=false; n4.Enabled:=true; n13.Enabled:=false;  
    toolbutton1.Enabled:=false;  
    toolbutton2.Enabled:=true;  
    toolbutton3.Enabled:=true;  
    toolbutton4.Enabled:=true;
```

```

        toolbutton5.Enabled:=true;
        toolbutton6.Enabled:=true;
    groupbox1.Visible:=true;
end;

```

Центральное место в приведенном коде занимает тело цикла конструкции `for`. Оно: задает ширину текущего столбца в таблице – свойство `ColWidths[i]`; задает в качестве заголовка этого столбца значение от 1 до n (`Cells[i,0]:=inttostr(i)`); определяет согласно условия случайное значение для текущего элемента массива; переносит это значение в таблицу; добавляет в таблицу дополнительный столбец – увеличивает значение свойства `ColCount` на 1.

Следующий блок команд также играет существенную роль.

✚ Мы не будем разбирать остальные строки приведенного кода (оставим для самостоятельного разбора), однако приведем описание объектов: N1 – пункт меню «Задача», N4 – пункт меню «Обработка», N13 – подпункт «Исходный массив» пункта меню «Обработка», `ToolButton(i)` – кнопки на панели инструментов, `groupbox` – панель группирования.

II.7. Пункт меню «Обработка».

Пункт меню «Обработка» имеет несколько подпунктов. Рассмотрим код подпункта «Максимальное значение»:

```

procedure TForm1.N5Click(Sender: TObject);
var
    max, i : integer;
    st : string;
begin
    max:= mas[1]; edit2.Text:="";
    for i:=2 to n do
        if max<mas[i] then max:=mas[i];
    for i:=1 to n do
        if mas[i]=max then
            begin
                edit2.Text:=edit2.Text+inttostr(i)+' ';
            end;
    label1.Visible:=true;
    label1.Caption:='Максимальное значение';
    label2.Visible:=true;
    label2.Caption:='Дни, когда достигнут максимум:';

```



```
edit1.Visible:=true;  
edit1.Text:=inttostr(max)+' мм.рт.ст.';  
edit2.Visible:=true;  
combobox1.Visible:=false;  
st:=edit2.Text;  
delete(st, length(edit2.Text)-1, 1);  
edit2.Text:=st;
```

```
end;
```

Код обработки события OnClick подпункта «Минимальное значение»:

```
procedure TForm1.N6Click(Sender: TObject);
```

```
var
```

```
    min, i : integer;
```

```
    st : string;
```

```
begin
```

```
    min:= mas[1]; edit2.Text:="";
```

```
    for i:=2 to n do
```

```
        if min>mas[i] then min:=mas[i];
```

```
    for i:=1 to n do
```

```
        if mas[i]=min then
```

```
            begin
```

```
                edit2.Text:=edit2.Text+inttostr(i)+' , ';
```

```
            end;
```

```
    label1.Visible:=true;
```

```
    label1.Caption:='Минимальное значение';
```

```
    label2.Visible:=true;
```

```
    label2.Caption:='Дни, когда достигнут минимум:';
```

```
    edit1.Visible:=true;
```

```
    edit1.Text:=inttostr(min)+' мм.рт.ст.';
```

```
    edit2.Visible:=true;
```

```
    combobox1.Visible:=false;
```

```
    st:=edit2.Text;
```

```
    delete(st, length(edit2.Text)-1, 1);
```

```
    edit2.Text:=st;
```

```
end;
```

Код обработки события OnClick подпункта «Среднее значение»:

```
procedure TForm1.N8Click(Sender: TObject);
```

```
var
```

```
    sred, i : integer;
```

```
begin
```

```

sred:=0;
for i:=1 to n do sred:=sred+mas[i];
sred:=round(sred/n);
label1.Visible:=true;
label1.Caption:='Среднее значение';
label2.Visible:=false;
edit1.Visible:=true;
edit1.Text:=inttostr(sred)+' мм.пт.ст.';
edit2.Visible:=false;
combobox1.Visible:=false;

```

end;

Код обработки события OnClick подпункта «>d»:

```

procedure TForm1.d1Click(Sender: TObject);
begin
    combobox1.Visible:=true;
    label1.Visible:=true;
    label1.Caption:='Значение параметра d';
    label2.Visible:=false;
    edit2.Visible:=false;

```

end;

Заметим, что приведенный код не решает соответствующего пункта задачи. Для решения же задачи в виду особенности компонента ComboBox предусмотрен обработчик события OnChange:

```

procedure TForm1.ComboBox1Change(Sender: TObject);
var
    k, i : integer;
begin
    k:=0;
    for i:=1 to n do
        if mas[i]>strtoint(combobox1.Text) then k:=k+1;
    label2.Visible:=true;
    label2.Caption:='Число дней:';
    edit2.Visible:=true;
    edit2.Text:=inttostr(k)+' дней'

```

end;

Код обработки события OnClick подпункта «Сортировка»:

```

procedure TForm1.N11Click(Sender: TObject);
var
    m : tMas;

```

```
        i, j, min, buf : integer;
    begin
        m:=mas;
        for i:=1 to n-1 do
            begin
                min:=i;
                for j:=i+1 to n do
                    if m[j] < m[min] then min:=j;
                buf:=m[i]; m[i]:=m[min]; m[min]:=buf;
            end;
        for i:=1 to n do
            stringgrid1.Cells[i,1]:=inttostr(m[i]);
        toolbutton7.Enabled:=true;
        toolbutton6.Enabled:=false;
        n13.Enabled:=true;
        n11.Enabled:=false;
        combobox1.Visible:=false;
        label1.Visible:=true;
        label1.Caption:='Данные отсортированы!';
        label2.Visible:=false;
        edit1.Visible:=false;
        edit2.Visible:=false;
    end;
```

Для сортировки данных в массиве в подпрограмме описана дополнительная переменная типа массив, которой передан весь набор значений из исходного массива. Данные могли бы быть отсортированы и в исходном массиве, но тогда мы не смогли бы привести исходный набор. Для сортировки был использован алгоритм сортировки выбором [8]

Код обработки события OnClick подпункта «Исходный массив»:

```
procedure TForm1.N13Click(Sender: TObject);
var
    i:integer;
begin
    toolbutton6.Enabled:=true;
    toolbutton7.Enabled:=false;
    n11.Enabled:=true;
    n13.Enabled:=false;
    for i:=1 to n do
        stringgrid1.Cells[i,1]:=inttostr(mas[i]);
```

```
combobox1.Visible:=false;  
label1.Visible:=true;  
label1.Caption:='Исходный набор данных!';  
label2.Visible:=false;  
edit1.Visible:=false;  
edit2.Visible:=false;  
end;
```

➤ Следующий пункт меню «Выход» не имеет подпунктов. Ее назначение – завершение программы. Описание кода обработки соответствующего события оставим читателю в качестве самостоятельного задания.

П.8. Связь между меню и панелью инструментов.

В решениях предыдущих задач мы уже говорили о том, как назначать обработчику события объекта описанный ранее код другого объекта.

В данной задаче кнопки на панели инструментов, как отмечалось ранее, дублируют команды пунктов меню. Выберем первую кнопку панели инструментов (объект ToolButton1, предназначена для дублирования подпункта «Загрузить данные» пункта меню «Задача»). На окне Инспектора объекта переходим на вкладку События.

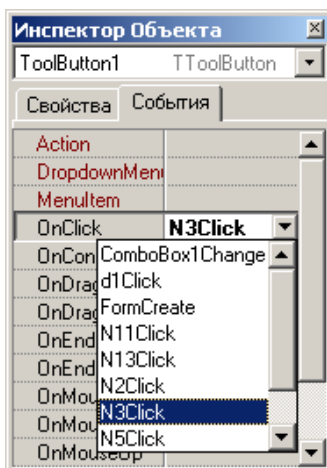


Рисунок 17

В выпадающем списке напротив события OnClick (рис. 17) выбираем метод, соответствующий обработчику события OnClick подпункта «Загрузить данные» пункта меню «Задача» (в нашем случае N3Click). Теперь первая кнопка панели инструментов (рис. 15) выполняет то же действие, что и упомянутый пункт меню.

Осталось повторить указанное действие для остальных кнопок, установив соответствие между пунктами меню и кнопками панели инструментов нашего приложения.

ЛИТЕРАТУРА

1. Delphi/ Программирование на языке высокого уровня: Учебник для вузов / В.В. Фаронов. – СПб.: Питер, 2003. – 640 с.: ил.
2. Turbo Pascal / С. А. Немнюгин. – СПб: Издательство «Питер», 2000. – 496 с.: ил.
3. Дарахвелидзе П. Г., Марков Е. П. Программирование в Delphi 7. — СПб.: БХВ-Петербург, 2003. – 784 с : ил.
4. Златопольский Д. М. Сборник задач по программированию – 2-е изд. перераб. и доп. – СПб: БХВ-Петербург, 2007. – 240 с.: ил..
5. Информатика и ИКТ. Задачник по моделированию 9-11 класс. Базовый уровень / Под ред. проф. Н.В. Макаровой. – Питер, 2007. – 192 с.: ил.
6. Информатика и информационные технологии. Учебник для 10-11 классов / Н.Д. Угринович. – 4-е изд. – М.: Бином. Лаборатория знаний, 2007. – 511 с.: ил.
7. Карпов Б. Delphi: специальный справочник. – СПб.: Питер, 2001. – 688 с.: ил.
8. Методы поиска и сортировки / М.С. Густокашин. – М.: Чистые пруды, 2007. – 32 с. – (Библиотечка «Первого сентября», серия «Информатика». Вып. 6 (18)).
9. Сборник задач по программированию / Д.А. Гуденко, Д.В Петроченко. – СПб.: Питер, 2003. – 475 с.: ил. – (Серия «Компас»).