

**Министерство сельского хозяйства Российской Федерации**  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Государственный университет по землеустройству»

Кафедра информатики

Е.Е. Дмитриева  
М.И. Коробочкин

**Учебно-методическое пособие**  
**ПО ИНФОРМАТИКЕ**  
**Для заочников**

**Часть II. Программирование в среде Delphi**

Допущено Научно-методическим советом по информатике  
при Министерстве образования и науки РФ в качестве учебного пособия  
по дисциплине «Информатика» для студентов высших учебных заведений,  
обучающихся по направлениям: «Землеустройство и кадастры»,  
«Менеджмент», «Прикладная геодезия».

Москва 2016

## **Введение**

Цель данного учебно-методического пособия – помочь студентам заочного факультета изучить основы программирования на примерах программ, реализующих линейные, разветвляющиеся, циклические алгоритмические структуры и их комбинации. Читатель получит представление о том, как разрабатываются программы, работающие под управлением операционных систем из семейства Windows, и сможет создавать свои программные приложения под Windows. Разработка алгоритмов и программ, отладка синтаксических и логических ошибок, помимо прочего, помогают развивать логическое и аналитическое мышление, что принципиально важно для студентов и выпускников технических и экономических специальностей.

В качестве языка, на котором читатель будет осваивать технологию современного программирования, выбран популярный язык Delphi. Важными достоинствами языка Delphi для начинающего программиста являются простота и дружелюбие, за счет которых достигается высокая скорость разработки приложений. Язык программирования является составной частью системы быстрой разработки программных приложений RAD Delphi. (До 7 версии язык программирования, входящий в систему Delphi назывался Object Pascal. Начиная с Delphi 7, язык программирования официально получил название Delphi.).

### **Содержание контрольной работы**

Контрольная работа заключается в выполнении 5 индивидуальных задач (1 задача из одной главы). Номер индивидуальной задачи определяется номером варианта (последняя цифра номера зачетной книги, либо две последние цифры, если они  $\leq 20$ ).

Оформляется отчет:

1. Титульный лист контрольной работы (образец в деканате)
2. По каждой из 5 задач:
  - 1) постановка задачи;
  - 2) алгоритм;
  - 3) экранная форма (интерфейс пользователя);

- 4) код программы;
- 5) результаты тестирования (три тестовых примера).

## Глава 1. Знакомство со средой программирования Delphi

Система быстрой разработки программных приложений Delphi основана на технологии визуального проектирования и событийного программирования. Разработчик создает интерфейс для программы путем выбора готовых компонентов–объектов и размещает их с учетом удобства пользователей программы. С объектами могут происходить разные события, например клик по кнопке, потеря фокуса, перемещение. Событийное программирование – это связывание программного кода, реализующего нужные действия, с определенным событием, происходящим с компонентом–объектом.

Разберем структуру среды на примерах версий Delphi 6,7 и Delphi XE, независимо от конкретно используемой версии, чтобы подготовить читателя к самостоятельному освоиванию новых версий, которые будут появляться на рынке программных продуктов.

### Среда программирования Delphi

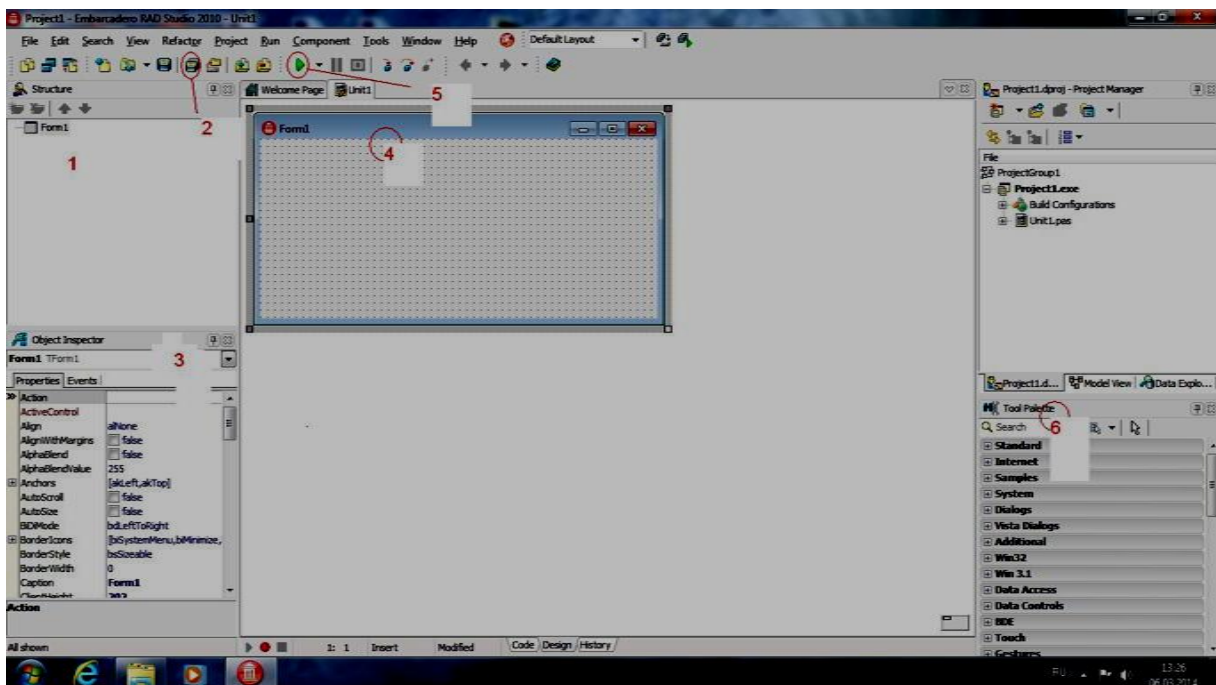


Рис. 1–1. Окно среды Delphi XE

Рассмотрим основные составные части Delphi, представленные на Рис. 1–1:

– Главное меню. Находится под заголовком окна, где указано имя проекта (по умолчанию Project1) и название версии Delphi, с которой вы работаете.

– Панель кнопок быстрого доступа. На ней находятся, помеченные цифрами 2 и 5 кнопки: 2 – кнопка связанная с командой, Сохранить все, 5 – кнопка Выполнить.

– Дизайнер Форм (Form Designer), обозначен цифрой 4.

– Окно Редактора Кода (Editor Windows) находится позади окна Дизайнера Форм. Его можно увидеть, если отодвинуть окно Form1 или щелкнуть по вкладке Unit1.

– Палитра Компонентов (Tool Palette или Component Palette). Tool Palette в Delphi XE находится справа внизу и обозначена цифрой 6 на рис. 1–1. Палитра Компонентов в Delphi 6, 7 находится справа от панели кнопок быстрого доступа (рис.1– 4).

– Инспектор Объектов (Object Inspector) помечен цифрой 3 на рис. 1–1. Для Delphi 6, 7 показан на рис.1– 5.

– Структура проекта (Structure) помечена цифрой 1 на рис. 1–1. В терминологии Delphi 6, 7 это Дерево Объектов (Object Tree View).

– Справочник (On-line help). Клавиша F1.

Главное или системное меню (Menu System) предоставляет быстрый и гибкий интерфейс к среде Delphi, потому что может управляться по набору “горячих клавиш”. Это удобно еще и тем, что здесь используются слова или короткие фразы, более точные и понятные, нежели иконки или пиктограммы. Например, в меню File собраны команды, позволяющие производить действия с файлами проекта. Здесь находятся команды сохранения проекта, сохранения модуля и команда Сохранить все (**Save All**), которую все должны запомнить.

Дизайнер Форм отдельно показан на рис.1–2.



Рис. 1–2. Дизайнер форм

Дизайнер Форм первоначально состоит из одного пустого окна, которое заполняется всевозможными объектами, выбранными на Палитре Компонентов.

Окно редактора кода (Рис. 1–3) в некоторых версиях Delphi, например 6 и 7, находится прямо позади окна стартовой формы, которую можно просто отодвинуть или щелкнуть по вкладке Unit1. В Delphi XE для доступа к окну редактора кода, можно щелкнуть по вкладке Code, находящейся внизу окна среды Delphi (Рис. 1–1). Для возврата в окно формы – щелкнуть по вкладке Design.

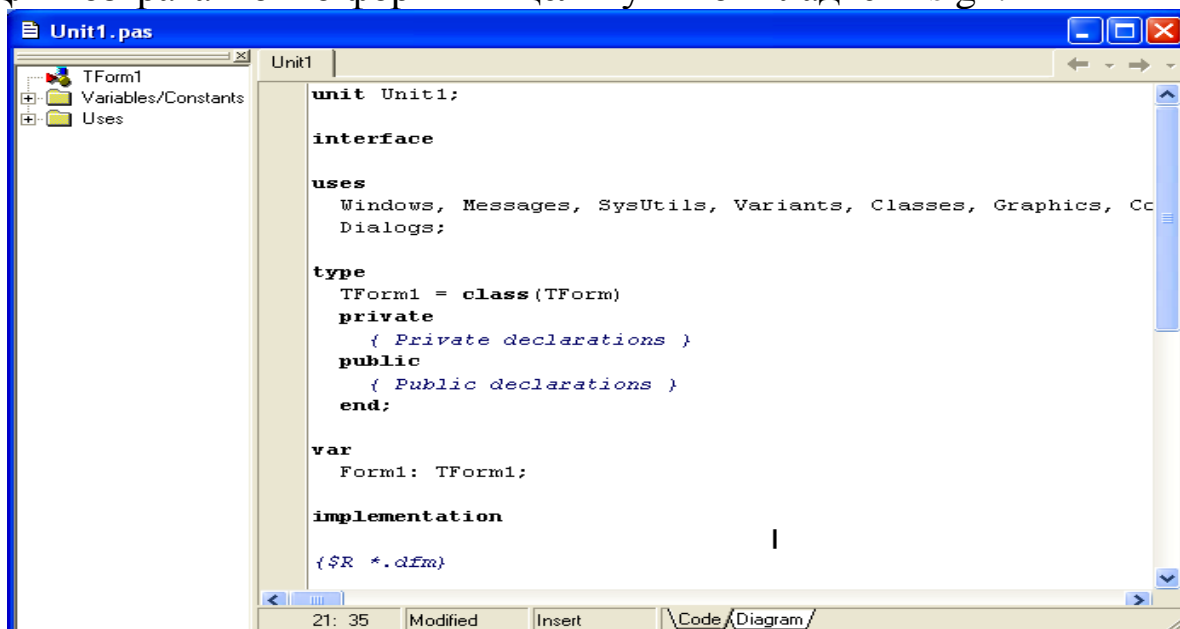


Рис. 1–3. Окно Редактора Кода

Палитра Компонентов (рис. 1–4) позволяет выбрать нужные объекты для размещения их на Дизайнере Форм. Для использования Палитры Компонентов просто один раз щелкните мышкой на одном из объектов, и потом второй раз – на Дизайнере Форм. Выбранный объект появится на проектируемом окне в месте щелчка и им можно манипулировать с помощью мыши: двигать его с места на место, использовать границу, прорисованную вокруг объекта, для изменения его размеров.

Палитра Компонентов использует постраничную группировку объектов. В нижней части Палитры находится набор закладок – Standard, Additional, Dialogs и т.д. Если щелкнуть мышью на одной из закладок, то можно перейти на соответствующую страницу Палитры Компонентов.



Рис. 1–4: Палитра Компонентов – место, где выбираются объекты, которые будут помещены на форму.

Ниже перечислены стандартные компоненты Delphi, непосредственно используемые при изучении темы 1, с некоторыми комментариями по их применению.



**Курсор** – не компонент, просто пиктограмма для быстрой отмены выбора какого-либо объекта.



**Label** (метка) служит для отображения текста на экране. Можно изменить шрифт и цвет метки, если дважды щелкнуть на свойство Font в Инспекторе Объектов.



**Edit** – стандартный компонент для ввода. Он может быть использован для отображения короткого фрагмента текста и позволяет пользователю вводить данные во время выполнения программы.



**Button** (кнопка) позволяет выполнять какие-либо действия при нажатии кнопки во время выполнения программы.

Если навести указатель мыши на компонент, то появится подсказка, которая указывает тип компонента. Например, если

мы наведем указатель мыши на компонент ОК, появится подсказка **TButton**.

В Delphi все делается очень просто. Поместив Button на форму, по двойному щелчку можно вызвать заготовку обработчика события нажатия кнопки. Далее нужно заполнить заготовку нужным кодом.

Слева от Дизайнера Форм находится Инспектор Объектов (рис. 1–5). Информация в Инспекторе Объектов меняется в зависимости от объекта (компонента), выбранного на форме (дизайнере форм). Каждый компонент является объектом и можно менять его вид и поведение с помощью Инспектора Объектов.

Инспектор Объектов состоит из двух страниц, каждую из которых можно использовать для определения поведения данного компонента. Первая страница – это список свойств, вторая – список событий. Если нужно изменить что-нибудь, связанное с определенным компонентом, то обычно это делается в Инспекторе Объектов. Например, можно изменить имя, положение и размер компонента Label на форме, изменяя свойства Caption, Left, Top, Height и Width.

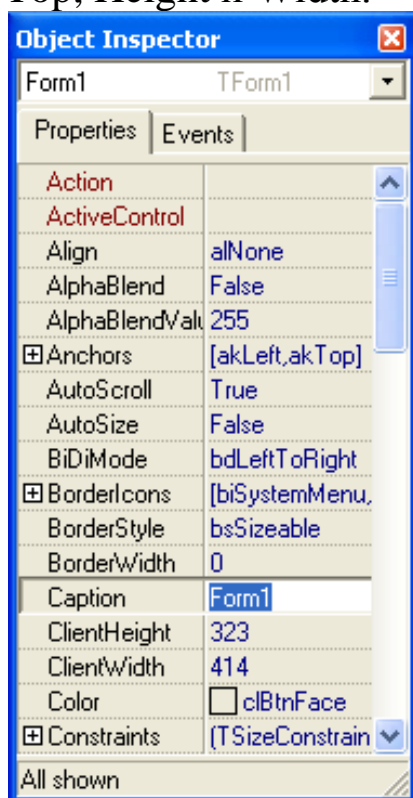


Рис. 1–5. Инспектор объектов

Можно использовать закладки Инспектора Объектов для переключения между страницами свойств (Properties) и событий (Events). Страница событий связана с Редактором; если дважды щелкнуть мышкой на правую сторону какого-нибудь пункта, то соответствующий данному событию код автоматически запишется в Редактор, сам Редактор немедленно получит фокус, и можно сразу добавить код обработчика данного события.

Последняя, важная часть среды Delphi – Справочник (on-line help). Для доступа к этому инструменту нужно просто выбрать в системном меню пункт Help (F1).

Справочник является контекстно-зависимым; при нажатии клавиши F1, на экране появляется подсказка, соответствующая текущей ситуации. Например, находясь в Инспекторе Объектов, выберите какое-нибудь свойство и нажмите F1 – получите справку о назначении данного свойства. Если в любой момент работы в среде Delphi возникает неясность или затруднение – нажмите F1 и необходимая информация появится на экране.

Панель с кнопками для быстрого доступа (SpeedBar) для Delphi XE находится непосредственно под меню (Рис 1–1). В Delphi 6, 7 SpeedBar (рис.1–6) тоже находится под меню, слева от Палитры Компонентов. SpeedBar выполняет многое из того, что можно сделать через меню. Если задержать мышь над любой из кнопок на SpeedBar, то появится подсказка, объясняющая назначение данной кнопки.



Рис. 1–6: Панель с кнопками для быстрого доступа (SpeedBar) для Delphi 6, 7

### **Задача**

Составить программу (проект) для вычисления площади треугольника по основанию и высоте.

Прежде чем решать задачу на компьютере, надо понять, какие **исходные данные** нам нужны, какие **результаты** мы должны получить на выходе программы и определить какие действия должен выполнить компьютер, чтобы решить задачу.

В условии задачи сказано, что вычисление площади нужно выполнить по основанию и высоте. Будем считать, что мы имеем дело с земельным участком треугольной формы, для которого измерено основание  $a$  в метрах и высота  $h$  в метрах с точностью до сантиметров. Площадь  $S$  требуется вычислить в гектарах.

Какие действия должен выполнить компьютер?

1. Прочитать в память исходные данные: основание  $a$  и высоту  $h$ .
2. Вычислить по известной из геометрии формуле площадь треугольника



$$S = \frac{1}{2}ah.$$

3. Перевести площадь в гектары

$$S_{га.} = S_{кв.м.}/10000.$$

4. Вывести на экран результат вычислений.

Теперь мы можем использовать инструменты визуального проектирования Delphi для создания интерфейса пользователя в виде окна, как принято в операционных системах семейства Windows. Для создания окна используется компонент Form. При запуске Delphi 6, 7 стартовая или главная форма появляется на экране автоматически.

Опишем процесс проектирования главной формы для среды Delphi XE. При запуске Delphi XE на экране появляется окно (Рис. 1–7).

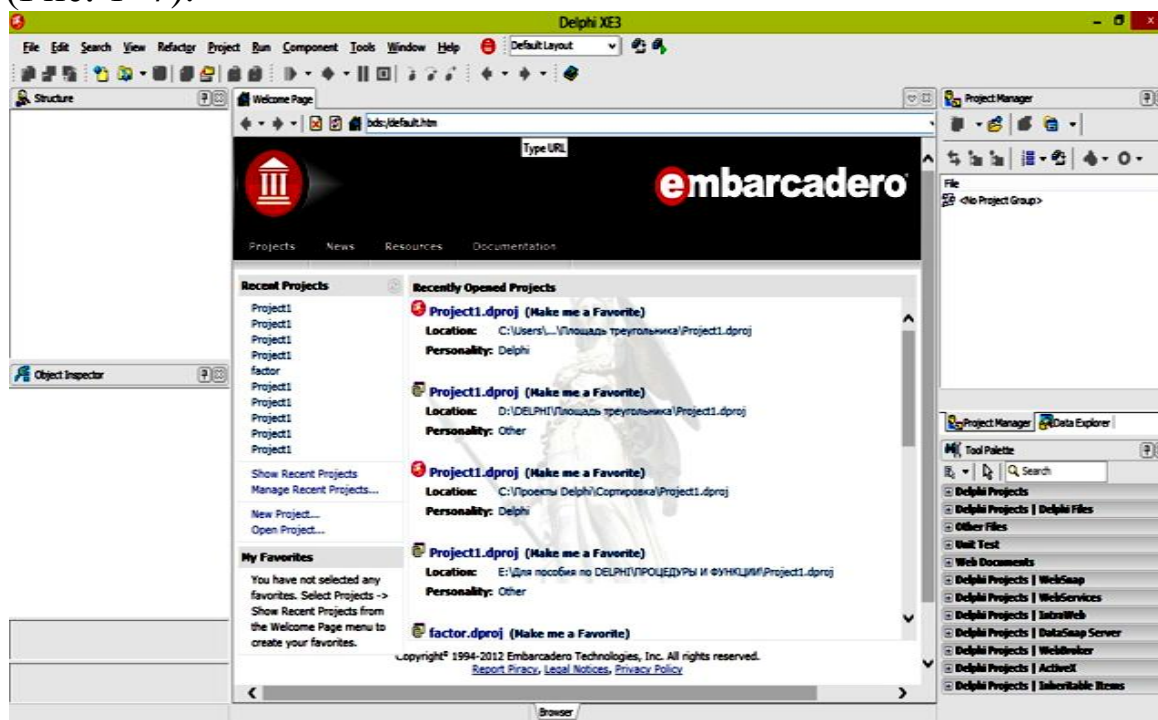


Рис. 1–7. Стартовое окно Delphi XE

На приветственной странице (Welcome Page) указаны названия последних разрабатываемых проектов в среде на данном компьютере. Создать новый проект можно следующими способами.

1. Закрывать окно Welcome Page. Перейти в главное меню. Пройти по цепочке команд:

File → New → VCL Forms Application Delphi. На экране появится стартовая форма (рис. 1–8).

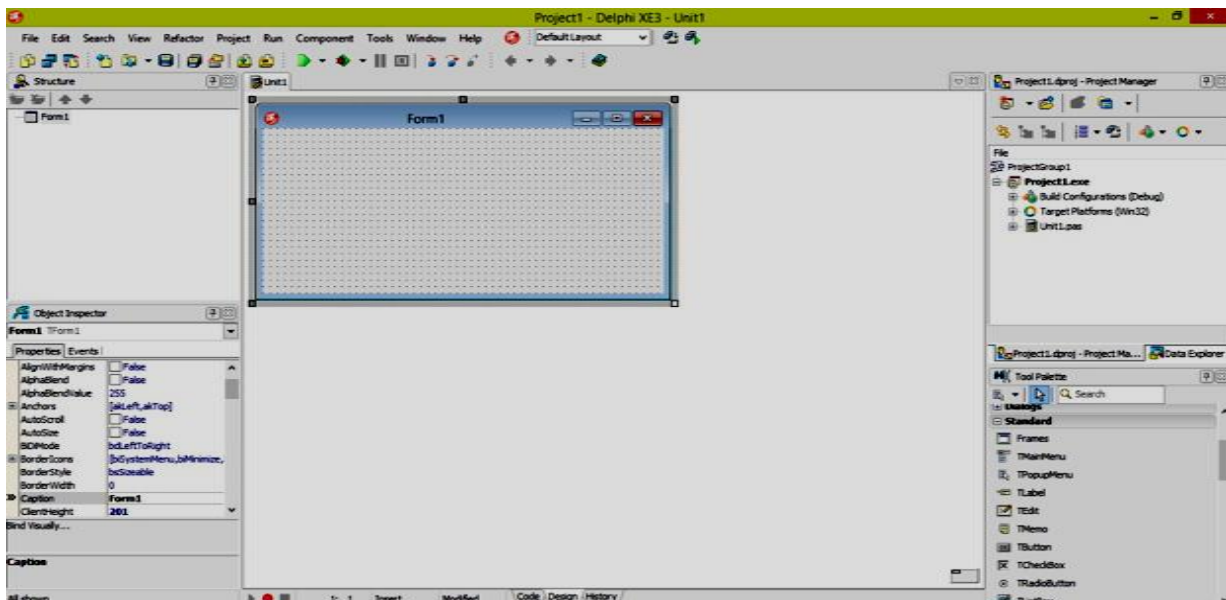


Рис. 1–8. Окно стартовой формы

2. В меню Projects выбрать New Project. На экране появится окно, представленное на рис. 1–9.

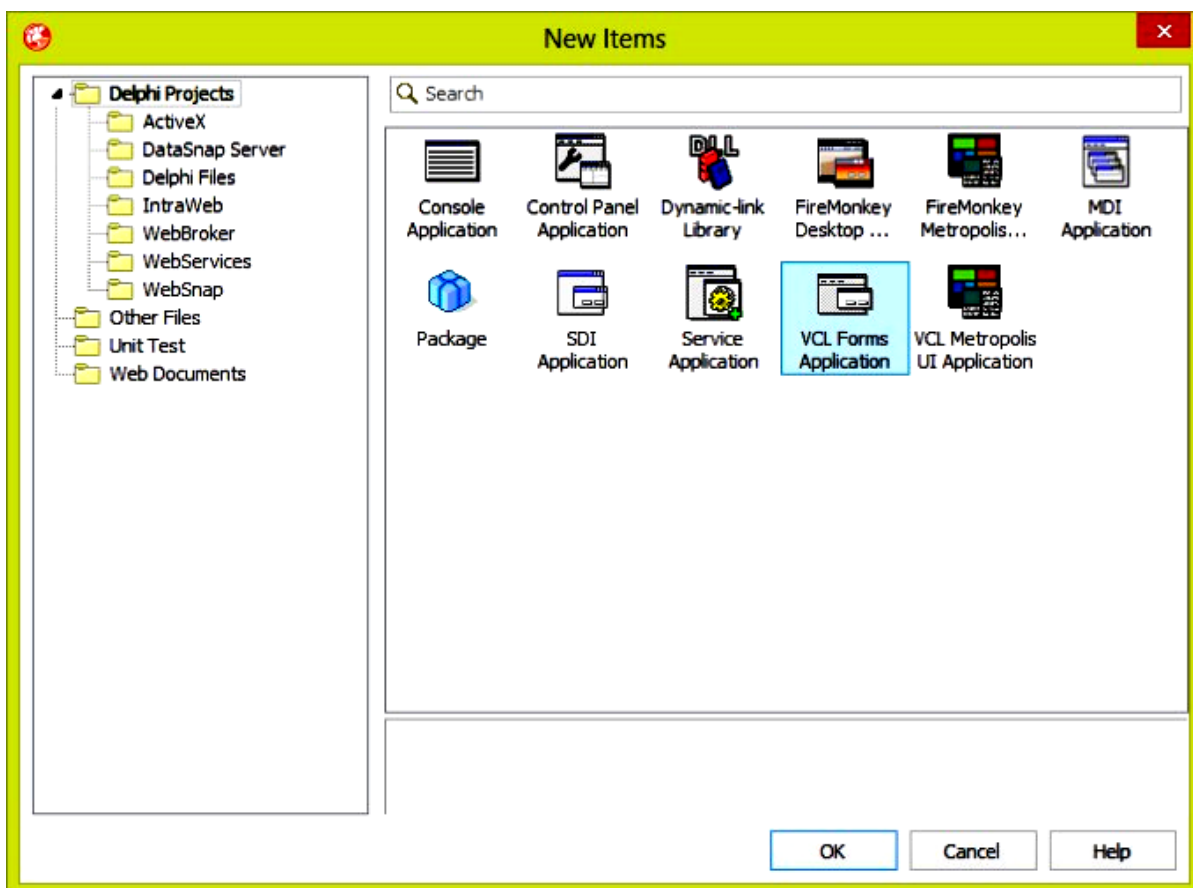


Рис. 1–9. Окно, где нужно выбрать вид создаваемого приложения

Мы создаем VCL Forms Application (приложение на основе библиотеки визуальных компонентов). Выбираем соответствующую

щий значок и нажимаем кнопку ОК. На экране появится окно стартовой формы (рис 1–8).

Теперь в окне стартовой формы Form1 мы должны разместить компоненты для ввода исходных данных с соответствующими подписями, чтобы пользователь программы понимал в какое поле ввести какое значение, и поместить компонент для вывода результатов работы программы. Предусмотрим две командные кнопки: одну, которая собственно запускает процесс вычислений, и другую, которая закрывает программу. Нужно запроектировать окно интерфейса примерно так, как показано на рис. 1–10.

Какие компоненты надо использовать и какие свойства для них установить?

Чтобы задать заголовок окна, нужно подсветить его на форме Form1, так чтобы в инспекторе объектов (Object Inspector) появился объект Form1 и его свойства. Надо выделить свойство Caption и вместо того, что там написано по умолчанию (Form1), написать то, что нам нужно, а именно – Площадь треугольника.

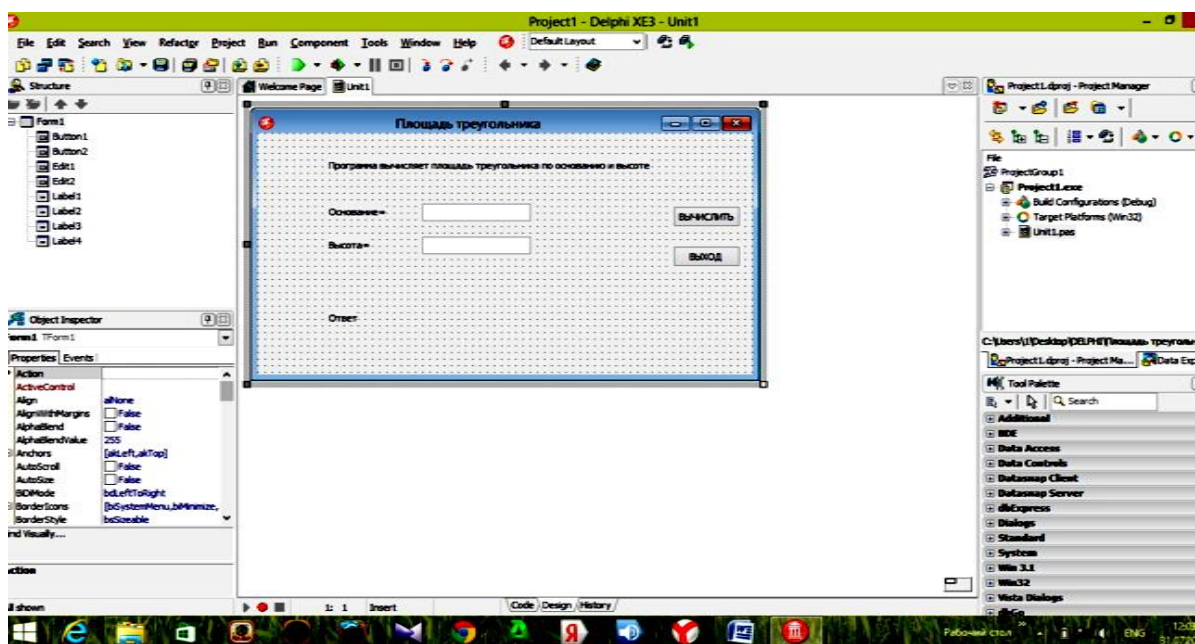


Рис. 1-10. Окно проекта «Площадь треугольника»

Для того чтобы сделать надпись «Программа вычисляет площадь треугольника по основанию и высоте» используем компонент Label. На палитре компонентов (в Delphi XE она находится справа внизу – Tool Palette) раскроем вкладку Standard. Найдем компонент Label, щелкнем по нему, перейдем на форму и щелкнем примерно в

том месте, где должна находиться эта надпись. В инспекторе объектов выделен объект Label1 и выведены его свойства. Мы выбираем Caption (Заголовок), стираем то, что там написано по умолчанию и пишем – Программа вычисляет площадь треугольника по основанию и высоте.

*Замечание.* Необходимые для осваивания данного учебного пособия компоненты и их основные свойства приведены в Приложении 1.

Аналогично мы делаем надписи «Основание=» (Label2), «Высота=» (Label3), «Ответ» (Label4). Обратите внимание! В дереве объектов проекта (Structure) появляются новые объекты.

Для организации ввода используем компонент Edit. Раскроем вкладку Standard и щелкнем по компоненту Edit, потом щелкнем по нужному месту (рядом с надписью «Основание=») на форме. У нас появилось поле, внутри которого написано «Edit1». Очистим это поле. В инспекторе объектов выделен объект Edit1 и выведены его свойства. Найдем свойство Text и удалим то, что там написано по умолчанию. Подобным образом создадим второе поле ввода (Edit2).

Запроектируем командные кнопки. На вкладке Standard щелкнем по компоненту Button, затем щелкнем на форме в том месте, где должна находиться командная кнопка «Вычислить». В инспекторе объектов для компонента Button1 ищем свойство Caption, стираем Button1 и пишем – ВЫЧИСЛИТЬ. Аналогично делаем кнопку ВЫХОД (Button2).

Даже в начале создания проекта его нужно сохранять, чтобы не потерять уже выполненную работу. В меню File или на панели кнопок быстрого доступа найдите команду **Save All** (сохранить всё) и щелкните по ней. Первый раз откроется окно (рис. 1–11), где нужно выбрать носитель или месторасположение проекта (по умолчанию выбрано Delphi).

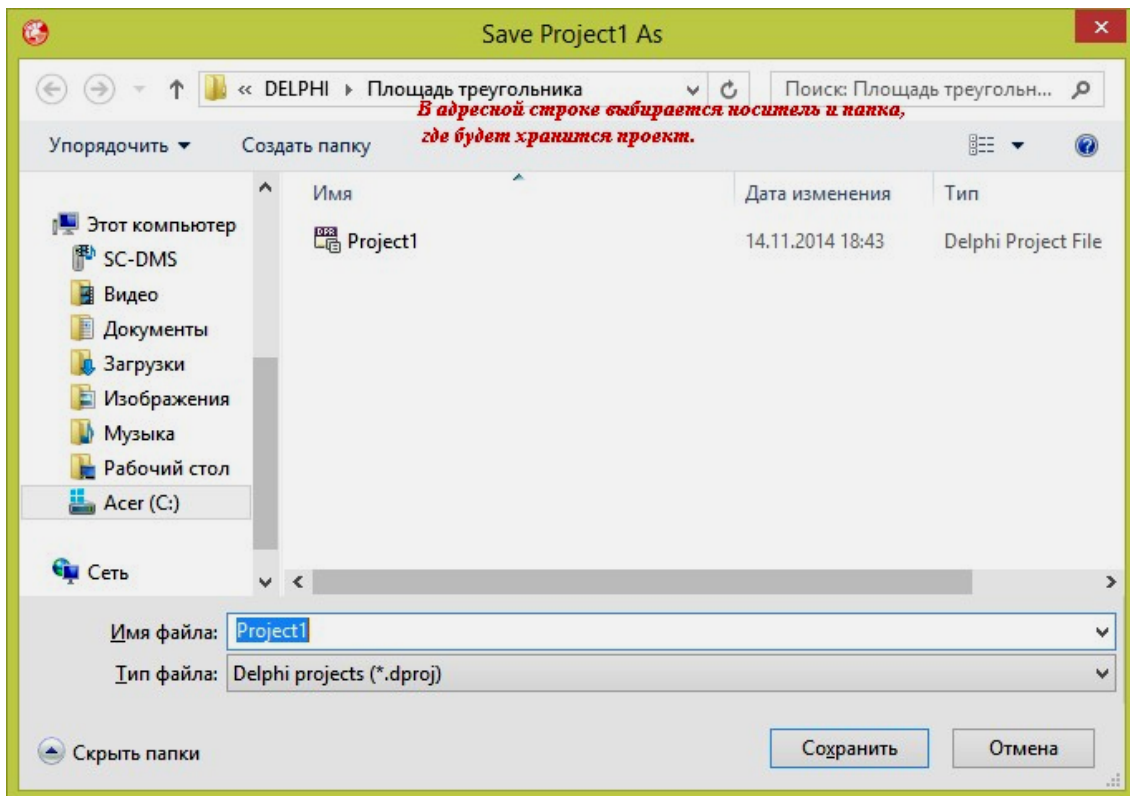


Рис.1–11. Окно сохранения проекта

Обязательно для сохраняемого проекта нужно создать новую папку (создается в том же окне). Дать имя папке, отражающее содержание проекта, например «Площадь треугольника». Открыть папку. Подтвердить сохранение файла Unit1, подтвердить сохранение файла Project1.

Стандартные имена файлов проекта Unit1, Project1 начинающим программиста рекомендуется оставить без изменения. Когда вы будете сохранять проект второй и последующие разы, проект автоматически сохраняется там же, где был сохранен первый раз.

Теперь перейдем собственно к программированию, реализующему алгоритм. Щелкните по вкладке Code и вы попадете в код будущей программы (Рис. 1–12). Причем часть кода сгенерировала сама система, когда вы проектировали форму.

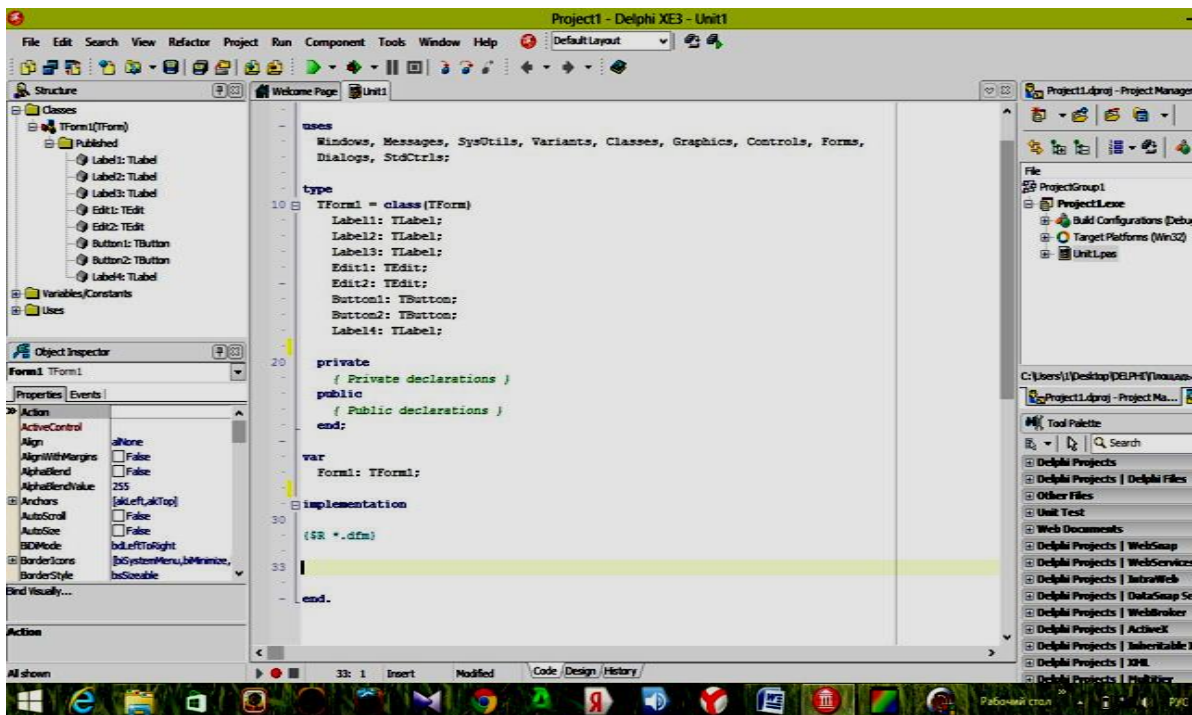


Рис. 1–12. Код будущего проекта «Площадь треугольника» после визуального проектирования окна интерфейса

Нам нужно связать с кнопкой **ВЫЧИСЛИТЬ** алгоритм программы, переведенный в инструкции языка Delphi. Перейдем в окно стартовой формы (рис. 1–10), щелкнув по вкладке Design внизу окна Delphi. С кнопками удобно связывать событие Click (щелчок). Для того чтобы создать событие Click для кнопки **ВЫЧИСЛИТЬ** (Button1), два раза щелкнем по изображению кнопки на форме. Откроется окно редактора кода, где будет сгенерирована системой заготовка будущей процедуры:

```

procedure TForm1.Button1Click(Sender: TObject);
begin

end;

```

Если в инспекторе объектов при выделенном объекте Button1, вы перейдете на вкладку Events (События), то там (рис. 1–13) напротив события OnClick будет записана кнопка Button1 вместе с событием Click. В тексте модуля автоматически появляется описание процедуры:

```

procedure Button1Click(Sender: TObject);

```

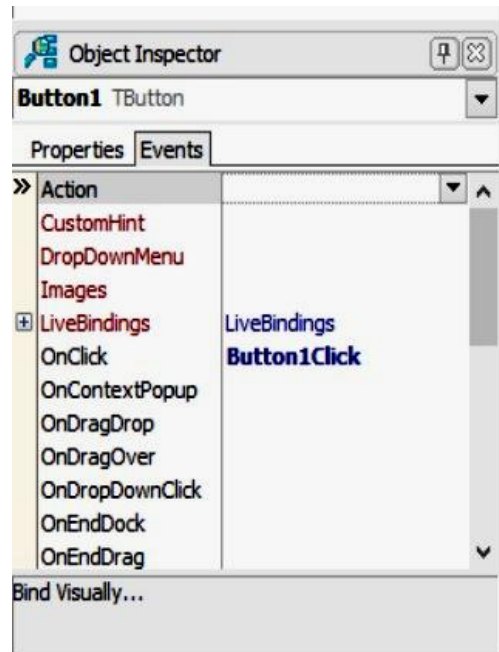


Рис. 1–13. Вкладка событий в инспекторе объектов

Что теперь нужно написать в тексте модуля Unit1? Пока четко следуйте рекомендациям учебного пособия. В следующих параграфах вы изучите инструкции языка и правила составления программы. Ниже приведен текст модуля проекта «Площадь треугольника». Указания авторов включены в текст. Курсивом выделены инструкции, сгенерированные самой системой. Итак, мы остановились на том, что система сгенерировала заготовку для будущей процедуры. Прежде чем писать код процедуры, загляните в раздел описаний, начинающийся со служебного слова `Var`, и впишите туда описание переменных, как сделано в приведенном ниже тексте модуля.

**Замечание.** Со знака `//` начинаются операторы однострочного комментария. Они носят пояснительный характер и не исполняются.

В заготовку процедуры, связанную с событием клика по кнопке `Button1`, между *begin* и *end* впишите инструкции ввода данных, вычисления по формулам и вывода ответа из текста модуля, приведенного ниже.

Создайте заготовку процедуры `ВЫХОД`, два раза щелкнув по изображению кнопки на форме, в которую запишите одну единственную инструкцию `Form1.Close`.

Текст модуля:

*unit Unit1;*

*interface*

*uses*

*Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
Dialogs, StdCtrls;*

*type*

*TForm1 = class(TForm)*

*Label1: TLabel;*

*Label2: TLabel;*

*Label3: TLabel;*

*Edit1: TEdit;*

*Edit2: TEdit;*

*Button1: TButton;*

*Button2: TButton;*

*Label4: TLabel;*

*procedure Button1Click(Sender: TObject);*

*procedure Button2Click(Sender: TObject);*

*private*

*{ Private declarations }*

*public*

*{ Public declarations }*

*end;*

*var*

*Form1: TForm1;*

*// описание переменных, участвующих в программе*

*a, h, S: Real;*

*implementation*

*{ \$R \*.dfm }*

*procedure TForm1.Button1Click(Sender: TObject);*

*begin*

*// ввод исходных данных из полей ввода Edit*

*a:=StrToFloat(Edit1.Text);*

*h:=StrToFloat(Edit2.Text);*



```
//вычисление площади треугольника
S:=1/2*a*h;
//перевод площади в гектары
S:=S/10000;
//вывод на форму ответа
Label4.Caption:='Площадь треугольника равна '+FloatToStr(S)+'ГА'
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
// закрытие формы – окна интерфейса
Form1.Close
end;
```

end.

Не забывайте сохранять свой проект.

Теперь нужно запустить проект на выполнение. О процессе компиляции и выполнении программы будет рассказано в следующем параграфе. А сейчас просто нажмите на зеленую стрелку (помечена цифрой 5 на рис. 1–1). Если вы сделали все точно по данному пособию, в вашем проекте ошибок не будет. Программа начнет выполняться. На экране появится окно, куда нужно ввести исходные данные, затем щелкнуть по кнопке вычислить и получить результат (Рис. 1–14).

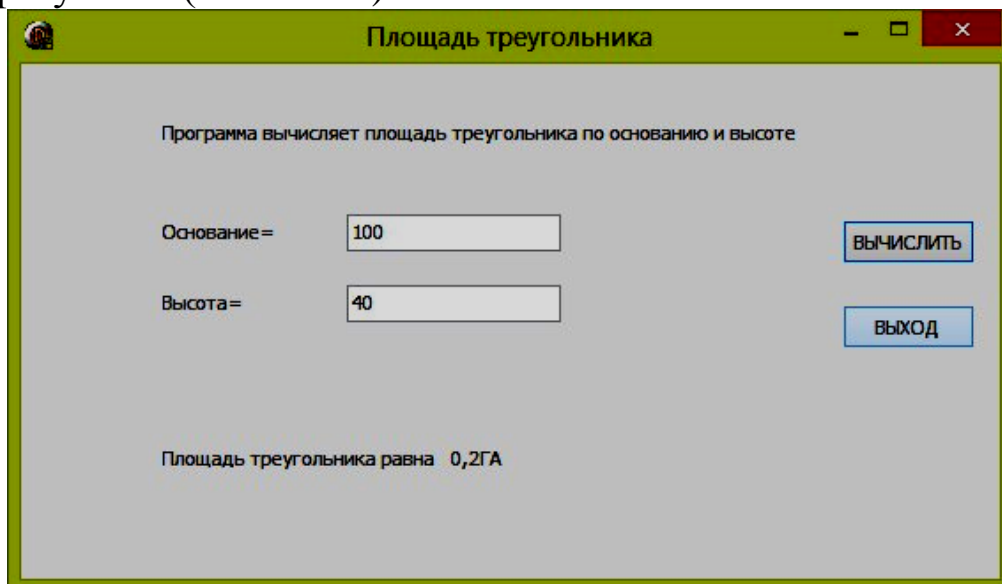


Рис. 1–14. Результат работы программы

**Рекомендуемый интернет–ресурс:**

**<http://www.youtube.com/watch?v=onRhDSpHZOA>**

### ***Вопросы для самоконтроля***

1. Какие основные части среды Delphi вы знаете?
2. Где находится главное меню?
3. Где находится дизайнер форм (стартовая форма)?
4. Где находится панель кнопок быстрого доступа?
5. Где находится палитра компонентов?
6. Какие компоненты вы знаете на вкладке Standard?
7. Где находится инспектор объектов?
8. Что можно изменять с помощью инспектора объектов?
9. Как сохранить проект?
10. Как запустить программу на выполнение?
11. Как запроектировать командную кнопку?
12. Как запроектировать поле для ввода исходных данных?
13. Как запроектировать поле для вывода результатов работы программы?
14. Где записывается код модуля программы?
15. Что нужно сделать, чтобы записать инструкции, которые выполняются при щелчке по кнопке ВЫЧИСЛИТЬ в программе «Площадь треугольника»?
16. Для чего предназначена кнопка ВЫХОД в программе «Площадь треугольника»?

## **Глава 2. Разработка линейных программ**

***В данной главе рассматриваются следующие вопросы:*** понятия алгоритм и программа; этапы разработки программы; типы данных языка Delphi, переменные и константы, оператор присваивания; некоторые функции преобразования типов, структура процедуры обработки события, создание и сохранение простейшего проекта, компиляция, запуск на выполнение.

### **Теория**

#### **Алгоритм и программа**

***Алгоритм*** – это точный порядок выполнения операций, преобразующих исходные данные в нужный результат. Алгоритм дол-

жен обладать определенными свойствами: дискретностью, однозначностью, массовостью, результативностью.

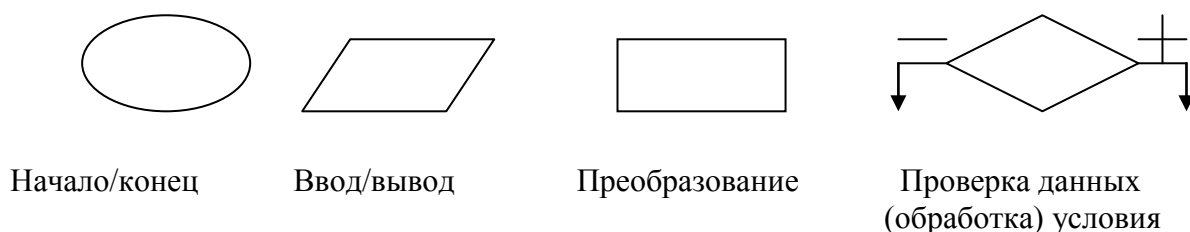
**Дискретность** – предполагает, что алгоритм состоит из отдельных операций. Работа операций, которые зависят от других операций, должна выполняться только после завершения операций, от которых они зависят. Независимые операции могут выполняться в любом порядке, в том числе параллельно.

**Однозначность** – это единственность толкования правил и порядка выполнения операций.

**Массовость** – это возможность применения алгоритма для решения класса задач, что предполагает его правильную работу при меняющихся в заданных пределах значениях исходных данных.

**Результативность** – выполнение алгоритма должно приводить к получению нужного результата за конечное число шагов.

Алгоритм решения задачи может быть представлен в виде словесного описания или графически в виде блок-схемы. В блок-схемах используются следующие специальные символы:



Программа – это последовательность команд описывающих алгоритм на языке программирования.

### Этапы разработки программы

**1. Определение требований к программе.** На этом этапе подробно описывается исходная информация, формулируются требования к результату, описывается поведение программы в особых случаях. Разрабатываются диалоговые окна, обеспечивающие взаимодействие пользователя и программы.

**2. Разработка алгоритма.** Определяется последовательность действий, которые надо выполнить для получения результата. Результатом этапа разработки алгоритма является подробное словесное описание алгоритма или его блок-схема.

**3. Кодирование.** Алгоритм записывается на выбранном языке программирования.

4. *Отладка* – это процесс поиска и устранения синтаксические ошибки (ошибок в тексте), который выполняется в процессе компиляции программы.

5. *Тестирование*. На этом этапе проверяют, как ведет себя программа на разных наборах исходных данных, в том числе и на заведомо неверных. Такая проверка выполняется с помощью заранее подготовленных тестовых задач с известными ответами с целью выявления алгоритмических и логических ошибок в программе.

## Типы данных языка Delphi

*Delphi* строгий язык, требующий обязательного описания типов всех данных, используемых в программе. Классификация типов данных Delphi разделяет все типы на: **простой, строковый, структурный, указательный, процедурный, вариантный**. Мы рассмотрим здесь только те типы данных, которые необходимы для программирования на начальном уровне.

В *Delphi* **простой** тип данных разбит на две группы: **порядковые**, представляющие данные разных объемов, которыми процессор может легко манипулировать, и **действительные**, представляющие действительные числа. В **порядковых** типах информация представляется в виде отдельных элементов. В *Delphi* определены три стандартных порядковых типа (**целые, символьные и булевы**) и два типа, определяемых пользователем (**перечисления и поддиапазоны**).

Все значения любого порядкового типа образуют упорядоченную последовательность, и значение переменной порядкового типа определяется ее местом в этой последовательности. За исключением переменных целых типов, значения которых могут быть как положительными, так и отрицательными, первый элемент любого порядкового типа имеет номер 0, второй элемент – номер 1 и т.д.

### *Целые типы*

В переменных целых типов информация представляется в виде целых чисел, т.е. чисел, не имеющих дробной части.

Таблица 2.1. Целые типы

Тип	Диапазон значений	Физический формат
Byte	0..255	1 байт, без знака
Word	0..65 535	2 байта, без знака
SmallInt, Int16	-32 768..32 767	2 байта, со знаком
LongInt, Integer, Int32	-2 147 483 648.. 2 147 483 647	4 байта, со знаком
Int64	$-2^{63}..2^{63}-1$	8 байт, со знаком

В Delphi XE типы Integer, Int32 аналогичны по размеру типу LongInt.

### *Символьный тип*

Delphi поддерживает универсальный символьный тип – **Char**. В Delphi 6, 7 Char соответствует типу данных AnsiChar. В Delphi XE тип char аналогичен типу WideChar. В кодировке Unicode один символ может занимать до 4 байт.

### *Булевый тип*

В Delphi информация о чем-либо, что можно представить как ИСТИНА (True) или ЛОЖЬ (False), хранится в переменных булевых типов. Всего таких типов четыре, но для Delphi предпочтителен тип **Boolean**, остальные определены для совместимости с другими языками программирования и операционными системами.

Булевый тип определен в Delphi так, как будто он объявлен следующим образом:

Type Boolean = (False, True);

Переменным типа Boolean можно присваивать только значения True (истина) и False (ложь).

С помощью типа Boolean в Delphi выполняются сравнения.

### *Перечислимые типы*

Type enum type = (first value, value2, value3, last value);

Обычно данные перечислимых типов содержат дискретные значения, представляемые не числами, а именами. Тип Boolean - простейший перечислимый тип в Delphi.

Большинство перечислимых типов – это просто списки уникальных имен или идентификаторов, зарезервированных с конкретной целью. Например, можно создать тип MyColor (мой

цвет) со значениями myRed, myGreen и myBlue (мой красный, мой зеленый, мой синий). Это делается совсем просто:

Type MyColor = (myRed, myGreen, myBlue);

В этой строке объявлены четыре новых идентификатора: MyColor, myRed, myGreen и myBlue. Идентификатором MyColor обозначен порядковый тип, следовательно, в синтаксисе Delphi можно применять этот идентификатор везде, где разрешены перечислимые типы. Остальные три идентификатора - это значения типа MyColor. Подобно символьным и булевым типам перечислимые не являются числами, и использовать их наподобие чисел не имеет смысла. Однако, перечислимые типы относятся к порядковым, так что значения любого такого типа упорядочены. Идентификаторам в списке присваиваются в качестве порядковых номеров последовательные числа. Первому имени присваивается порядковый номер 0, второму – 1 и т.д.

### *Действительные типы*

В переменных действительных типов содержатся числа, состоящие из целой и дробной частей. В Delphi определено шесть действительных типов. Все типы могут представлять число 0, однако они различаются пороговым (минимальным положительным) и максимальным значениями, которые могут представлять, а также точностью (количеством значащих цифр) и объемом. Действительные типы описываются в табл. 1..

Таблица 2.2. **Действительные типы**

Тип	Диапазон значений	Количество значащих цифр	Объем (байт)
Single	1.5E-45..3.4E38	7-8	4
Real48	2.9E-39..1.7E38	11-12	6
Double, Real	5.0E-324..1.7E308	15-16	8
Comp	-2E63+1..2E63-1	19-20	8
Currency	-922337203685477.5808... ..922337203685477.5807	19-20	8
Extended	3.6E-4932 ..1.1E4932	19-20	10

### ***Строковый тип***

Строковый тип предназначен для хранения строки символов. В Delphi для обозначения строкового типа допускается, пришедшее из Pascal, название **String**. Тип String – это статически размещаемые в памяти компьютера строки. Первый элемент задает длину строки, которая может принимать значения от 0 до 255 символов.

**Замечание.** Для выполнения заданий из данного учебного пособия можно ограничиться применением следующих типов данных: Real, Integer, Boolean, String, Char.

### ***Переменные и константы***

**Переменная** – это именованная ячейка памяти компьютера, где находятся данные, которые могут изменяться при работе программы. Чтобы программа могла обратиться к переменной (получить данные для расчета, запомнить результат вычислений), переменная должна иметь имя (**идентификатор**). Имена переменных составляют из последовательности букв латинского алфавита, цифр и некоторых специальных символов. Первой в имени обязательно должна быть буква. Пробел в имени переменной использовать нельзя. Прописные и строчные буквы в именах переменных компилятором Delphi не различаются. Имена переменных желательно задавать так, чтобы они соответствовали логическому смыслу переменной. Например: Summa, x1, skidka. Все переменные и их типы должны быть описаны в разделе описания переменных (Var) и в программе используются уже их имена (идентификаторы). Например,

```
Var Summa : Integer;  
    Skidka : Real;  
    Flag   : Boolean;
```

**Константа** – это ячейка памяти компьютера, где находятся данные, которые не могут изменяться при работе программы.

**Константы** подразделяются на обычные и именованные. Обычная константа представляет собой целое или дробное число, строку символов или отдельный символ, логическое значение. Числовые константы могут выглядеть в программе так:

0.001

3.14

-347

0

-1.2456000000E+06 // запись в форме с плавающей точкой.

Строковые и символьные константы заключаются в апострофы. Примеры строковых и символьных констант:

'Язык программирования Delphi'

'3,14'

'e'

Логическая константа может принимать только одно значение из двух: ложь (False) и истина (True).

Именованная константа описывается в разделе описания констант (Const) и в программе используется уже ее имя. Например,

```
Const Bound = 10;
```

```
    Flag = True;
```

```
    Pi = 3.1415926;
```

```
    W = 'WORD';
```

## Оператор присваивания

В общем виде оператор присваивания выглядит так:

Имя переменной := Выражение;

В результате выполнения оператора присваивания переменная, стоящая в левой части, получает значение выражения, стоящего справа (часто формула).

Пример:

```
Skidka := 12.5;
```

```
Flag := False;
```

```
S := 1/2*a*b;
```

Выражение состоит из операндов и операторов. Операндами обычно бывают: переменные, константы, функции. Операторы находятся между операндами и обозначают действия:

+ сложение

- вычитание

\* умножение

/ деление



DIV деление нацело  
MOD остаток от деления нацело  
Примеры выражений:  
 $A+B/C+0.1*C$   
Summa\*0.85  
A MOD 100

## Структура программы на языке Delphi

Исходный текст программы состоит из последовательности строк, в которой каждая строка может начинаться с любой позиции. Структурно программа состоит из заголовка и блока. *Заголовок* находится в начале программы и имеет вид:

Program <Имя программы>;

*Блок* состоит из двух частей: описательной и исполнительной. В *описательной части* содержится описание элементов программы, в *исполнительной части* указываются действия с различными элементами программы, позволяющие получить требуемый результат.

В общем случае *описательная часть* состоит из следующих разделов:

- подключения модулей;
- объявления меток;
- объявления констант;
- описания типов данных;
- объявления переменных;
- описания процедур и функций.

В конце каждого из указанных разделов указывается точка с запятой.

*Структуру программы* в общем случае можно представить следующим образом:

```
Program <Имя программы>;  
Uses <Список модулей>;  
Label <Список меток>;  
Const <Список констант>;  
Type <Описание типов>;  
Var <Объявление переменных>;  
<Описание процедур>;
```

```
<Описакие функций>;  
Begin  
<операторы>;  
End;  
End.
```

В структуре конкретной программы любой из разделов описания и объявления может отсутствовать. Разделы описаний и объявлений, кроме раздела подключения модулей, который располагается сразу после заголовка программы, могут встречаться в программе произвольное число раз и следовать в произвольном порядке. При этом все описания и объявления элементов программы должны быть сделаны до того, как они будут использованы. Рассмотрим отдельные разделы программы.

Раздел *подключения модулей* состоит из зарезервированного слова `Uses` и списка имен подключаемых стандартных и пользовательских библиотечных модулей. Формат этого раздела:

```
Uses <Имя1>, <Имя2>, ... , <ИмяN>,-
```

**Пример.** Подключение модулей.

```
Uses Crt, Dos, MyLib;
```

Раздел *объявления меток* начинается зарезервированным словом `Label`, за которым следуют имена меток, разделенные запятыми. Формат этого раздела:

```
Label <Имя1>, <Имя2>, ... , <ИмяN>;
```

**Пример.** Объявление меток.

```
Label metka1, metka2, 10, 567;
```

В разделе *объявления констант* производится присваивание идентификаторам именованных констант их постоянных значений. Раздел начинается ключевым словом `const`, за которым следует ряд конструкций, присваивающих константам значения. Эти конструкции представляют собой имя константы и выражение, значение которого присваивается константе. Имя константы отделено от выражения знаком равенства, в конце конструкции ставится точка с запятой. Формат этого раздела:

```
Const <Идентификатор1> = <Выражение>;  
<ИдентификаторN> = <Выражение>;
```

**Пример.** Объявление констант.

```
Const st1 = 'WORD'; ch = '5'; n34 = 45.8;
```

Тип константы распознается компилятором автоматически на основании типа выражения.

В Delphi имеется большое количество констант, которые можно использовать без их предварительного объявления, например, False и True.

В разделе *описания типов* описываются типы данных пользователя.

Этот раздел не является обязательным, и типы могут быть описаны неявно в разделе объявления переменных. Раздел описания типов начинается ключевым словом Type, за которым располагаются имена типов и их описания, разделенные знаком равенства. Каждое имя типа и его описание отделяется точкой с запятой. Формат раздела:

```
Type <Имя типа1> = <Описание типа>;  
<Имя типаN> = <Описание типа>;
```

**Пример.** Описание типов.

```
Type char2 = ('a' .. 'z') ;  
NumberArray = array[1 .. 100] of real;  
Month = 1 .. 12;
```

В Delphi имеется много *стандартных* типов, не требующих предварительного описания, например, Real, Integer, Char или Boolean.

**Каждая переменная программы должна быть объявлена.** Объявление обязательно предшествует использованию переменной. Раздел *объявления переменных* начинается с ключевого слова Var, после которого через запятые перечисляются имена переменных и через двоеточие их тип. Формат раздела:

```
Var <Идентификаторы> : <Тип>;  
<Идентификаторы> : <Тип>;
```

**Пример.** Объявление переменных.

```
Var a, bhg, u7: real;  
symbol: char;  
n1,n2: integer;
```

**Подпрограммой** называют логически законченную и специальным образом оформленную часть программы, которая по ее имени может вызываться для выполнения из других точек программы неограниченное число раз. Подпрограммы в Delphi разделяются на два вида: *процедуры* и *функции*. Каждая подпрограмма представляет собой блок и должна быть определена в разделе *описания процедур и функций*. Описание процедур и функций рассматривается ниже.

Раздел *операторов* начинается с ключевого слова Begin, после которого следуют операторы языка, разделенные точкой с запятой. Завершает этот раздел ключевое слово End, после которого указывается точка с запятой. Формат раздела:

```
Begin  
  <Оператор1>;  
  <ОператорN>;  
End;
```

Вся программа (проект) заканчивается ключевым словом End, после которого ставится точка.

### **Некоторые функции преобразования типов**

- IntToStr (Value: integer): string – преобразование значения целочисленного выражения value в строку;
- StrToInt (s: string): integer – преобразование строки s в целое число;
- FloatToStr(Value: Extended): string – преобразование значения вещественного выражения value в строку;
- StrToFloat (s: string): Extended – преобразование строки s в вещественное число.

### **Структура проекта**

Создаваемое в среде Delphi приложение состоит из нескольких элементов, объединенных в проект. В состав проекта входят следующие элементы (в скобках указаны расширения имен файлов):

- головной файл с кодом проекта (DPR);
- описания форм (DFM);
- модули (PAS);
- параметры проекта (DOF);

- описание ресурсов (RES);
- исполняемый файл (EXE).

Если открыть созданный выше проект «Площадь треугольника», все эти файлы будут видны (Рис. 2-1).

Delphi XE создает в проекте файл с расширением DPROJ, файл в формате расширяемого языка разметки, содержащий основные конфигурационные и справочные сведения о проекте. Это главный файл проекта.

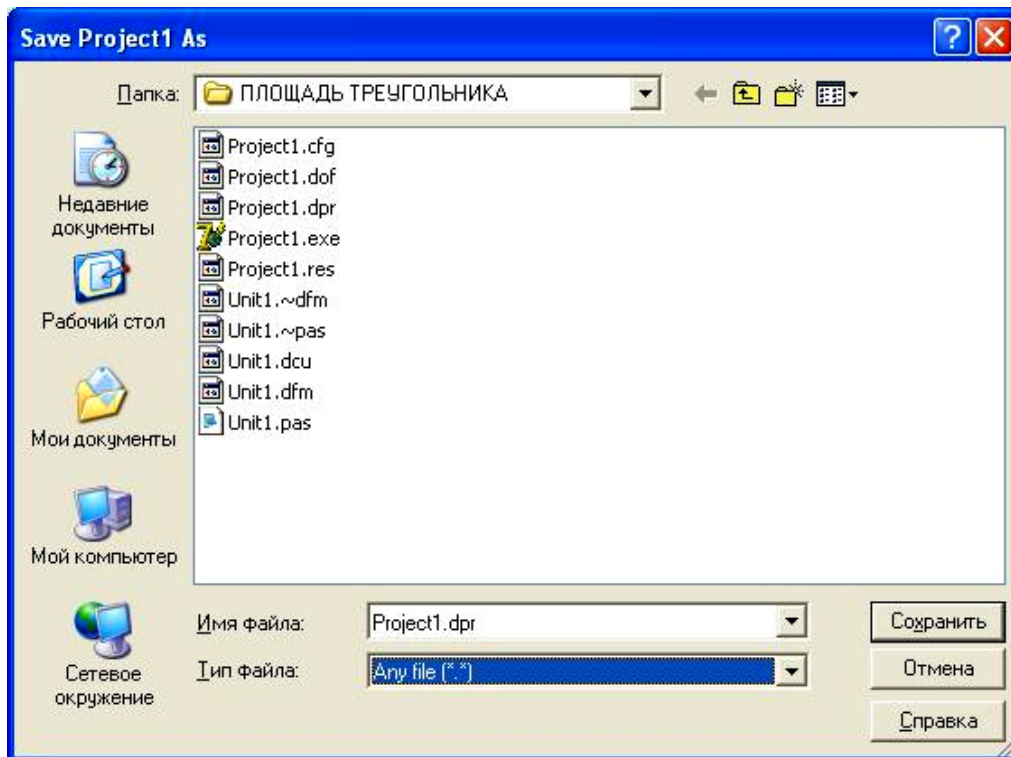


Рис. 2–1. Содержание проекта «Площадь треугольника»

В зависимости от сложности проекта могут присутствовать и другие файлы.

Любой проект всегда содержит главный модуль (файл с расширением .dpr). Помимо главного модуля, любой проект включает в себя как минимум один модуль формы (файл с расширением .pas). Сложные проекты состоят из нескольких модулей. Главный модуль можно увидеть, выбрав в главном меню команду Project→ View Source. Главный модуль программы «Бак для летнего душа», которая будет рассмотрена ниже в этой главе, выглядит так:

```
program Project1;
```

```
uses
```

```

Forms,
Unit1 in 'Unit1.pas' {Form1 };

{$R *.res}

begin
Application.Initialize;
Application.CreateForm(TForm1, Form1);
Application.Run;
end.

```

Текст программы начинается зарезервированным словом **program** и заканчивается словом **end** с точкой за ним.

После слова **uses** указываются модули, которые используются в программе. Модуль Forms является уже известным Delphi, а модуль Unit1 – новым, ранее неизвестным, и поэтому система указывает также имя файла с текстом модуля (in 'Unit1.pas') и имя связанного с модулем файла описания формы {Form1}.

Тело программы начинается со слова **begin** (начать) и ограничивается **end** с точкой. В теле записываются операторы. Каждый из них обращается к одному из методов объекта Application. *Объект* – это специальным образом оформленный фрагмент программы, состоящий из данных и подпрограмм для их обработки. Данные называются *полями объекта*, а подпрограммы – методами. Назначение метода Application.Initialize – выполнить подпрограмму, имя которой размещается в системной переменной InitProc. Метод Application.CreateForm(TForm1, Form1); создает и показывает на экране окно главной формы. Метод Application.Run; получает и обрабатывает поток поступающих от Windows сообщений о действиях пользователя.

Файл проекта формируется самой средой Delphi и в большинстве случаев не подлежит редактированию.

### Сохранение проекта

Для каждого проекта рекомендуется создавать отдельную папку.

Для сохранения проекта можно из меню **File** выбрать команду **Save Project As**. Если проект сохраняется первый раз, то Delphi сначала предложит сохранить содержимое окна редактора кода, поэто-

му на экране появится окно **Save Unit1 As**. В этом окне надо указать носитель (диск, флешку), **создать новую папку**, предназначенную для файлов проекта, дать ей имя, отражающее содержание проекта, и открыть эту папку. Имя модуля Unit1, сгенерированное компьютером, можно не менять. После нажатия кнопки **Сохранить**, появляется следующее окно, в котором сгенерировано имя файла проекта Project1. Его тоже нужно **Сохранить**. Имена файлов модуля (pas-файл) и проекта (dpr – файл) должны быть разными. Начинающим советуют оставлять сгенерированные системой имена.

В последующие разы, проект автоматически сохраняется там же, где был сохранен первый раз.

Часто, для избегания путаницы, программисты пользуются командой **Save All** из меню **File** или кнопкой быстрого доступа



Особенно это важно при сохранении проектов, состоящих из нескольких форм и модулей.

Переписать проект, в другое место или под другим именем можно с помощью двух команд **Save Project As** (сохранить как) и **Save As** (сохранить как).

## Структура модуля

Начинается модуль со служебного слова Unit

Модуль состоит из следующих разделов: интерфейса, реализации и инициализации.

В раздел интерфейса (начинается словом interface) компилятору сообщается, какая часть является доступной для других модулей программы. В этом разделе подключаются библиотечные модули (после Uses) , используемые данным модулем. В нем находится сформированное Delphi описание формы, которое следует за словом type.

Раздел реализации открывается словом implementation и содержит объявления локальных переменных, процедур и функций, поддерживающих работу формы.

Начинается раздел реализации директивой {\$R \*.dfm}, указывающей компилятору, что в процессе генерации выполняемого файла надо использовать описание формы. Описание формы находится в

файле `dfm`, имя которого совпадает с именем модуля. Файл описания формы генерируется средой Delphi на основе внешнего вида формы.

За директивой `{R *.dfm}` следуют процедуры обработки событий для формы и ее компонентов. Сюда же программист может помещать другие процедуры и функции.

### Компиляция и выполнение проекта

В процессе компиляции проекта создается готовый к использованию файл, которым может быть приложение (EXE). Процесс компиляции состоит из двух шагов: непосредственно компиляции и компоновки. Запуск процесса компиляции выполняется по команде **Project** → **Compile** <Project1> (Проект → Компилировать <проект>) или нажатием комбинации клавиш <Ctrl>+<F9>. В этой команде содержится имя проекта, разработка которого выполняется в настоящий момент, первоначально это Project1.

При сохранении проекта под другим именем соответственно должно измениться имя проекта в команде меню Сохранить проект как (**Save Project As**).

Из проекта получается загрузочный модуль только в том случае, если в нем нет синтаксических ошибок. Первоначально практически любой проект содержит ошибки, которые программист должен исправить. На рис. 2–2 приведен вид экрана при компиляции программы, когда компилятор нашел ошибки. В центре окна в поле **Compiling** выводится сообщение **There are errors** и отображается информация о количестве синтаксических (поле **Errors**) и семантических (поле **Warnings**) ошибок, а также о количестве неточностей (поле **Hints**). Сами же сообщения об ошибках, предупреждения и подсказки отображаются в окне **Messages** (расположено в нижней части экрана). После того как окно **Compile** будет закрыто, в редакторе кода будет выделена строка, в которой находится первая (от начала файла) обнаруженная ошибка.



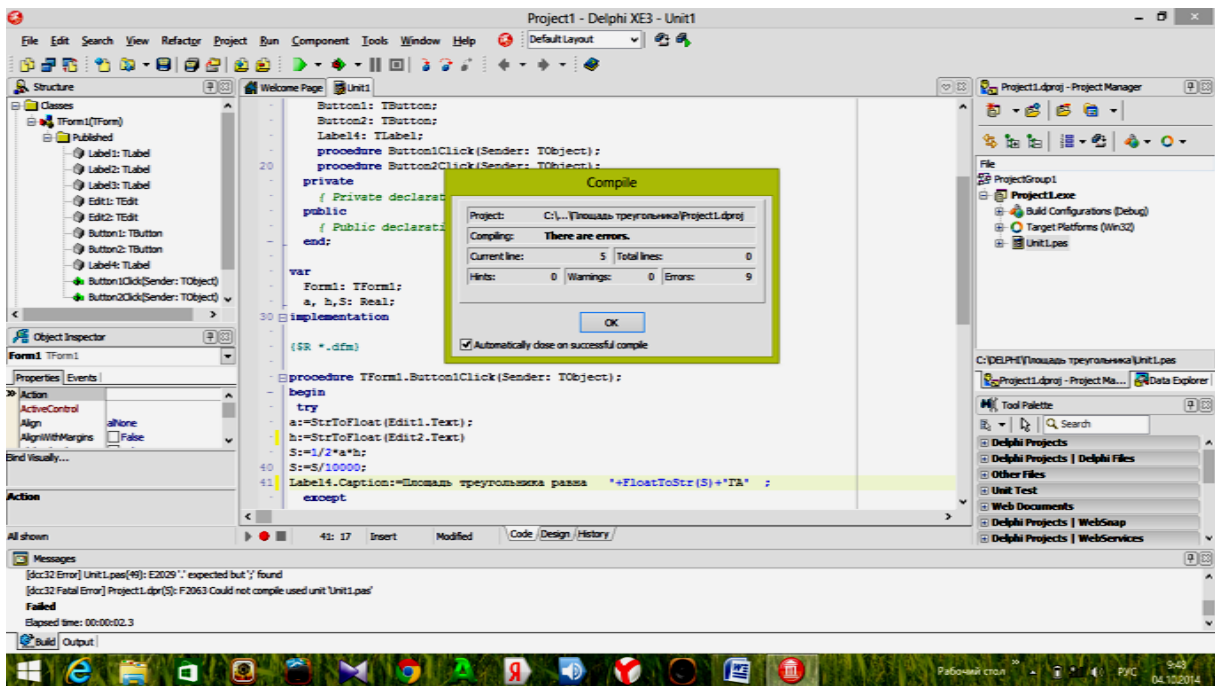


Рис. 2–2. Окно проекта Delphi после компиляции. Найденны ошибки

Выполнение проекта *из среды Delphi* осуществляется командой **Run** (Выполнить) или нажатием клавиши <F9>. При этом – созданное приложение начинает свою работу. Часто используют быструю кнопку на панели быстрых кнопок (на ней изображена зеленая стрелка). Если в файлы проекта вносились изменения, то предварительно выполняется компиляция проекта.

## Практика

### *Задача 2а*

Разработать проект для вычисления объема бака с заданными сторонами: a, b, c (в метрах) для душа на даче и определения количества материала, которое пойдет на изготовление такого бака.

Метод решения:

$V=abc$  – объем бака

$S=2(ab+bc+ca)$  – площадь поверхности бака (количество материала в квадратных метрах).

Составим алгоритм программы на естественном языке в виде пронумерованных пунктов.

1. Введем исходные данные – стороны бака (a, b, c) в память компьютера.

2. Вычислим объем бака по формуле, указанной в методе решения.

3. Выведем на экран подсчитанный объем бака  $V$  куб.м.

4. Вычислим площадь поверхности бака, которая равна необходимому количеству материала для строительства, по формуле, указанной в методе решения.

5. Выведем на экран необходимое количество материала  $S$  кв. м.

На этом алгоритм закончен. Приступим к визуальному конструированию интерфейса с пользователем в виде окна. В данном проекте предусмотрим пять командных кнопок: для ввода данных в память компьютера, для вычисления объёма бака, для вычисления необходимого количества материала, для очищения полей с данными и набора новых значений сторон и для завершения программы.

### **Компоненты:**

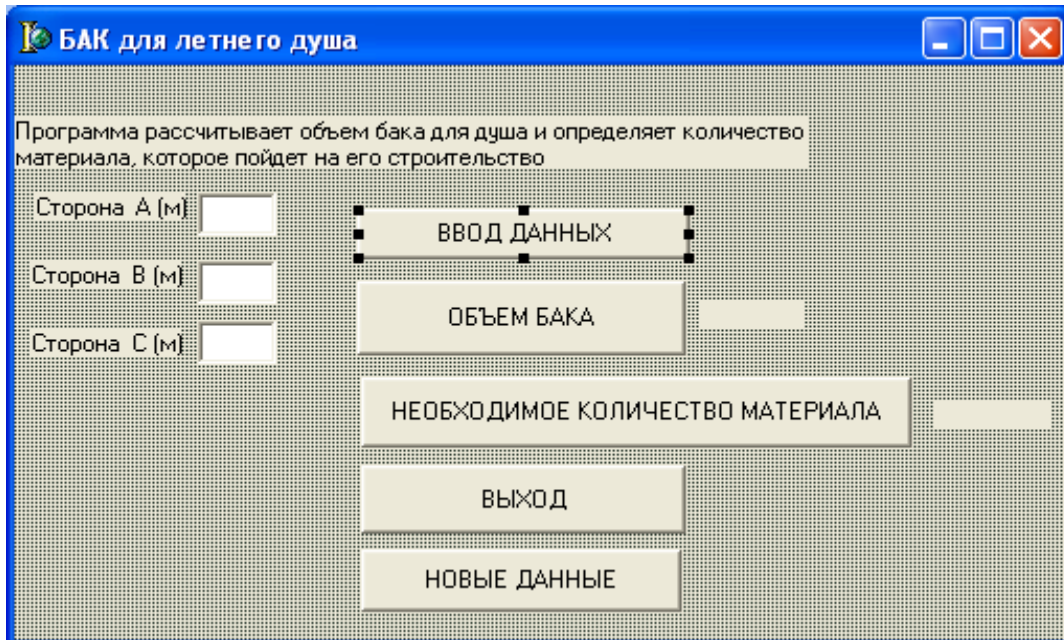
Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	БАК для летнего душа	Заголовок формы
Label1	Caption	Программа рассчитывает объем бака для душа и определяет количество материала, которое пойдет на его изготовление	Справочная информация для пользователя программы
Label2	Caption	Сторона А (м)	Подсказка пользователю
Label3	Caption	Сторона В (м)	Подсказка пользователю
Label4	Caption	Сторона С (м)	Подсказка пользователю
Label5	Caption	Должно быть очищено.	Поле для вывода объема
Label6	Caption	Должно быть очищено.	Поле для вывода количества материала
Edit1	Text	Должно быть очищено.	Поле для ввода стороны А
Edit2	Text	Должно быть очищено.	Поле для ввода стороны В
Edit3	Text	Должно быть очищено.	Поле для ввода стороны С
Button1	Caption	ВВОД ДАННЫХ	Кнопка для ввода данных
Button2	Caption	ОБЪЕМ БАКА	Кнопка для вычисления объема бака
Button3	Caption	НЕОБХОДИМОЕ КОЛИЧЕСТВО МАТЕРИАЛА	Кнопка для вычисления необходимого количества материала
Button4	Caption	ВЫХОД	Кнопка закрытия формы и выхода из программы
Button5	Caption	НОВЫЕ ДАННЫЕ	Кнопка для очистки полей ввода для новых данных

### **Переменные**

Обозначение в программе	Содержание	Тип
a	сторона А	вещественный
b	сторона В	вещественный

c	сторона С	вещественный
V	объем бака	вещественный
S	Площадь поверхности – количество материала	вещественный

## Проект формы



### ***Замечание***

*В текстах модулей в учебном пособии, описания и операторы, генерируемые самой системой, выделены курсивом.*

*В следующих темах в задачах приведены не все инструкции, генерируемые системой. Бездумное удаление инструкций, генерируемых самой системой, приводит к возникновению ошибок в проекте.*

### **Текст модуля**

*unit Unit1;*

*interface*

*uses*

*Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
Forms, Dialogs, StdCtrls;*

*type*

*TForm1 = class(TForm)*

*Label1: TLabel;*

*Label2: TLabel;*

*Label3: TLabel;*

*Label4: TLabel;*

```

Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Button1: TButton;
Button2: TButton;
Button3: TButton;
Label5: TLabel;
Label6: TLabel;
Button4: TButton;
procedure Button1Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
  a,b,c,V,S: real;
  {a, b, c – стороны бака; V – объем бака;
   S – площадь поверхности}
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
  // ВВОД ИЗ ПОЛЕЙ ФОРМЫ ЗНАЧЕНИЙ СТОРОН БАКА
  a:=StrToFloat(Edit1.Text);
  b:=StrToFloat(Edit2.Text);
  c:=StrToFloat(Edit3.Text);
  end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  // ВЫЧИСЛЕНИЕ ОБЪЕМА
  V:=a*b*c;
  // ВЫВОД ЗНАЧЕНИЯ ОБЪЕМА НА ФОРМУ
  Label5.Caption:=FloatToStrF(V,Ffixed,8,2)+' куб. м'
  end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  // ВЫЧИСЛЕНИЕ ПЛОЩАДИ ПОВЕРХНОСТИ БАКА

```

```

S:=2*(a*b+b*c+c*a);
// вывод значения поверхности бака на форму
Label6.Caption:=FloatToStrF(S,Ffixed,9,2)+' кв. м'
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
// закрытие формы
Form1.Close
end;
procedure TForm1.Button5Click(Sender: TObject);
begin
// очистка полей формы для ввода новых данных
Edit1.Clear;
Edit2.Clear;
Edit3.Clear;
Label5.Caption:=' ';
Label6.Caption:=' ';
end;
end.

```

## Результаты работы программы

БАК для летнего душа

Программа рассчитывает объем бака для душа и определяет количество материала, которое пойдет на его строительство

Сторона А (м)

Сторона В (м)

Сторона С (м)

ВВОД ДАННЫХ

ОБЪЕМ БАКА 2,93 куб. м

НЕОБХОДИМОЕ КОЛИЧЕСТВО МАТЕРИАЛА 12,30 кв.м

ВЫХОД

НОВЫЕ ДАННЫЕ

## Рекомендуемый интернет-ресурс

[http://www.youtube.com/watch?v=TL2obDWmF7o&index=2&list=P LDDb9D4A0E37B4953&src\\_vid=96N8dQJWEao&feature=iv&annotation\\_id=annotation\\_261495](http://www.youtube.com/watch?v=TL2obDWmF7o&index=2&list=P LDDb9D4A0E37B4953&src_vid=96N8dQJWEao&feature=iv&annotation_id=annotation_261495)

### ***Вопросы для самоконтроля***

1. Что такое алгоритм?
2. Дайте определение программе.
3. Какие этапы предусматривает разработка программы.
4. Дайте определение переменной.
5. Что могут хранить константы?
6. Какие данные можно хранить в переменной, описанной типом Integer?
7. Какие действительные типы вы знаете в языке Delphi?
8. Для хранения каких данных используется тип Char?
9. Для хранения каких данных используется тип String?
10. Какие значения может принимать переменная описанная типом Boolean?
11. Назначение оператора присваивания.
12. Какие арифметические операторы есть в языке Delphi?
13. Какую структуру в общем виде имеет программа, написанная на языке Delphi?
14. Какие файлы может содержать проект программы на языке Delphi?
15. Как сохранить, разработанный проект? Опишите последовательность действий.
16. Какие функции преобразования типов вы знаете?
17. Как откомпилировать и выполнить проект?
18. Для каких целей предназначен компонент Button3 в программе БАК?
19. Что нужно сделать, чтобы программа БАК прочитала в память компьютера исходные данные?
20. Что произойдет, если щелкнуть по кнопке **НОВЫЕ ДАННЫЕ**?

## *Задачи для самостоятельного решения*

### Группа А

№ 2.1 Дан радиус шара ( $R$ ) в метрах. Составить программу для вычисления площади полной поверхности:

$$S = 4\pi R^2$$

и объема:

$$V = 4\pi R^3 / 3.$$

№ 2.2 Известны значения радиуса шара ( $R$  в см.), хорды шарового сектора в сантиметрах ( $a$  в см.) и стрелы шарового сектора ( $h$  в см). Составить программу для вычисления площади полной поверхности шарового сектора:

$$S = \pi R(2h + a)$$

и объема шарового сектора:

$$V = 2\pi R^2 h / 3.$$

№ 2.3 Дан круговой прямой цилиндр. Известен радиус цилиндра  $R$  и высота  $h$  в метрах. Составить программу для вычисления площади боковой поверхности:

$$M = 2\pi R h,$$

площади полной поверхности:

$$S = 2\pi R(R + h),$$

объема:

$$V = \pi R^2 h.$$

№ 2.4 Дан усеченный круговой цилиндр. Известны: радиус основания цилиндра –  $R$  в метрах;  $h_1$  и  $h_2$  – высоты усеченного цилиндра (м). Написать программу для вычисления площади боковой поверхности:

$$M = \pi R(h_1 + h_2),$$

площади полной поверхности:

$$S = \pi R \left[ h_1 + h_2 + R + \sqrt{R^2 + \left( \frac{h_2 - h_1}{2} \right)^2} \right],$$

объема

$$V = \pi R^2 \frac{h_1 + h_2}{2}.$$

№ 2.5 Известны:  $r$  и  $R$  – внутренний и внешний радиусы кругового кольца в метрах; центральный угол  $\varphi$  (в градусах) час-

ти кольца. Составить программу для вычисления площади кругового кольца ( $S_1$ ) и площади части кольца ( $S_2$ ) с заданным центральным углом по формулам:

$$S_1 = \pi(R^2 - r^2), S_2 = \frac{\varphi\pi}{360}(R^2 - r^2).$$

№ 2.6 Составить программу для вычисления объема трехгранной усеченной пирамиды. Известны стороны оснований пирамиды:  $a_1, b_1, c_1, a_2, b_2, c_2$  и высота пирамиды  $h$ . Объем вычисляется по формуле:

$$V = h[S_1 + S_2 + \sqrt{S_1 S_2}] / 3,$$

где  $S_1$  – площадь нижнего основания;

$S_2$  – площадь верхнего основания.

Площади оснований рассчитать по формуле Герона.

№ 2.7 Написать программу для вычисления расстояния  $L$  между двумя населенными пунктами, если автомобиль со скоростью  $V$  км/час проезжает его за  $T$  часов.

Расстояние  $L$  между населенными пунктами равно  $L = V * T$ .

Предусмотреть запросы при вводе исходных данных:

Скорость автомобиля км/час:

Время проезда в часах:

Форма вывода результата:

Расстояние между населенными пунктами равно <значение  $L$ > км.

№ 2.8 Составить программу для вычисления силы тока  $I = q/t$ , где  $q$  – величина электрического заряда (измеряется в кулонах);  $t$  – время прохождения заряда через поперечное сечение проводника в секундах.

№ 2.9 Написать программу для определения объема усеченного конуса, с высотой  $H$  м и параллельными основаниями с радиусами  $R_1$  м и  $R_2$  м. В программе предусмотреть округление объема до  $0,001 \text{ м}^3$ . Формула для вычисления объема:

$$V = H / 3(S_1 + \sqrt{S_1 S_2} + S_2),$$

где

$$S_1 = \pi R_1^2, S_2 = \pi R_2^2.$$

Форма выдачи результата:

ОБЪЕМ КОНУСА= значение  $V$  КУБ.М.



№ 2.10 Дан обелиск. Нижнее и верхнее основание являются прямоугольниками, расположенными в параллельных плоскостях; противоположные боковые грани одинаково наклонены к основанию, но не пересекаются в одной точке. Даны  $a, b$  и  $a_1, b_1$  – стороны оснований,  $h$  – высота. Найти объем обелиска по формуле

$$V=h[ab+(a+a_1)(b+b_1)+a_1b_1]/6$$

Форма выдачи результата:

ОБЪЕМ ОБЕЛИСКА= значение  $V$  КУБ.М.

### Группа Б

№ 2.11 Составить программу для исследования величины поправки ( $\Delta S$ ) за редуцирование расстояния ( $S$  в метрах) при переходе с эллипсоида на плоскость в проекции Гаусса. Для этого рассчитать поправку:

$$\Delta S = S \frac{y^2}{2R^2}$$

для случаев :

- 1) линия располагается на осевом меридиане ( $y = 0$  км);
- 2) линия находится на краю шестиградусной зоны ( $y \approx 330$  км).

$R$  – радиус Земли.  $R \approx 6371,1$  км.

$y$  – ордината середины редуцируемого отрезка в км.

Поправку вычислить в метрах с точностью до 0,1 м.

Замечание.  $Y$  можно задавать в пределах  $0 \leq y \leq 330$ .

№ 2.12 Составить программу для вычисления поправки  $\Delta P$  в площадь за переход с поверхности шара на плоскость в проекции Гаусса:

$$\Delta P = P \frac{y^2}{R^2}$$

$P$  – площадь участка на поверхности шара (га).

$R$  – радиус Земли.  $R \approx 6371,1$  км.

$y$  – ордината середины редуцируемой площади (0–300 км. См. замечание к 1.11).

Поправку округлить до 0,01 га.

№ 2.13 Составить программу для расчета цены деления уровня по формуле:

$$\tau'' = \frac{(l_1 - l_2)206''}{nd},$$

где  $l_1 - l_2$  – разность отсчетов по рейке в мм;

$n$  – число делений, на которое сместился пузырек уровня;

$d$  – расстояние от нивелира до рейки в метрах.

Перевод мм в м в рабочей формуле учтен константой 206''.

Результат вывести с точностью до 0,01 секунды.

№ 2.14 Дана арифметическая прогрессия 1–го порядка. Известны: первый член  $a_1$  арифметической прогрессии,  $d$  – разность арифметической прогрессии. Составить программу для вычисления  $n$ –го члена прогрессии

$$a_n = a_1 + d(n-1)$$

и суммы  $n$ –членов прогрессии:

$$S_n = (a_1 + a_n)n/2.$$

№ 2.15 Дана геометрическая прогрессия. Известны: первый член геометрической прогрессии  $b_1$ ,  $q$  ( $q \neq 1$ ) – знаменатель геометрической прогрессии. Составить программу для вычисления суммы геометрической прогрессии:

$$S_n = b_1 \frac{q^n - 1}{q - 1}$$

№ 2.16 Написать программу для вычисления поправки за кривизну Земли в длину измеренной линии:

$$k = \frac{S^2}{2R},$$

где  $S$  – длина линии в метрах,  $R$  – радиус Земли.  $R \approx 6371,1$  км. Поправку вычислить в сантиметрах и округлить до 0,1 см.

Перевод в единые единицы измерения предусмотреть в формуле самостоятельно.

№ 2.17 Если известны координаты двух точек  $X_1Y_1$  и  $X_2Y_2$  (рис. 7–1), то координаты  $X, Y$  точки, делящей отрезок 1–2 в отношении  $\frac{m}{n}$ , можно определить по формулам:

$$X = \frac{nX_1 + mX_2}{m + n}, Y = \frac{nY_1 + mY_2}{m + n}.$$

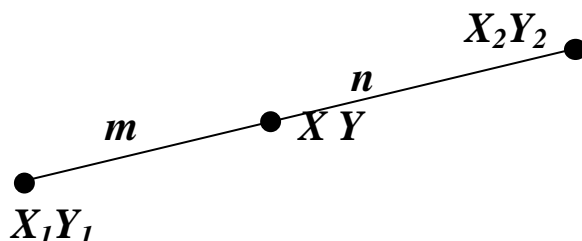


Рис. 2–3

Написать программу для определения  $X, Y$ . Значения  $X_1, Y_1, X_2, Y_2$  вводить в метрах. Координаты  $X, Y$  выводить в метрах с точностью до 0,1м.

Форма выдачи результата:

КООРДИНАТЫ ДЕЛЯЩЕЙ ТОЧКИ:

$X$  = значение  $X, Y$  = значение  $Y$ .

№ 2.18 Написать программу для вычисления поправки превышения за кривизну Земли и рефракцию:

$$f = 0,42 \frac{S_i^2}{R},$$

$f$  – поправка измеряется в метрах. Рефракция – преломление светового луча в атмосфере (поправку вычисляют для расстояний более 300 м);  $R$  – радиус Земли ( $R \approx 6371,1$  км);  $S_i$  – длина линии в метрах между точками, где измеряется превышение.

Поправку округлять до 0.01 м.

Форма вывода результата: ПОПРАВКА= значение (м).

№ 2.19 Написать программу для расчета общих годовых издержек на 1 га угодий в зависимости от площади территории  $P$  и пространственного расположения земель и хозяйственного центра:

$$Z = \frac{2260}{P} + 1,48K_1K_2\sqrt{P} + 67,4,$$

где  $K_1$  – коэффициент, характеризующий конфигурацию земельной площади и положение хозяйственного центра;

$K_2$  – коэффициент, показывающий, во сколько раз путь по дорогам к данному участку длиннее, чем путь по прямой.

Значение  $Z$  округлить до 0,01.

Форма вывода результата:

ГОДОВЫЕ ИЗДЕРЖКИ= значение  $Z P/ГА$ .

№ 2.20 Составить программу для вычисления площади сектора  $S_1$  и площади сегмента  $S_2$  если известны  $r$  – радиус круга,  $\alpha$  – центральный угол (в градусах) по формулам:

$$S_2 = r^2(\pi\alpha/180 - \sin(\pi\alpha/180))/2;$$

$$S_1 = r^2\pi\alpha/360.$$

Форма вывода результатов:

ПЛОЩАДЬ СЕГМЕНТА КРУГА = значение  $S_1$

ПЛОЩАДЬ СЕКТОРА КРУГА = значение  $S_2$ .

Радиус круга задан в метрах. Площади определить с точностью до 0,1 м<sup>2</sup>.

### Глава 3. Стандартные и библиотечные функции

*В данном параграфе рассматриваются следующие вопросы:* некоторые стандартные библиотечные функции (их библиотеки подключены автоматически), библиотечные функции (некоторые математические функции), функции даты, использование окна ввода, процедура и функции вывода окон сообщений пользователю.

#### Теория

##### *Некоторые стандартные библиотечные функции*

Пользоваться стандартными функциями можно без подключения библиотек. Библиотеки, где они описаны, подключаются автоматически.

Таблица 3.1 Стандартные математические функции

Функция	Возвращаемое значение
Abs (x)	абсолютная величина x
Arctan(x)	арктангенс x (результат в радианах)
Cos (x)	косинус x (x выражается в радианах, а не в градусах)
Dec(n)	аналогична оператору n:=n-1
Exp (x)	экспоненциальная функция от x ( $e^x$ )
Frac(x)	дробная часть x
Inc(n)	аналогична оператору n:=n+1
Int (x)	целая часть x. Несмотря на название, возвращает действительное значение (с плавающей запятой), т.е. просто устанавливает нуль в дробной части
Ln (x)	натуральный логарифм от x
Odd (X)	Возвращает булево True (истина), если X – нечетное целое, и False (ложь) – в противном случае
pi	Константа $\pi$
Round (x)	Возвращает ближайшее к x целое значение. Условие «ближайшее к x» не работает, если верхнее и нижнее значения оказываются равноудаленными (например, если дробная часть точно равна 0,5). В этих случаях Delphi перекладывает решение на операционную систему. Обычно процессоры Intel решают эту задачу в соответствии с рекомендацией IEEE округлять в сторону ближайшего четного целого числа.
Sin(x)	синус x (x выражается в радианах)
Sqr(x)	квадрат x, т.е. $X*X$
Sqrt (x)	квадратный корень от x
Trunc (x)	целая часть x. В отличие от Int, возвращающей действительное значение, Trunc возвращает целое
X Div Y	Возвращает целую часть частного деления X на Y
X Mod Y	Возвращает остаток частного деления X на Y

**Замечание.** В библиотеках Delphi, которые автоматически подключаются к модулю при создании формы, нет арифметического действия – возведение в степень. Поэтому для возведения числа  $a$  в степень  $x$  можно воспользоваться следующими функциями:

$a^x$  соответствует  $exp(x*ln(a))$ .

См. также раздел Некоторые математические функции.

Таблица 3.2 **Функции преобразования**

<b>Функция</b>	<b>Возвращаемое значение</b>
Chr(n)	символ с номером n.
DateToStr (Date)	результат преобразования значения даты в выражении Date в строку.
FloatToStr(n)	строка, являющаяся изображением значения вещественного n. Различие между функциями описано ниже.
FloatToStrF(n,f,l,m)	строка, являющаяся изображением значения вещественного n с форматом. Различие между функциями описано ниже.
IntToStr(n)	строка, являющаяся изображением значения целого n.
Pred(n)	возвращает предыдущее значение n.
StrToDate(S)	результат преобразования строки S в дату.
StrToFloat(s)	вещественное число, изображаемое строкой s.
StrToInt(s)	целое число, изображаемое строкой s.
StrToTime(S)	результат преобразования строки S во время.
TimeToStr(Time)	результат преобразования значения времени в выражении Time в строку.

Функция *FloatToStrF(n,f,l,m)* обеспечивает возможность форматного вывода вещественных чисел. Её аргументы имеют следующий смысл:

- n* – преобразуемое значение;
- f* – формат (способ изображения);
- l* – точность (общее количество цифр);
- m* – количество цифр после десятичной точки.

Поле «формат» может принимать следующие значения

**ffGeneral** – Общий числовой формат. Значение преобразовывается в изображение десятичного числа либо с фиксированной точкой, либо записанного в научном формате. Лишние нули удаляются из результирующей строки, и в случае необходимости вставляется десятичная точка. Запись числа с фиксированной точкой используется, если количество цифр в числе не превышает общее количество цифр, заданное пользователем, а само число не меньше 0.00001. В противном случае используется научный формат.

**ffExponent** – Научный формат. Число преобразовывается в строку вида «*-d.ddd...E+dddd*». Если число отрицательное, строка начинается знаком минус. Количество цифр в показателе экспоненты от 0 до 4.

***ffFixed*** – Формат с фиксированной точкой. Число преобразуется в строку вида: «*-ddd.ddd...*». Если число отрицательное, строка начинается знаком минус. Количество цифр после десятичной точки не более 18. Если количество цифр превышает величину, указанную пользователем, изображение числа преобразуется в научный формат.

***ffNumber*** – Числовой формат возвращает строку в виде «*-d,ddd,ddd.ddd...*». Он отличается от формата с фиксированной точкой только разделителями групп разрядов.

***ffCurrency*** – Денежный формат. Преобразует результат и представляет его в виде суммы денег.

### ***Некоторые строковые функции***

Таблица 3.3 **Некоторые строковые функции**

<b>Функция</b>	<b>Описание</b>
Concat(s1, s2, s3)	Возвращает последовательное соединение строк. Эквивалентна оператору s1+s2+s3
Copy(s, pos, len)	Возвращает подстроку длиной максимум len символов, начинающуюся в позиции pos строки s
Delete(s, pos, len)	Удаляет максимум len символов из строки s, начиная с позиции pos
Insert(source, target, pos)	Вставляет строку source в строковую переменную target, начиная с позиции pos
Length (s)	Возвращает динамическую длину строки.
Pos(substring, s)	Возвращает место первого вхождения подстроки substring в строку s.
Str(x, s)	Преобразует численное значение x в строковую переменную s
Val (s, v, code)	Преобразует строку s в соответствующее численное представление v. Если преобразование успешно code=0.

### ***Некоторые библиотечные математические функции***

Библиотечные математические функции находятся в библиотеке Math. Библиотеку Math нужно подключать в разделе Uses:

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,
  Dialogs, StdCtrls, Math;
```

Таблица 3.4 Библиотечные математические функции

Функция	Возвращаемое значение
ArcCos(x)	Арккосинус x (результат в радианах)
ArcSin(x)	Арсинус x (результат в радианах)
ArcTan2(y/x)	Арктангенс y/x (результат в радианах). Возвращает значение угла в несущем квадранте от -π до π. Значение x не может быть нулем
DegToRad	Преобразует углы, заданные в градусах в радианы, аналогично преобразованию: radians = degrees(pi/180)
Log10(x)	Десятичный логарифм от x
Power(X,Y)	Возведение X в вещественную степень Y
RadToDeg	Перевод радиан в градусы, аналогично преобразованию: degrees = radians(180/pi)
Randomize	Инициализация генератора случайных чисел
RandG (Mean,StdDev)	Случайное число, сгенерированное по закону Гаусса Mean – среднее, StdDev – стандартное отклонение. Mean, StdDev вещественные числа (Extended).
Random(Range)	Случайное число в пределах от 0 до Range. Range – целое (Integer).
Tan(x)	Тангенс x (x выражается в радианах)

### *Некоторые функции и процедуры даты*

Функции даты находятся в библиотеке DateUtils, которую нужно подключать в разделе Uses.

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, *DateUtils*;

Таблица 3.5 Функции и процедуры даты

Функция	Возвращаемое значение
Date	Возвращает текущую дату, установленную на компьютере. Функция не имеет аргументов
DecodeDate(Dat, Y,M,D)	Процедура возвращает дату в виде трех величин Y(год), M(месяц), D(день)
DaySpan(An,At)	Функция возвращает количество дней, прошедших между двумя датами An и At.
EncodeDate(Y,M,D)	Функция преобразует три величины Y(год), M(месяц), D(день) в дату
YearsBetween (An,At)	Функция возвращает количество лет, прошедших между двумя датами An и At



Если в программе необходимо использовать переменные, содержащие дату, то их нужно описать типом `TDateTime` в разделе описания переменных. `TDateTime` – это стандартный объект Delphi.

### Ввод из окна ввода

Функция `InputBox` выводит на экран стандартное диалоговое окно – окно ввода. Значение функции `InputBox` – строка, которую ввел пользователь. Формат функции `InputBox` следующий:

Переменная:= `InputBox`(Заголовок, Подсказка, Значение);

где:

Переменная – переменная строкового типа, значение которой должно быть получено от пользователя;

Заголовок – текст, выводимый в заголовок окна;

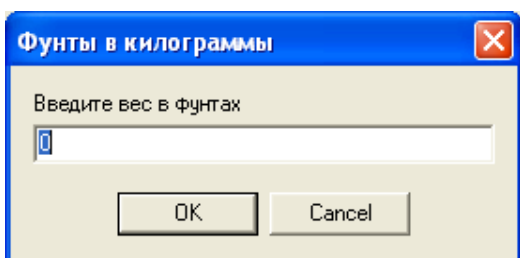
Подсказка – текст поясняющего сообщения;

Значение – текст, который будет находиться в поле ввода, когда окно ввода появится на экране.

Например, следующая инструкция вызывает появление на экране окна ввода, показанного ниже.

```
S:=InputBox('Фунты в килограммы, 'Введите вес в фунтах','0');
```

```
funt:=StrToFloat(S);
```



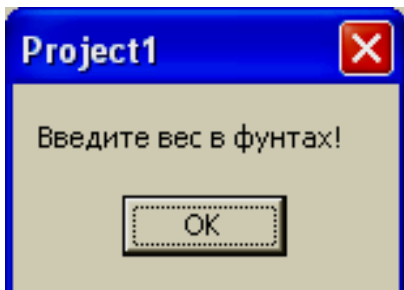
Если во время работы программы пользователь введет строку и щелкнет на кнопке `OK`, то значением функции `InputBox` будет введенная строка, если – на кнопке `Cancel`, то значением функции будет строка, переданная функции в качестве параметра `Значение` (в примере 0).

### Вывод в окно сообщения

Вывести на экран окно с сообщением можно при помощи процедуры `ShowMessage` или функции `MessageDlg`.

Процедура `ShowMessage` выводит на экран окно с текстом и командной кнопкой `OK`.

Формат процедуры ShowMessage :  
 ShowMessage (Сообщение);  
 где Сообщение – текст, который будет выведен в окне.  
 Например,  
 ShowMessage ('Введите вес в фунтах');



Функция MessageDlg позволяет поместить в окно с сообщением один из стандартных значков, задать количество и тип командных кнопок и определить, какую из кнопок нажал пользователь. Формат функции MessageDlg следующий:

Выбор:=MessageDlg(Сообщение, Тип, Кнопки, КонтекстСправки);

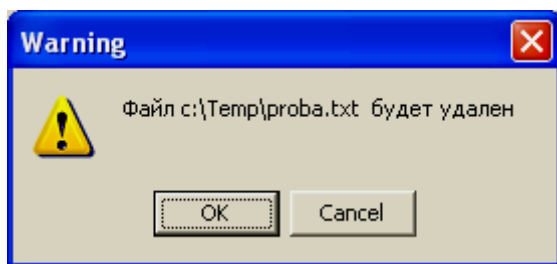
Где Сообщение – текст сообщения; Тип – тип сообщения. Тип сообщения задается именованной константой:

Константа	Тип сообщения
MtWarning	Внимание
MtError	Ошибка
MtInformation	Информация
MtConfirmation	Подтверждение
MtCustom	Обычное

Кнопки – список кнопок, отображаемых в окне сообщения. Список может состоять из нескольких разделенных запятыми именованных констант. Весь список заключается в квадратные скобки.

Константа	Кнопка	Константа	Кнопка
mbYes	Yes	mbAbort	Abort
mbNo	No	mbRetry	Retry
mbOk	Ok	mbIgnore	Ignore
mbCancel	Cancel	mbAll	All
mbHelp	Help		

Например, оператор:  
 r:=MessageDlg('Файл'+ Fname + ' будет удален',  
 mtWarning, [mbOk,mbCancel], 0);  
 вызовет появление на экране окна:



### Замечание

Переменная  $r$  должна быть описана типом Word.

## Практика

### Задача 3а

Дано: гипотенуза и прилежащий к гипотенузе угол прямоугольного треугольника. Составить программу для вычисления катета, противолежащего заданному углу. Формула из тригонометрии:  $a=c*\sin A$ .

Алгоритм решения задачи выглядит так:

1. Ввод исходных данных: гипотенузы и угла, представленного в трех частях (градусы, минуты и секунды)

2. Перевод угла в радианы. Это необходимо, так как тригонометрические функции Delphi работают с углами, заданными в радианах.

Угол в радианах = (Градусы+Минуты/60+Секунды/3600)\*PI/180

3. Вычисление катета по формуле:

катет = гипотенуза \*  $\sin(\text{Угол в радианах})$

4. Вывод катета в окно интерфейса.

### Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Катет прямоугольного треугольника	Заголовок формы
Label1	Caption  WordWrap	Программа вычисляет катет прямоугольного треугольника по гипотенузе и противолежащему углу  True	Справочная информация для пользователя программы  Перенос не уместившихся слов на новую строку
Label2	Caption	Гипотенуза=	Подсказка пользователю
Edit1	Text	Должно быть пустым	Поле для ввода значения гипотенузы

Label3	Caption	Противолежащий угол (гр., мин., сек.)=	Подсказка пользователю
Edit2	Text	Должно быть пустым	Поле для ввода градусов
Edit3	Text	Должно быть пустым	Поле для ввода минут
Edit4	Text	Должно быть пустым	Поле для ввода секунд
Label4	Caption	Должно быть пустым	Поле для вывода результата
Button1	Caption	ВЫЧИСЛИТЬ	Кнопка вычисления катета
Button2	Caption	ОЧИСТИТЬ	Кнопка очищения полей для ввода новых данных
Button3	Caption	ВЫХОД	Кнопка прекращения выполнения программы и закрытия формы

## Переменные

Обозначение в программе	Содержание	Тип
a	Определяемый катет прямоугольного треугольника	вещественный
c	Гипотенуза	вещественный
G	Градусы противолежащего определяемому катету угла	целый
M	Минуты противолежащего определяемому катету угла	целый
S	Секунды противолежащего определяемому катету угла	вещественный

## Проект формы

Катет прямоугольного треугольника

Программа вычисляет катет прямоугольного треугольника по гипотенузе и противолежащему углу

Гипотенуза=

Противолежащий угол (гр., мин., сек.)=

## Текст модуля

```
unit Unit1;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label4: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  a:Real;
  c:Real;
  G,M:Integer;
  S:Real;
  Rad:Real;
```

implementation

```
{ $R *.dfm }
```

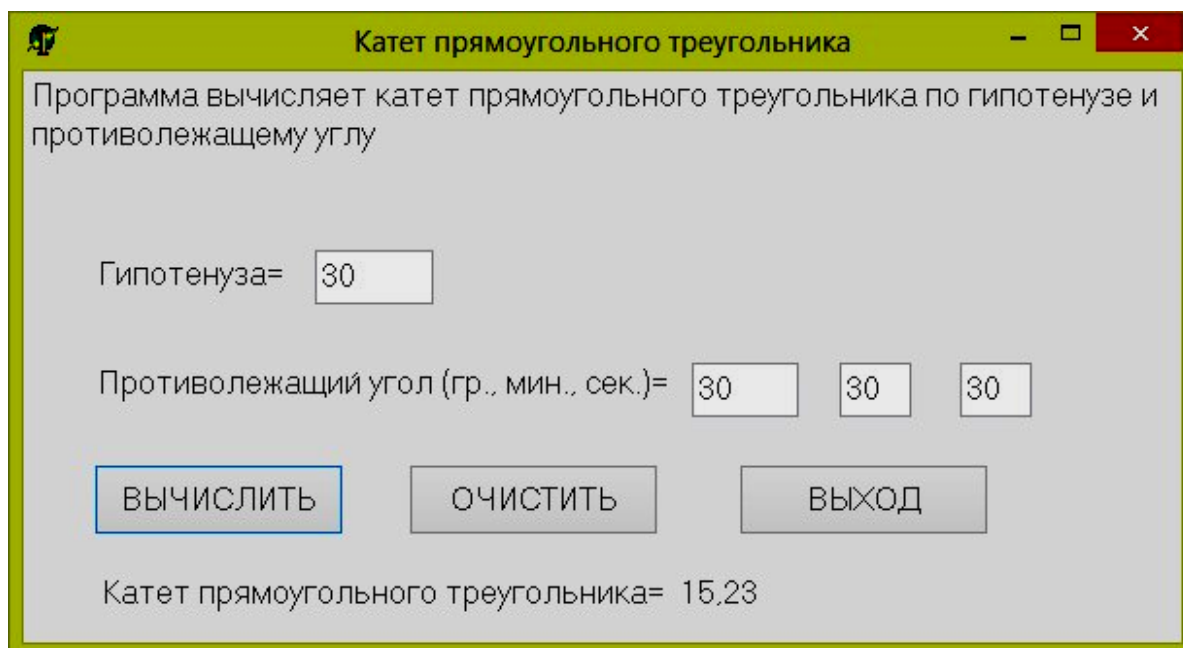
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  c:=StrToFloat(Edit1.Text);
  G:=StrToInt(Edit2.Text);
  M:=StrToInt(Edit3.Text);
  S:=StrToFloat(Edit4.Text);
  Rad:=(G+(M+S/60)/60)*PI/180;
  a:=c*sin(Rad);
  Label4.Caption:='Катет прямоугольного треугольника= '+
FloatToStrF(a, Ffixed,10,2);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Edit1.Clear;
  Edit2.Clear;
  Edit3.Clear;
  Edit4.Clear;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Form1.Close
end;

end.
```

## Результаты работы программы



### ***Задача 3b***

Составить программу проверяющую работу функций генерации случайных чисел `Random(Range)` и `RandG(Mean,StdDev)` в Delphi. Эта задача на использование функций из библиотеки *Math*. Библиотеку *Math* нужно подключить в разделе `Uses`.

### **Замечание**

`Random(Range)` – функция генерирует случайные числа в интервале от 0 до заданного пользователем предела `Range`.

`RandG(Mean,StdDev)` – функция генерирует случайные числа, используя распределение Гаусса. Параметры распределения (`Mean` – среднее и `StdDev` – стандартное отклонение) задаются пользователем программы.

Перед использованием генератора случайных чисел нужно его проинициализировать с помощью функции `Randomize`.

### **Компоненты**

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Генерация случайных чисел	Заголовок формы
Label1	Caption	Для генерации случайного числа в интервале от 0 до RANGE (число) введите	Подсказка пользователю.

Label2	Caption	Ранг=	Подсказка пользователю. После = вводится крайняя правая граница диапазона изменения случайных чисел (ранг). Крайняя левая граница равна 0.
Edit1	Text	Должно быть очищено	Поле для ввода ранга
Label3	Caption	Случайное число=	Поле для вывода результата
Label4	Caption WordWrap	Для генерации случайного числа по закону Гаусса введите среднее значение и стандартное отклонение True	Подсказка пользователю.  Перенос слов на новую строку.
Label5	Caption	Среднее	Подсказка пользователю. После = вводится параметр – среднее значение для генерации числа по закону Гаусса
Edit2	Text	Должно быть очищено	Поле для ввода параметра – среднее значение для генерации числа по закону Гаусса
Label6	Caption	Стандарт отклонение	Подсказка пользователю. После = вводится параметр – стандартное отклонение для генерации числа по закону Гаусса
Edit3	Text	Должно быть очищено	Поле для ввода параметра – стандартное отклонение для генерации числа по закону Гаусса
Label7	Caption	случайное число по закону Гаусса=	Поле для вывода результата
Button1	Caption	ГЕНЕРАЦИЯ	Кнопка для запуска генерации случайных чисел с помощью двух функций
Button2	Caption	ВЫХОД	Кнопка прекращения выполнения программы и закрытия формы



## Проект формы

Генерация случайных чисел

Для генерации случайного числа в интервале от 0 до RANGE (число) введите

РАНГ=

случайное число=

Для генерации случайного числа по закону Гаусса введите среднее значение и стандартное отклонение

СРЕДНЕЕ:

СТАНДАРТНОЕ ОТКЛОНЕНИЕ

случайное число по закону Гаусса=

ГЕНЕРАЦИЯ      ВЫХОД

## Текст модуля

*unit Unit1;*

*interface*

*uses*

*Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  
Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Math;*

*type*

*TForm1 = class(TForm)*

*Label1: TLabel;*

*Edit1: TEdit;*

*Label2: TLabel;*

*Label3: TLabel;*

*Edit2: TEdit;*

*Label4: TLabel;*

*Label5: TLabel;*

```

    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form1: TForm1;
    Mean, StdDev: Extended;
    Rang: Integer;
implementation

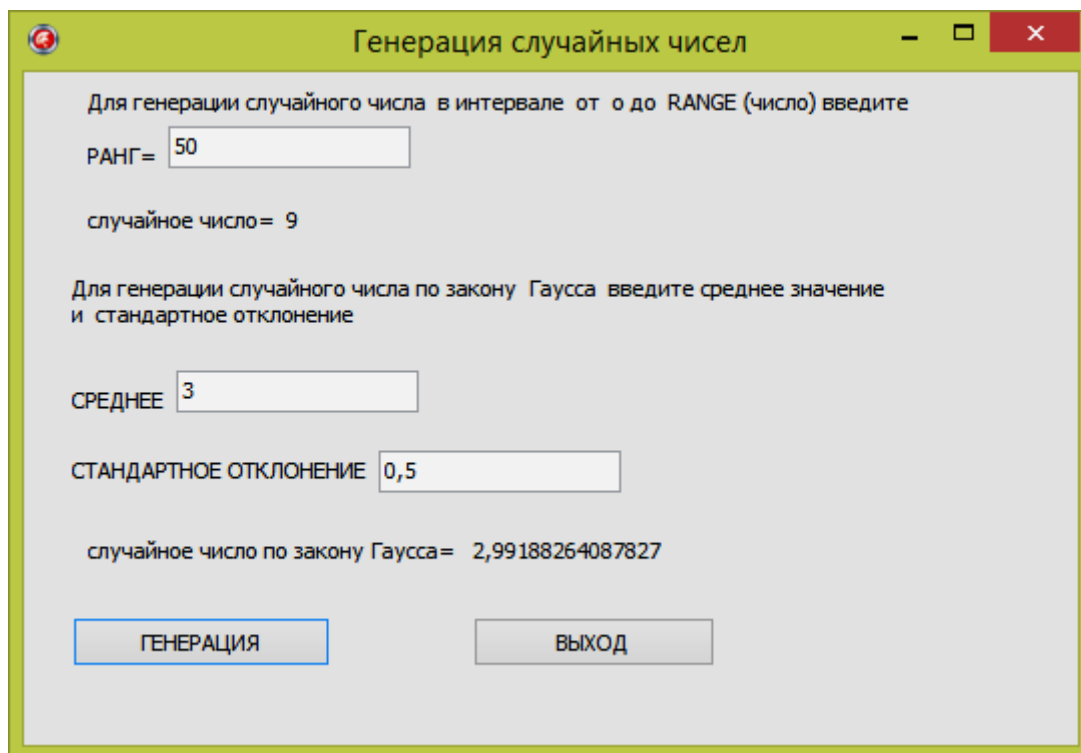
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    Randomize;
    Rang:=StrToInt(Edit1.Text);
    Label3.Caption:= Label3.Caption+ ' '+IntToStr(Random(Rang));
    Mean:=StrToFloat(Edit2.Text);
    StdDev:= StrToFloat(Edit3.Text);
    Label7.Caption:= Label7.Caption+ ' ' + Float-
ToStr(RandG(Mean,StdDev));
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    Form1.Close
end;
end.

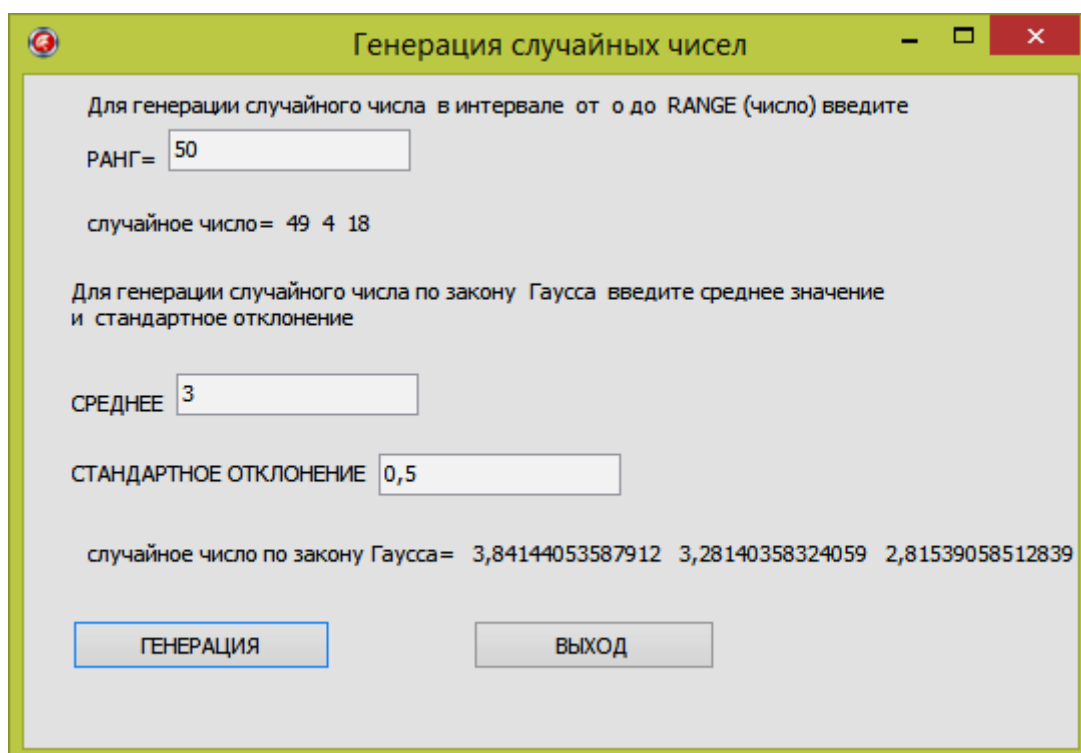
```

Запустите программу на выполнение. Введите исходные данные. Нажмите кнопку ГЕНЕРАЦИЯ.



Так как генерируются случайные числа, результат у вас получится другой. Можете понажимать несколько раз кнопку ГЕНЕРАЦИЯ, чтобы посмотреть как программа генерирует случайные числа.

При следующем вызове программы и трехкратном нажатии на кнопку ГЕНЕРАЦИЯ мы получили следующий результат



### Задача 3с

Составить программу, определяющую сколько дней прожил пользователь программы.

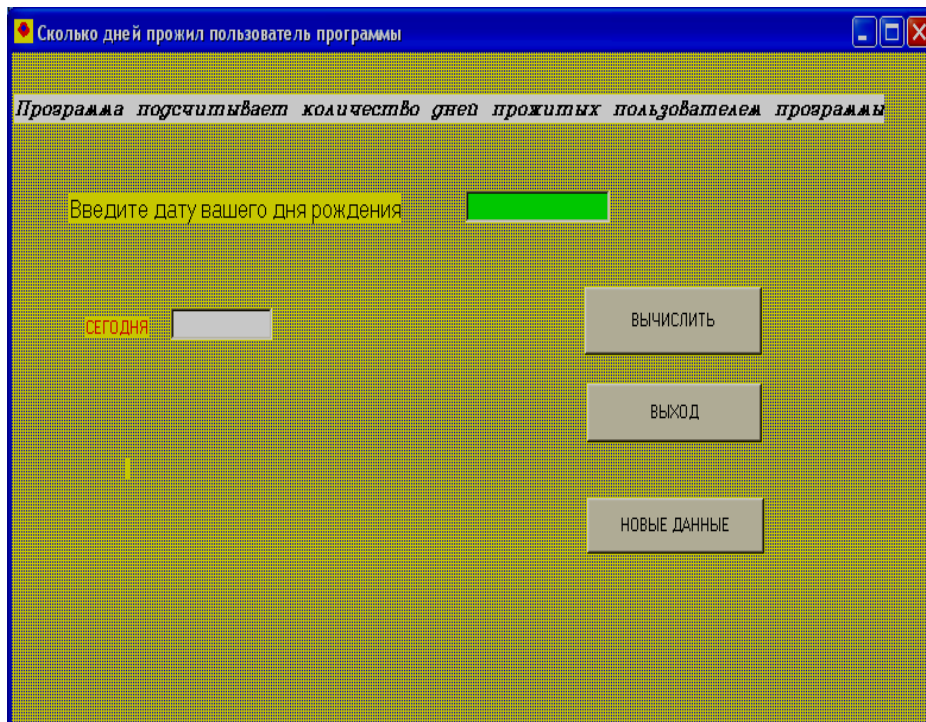
Эта задача на использовании функций работы с датами. Не забудьте в раздел Uses добавить библиотечный модуль **DateUtils**.

### Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Сколько дней прожил пользователь программы	Заголовок формы
Label1	Caption	Программа подсчитывает сколько дней прожил пользователь программы	Справочная информация для пользователя программы
	WordWrap	True	Перенос не уместившихся слов на новую строку
Label2	Caption	Введите дату вашего рождения	Подсказка пользователю
Edit1	Text		Поле для ввода даты рождения
Label3	Caption	Сегодня	Подсказка пользователю
Edit2	Text		Поле, в которое выводится дата, установленная на компьютере
Label4	Caption	Должно быть очищено	Поле для вывода результата
Button1	Caption	ВЫЧИСЛИТЬ	Кнопка вычисления количества дней
Button2	Caption	ВЫХОД	Кнопка прекращения выполнения программы и закрытия формы
Button3	Caption	ОЧИСТИТЬ	Кнопка для очищения полей для ввода новых данных

Эту задачу можно решить, оперируя одними объектами Delphi, без переменных.

## Проект формы



### Текст модуля

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,  
Dialogs, StdCtrls, DateUtils;
```

```
type
```

```
TForm1 = class(TForm)
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
Edit1: TEdit;
```

```
Label3: TLabel;
```

```
Edit2: TEdit;
```

```
Button1: TButton;
```

```
Button2: TButton;
```

```
Label4: TLabel;
```

```

Button3: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin
  Edit2.Text:=DateToStr(Date());
  Label4.Caption:='ВЫ ПРОЖИЛИ '+
  FloatToStr(DaySpan(Date(),StrToDate(Edit1.Text)))+ ' дней'
end;

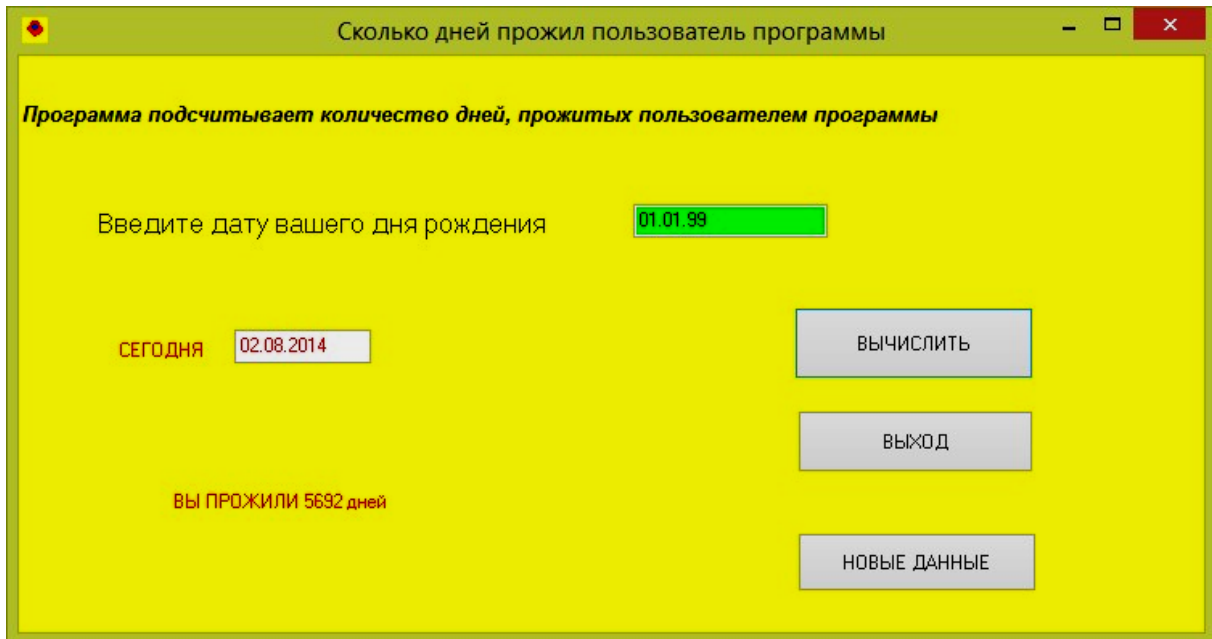
procedure TForm1.Button2Click(Sender: TObject);
begin
  Form1.Close
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Edit1.Clear;
  Label4.Caption:=' ';
end;

end.

```

## Результаты работы программы



### **Задача 3d**

Составить программу для решения прямой и обратной геодезических задач.

Прямая геодезическая задача состоит в том, что по координатам одного конца А ( $X_A$  и  $Y_A$ ) линии АВ, по дирекционному углу этой линии  $\alpha_{AB}$  и ее горизонтальному проложению  $S_{AB}$  вычисляют координаты другого конца В этой линии ( $X_B$   $Y_B$ ).

$$X_B = X_A + S_{AB} \cos \alpha_{AB},$$

$$Y_B = Y_A + S_{AB} \sin \alpha_{AB}.$$

*Горизонтальное проложение* – ортогональная проекция линии местности на горизонтальную плоскость.

*Дирекционный угол* – горизонтальный угол, отсчитываемый от северного направления линии, параллельной оси абсцисс, по ходу часовой стрелки до направления данной линии.

Обратная геодезическая задача состоит в том, что по координатам концов линии АВ вычисляют дирекционный угол и горизонтальное проложение этой линии. То есть известны:  $X_A$ ,  $Y_A$ ,  $X_B$ ,  $Y_B$ . Надо вычислить  $\alpha_{AB}$  и  $S_{AB}$ .

$$\operatorname{tg} \alpha_{AB} = \frac{Y_B - Y_A}{X_B - X_A}$$

$$S_{AB} = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$$

## Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Прямая и обратная геодезические задачи	Заголовок формы
Label1	Caption	Обратная геодезическая задача – вычисление по координатам двух точек горизонтального проложения и дирекционного угла линии между ними	Справочная информация для пользователя программы
	Visible	False	невидимый
Label2	Caption	XA	Подсказка пользователю
	Visible	False	невидимый
Label3	Caption	YA	Подсказка пользователю
	Visible	False	невидимый
Label4	Caption	XB	Подсказка пользователю
	Visible	False	невидимый
Label5	Caption	YB	Подсказка пользователю
	Visible	False	невидимый
Label6	Caption	Горизонтальное проложение линии АВ в метрах	Подсказка пользователю
	Visible	False	невидимый
Label7	Caption	Должно быть очищено	Поле для вывода горизонтального проложения
	Visible	False	невидимый
Label8	Caption	Дирекционный угол (гр. мин. сек.)	Подсказка пользователю
	Visible	False	невидимый
Label9	Caption	Должно быть очищено	Поле для вывода дирекционного угла (градусы)
	Visible	False	невидимый
Label10	Caption	Должно быть очищено	Поле для вывода дирекционного угла (минуты)
	Visible	False	невидимый
Label11	Caption	Должно быть очищено	Поле для вывода дирекционного угла (секунды)
	Visible	False	Невидимый
Edit1	Text	Должно быть очищено	Поле для ввода координаты XA
	Visible	False	невидимый



Edit2	Text	Должно быть очищено	Поле для ввода координаты YA
	Visible	False	невидимый
Edit3	Text	Должно быть очищено	Поле для ввода координаты XB
	Visible	False	невидимый
Edit4	Text	Должно быть очищено	Поле для ввода координаты YB
	Visible	False	невидимый
Button1	Caption	ПРЯМАЯ ГЕОДЕЗИЧЕСКАЯ ЗАДАЧА	Кнопка запускает на выполнение прямую геодезическую задачу
Button2	Caption	ОБРАТНАЯ ГЕОДЕЗИЧЕСКАЯ ЗАДАЧА	Кнопка делает видимыми поля для решения обратной геодезической задачи
Button3	Caption	ВЫХОД	Кнопка закрытия формы и выхода из программы
Button4	Caption	Вычислить	Кнопка вычисления обратной геодезической задачи
	Visible	False	невидимый

### Переменные

Обозначение в программе	Содержание	Тип
XA	Координата X точки A	вещественный
YA	Координата Y точки A	вещественный
XB	Координата X точки B	вещественный
YB	Координата Y точки B	вещественный
G	Часть дирекционного угла линии AB в градусах	целый
M	Часть дирекционного угла линии AB в минутах	целый
S	Часть дирекционного угла линии AB в секундах	вещественный
L	Горизонтальное проложение линии AB	вещественный
DX	Приращение координат по оси X	вещественный
DY	Приращение координат по оси Y	вещественный
DU	Дирекционный угол в градусах	вещественный

**Замечание.** При проектировании компонентов для решения обратной геодезической задачи на той же форме, свойство Visible компонентов целесообразно установить в положение False.

## Текст модуля

*unit Unit1;*

*interface*

*uses*

*Windows, Messages, SysUtils, Variants, Classes, Graphics,  
Controls, Forms,*

*Dialogs, StdCtrls, **Math**;*

{ библиотечный модуль Math нужно добавить в стандартный набор модулей для подключения математической библиотеки }

.....

*var*

*Form1: TForm1;*

*XA, YA: Extended; // координаты точки А*

*XB, YB: Extended; // координаты точки В*

*G: Integer; // дирекционный угол в градусах*

*M: Integer; // часть дирекционного угла в минутах*

*S: Real; // часть дирекционного угла в секундах*

*L: Extended; // горизонтальное проложение линии АВ*

*DX, DY: Extended; // приращения координат по осям X, Y*

*DU: Extended; // дирекционный угол в радианах*

*implementation*

*{ \$R \*.dfm }*

*procedure TForm1.Button1Click(Sender: TObject);*

*begin*

*ShowMessage('Прямая геодезическая задача – определение координаты'+*

*' точки по координатам исходной точки, горизонтальному проложению '*

*' и дирекционному углу.'+#13*

*+'Дирекционный угол задается в градусах, минутах и секундах. Коор-ты в метрах.');*

*// ввод координат исходной точки, дирекционного угла*

*// и горизонтального проложения линии АВ из окон ввода*

*XA:=StrToFloat(InputBox('Прямая геодезическая задача',*

*'Введите координату X точки А','0'));*

*YA:=StrToFloat(InputBox('Прямая геодезическая задача',*

*'Введите координату Y точки А','0'));*

*G:=StrToInt(InputBox('Прямая геодезическая задача',*

*'Введите часть дирекционного угла линии АВ в градусах' + #13*

*+'Значение < 360','0'));*

*M:=StrToInt(InputBox('Прямая геодезическая задача',*

*'Введите часть дирекционного угла линии АВ в минутах'+*

*#13+'Значение <60','0'));*

*S:=StrToFloat(InputBox('Прямая геодезическая задача',*

```

    'Введите часть дирекционный угол линии АВ в секундах'
+#13+'Значение <60','0'));
    L:=StrToFloat(InputBox('Прямая геодезическая задача',
    'Введите горизонтальное проложение линии АВ в метрах','0'));
    // вычисление координат точки В
    XB:=XA+L*cos((g+m/60+s/3600)/180*pi);
    YB:=YA+L*sin((g+m/60+s/3600)/180*pi);
    // в нижеприведенном преобразовании можно использовать
    // функцию преобразования DegToRad
    {XB:=XA+L*cos(DegToRad(g+m/60+s/3600));
    YB:=YA+L*sin(DegToRad (g+m/60+s/3600));}
    // округление вычисленных координат
    // до двух знаков после запятой
    XB:=Trunc(XB*100+0.5)/100;
    YB:=Trunc(YB*100+0.5)/100;
    // вывод координат точки В окно сообщения
    ShowMessage('Прямая геодезическая задача ' + #13 + 'XB='
+FloatToStrF(XB,Ffixed,10,2)+ #13
    + 'YB='+FloatToStrF(YB,Ffixed,10,2))
    end;
    procedure TForm1.Button3Click(Sender: TObject);
    begin
    Form1.Close
    end;
    procedure TForm1.Button2Click(Sender: TObject);
    begin
        // сделать видимыми компоненты, относящиеся к обратной
        // геодезической задаче
        Label1.Visible:=True;
        Label2.Visible:=True;
        Label3.Visible:=True;
        Label4.Visible:=True;
        Label5.Visible:=True;
        Label6.Visible:=True;
        Label7.Visible:=True;
        Label8.Visible:=True;
        Label9.Visible:=True;
        Label10.Visible:=True;
        Label11.Visible:=True;
        Edit1.Visible:=True;
        Edit2.Visible:=True;
        Edit3.Visible:=True;
        Edit4.Visible:=True;

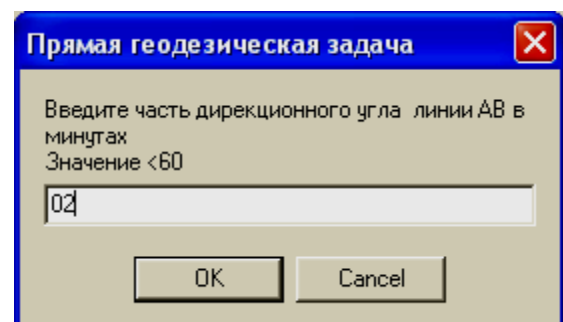
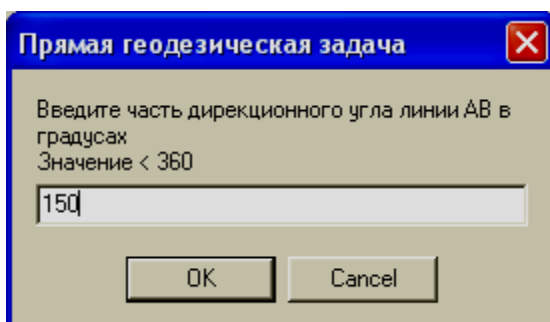
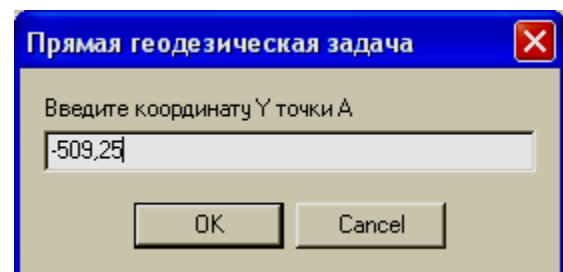
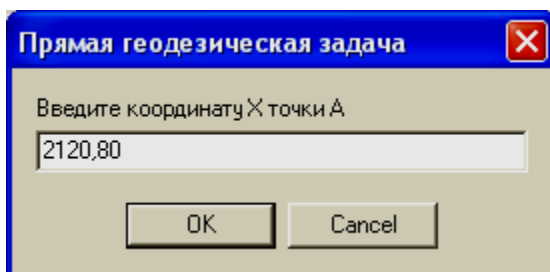
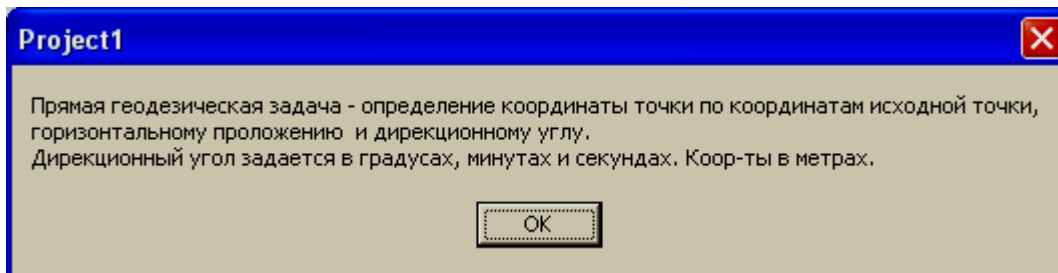
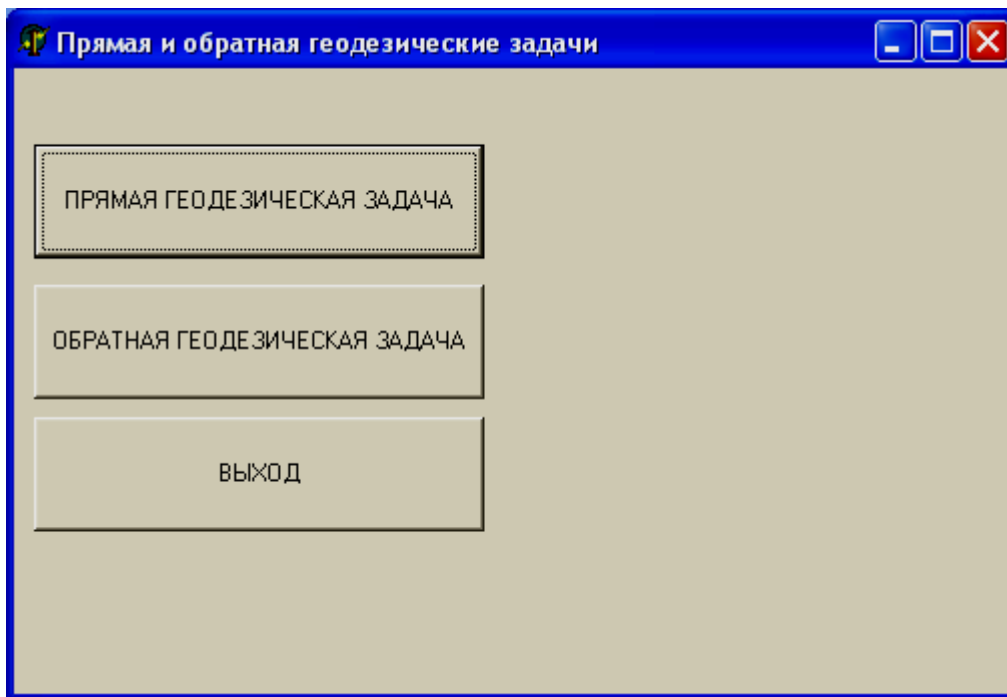
```

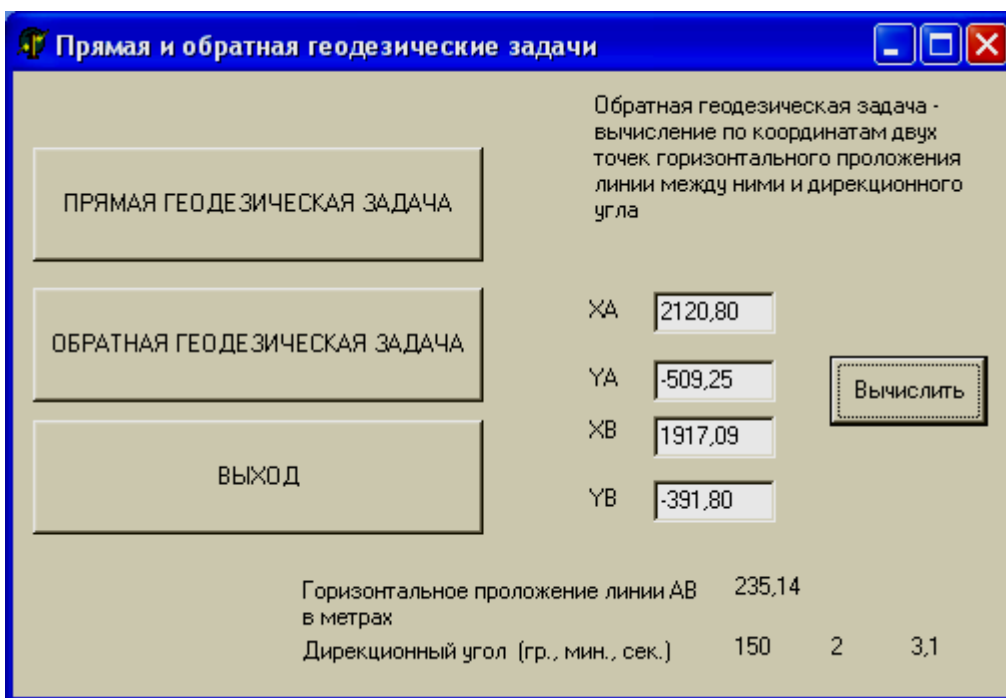
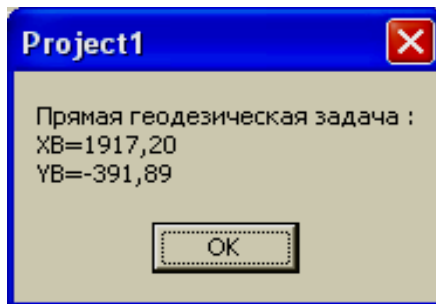
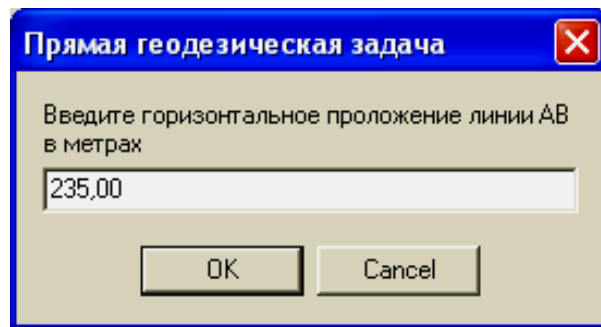
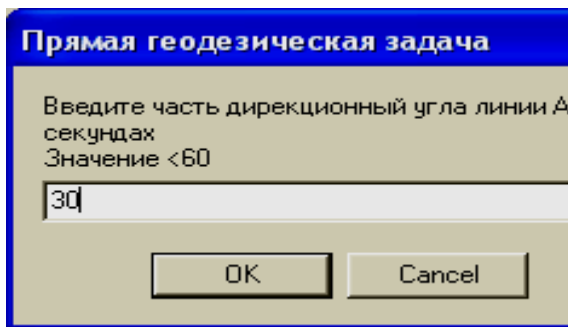
```

// перенести фокус в первое поле ввода на форме
Edit1.SetFocus;
Button4.Visible:=True;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
// ввод координат точек А и В из полей ввода на форме
XA:=StrToFloat(Edit1.Text);
YA:=StrToFloat(Edit2.Text);
XB:=StrToFloat(Edit3.Text);
YB:=StrToFloat(Edit4.Text);
DX:=XB-XA;
DY:=YB-YA;
// вычисление горизонтального проложения линии АВ
L:=Sqrt(Sqr(DX)+Sqr(DY));
// округление горизонтального проложения
// до двух знаков после запятой
L:=Trunc(L*100+0.5)/100;
// вычисление дирекционного угла и перевод
// его значения из радиан в градусы
DU:=ArcTan2(DY,DX)/Pi*180;
{ в данном преобразовании можно было использовать функцию
RadToDeg: DU:= RadToDeg (ArcTan2(DY,DX)); }
//ArcTan2(Y/X) - библиотечная функция находит угол X/Y
//в несущем квадранте по знакам X и Y от -Pi до Pi в радианах.
//если угол со знаком минус к нему прибавляется 360 градусов
if DU<0 Then DU:=360+DU;
//выделение градусной части дирекционного угла
G:=Trunc(DU);
// выделение минутной части дирекционного угла
M:=Trunc((DU-G)*60);
// выделение секундной части дирекционного угла
S:=((DU-G)*60-M)*60;
{ вывод горизонтального проложения и дирекционного угла в поля
вывода }
Label7.Caption:=FloatToStrF(L,Ffixed,10,2);
Label9.Caption:=IntToStr(G);
Label10.Caption:=IntToStr(M);
Label11.Caption:=FloatToStrF(S,Ffixed,5,1);
end;
end.

```

Такие окна будут появляться на экране при выполнении проекта:





### ***Вопросы для самоконтроля***

1. Какие вы знаете стандартные математические функции?
2. Как вычислить  $a^x$ ?
3. С помощью какой функции можно округлить вещественное число до ближайшего целого?
4. Как выделить целую часть из вещественного числа?
5. Как выделить дробную часть из вещественного числа?
6. Как преобразовать строку символов в вещественное число с форматом?

7. Какие функции преобразования вы знаете?
8. Как получить предыдущее значение переменной? (см. табл. 3.2)
9. Какие функции работы со строками вы знаете?
10. Как подсчитать количество символов в строке?
11. Какие библиотечные математические функции вы знаете?
12. Как преобразовать градусы в радианы на языке Delphi?
13. Как осуществить ввод из окна ввода?
14. Какие функции генерации случайных чисел вы знаете?
15. Как осуществить ввод из окна ввода?
16. Как осуществить вывод в окно сообщения?
17. Опишите действия, которые выполняет программа «катет прямоугольного треугольника», когда вы нажимаете на кнопку ВЫЧИСЛИТЬ.
18. Опишите действия, связанные с событием клик по кнопке Button1 в программе «Генерация случайных чисел».
19. Объясните, что происходит в результате выполнения оператора  
`Edit2.Text:=DateToStr(Date());`  
 в программе «Сколько дней прожил пользователь».
20. Объясните, какие действия связаны с событием клик по командной кнопке Button2 в программе «Прямая и обратная геодезические задачи».
21. В каких единицах измерения должен быть задан угол, если в программе на языке Delphi нужно вычислить  $\sin$  угла?
22. Как вычислить тангенс угла в программе на языке Delphi?

### *Задачи для самостоятельного решения*

#### Группа А

№ 3.1 Даны  $x, y, z$ . Составить программу для вычисления  $a, b$  по формулам:

$$a = \frac{\sqrt{|x-1|} - \sqrt[3]{|y|}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}, b = x(\arctg(z) + e^{-(x+3)}).$$

№ 3.2 Даны  $x, y, z$ . Составить программу для вычисления  $a, b$  по формулам:

$$a = \frac{3 + e^{y-1}}{1 + x^2 |y - \operatorname{tg}(z)|}, b = 1 + |y - x| + \frac{(y - x)^2}{2} + \frac{|y - x|^3}{3}.$$

№ 3.3 Даны  $x, y, z$ . Составить программу для вычисления  $a, b$  по формулам:

$$a = (1 + y) \frac{x + y/(x^2 + 4)}{e^{-x-2} + 1/(x^2 + 4)}, b = \frac{1 + \cos(y - 2)}{x^4/2 + \sin^2(z)}.$$

№ 3.4 Даны  $x, y, z$ . Составить программу для вычисления  $a, b$  по формулам:

$$a = y + \frac{x}{y^2 + \left| \frac{x^2}{y + x^3/3} \right|}, b = 1 + \operatorname{tg}^2\left(\frac{z}{2}\right).$$

№ 3.5 Даны  $x, y, z$ . Составить программу для вычисления  $a, b$  по формулам:

$$a = \frac{2 \cos(x - \pi/6)}{1/2 + \sin^2(y)}, b = 1 + \frac{z^2}{3 + z^2/5}.$$

№ 3.6 Даны  $x, y, z$ . Составить программу для вычисления  $a, b$  по формулам:

$$a = \frac{1 + \sin^2(x + y)}{2 + |x - 2x/(1 + x^2 y^2)|} + x, b = \cos^2\left(\operatorname{arctg}\left(\frac{1}{z}\right)\right).$$

№ 3.7 Составить программу, подсчитывающую сколько дней осталось до каникул.

№ 3.8 Даны действительные числа:  $\delta$  и  $\sigma$ . Найти  $p(\delta, \sigma)$  по формуле:

$$p(\delta, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{\delta^2}{2\sigma^2}}$$

№ 3.9 Составить программу, подсчитывающую количество символов в полном имени пользователя программы, включая пробелы.

№ 3.10 Составить программу, определяющую сколько лет прожил пользователь программы.

## **Группа Б**

№ 3.11 Написать программу для решения прямой геодезической задачи: по заданным координатам точки А ( $X_A, Y_A$ ), горизонтальному проложению ( $L$  в м) от точки А до точки В и дирекционному углу ( $\alpha$ ) этого проложения найти координаты точки В:



$$X_B = X_A + L \cdot \cos \alpha$$

$$Y_B = Y_A + L \cdot \sin \alpha$$

Угол вводить в градусах, минутах и секундах, координаты, вводить с точностью до 1 м.

В программе предусмотреть следующие запросы при вводе исходных данных:

XA =, YA =

ПРОЛОЖЕНИЕ АВ =

ДИР. УГОЛ АВ: G, M, S =

Форма вывода результата:

XB = значение XB    YB = значение YB.

№ 3.12 Написать программу, которая по заданному значению вертикального угла  $\nu$  и расстоянию  $S$  (в м) между точками А и В определяет превышение точки В над точкой А:

$$R = S \cdot \sin \nu$$

и горизонтальное проложение между А и В:

$$D = S \cdot \cos \nu$$

Значение округлять до 0,01 м, значение  $D$  – до 0,1 м.

В программе предусмотреть следующие запросы при вводе исходных данных:

ВЕРТИКАЛЬНЫЙ УГОЛ: G, M, S =

РАССТОЯНИЕ =

Форма вывода результатов:

ПРЕВЫШЕНИЕ = значение R

ГОР. ПРОЛОЖЕНИЕ = значение D

(См. примечание к задаче 2.11).

№ 3.13 Написать программу для вычисления поправки дирекционного угла в сек. по значениям: дирекционного угла  $\alpha$ , расстояния  $S$  в метрах и поправок координат концов отрезка  $\delta'_x, \delta'_y, \delta''_x, \delta''_y$  в метрах:

$$\delta_\alpha = \frac{206000}{S} (\sin \alpha \cdot (\delta'_x - \delta'_y) + \cos \alpha \cdot (\delta''_x - \delta''_y)).$$

Поправку округлить до 0,01 с.

В программе предусмотреть следующие запросы при вводе исходных данных:

ДИР. УГОЛ: ГР., МИН., СЕК. =  
РАССТОЯНИЕ =  
ПОПРАВКИ КООРДИНАТ:

DX1 = вводить  $\delta'_x$

DY1 = вводить  $\delta'_y$

DX2 = вводить  $\delta''_x$

DY2 = вводить  $\delta''_y$

Форма вывода результата:

ПОПРАВКА ДИР. УГЛА= значение в сек.

№ 3.14 Написать программу для вычисления площади треугольного участка по длине стороны А и прилежащих к ней углов  $\beta$  и  $\gamma$ . Значения углов вводить в градусах, минутах и секундах, длину стороны А – в метрах. Площадь вывести в гектарах с точностью до 0,1 га.

Формула для вычисления площади:

$$S = \frac{A^2 \cdot \sin \beta \cdot \sin \gamma}{2 \cdot \sin(\beta + \gamma)}$$

В программе предусмотреть следующие запросы при вводе исходных данных:

УГОЛ В: ГР., МИН., СЕК. =

УГОЛ С: ГР., МИН., СЕК. =

СТОРОНА А =

Форма вывода результата:

ПЛОЩАДЬ УЧАСТКА = значение S ГА.

№ 3.15 Написать программу для перевода декартовых координат (X,Y) некоторой точки в полярные ( $\rho, \Theta$ ) при условии, что полюс совпадает с началом декартовой системы, а полярная ось совпадает с ось ОХ:

$$\rho = \sqrt{X^2 + Y^2}$$

$$\Theta = \arctg \frac{Y}{X}.$$

Значение  $\rho$  округлять до 0,1 м. Значение  $\Theta$  выводить в градусах, минутах и секундах, округляя секунды до 0,1.

В программе предусмотреть следующие запросы при вводе исходных данных:

ДЕКАРТОВЫ КООРДИНАТЫ (X,Y) =

Форма вывода результатов:

ПОЛЯРНЫЕ КООРДИНАТЫ:

УГОЛ (ГР., МИН., СЕК.) = значение

РАССТОЯНИЕ = значение

№ 3.16 Написать программу для перевода координат точки М из полярной системы в декартову, считая, что полюс совпадает с началом декартовых координат, а полярная ось совпадает с осью ОХ:

$$X = \rho \cos \Theta, Y = \rho \sin \Theta.$$

Здесь  $\rho$  и  $\Theta$  – координаты точки М в полярной системе, где  $\Theta$  в градусах, минутах и секундах.

Значения X,Y при выводе округлить до 0,01 м.

В программе предусмотреть следующие запросы при вводе исходных данных:

ПОЛЯРНЫЕ КООРДИНАТЫ:

УГОЛ (ГР., МИН., СЕК.) =

РАССТОЯНИЕ (МЕТРЫ) =

Форма вывода результатов:

ДЕКАРТОВЫ КООРДИНАТЫ:

X = значение X МЕТРОВ Y = значение Y МЕТРОВ.

№ 3.17 Написать программу для вычисления коэффициента нагрузки лесной полосы (R) по заданному значению угла  $\alpha$ , образуемого осью лесной полосы и направлением ветра:

$$R = \sin(\alpha + 6^\circ)(1 - \operatorname{tg}(45^\circ - \frac{\alpha}{2})).$$

Значение  $\alpha$  вводить в градусах и минутах.

В программе предусмотреть следующие запросы при вводе:

УГОЛ (ГР., МИН.) =

Форма вывода результатов:

КОЭФФИЦИЕНТ НАГРУЗКИ = значение R.

№ 3.18 Написать программу для вычисления площади треугольного участка по длине стороны А и прилежащих к ней углов  $\beta$  и  $\gamma$ . Значения углов вводить в градусах, минутах и секундах, длину стороны А – в метрах. Площадь вывести в гектарах с точностью до 0,1 га.

Формула для вычисления площади:

$$S = \frac{A^2 \sin \beta \sin \gamma}{2 \sin(\beta + \gamma)}$$

В программе предусмотреть следующие запросы при вводе исходных данных:

УГОЛ В: ГР., МИН., СЕК. =

УГОЛ С: ГР., МИН., СЕК. =

СТОРОНА А =

Форма вывода результатов:

ПЛОЩАДЬ УЧАСТКА = значение S ГА

№ 3.19 Написать программу для определения угла (рис. 3–1) по заданным значениям X и S. Угол  $\alpha$  определять в градусах, минутах и секундах, округлив секунды до 0,1 сек.

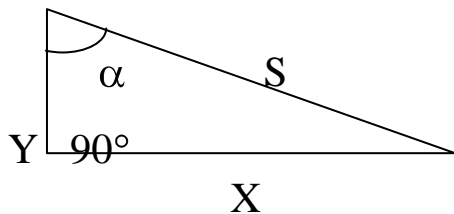


Рис. 3-1

В программе предусмотреть запросы при вводе исходных данных:

ГИПОТЕНУЗА = ?

ПРОТИВОЛЕЖАЩИЙ КАТЕТ = ?

Форма вывода результата:

УГОЛ – количество ГР., количество минут МИН., количество секунд СЕК.

№ 3.20 Написать программу для определения угла  $\alpha$  (рис. 3–1) по заданным значениям Y и S. Угол определять в градусах, минутах и секундах, округляя секунды до 0,1.

В программе предусмотреть запросы при вводе исходных данных:

ГИПОТЕНУЗА =

ПРИЛЕЖАЩИЙ КАТЕТ =

Форма вывода результата:

Угол = градусы ГР., минуты МИН., секунды СЕК.

## Глава 4. Программы с разветвлениями

*В данной главе* рассматриваются следующие вопросы: условный оператор, оператор выбора (варианта), оператор перехода, компоненты: ListBox, ComboBox, некоторые приемы работы с отладчиком, обработка исключительных ситуаций.

### Теория

#### Логические выражения

Логическое выражение в общем виде выглядит так:

ОП1 оператор ОП2,

где ОП1 и ОП2 – операнды логического выражения, в качестве которых может выступать переменная, константа, функция или выражение; оператор – это оператор сравнения.

Результатом *логического выражения* является логическое значение True или False. Логические выражения чаще всего используются в условном операторе и в операторах цикла и состоят из:

- логических констант True и False ;
- операндов (переменные, элементы массивов, выражения)
- логических переменных типа boolean;
- операций сравнения (отношения);
- логических операций;
- круглых скобок.

Для установления отношения между двумя значениями, заданными выражениями, переменными или константами, используются следующие операции сравнения:

- = – равно,
- < – меньше,
- > – больше,
- <= – меньше или равно,
- >= – больше или равно,
- <> – не равно,
- in – принадлежность множеству.

Операции сравнения выполняются после вычисления соответствующих выражений. Результатом операции сравнения является значение False, если соответствующее отношение не имеет места. И значение True, если соответствующее отношение имеет место.

**Замечание.** Приоритет операций сравнения меньше, чем приоритет логических операций. Поэтому, если логическое выражение, содержащее операцию сравнения, является операндом логической операции, то его нужно заключить в круглые скобки.

В сложных логических выражениях используются следующие логические операторы:

Таблица 4.1 Логические операторы

Операция	Описание	Операнд 1	Операнд 2	Результат
not	Отрицание	False True		True False
and	Логическое и	False False True True	False True False True	False False False True
or	Логическое или	False False True True	False True False True	False True True True
xor	Исключающее или	False False True True	False True False True	False True True False

В таблице 4.1 логические операторы приведены по старшинству приоритетов. Высший приоритет имеет оператор отрицания (инверсия), на втором месте конъюнкция (логическое умножение или логическое и), на третьем – дизъюнкция (логическое сложение или логическое или), на последнем месте в таблице 4.1 исключающее или.

### ***Примеры логических выражений***

#### Простые логические выражения

$$x < 10$$

$$x + 17 \geq y$$

Если переменная  $x=5$ , а переменная  $y=24$ , то результатом первого логического выражения будет TRUE (истина), а второго FALSE (ложь).

#### Сложные логические выражения

$$(x > a) \text{ and } (x < b) \text{ or } (y < c)$$

Допустим значения переменных следующие:  $x=5$ ;  $a=3$ ;  $b=8$ ;  $y=5$ ;  $c=4$ . Значения простых логических выражений:  $5 > 3 = \text{TRUE}$ ;

5<8= TRUE;5<4= FALSE. Согласно приоритету логических операторов сначала выполняется оператор and:

TRUE and TRUE= TRUE.

Следующим выполняется логический оператор or:

TRUE or FALSE= TRUE.

Таким образом, результатом рассмотренного сложного логического выражения является значение TRUE (истина).

Переменные и операнды, участвующие в логических выражениях могут быть любого типа из существующих в Delphi. Только типы сравниваемых между собой операндов должны быть одинаковыми. Рассмотрим ещё одно сложное логическое выражение:

**Not (color = 'красный' or color = 'желтый').**

Результатом данного логического выражения будет истина (TRUE), если переменная color примет какое-нибудь другое значение и не будет равна красному или желтому.

### Условный оператор

*Условный оператор* обеспечивает выполнение или невыполнение некоторых операторов в зависимости от соблюдения определенных условий. Условный оператор в общем случае предназначен для организации разветвления программы на два направления и имеет формат:

```
if <Условие> then <Оператор1> [ else <Оператор2> ];
```

Условие представляет собой выражение логического типа. Оператор работает следующим образом: если условие истинно (имеет значение True), то выполняется Оператор1, в другом случае выполняется Оператор2. Оба оператора могут быть составными. Условный оператор может быть записан в сокращенной форме, когда слово else и оператор после него отсутствуют.

```
if <Условие> then <Оператор1>;
```

В этом случае при невыполнении условия выполняется оператор, следующий за условным.

Для организации разветвлений на три направления и более можно использовать несколько условных операторов, вложенных друг в друга. При этом каждое else соответствует тому then, которое непосредственно ему предшествует. Из-за возможных ошибок следует избегать большой вложенности условных операторов друг в друга.

**Замечание.** По синтаксису условного оператора после слов **then** и **else** может стоять только один оператор, а по смыслу программы бывает нужно выполнить несколько операторов. В этом случае операторы, которые должны быть выполнены в ветвях **then** и **else**, заключаются в операторные скобки **begin** и **end** (составной оператор) .

Обратите внимание, что в операторе **if** перед служебным словом **else** нельзя ставить точку с запятой. Это вызовет ошибку.

**Примеры.** Условные операторы.

Условный оператор в полной форме:

```
if x > 0 then x := x + 1 else x := 0;
```

Если переменная  $x$  больше нуля, то ее значение увеличивается на единицу. В противном случае переменная  $x$  получает значение равное нулю.

Условный оператор в краткой форме (ветвь иначе отсутствует):

```
if q = 0 then a := 1;
```

Если переменная  $q$  равна нулю, то переменной  $a$  присваивается значение равное 1. В противном случае выполняется оператор следующий за оператором **if** после точки с запятой.

```
if not (color = 'красный' or color = 'желтый')
```

```
then ggo := 'можно переходить дорогу'
```

```
else ggo := 'Стоять. Ждать зеленый свет';
```

```
if x + y <> 0 then begin a := 1/(x + y); x:=x+1; y:=y-1 end;
```

В последнем примере по смыслу, если значение логического выражения  $x + y$  не равно нулю, нужно выполнить несколько операторов, поэтому они оформляются в виде составного оператора (заключаются в операторные скобки **begin** и **end**).

## Оператор выбора

**Оператор выбора** является обобщением условного оператора и позволяет сделать выбор из произвольного числа имеющихся вариантов, то есть организовать разветвления на произвольное число направлений. Этот оператор состоит из выражения, называемого *селектором*, списка вариантов и необязательной ветви **else**, имеющей тот же смысл, что и в условном операторе.

Формат оператора выбора:

```
case <Выражение-селектор> of  
<Список1> : <Оператор1>;
```



```
<СписокN> : <ОператорN>  
else <Оператор>;  
end;
```

**Выражение-селектор** должно быть порядкового типа. Каждый вариант представляет собой список констант, отделенных двоеточием от относящегося к данному варианту оператора, возможно, составного. Список констант выбора состоит из произвольного количества значений и диапазонов, отделенных друг от друга запятыми. Границы диапазона записываются двумя константами через разделитель « . . » . Тип констант должен совпадать с типом *выражения-селектора*.

Оператор выбора выполняется следующим образом:

- вычисляется значение выражения селектора;
- производится последовательный просмотр вариантов на предмет совпадения значения селектора с константами и значениями из диапазонов соответствующего списка;
- если для очередного варианта этот поиск успешный, то выполняется оператор этого варианта. После этого выполнение оператора выбора заканчивается;
- если все проверки оказались безуспешными, то выполняется оператор, стоящий после слова **else** (при его наличии).

### ***Замечание.***

При совпадении значения-селектора с константой из списка, по смыслу программы в некоторых случаях нужно выполнить несколько операторов, а по синтаксису оператора **Case** после списка констант должен стоять только один оператор. В таких случаях операторы, которые нужно выполнить оформляются, как составной оператор (закljučаются в служебные слова **begin** и **end**).

Иногда в программах не предусматривается никаких действий, если в операторе **Case** выражение-селектор не совпадает ни с одной константой из списка. В таких случаях ветвь **else** отсутствует в операторе.

В отличие от оператора **If** в операторе **Case** перед служебным словом **else** можно поставить точку с запятой. Компилятор не выдает сообщение об ошибке.

**Пример.** Оператор выбора.

```
case DayNumber of  
1 .. 5 : strDay := 'Рабочий день';  
6, 7 : strDay := 'Выходной день'  
else strDay := ' ';  
end;
```

В зависимости от значения целочисленной переменной DayNumber, содержащей номер дня недели, строковой переменной strDay присваивается соответствующее значение. Если DayNumber равно **3**, то strDay присваивается значение **Рабочий день**.

### Оператор перехода

Оператор безусловного перехода осуществляет переход на оператор, помеченный меткой. Формат оператора перехода следующий:

Goto метка, где метка – это идентификатор, находящийся перед оператором, который должен быть выполнен после оператора Goto.

Метка должна быть описана в разделе label, который располагается перед разделом описания переменных.

Например,  
Label NNN, 123,G12;

.....  
NNN: A:=D/S;

....  
Goto NNN;

#### **Замечание**

Современный стиль программирования предполагает составление программ без использования оператора Goto.

### Компонент ListBox



ListBox (список) нужен для показа прокручиваемого списка. Классический пример ListBox'а в среде Windows – выбор файла из списка в пункте меню File | Open многих приложений. Названия файлов или директорий находятся в ListBox'е.

## Главные свойства компонента TListBox

Свойство	Назначение
Name	Имя компонента. В программе используется для доступа к свойствам компонента
Items	Элементы списка
ItemIndex	Номер выбранного элемента списка. Номер первого элемента списка равен нулю

Список может быть сформирован во время создания формы или во время работы программы.

Для формирования списка во время создания формы надо в окне Инспектора Объектов (Object Inspector) выбрать свойство Items и щелкнуть на кнопке запуска редактора списка строк. В открывшемся окне String List Editor нужно набрать список (каждый элемент списка в отдельной строке). После ввода очередного элемента списка, для перехода к новой строке, необходимо нажать клавишу ENTER. После ввода последнего элемента списка щелкнуть на кнопке ОК.


Добавить элементы в список непосредственно в программе можно с помощью метода Add, например:

```
ListBox1.Items.Add('Россия');
```

```
ListBox1.Items.Add('Австрия');
```

Если ни один из элементов списка не выбран, то значение свойства ItemIndex равно минус единице.

## Компонент ComboBox

 ComboBox во многом напоминает ListBox, за исключением того, что позволяет вводить информацию в маленьком поле ввода-редактирования сверху либо путем выбора из списка. Некоторые свойства: Text (текст, находящийся в поле ввода-редактирования); Items (элементы раскрывающегося списка); DropDownCount (количество отображаемых элементов в раскрытом списке). Список может быть сформирован таким же образом, как у компонента ListBox.

## Некоторые приемы работы с отладчиком

Отладчик позволяет выполнять программу построчно, по функциям или по блокам. Указав, какие части программы должны выполняться и когда выполнение программы должно приостановиться, можно быстро пропускать те участки программы, которые работают правильно, сосредоточившись на отладке тех частей, которые приводят к неверному выполнению программы. Для управления выполнением программы предназначены следующие команды, которые вызываются из выпадающего меню Run главного меню Delphi.

Run	Программа выполняется без остановки на каждой строке. Эта программа эквивалентна выполнению программы вне отладчика
Step Over	Программа выполняется построчно. Подпрограммы выполняются как одна строка
Trace Into	Программа выполняется построчно. Также выполняется каждая строка подпрограммы
Run To Cursor	Программа выполняется до той строки, на которой находится курсор в окне редактора кода
Program Pause	Выполнение программы приостанавливается
Program Reset	Выполнение программы завершается
Add Breakpoint	Можно указать точки прерывания при выполнении программы
Add Watch	Можно указать точки просмотра для слежения за изменением значений переменных или полей структур данных

Рассмотрим некоторые приемы отладки для Delphi версий 6,7. В процессе обучения программированию, очень удобно пользоваться пошаговым выполнением программы. Это позволяет посмотреть, как изменяются значения переменных, участвующих в программе в процессе пошагового выполнения. Какую последовательность действий предпринять? Установить курсор в тексте программы на вторую исполняемую строку после Begin и запустить команду Run To Cursor (F4). Программа выполнит этот оператор и остановится. В окне редактора кода выполненная строка будет помечена зеленой стрелкой слева и выделена синим цветом. Если навести курсор на название какой-нибудь переменной из предшествующей части программы, рядом с курсором появится справочное окошко, где приведено значение этой переменной. Далее командой Step Over (F8) запустить следующий шаг выполнения. Будет помечена следующая строка при выполнении. С помощью курсора мыши можно посмотреть, как изменились значения переменных при выполнении этого шага программы.

Часто при отладке пользуются окном точек просмотра. Пометить переменные для просмотра можно, наведя на них курсор и выполнив команду Add Watch (Ctrl+F5). Далее запустить пошаговое выполнение программы. Часто окно Watch List бывает закрыто окном редактора кода. Его надо отодвинуть.

Отладочные настройки отменяются командой Program Reset.

В Delphi XE с первого взгляда все выглядит иначе. Но принцип использования команд отладки остался тем же. Окно, в котором можно просматривать значения переменных при пошаговом выполнении программы, находится слева и называется Local Variables (см. рис 4.1).

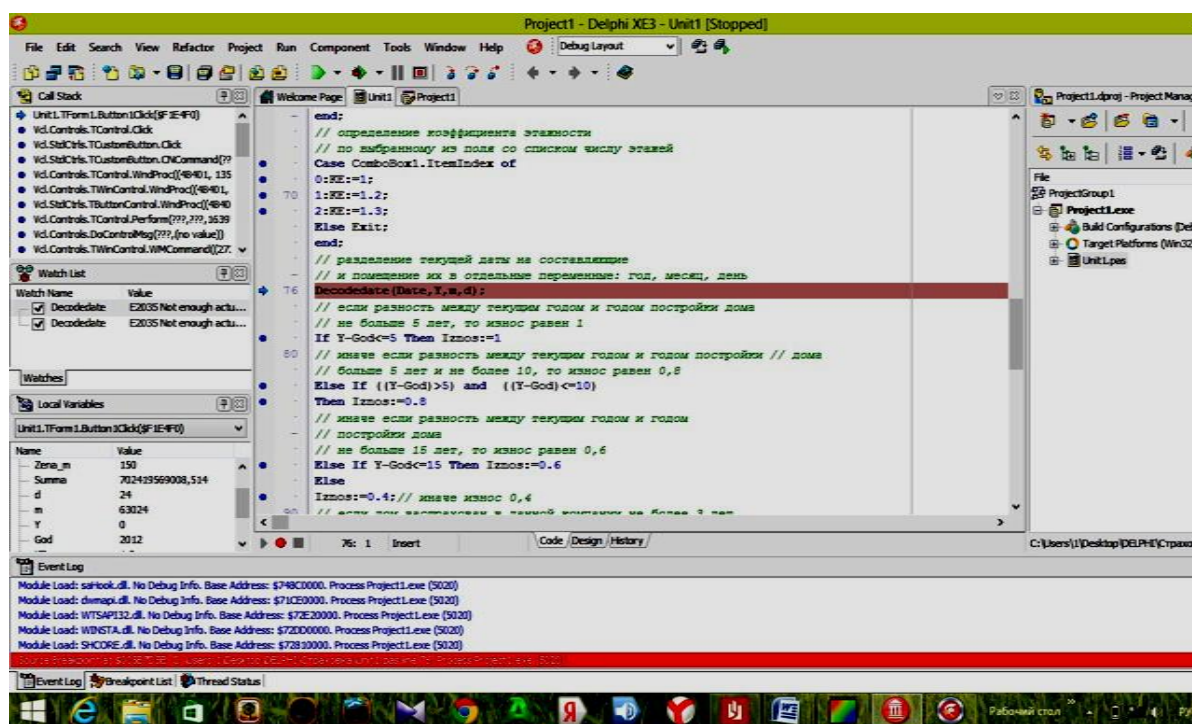


Рис. 4.1 Окно Delphi XE при использовании команд отладчика

**Замечание.** До процесса отладки программы на стадии выполнения (программа работает неправильно) нужно откомпилировать программу – исправить синтаксические ошибки. Наиболее часто встречающиеся сообщения об ошибках приведены в Приложении 2.

### Обработка исключительных ситуаций

Практически в любой даже высоко профессионально написанной программе можно найти «подводные камни», не говоря уже о студенческой программе. Причины возникающих при вы-

полнении программы ошибок могут быть разными. Например, абсурдные действия пользователя программы, сбой оборудования, бреши в системе безопасности и др. Поэтому разработчики Delphi придумали конструкции, которые могут помочь избежать прерывания работы программы с системным сообщением об ошибке выполнения.

Конструкция `try..except` позволяет создать в программе специальный участок кода, который выполняется при возникновении исключительной ситуации и пропускается в штатном режиме программы.

Синтаксис конструкции:

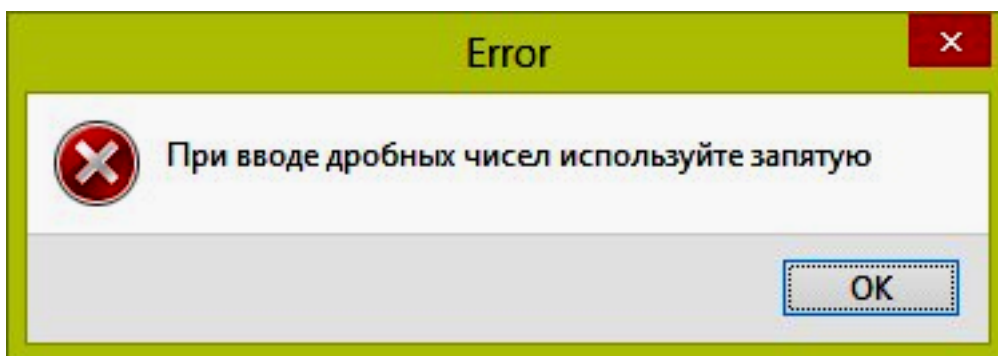
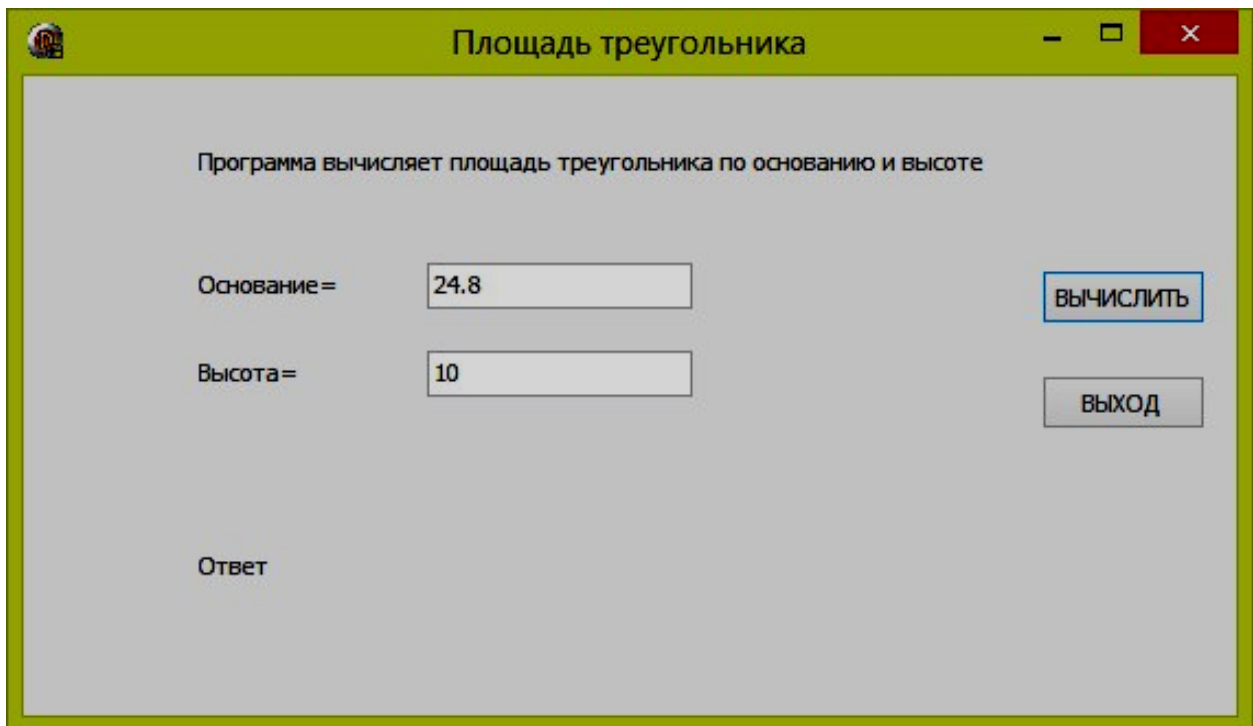
```
try
  {операторы защищенной секции}
except
  {операторы секции обработки исключительной ситуации}
end;
```

Например, пользователи программ часто забывают вводить данные в поля ввода или вместо десятичного разделителя запятой в вещественных числах ставят точку. И то и другое является исключительной ситуацией. Программа прекращает работу с системным сообщением об ошибке выполнения, которое в большинстве случаев непонятно пользователю. Покажем, как сделать программу более дружелюбной на примере вычисления площади треугольника из §1 данного учебного пособия. Код процедуры, связанной с событием щелчок по кнопке `Button1`: выглядит теперь так:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  try
    a:=StrToFloat(Edit1.Text);
    h:=StrToFloat(Edit2.Text);
    S:=1/2*a*h;
    S:=S/10000;
    Label4.Caption:='Площадь треугольника равна
'+FloatToStr(S)+'Г A';
  except
    if (Length(Edit1.Text) = 0) or (Length(Edit2.Text)=0)
```

```
then MsgBox('Надо ввести данные в оба поля', mtWarning, [mbOk], 0)
else MsgBox('При вводе дробных чисел используйте запятую', mtError, [mbOk], 0);
end;
```

В секцию try включены операторы, выполняющиеся, если пользователь заполнил все поля ввода правильно. Если при заполнении полей допущены ошибки, в секции exsert производится их анализ. Если некоторые поля не заполнены выдается окно с сообщением «Надо ввести данные в оба поля». Если целая и дробная части числа разделены точкой, выдается сообщение 'При вводе дробных чисел используйте запятую'. Можно нажать кнопку ОК и исправить ошибки в полях ввода.



Конструкция `try..finally` также предназначена для обработки исключительных ситуаций. В защищенную секцию `try` заносится потенциально опасный код. А в секцию завершения `finally` заносятся операторы, которые выполняются в любом случае.

## Практика

### *Задача 4а*

Составить программу контроля веса, определяющую по росту и весу пользователя программы, нужно ли ему похудеть или поправиться и на сколько килограммов.

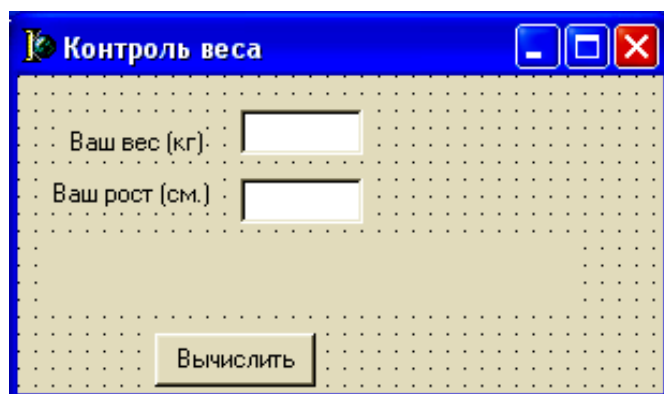
### Компоненты

Имя компонента	Свойства компонентов	Значение	Назначение
Form1	Caption	Контроль веса	Заголовок формы
Label1	Caption	поле для ввода веса	Подсказка пользователю
Label2	Caption	поле для ввода роста	Подсказка пользователю
Label3	Caption	Поле должно быть очищено от значения, заданного по умолчанию	поле вывода результатов работы программы
Button1	Caption	Вычислить	Кнопка для вычисления

### Переменные

Обозначение в программе	Содержание	Тип
wes	вес	вещественный
Rost	рост	вещественный
opt	оптимальный вес	вещественный
d	отклонение от оптимального веса	вещественный

### Проект формы

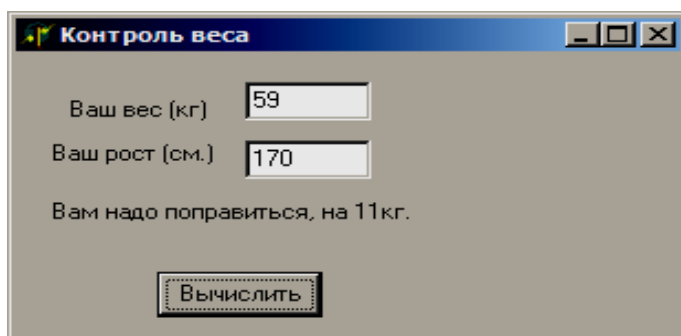




## Текст модуля

```
unit wtest;
.....
var
  Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
var
  wes:real; {вес}
  Rost:real; {рост}
  opt:real; {ОПТИМАЛЬНЫЙ вес}
  d:real; {ОТКЛОНЕНИЕ ОТ ОПТИМАЛЬНОГО веса}
begin
  wes:=StrToFloat(Edit1.text);
  Rost:=StrToFloat(Edit2.Text);
  opt:=Rost-100;
  if wes=opt
  then
    Label3.caption:='Ваш вес оптимален!'
  else
    if wes<opt
    then
      begin
        d:=opt-wes;
        Label3.caption:='Вам надо поправиться, на'+ FloatToStr(d)+ 'кг.';
      end
    else
      begin
        d:=wes-opt;
        Label3.caption:='Надо немного похудеть, на '+ FloatToStr(d)+ 'кг.';
      end;
  end;
end;
end.
```

## Результаты работы программы



### **Задача 4b**

Некоторая страховая компания страхует индивидуальные дома по следующей схеме.

Стоимость страховки = Площадь \* Цена\_кв.м \* Коэф.этажности \* \*Износ/100\*Проценты\*Курс\_у.е..

Цена\_кв.м:

Кирпичный коттедж – 200 у.е.

Дом из оцилиндрованного бревна – 250 у.е.

Дом из бруса – 150 у.е.

Дом из бревен – 100 у.е.

Каркасно-щитовой – 50 у.е.

Коэффициент этажности:

1 этаж – 1

2 этажа – 1,2

3 этаж – 1,3

Коэффициент износа:

Возраст дома:

Более 15 лет – 0,4

От 11 до 15 лет – 0,6

от 6 до 10 лет – 0,8

до 5 лет – 1

Проценты страхового сбора:

Количество лет страхования в данной страховой компании

Превышает 7 лет – 1 процент

От 3 до 7 лет – 2 процента

До 3 лет (включительно) – 3 процента

### **Компоненты**

Имя компонента	Свойства компонентов	Значение	Назначение
Form1	Caption	Сумма страховки индивидуального дома	Заголовок формы
Label1	Caption WordWrap	Программа для расчета суммы страхования индивидуального дома в некоторой страховой компании True	Справочная информация для пользователя программы
Label2	Caption	Общая площадь (кв.м)	Подсказка пользователю
Label3	Caption	Год постройки	Подсказка пользователю
Label4	Caption	Курс у.е.	Подсказка пользователю
Label5	Caption	Сколько лет дом страховался в данной страховой компании	Подсказка пользователю

Label6	Caption	Сумма страховки	Поле для вывода суммы страховки дома
Label7	Caption	Дата	Поле для вывода текущей даты установленной на компьютере
Edit1	Text	Поле должно быть очищено от значения, заданного по умолчанию	Поле для ввода общей площади
Edit2	Text	Поле должно быть очищено от значения, заданного по умолчанию	Поле для ввода года постройки дома
Edit3	Text	Поле должно быть очищено от значения, заданного по умолчанию	Поле для ввода курса у.е.
Edit4	Text	Поле должно быть очищено от значения, заданного по умолчанию	Поле для ввода количества лет страхования в данной компании
Button1	Caption	РАСЧЕТ	Кнопка для вычисления суммы страховки
Button2	Caption	ВЫХОД	Кнопка закрытия формы и выхода из программы
ListBox1	Items	Кирпичный коттедж Дом из оцилиндрованного бревна Дом из бруса Дом из бревен Каркасно-щитовой	Список для выбора типа дома
ComboBox1	Text Items	Количество этажей 1 2 3	Поле со списком

## Переменные

Обозначение в программе	Содержание	Тип
S	Общая площадь индивидуального дома	вещественный
Zena_m	Цена кв. метра	денежный
Summa	Сумма страхового взноса	денежный
d	Текущий день	Целый беззнаковый
m	Текущий месяц	Целый беззнаковый
Y	Текущий год	Целый беззнаковый
God	Год постройки дома	Целый беззнаковый
KE	Коэффициент этажности	вещественный
Iznos	Коэффициент износа	вещественный
Prozent	Процент страхового взноса	вещественный
Kurs	Курс у.е.	вещественный

## Проект формы

Сумма страховки индивидуального дома

Программа для расчета суммы страхования индивидуального дома в некоторой страховой компании

Кирпичный коттедж  
Дом из оцилиндрованного бруса  
Дом из бруса  
Дом из бревен  
Каркасно-щитовой

Количество этажей

РАСЧЕТ

ВЫХОД

Общая площадь (кв. м)

Год постройки

Курс у.е.

Сколько лет дом страховался в данной страховой компании

Сумма страховки

Дата

## Текст модуля

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  
Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;
```

```
type
```

```
TForm1 = class(TForm)  
Label1: TLabel;  
ListBox1: TListBox;  
ComboBox1: TComboBox;  
Label2: TLabel;  
Label3: TLabel;  
Label4: TLabel;  
Label5: TLabel;  
Label6: TLabel;
```

```

Label7: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Edit4: TEdit;
Button1: TButton;
Button2: TButton;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;

implementation

  {$R *.dfm}

  procedure TForm1.Button1Click(Sender: TObject);

    Var S:Real; //общая площадь дома
    Zena_m, Summa: currency;// цена кв.м, сумма страховки дома
    d,m,Y,God: Word; // день, месяц, год страхования; год
по//стройки дома
    KE:Real; // коэффициент, учитывающий этажность
    Iznos:Real;// коэффициент износа
    Prozent:Integer; // процент страхового взноса
    Kurs:Real; // курс у.е.
  begin
    // ввод площади дома, года постройки и курса у.е.
    S:=StrToFloat(Edit1.Text);
    God:=StrToInt(Edit2.Text);
    Kurs:=StrToFloat(Edit3.Text);
    //вывод текущей даты на форму

```

```

Label7.Caption:='Сегодня : '+DateToStr(Date);
// по выбранному из списка типу дома определяется цена за
//кв.м
Case ListBox1.ItemIndex of
0:Zena_m:=200;
1:Zena_m:=250;
2:Zena_m:=150;
3:Zena_m:=100;
4:Zena_m:=50;
Else Exit;
end;
// определение коэффициента этажности
// по выбранному из поля со списком числу этажей
Case ComboBox1.ItemIndex of
0:KE:=1;
1:KE:=1.2;
2:KE:=1.3;
Else Exit;
end;
// разделение текущей даты на составляющие
// и помещение их в отдельные переменные: год, месяц, день
Decodedate(Date,Y,m,d);
// если разность между текущим годом и годом постройки дома
// не больше 5 лет, то износ равен 1
If Y-God<=5 Then Iznos:=1
// иначе если разность между текущим годом и годом постройки
// дома
// больше 5 лет и не более 10, то износ равен 0,8
Else If ((Y-God)>5) and ((Y-God)<=10)
Then Iznos:=0.8
// иначе если разность между текущим годом и годом
// постройки дома
// не больше 15 лет, то износ равен 0,6
Else If Y-God<=15 Then Iznos:=0.6
Else
Iznos:=0.4;// иначе износ 0,4
// если дом застрахован в данной компании не более 3 лет,
// то процент страхового взноса равен 3

```

```

If StrToInt(Edit4.Text)<=3 Then Prozent:=3
// иначе дом застрахован в данной компании более 7 лет,
// то процент страхового взноса равен 1
Else If StrToInt(Edit4.Text)>7 Then Prozent:=1
// иначе дом страхуется от 3 до 7 лет,
// процент страхового взноса равен 2
Else Prozent:=2;
// расчет суммы страхового взноса
Summa:=S*Zena_m*KE*Iznos/100*Prozent*Kurs;
// вывод суммы страховки на форму
Label6.Caption:='Сумма страховки Вашего дома составляет: '
+FloatToStrF(Summa,Ffixed,10,2)+' руб.'

```

end;

```

procedure TForm1.Button2Click(Sender: TObject);

```

```

begin

```

```

Form1.Close;

```

```

end;

```

end.

## Результаты работы программы

Сумма страховки индивидуального дома

Программа для расчета суммы страхования индивидуального дома в некоторой страховой компании

Кирпичный коттедж  
Дом из оцилинрованного бруса  
Дом из бруса  
Дом из бревен  
Каркасно-щитовой

2

РАСЧЕТ

ВЫХОД

Общая площадь (кв. м) 128,7

Год постройки 2012

Курс у.е. 35,5

Сколько лет дом страховался в данной страховой компании 1

Сумма страховки Вашего дома составляет: 24671,79 руб.

Сегодня : 08.08.2014

## Рекомендуемый интернет-ресурс

<http://www.youtube.com/watch?v=96N8dQJWEao&list=PLDDB9D4A0E37B4953&index=3>

### *Вопросы для самоконтроля*

1. По каким правилам строятся логические выражения в Delphi?
2. Какой логический оператор имеет самый высокий приоритет?
3. Какой логический оператор or или and выполнится первым в логическом выражении?
4. Для чего предназначен оператор If?
5. Опишите синтаксис оператора If.
6. Опишите синтаксис краткой формы оператора If.
7. Для чего предназначен оператор перехода?
8. Для чего предназначен оператор выбора?
9. Опишите синтаксис оператора выбора.
10. Переменные какого типа могут быть переменными-селекторами в операторе выбора?
12. Для чего предназначен компонент ListBox?
13. Для чего предназначен компонент ComboBox?
14. Опишите конструкции для обработки исключительных ситуаций.
15. Расскажите как работает программа «Контроль веса».
16. По какой формуле рассчитывается оптимальный вес в программе «Контроль веса» и какая переменная его хранит?
17. Какое значение примет переменная Iznos в программе «Сумма страховки индивидуального дома», если дом был построен в 2008 году?
18. Какое значение примет коэффициент этажности в программе «Сумма страховки индивидуального дома», если страхуется двухэтажный дом?
19. Укажите фрагменты программы «Сумма страховки индивидуального дома», где используется оператор выбора?
20. Укажите фрагмент программы «Сумма страховки индивидуального дома», где определяется коэффициент износа?



## *Задачи для самостоятельного решения*

### Группа А

№ 4.1 Дано действительное число  $x$ . Вычислить  $f(x)$ , если

$$f(x) = \begin{cases} x^2 & \text{при } -2 \leq x \leq 2, \\ 4 & \text{в противном случае} \end{cases}$$

№ 4.2 Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу  $(1,3)$ .

№ 4.3 Даны действительные числа  $x$ ,  $y$  ( $x \neq y$ ). Меньшее из этих двух чисел заменить их полусуммой, а большее – их удвоенным произведением.

№ 4.4 Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны.

№ 4.5 Дано действительное число  $x$ . Вычислить  $f(x)$ , если

$$f(x) = \begin{cases} x^2 + 4x + 5 & \text{при } x \leq 2, \\ 1/(x^2 + 4x + 5) & \text{в остальных случаях} \end{cases}$$

№ 4.6 Дано действительное число  $x$ . Вычислить  $f(x)$ , если

$$f(x) = \begin{cases} 0, & \text{при } x \leq 0 \\ x & \text{при } 0 \leq x \leq 1 \\ x^4 & \text{в остальных случаях} \end{cases}$$

№ 4.8 Даны действительные числа  $a$ ,  $b$ ,  $c$ . Проверить, выполняются ли неравенства  $a < b < c$ .

№ 4.9 Дана оценка в баллах, которую получил студент при прохождении теста. Составить программу для перевода баллов тестирования в оценки по пятибалльной системе, если известно:

85 – 100 баллов – 5 (отлично)

65 – 84 баллов – 4 (хорошо)

51 – 64 баллов – 3 (удовлетворительно)

$\leq 50$  баллов – 2 (неудовлетворительно).

№ 4.10 Составить программу–калькулятор, которая вводит два числа и знак арифметического действия (+, -, \*, /). В зависимости от того, какой знак введен, программа складывает числа, вычитает из одного другое, умножает или делит.

### Группа Б

№ 4.11 Написать программу для определения площади треугольного участка по заданным значениям длин его сторон А, В и С. Для вычисления площади использовать формулу Герона

$$S = \sqrt{P(P - A)(P - B)(P - C)},$$

где  $P = \frac{A + B + C}{2}$  – полупериметр.

В программе предусмотреть проверку существования треугольника со сторонами А, В и С. Треугольник со сторонами А, В и С возможен лишь в том случае, если одновременно выполняются неравенства  $A + B > C$ ,  $A + C > B$ ,  $B + C > A$ . В случае, если треугольник с заданными значениями сторон не существует, выдавать сообщение: ОШИБКА, ПРОВЕРЬТЕ ИСХОДНЫЕ ДАННЫЕ и передавать управление в начало программы. Длины сторон вводить в метрах, площадь определять в гектарах с точностью до 1 га.

№ 4.12 Из вершины О трапециевидного участка (рис. 4–1) измерены расстояния А, С и В до трех остальных вершин и угол между сторонами А и В. Написать программу для вычисления площади этого участка по формуле:

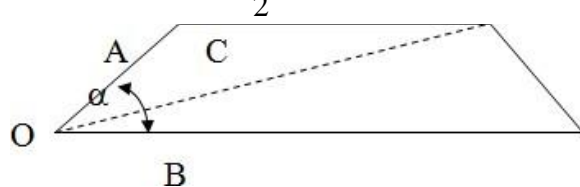
$$S = \frac{B + \sqrt{C^2 - A^2 \sin^2 \alpha} - A \cos \alpha}{2}$$


Рис. 4–1

Расстояния А, В и С измерены в метрах, угол  $\alpha$  в градусах, минутах и секундах. Площадь вычислить в гектарах, с точностью до 0.001 га.

Форма вывода результата:

Площадь = значение S в га

В программе предусмотреть следующие запросы при вводе:  
БОКОВАЯ СТОРОНА =?

ОСНОВАНИЕ =?

ДИАГОНАЛЬ =?

УГОЛ =?

В программе предусмотреть также проверку существования трапеции с введенными параметрами. Случаи, в которых трапецию построить нельзя:

– угол  $\alpha \geq 180^\circ$

$C < A * \sin \alpha$

$0 < \alpha \leq 90^\circ$  и  $C < A$ .

Во всех указанных ситуациях выводить сообщение ТРАПЕЦИИ НЕ СУЩЕСТВУЕТ, и передавать управление оператору ввода исходных данных.

№ 4.13 Написать программу для определения площади четырехугольника по значениям его четырех сторон A, B, C, D и диагонали (рис. 4–2) по формуле:

$$S = \sqrt{P_1(P_1 - A)(P_1 - B)(P_1 - L)} + \sqrt{P_2(P_2 - C)(P_2 - D)(P_2 - L)},$$

$$\text{где } P_1 = \frac{A + B + L}{2}, P_2 = \frac{C + D + L}{2}.$$

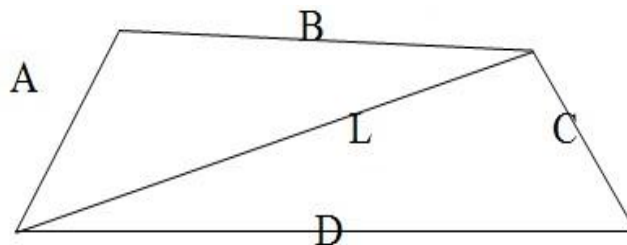


Рис. 4–2

Значения A, B, C, D, L вводить в метрах, площадь округлять до сотых гектара.

В программе предусмотреть проверку условия:

$$(A + B > L) \wedge (A + L > B) \wedge (B + L > A) \wedge (C + D > L) \wedge (C + L > D) \wedge (D + L > C)$$

В случае невыполнения этого условия выводить сообщение ОШИБКА В ИСХОДНЫХ ДАННЫХ и передавать управление оператору ввода исходных данных.

В программе предусмотреть следующие запросы при вводе исходных данных:

СТОРОНА СЛЕВА ОТ ДИАГОНАЛИ =?  
СТОРОНА СПРАВА ОТ ДИАГОНАЛИ =?  
ДИАГОНАЛЬ =?

Форма вывода результата:

ПЛОЩАДЬ = значение S ГА.

№ 4.14 Написать программу, которая определяет величину уклона на участке площадью  $P$  га по заданному сечению горизонталей  $H$  и длине всех горизонталей  $C$ :

$$I = \frac{H * C}{175P}.$$

Если вычисленное значение  $I \leq 3$  выводить сообщение:

ИЗМЕРЕНИЕ ПЛОЩАДЕЙ ПО ФОТОСХЕМЕ ДОПУСТИМО:

$I \leq 3$  гр.

В противном случае выводить сообщение:

ИЗМЕРЕНИЕ ПЛОЩАДЕЙ ПО ФОТОСХЕМЕ НЕДОПУСТИМО:

$I > 3$  гр.

Форма запросов при вводе исходных данных:

ПЛОЩАДЬ =?

СЕЧЕНИЕ ГОРИЗОНТАЛЕЙ =?

ДЛИНА ГОРИЗОНТАЛЕЙ =?

### Примечание

*Уклон линии* – отношение превышения  $h$  между точками к горизонтальному проложению  $S$  между ними.

*Превышение* – разность высот точек.

*Горизонтальное проложение* – ортогональная проекция линии местности на горизонтальную плоскость.

*Высоты горизонталей* всегда кратны высоте сечения рельефа. *Высота сечения рельефа* – это расстояние, на которое отстоят друг от друга уровенные поверхности, пересекающие земную поверхность.

№ 4.15 Написать программу для вычисления предполагаемого валового сбора зерна по формулам:

$$Y = \begin{cases} 218,36 + 73,14x_1 + 39,56x_2 + 56,84x_3 & \text{для } R < 56,34 \\ 431,45 + 132,6x_1 + 84,51x_2 + 112,59x_3 & \text{для } R \geq 56,34 \end{cases}$$

где  $x_1, x_2, x_3$  – площади почвенных групп (в га),

$R$  – текущая норма осадков (в мм).

Форма запросов при вводе исходных данных и сообщений при выводе результатов.

ПЛОЩАДИ ПОЧВЕННЫХ ГРУПП:

$X1=$

$X2=$

$X3=$

ТЕКУЩАЯ НОРМА ОСАДКОВ=

ПРЕДПОЛАГАЕМЫЙ ВАЛОВЫЙ СБОР ЗЕРНА = значение  $Y$  центнеров.

№ 4.16 Написать программу для определения положения точки с координатами  $X1$ ,  $Y1$  по отношению к полосе шириной  $P$ , ось которой описывается уравнением  $AX + BY + C = 0$ .

Расстояние от точки с координатами  $X1$ ,  $Y1$  до прямой можно вычислить по формуле:

$$R = \left| \frac{AX_1 + BY_1 + C}{\sqrt{A^2 + B^2}} \right|.$$

Условием нахождения точки в указанной полосе является неравенство  $R \leq P/2$ .

В программе предусмотреть запросы на ввод исходных данных:

$A$ ,  $B$ ,  $C=$

$P=$

$X1$ ,  $Y1=$

Результат выводить в виде одного из сообщений:

ТОЧКА В ПРЕДЕЛАХ ПОЛОСЫ или

ТОЧКА ЗА ПРЕДЕЛАМИ ПОЛОСЫ

№4.17 Написать программу для определения положения точки с координатами  $X1$ ,  $Y1$  по отношению к круглому участку радиуса  $R$  с центром в точке с координатами  $X_0$ ,  $Y_0$ .

Признаком нахождения точки внутри или на границе участка является выполнение неравенства

$$\sqrt{\Delta X^2 + \Delta Y^2} \leq R,$$

где  $\Delta X = X1 - X_0$ ,  $\Delta Y = Y1 - Y_0$ .

В программе предусмотреть выдачу запросов на ввод исходных данных:

$X_0, Y_0 =$

$R =$

$X_1, Y_1 =$

Результаты выводить в виде одного из сообщений:

ТОЧКА В ПРЕДЕЛАХ УЧАСТКА или

ТОЧКА ЗА ПРЕДЕЛАМИ УЧАСТКА

№4.18 Написать программу для определения положения точки с координатами  $X_1, Y_1, Z_1$  по отношению к сфере с радиусом  $R$  и координатами центра  $X_0, Y_0, Z_0$ . Признаком нахождения точки внутри сферы или на ее границе является выполнение неравенства

$$\sqrt{\Delta X^2 + \Delta Y^2 + \Delta Z^2} \leq R,$$

$$\Delta X = X_1 - X_0, \Delta Y = Y_1 - Y_0, \Delta Z = Z_1 - Z_0.$$

где в программе предусмотреть выдачу запросов на ввод исходных данных:

$X_0, Y_0, Z_0 =$

$R =$

$X_1, Y_1, Z_1 =$

Результаты выводить в виде одного из сообщений:

ТОЧКА ЗА ПРЕДЕЛАМИ СФЕРЫ

ТОЧКА В ПРЕДЕЛАХ СФЕРЫ

№4.19 Написать программу для определения положения точки с координатами  $X_1, Y_1$  по отношению к прямой, описываемой уравнением  $AX + BY + C = 0$ . Признаком того, что точка лежит на прямой будем считать выполнение неравенства :

$$|AX_1 + BY_1 + C| < 0,001.$$

Форма запросов при вводе исходных данных:

$A, B, C =$

$X_1, Y_1 =$

Форма вывода результата:

ТОЧКА НА ПРЯМОЙ или

ТОЧКА ВНЕ ПРЯМОЙ,

в зависимости от выполнения или невыполнения заданного условия.

№4.20 Написать программу, которая определяет положение прямой, заданной уравнением  $AX + BY + C = 0$  относительно прямоугольной системы координат. Возможные ситуации:

$A=0, B=0, C$  – любое – ошибка при вводе исходных данных, прямой не существует,

$A=0, B \neq 0, C=0$  – прямая совпадает с осью  $X$ ,

$A \neq 0, B=0, C=0$  – прямая совпадает с осью  $Y$ ,

$A=0, B \neq 0, C \neq 0$  – прямая проходит через точку  $Y = -C/B$   
параллельно оси  $X$ ,

$A \neq 0, B=0, C \neq 0$  – прямая проходит через точку  $Y = -C/A$   
параллельно оси  $Y$ ,

$A \neq 0, B \neq 0, C=0$  – прямая проходит через начало координат и точку  
с произвольным значением  $X$  (предусмотреть  
в случае возникновения указанной ситуации  
ввод значения  $X$ ) и  $Y = -AX/B$ .

$A \neq 0, B \neq 0, C \neq 0$  – прямая пересекает ось  $X$  в точке  $X = -C/A$  и ось  
 $Y$  в точке  $Y = -C/B$ .

Запросы при вводе исходных данных:

ПАРАМЕТРЫ  $A, B, C =$

Форма вывода результатов:

СОВПАДАЕТ С ОСЬЮ  $X$

или СОВПАДАЕТ С ОСЬЮ  $Y$

и т. п.

В случае  $A \neq 0, B \neq 0, C=0$  вначале выводить запрос: ВВЕДИТЕ  
ЛЮБОЕ ЗНАЧЕНИЕ  $X$ , а затем сообщение:

ПРОХОДИТ ЧЕРЕЗ НАЧАЛО КООРДИНАТ И ТОЧКУ

$X =$  значение,  $Y =$  значение.

№ 4.21 Написать программу, которая определяет, лежит ли  
точка с координатами  $X, Y$  на прямой, проведенной через точки с  
координатами  $X_1, Y_1$  и  $X_2, Y_2$ . Признаком, того что точка лежит  
на прямой, будем считать выполнение неравенства.

$$\left| \frac{Y - Y_1}{Y_2 - Y_1} - \frac{X - X_1}{X_2 - X_1} \right| < 0,01$$

Форма вывода результата:

ТОЧКА НА ПРЯМОЙ, если точка лежит на прямой, и

ТОЧКА ВНЕ ПРЯМОЙ, если точка лежит не на прямой.

Форма запроса при вводе исходных данных:

КООРДИНАТЫ  $X_1, Y_1 =$

КООРДИНАТЫ  $X_2, Y_2 =$

КООРДИНАТЫ  $X, Y =$

## Глава 5. Организация циклов

*В данной главе рассматриваются следующие вопросы:* алгоритмы циклических вычислений, операторы цикла (For do, While do, Repeat Until).

### Теория

#### Алгоритмы циклических вычислений

Прежде чем приступить к рассмотрению операторов цикла рассмотрим алгоритмы, для реализации которых они могут понадобиться.

**А.** Опишем алгоритм, реализующий суммирование ряда целых чисел в диапазоне значений от  $N1$  до  $N2$  с шагом единица.

1. Введем исходные данные  $N1$  и  $N2$  (крайние значения рассматриваемого диапазона целых чисел).

2. Заведем переменную  $S$ , в которой будет накапливаться сумма, и присвоим ей значение нуля.  $S:=0$ ;

3. Заведем переменную  $N$ , которая будет содержать текущее значение числа, и присвоим ей значение крайней левой границы диапазона чисел.  $N:=N1$ ;

4. Получим новое значение переменной  $S$ , прибавив к ее предыдущему значению текущее значение переменной  $N$ .

$S:=S + N$ ;

5. Увеличим значение переменной  $N$  на величину шага (в нашем случае на единицу).  $N:= N + 1$ ;

6. Проверим значение переменной  $N$ . Если  $N$  меньше или равно значению крайней правой границы диапазона чисел ( $N \leq N2$ ) вернемся к пункту 4 алгоритма. В противном случае перейдем к следующему пункту.

7. Выведем переменную  $S$ , которая содержит накопленную сумму чисел.

**В.** Рассмотрим алгоритм получения произведения ряда целых чисел, изменяющихся в диапазоне значений  $K1$  и  $K2$  с шагом  $Step$ , не равным единице ( $Step \neq 1$ ).

1. Введем исходные данные  $K1$ ,  $K2$ ,  $Step$ .



2. Заведем переменную  $P$ , которая будет накапливать произведение чисел, и присвоим ей значение 1, так как умножение на единицу не влияет на произведение.  $P:=1$ .

3. Заведем переменную  $K$ , которая будет содержать текущее значение числа, участвующего в накоплении произведения, и присвоим ей значение крайней левой границы диапазона перемножаемых чисел.  $K:=K_1$ .

4. Получим новое значение произведения  $P$ , как предыдущее значение  $P$ , умноженное на текущее значение числа  $K$ .  $P:=P * K$ ;

5. Увеличим текущее значение числа  $K$  на величину шага  $Step$ .

$K:=K + Step$ ;

6. Сравним значение текущего числа  $K$  с крайней правой границей диапазона чисел  $K_2$ . Если  $K$  меньше или равно  $K_2$ , то вернемся к пункту 4 данного алгоритма. Иначе перейдем к следующему пункту алгоритма.

7. Выведем значение переменной  $P$ , которая содержит накопленное произведение чисел.

**С.** Составим алгоритм получения и вывода значений функции  $F(x)$  для значений аргументов, меняющихся в диапазоне от  $X_1$  до  $X_2$  с шагом  $Step$ .

1. Введем исходные данные  $X_1, X_2, Step$ .

2. Заведем переменную  $X$  для хранения текущего значения аргумента и присвоим ей значение крайней левой границы изменения аргументов.  $X:=X_1$ ;

3. Вычислим  $F(X)$ .

4. Выведем  $X$  и  $F(X)$ .

5. Увеличим значение текущего аргумента функции  $X$  на величину шага.  $X:=X + Step$ ;

6. Если  $X$  меньше или равно правой границы диапазона изменения аргументов  $X_2$  ( $X \leq X_2$ ), то возвращаемся к пункту 3. В противном случае программа заканчивает свою работу.

**Д.** Разберем алгоритм для создания цикла, выход из которого произойдет при достижении заданной точности вычислений  $\varepsilon$ .

Допустим нам нужно подсчитать сумму разложения, где каждый следующий член меньше предыдущего.

$$\sum_{i=1}^{\infty} b_i = b_1 + b_2 + \dots + b_n + \dots$$

( $b_i$  рассчитывается по заданной формуле).

Вычисление членов разложения и их суммы нужно прекратить, как только модуль разности последнего и предпоследнего из вычисленных членов станет меньше, чем заданная величина точности  $\varepsilon$ , т.е.

$$|b_n - b_{n-1}| \leq \varepsilon.$$

1. Введем точность вычислений EPS.
2. Присвоим переменной, где будет храниться сумма, значение ноль. SUM:=0;
3. Начнем вычисления с первого члена разложения. Подсчитаем первый член  $b_1$  по формуле, известной из условия.
4. Прибавим к сумме значение  $b_1$ : SUM:=SUM +  $b_1$ ;
5. Вычислим следующий член разложения  $b_2$ , по формуле известной из условия.
6. Прибавим к сумме  $b_2$ : SUM:=SUM +  $b_2$ ;
7. Найдем разность текущего члена  $b_2$  и предыдущего  $b_1$ :  
R:=  $b_2 - b_1$ ;
8. Сравним модуль разности R с заданной точностью вычислений EPS. Если заданная точность не достигнута  $|R| > EPS$ , то предыдущему члену  $b_1$  присваиваем значение текущего  $b_2$ :  $b_1 := b_2$  и возвращаемся к пункту 5. Иначе, если заданная точность вычислений достигнута, переходим к пункту 9.
9. Выводим сумму разложения SUM.

### Операторы цикла

*Операторы цикла* используются для организации циклов (повторов). *Цикл* представляет собой последовательность операторов, которая может выполняться более одного раза. Группу повторяемых операторов называют телом цикла. Всего в Delphi имеется три вида операторов цикла:

- с параметром;
- с предусловием;
- с постусловием.

Обычно, если количество повторов известно заранее, то применяется оператор цикла с параметром, в противном случае – операторы с предусловием или с постусловием (чаще используются операторы с предусловием).

Операторы циклов могут быть вложены друг в друга.

### Оператор цикла с параметром

*Оператор цикла с параметром* имеет два следующих формата:

**for <Параметр> := <Выражение1> to <Выражение2> do <Оператор>;**

и

**for <Параметр> := <Выражение1> downto <Выражение2> do <Оператор>;**

*Параметр цикла* представляет собой переменную порядкового типа, которая должна быть определена в том же блоке, где находится оператор цикла, выражение1 и выражение2 являются соответственно начальным и конечным значениями параметра цикла и должны иметь тип, совместимый с типом параметра цикла. Оператор цикла обеспечивает выполнение тела цикла, которым является оператор после слова **do**, до полного перебора всех значений параметра цикла от начального до конечного с соответствующим шагом. Шаг параметра всегда равен 1 для первого формата цикла и -1 – для второго формата. То есть значение параметра последовательно увеличивается (for.. .to) или уменьшается (for.. .downto) на единицу при каждом повторении цикла. Цикл может не выполниться ни разу, если для цикла for...to значение начального выражения больше конечного, а для цикла for...downto, наоборот, значение начального выражения меньше конечного.

**Пример.** Вычислит сумму и произведение заданных рядов натуральных чисел.

```
var n, k: integer;
    s:Real;
    p:Extended;
begin
  {цикл вычисления суммы ряда 1+1/2+1/3+...+1/10}
  s := 0;
  for n := 1 to 10 do
```

```

s := s + 1/n;
{ цикл вычисления произведения членов ряда
1*1/2*1/3*...*1/20}
p:=1;
for k := 1 to 20 do
p:=p*1/k;

```

### Оператор цикла с предусловием

*Оператор цикла с предусловием* целесообразно использовать в случаях, когда число повторений тела цикла заранее неизвестно. (Например, когда параметр цикла не равен единице, или цикл должен выполняться до достижения заданной точности).

Формат оператора цикла с предусловием:

```
while <Условие> do <Оператор>;
```

Оператор тела цикла выполняется до тех пор, пока <Условие> (логическое выражение) не примет значение False. То есть цикл выполняется при значении логического выражения True.

**Замечание.** Если в теле цикла нужно выполнить больше одного оператора, то эти операторы оформляются в виде одного составного оператора, то есть заключаются в операторные скобки (служебные слова begin и end).

**Пример.** Оператор цикла с предусловием.

Рассмотрим расчет суммы натуральных чисел от 1 до 100 с шагом 3

```

var i: integer;
sum: integer;
begin
sum := 0;
i:=1;
while i <= 100 do
begin
sum := sum + i;
i := i + 3;
end;
Label2.Caption:='Сумма = ' + IntToStr(Sum);
end;

```

Если перед первым выполнением цикла условие не выполняется (значение логического выражения равно False), то тело цикла не выполняется ни разу, и происходит переход на оператор, следующий за оператором цикла (Label2.Caption).

### Оператор цикла с постусловием

Во многом этот оператор аналогичен оператору цикла с предусловием, но проверка условия выполняется в конце оператора. Выход из цикла происходит при равенстве условия константе True (Истина).

Формат оператора цикла с постусловием:

```
repeat <Операторы>; until <Условие>;
```

#### Пример

Найдем сумму  $\sum_{n=1}^{10} \frac{n}{(x-n)}$ , для нечетных значений  $n$ , при  $x-n \neq 0$ .

```
x:=strToInt(Edit1.text);
n:=1;
Sum := 0;
Repeat
if x-n <> 0 then
Sum := Sum + n/(x-n);
n:=n+2;
Until n>10;
Label2.Caption:='Сумма для нечетных n = ' + IntToStr(Sum);
```

В операторе Repeat Until условие проверяется после выполнения операторов тела цикла. Следовательно, операторы цикла в любом случае выполняются хотя бы один раз.

### Операторы break и continue

Операторы break и continue используются только с операторами цикла. Оператор break прерывает текущую итерацию цикла.

Например, если в цикле может возникнуть ситуация деления на ноль, ее можно пропустить с помощью оператора break:

```
x:=strToInt(Edit1.text);
n:=1;
```

```

Sum := 0;
Repeat
If (x-n) = 0 Then break;
Sum := Sum + n/(x-n);
n:=n+2;
Until n>10;

```

Оператор continue запускает внеочередное начало следующей итерации. В приведенном ниже фрагменте программы подсчитывается сумма нечетных чисел в диапазоне от 1 до 10.

```

Var y, Sum :Integer;
begin
Sum:=0;
For y:=1 To 10 do
begin
if (y MOD 2)=0 then continue; {пропускаются итерации с
четным значением y}
Sum:= Sum+y
end;
Label2.Caption:='Сумма нечетных чисел = ' + IntToStr(Sum);
end;

```

**Примечание.** Приведем программный код алгоритмов, рассмотренных в начале параграфа.

Суммирование ряда целых чисел в диапазоне значений от N1 до N2 с шагом один.

```

Var N1,N2, N, Sum: Integer;
begin
N1:=StrToInt(Edit1.Text);
N2:=StrToInt(Edit2.Text);
Sum:=0;
For N:=N1 To N2 do
Sum:=Sum+N;
Label3.Caption:='Сумма равна ' + IntToStr(Sum);
end;

```

Произведение ряда чисел из диапазона значений K1 и K2 с шагом Step.

```

Var K1, K2, K, Step, P : Real;

```

```

begin
K1:=StrToFloat(Edit1.Text);
K2:=StrToFloat(Edit2.Text);
Step:=StrToFloat(Edit3.Text);
P:=1; K:=K1;
While K<=K2 do
begin
P:=P*K;
K:=K+ Step;
end;
Label4.Caption:='Произведение равно ' +
FloatToStrF(P,FFixed,10,2 );
end;
end;

```

Расчет и вывод ряда значений функции  $F(x)=x^3$  для аргументов  $x$ .

```

Var X1, X2, X, Step, F : Real;
begin
X1:=StrToFloat(Edit1.Text);
X2:=StrToFloat(Edit2.Text);
Step:=StrToFloat(Edit3.Text);
X:=X1;
Label4.Caption:=' ';
While X<=X2 do
begin
F:=exp(3*ln(X));
Label4.Caption:=' Label4.Caption +#13+' X= ' +
FloatToStrF(X, FFixed,10,2) +' ' + 'F(X)= ' +
FloatToStrF(F, FFixed,10,2);
X:=X+ Step;
end;
end;

```

Этот же алгоритм можно перевести в программный код, используя оператор цикла Repeat Until.

```

Var X1, X2, X, Step, F : Real;
begin
X1:=StrToFloat(Edit1.Text);

```

```

X2:=StrToFloat(Edit2.Text);
Step:=StrToFloat(Edit3.Text);
X:=X1;
Label4.Caption:=' ';
Repeat
F:=exp(3*ln(X));
Label4.Caption:=' Label4.Caption +#13+' X= ' +
FloatToStrF(X, FFfixed,10,2 ) +' ' + 'F(X)=' +
FloatToStrF(F, FFfixed,10,2 );
X:=X+ Step;
Until X>X2;
end;

```

## Практика

### *Задача 5а*

Для N случайных целых чисел из интервала от 0 до M вычислить и вывести на форму аргументы  $x$  и соответствующие значения функции  $y$

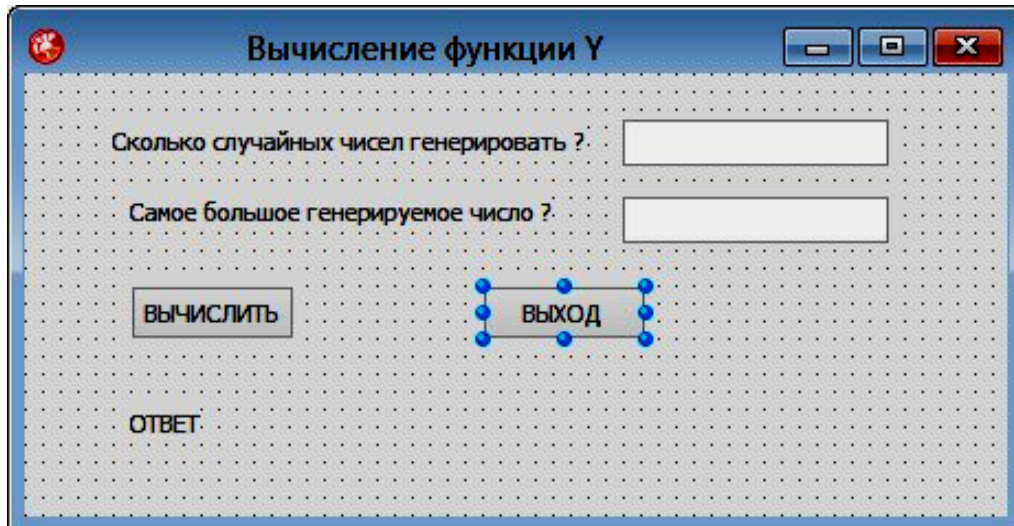
$$y = \begin{cases} x^2, & \text{для нечетных значений } x \\ x, & \text{для четных значений } x \end{cases}$$

### Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Вычисление функции Y	Заголовок формы
Label1	Caption	Сколько случайных чисел генерировать ?	Подсказка для пользователя программы
Label2	Caption	Самое большое генерируемое число?	Подсказка для пользователя программы
Label3	Caption	Ответ	Поле для вывода результатов
Edit1	Text	Должно быть очищено от значения по умолчанию	Поле для ввода начального значения n
Edit2	Text	Должно быть очищено от значения по умолчанию	Поле для ввода конечного значения n
Button1	Caption	Вычислить	Кнопка для вычисления
Button2	Caption	Выход	Кнопка закрытия формы и выхода из программы



## Проект формы



### Текст модуля

```
unit Unit1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, Sys-
  tem.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;
    Label3: TLabel;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```

var
  Form1: TForm1;

implementation

{$R *.dfm}

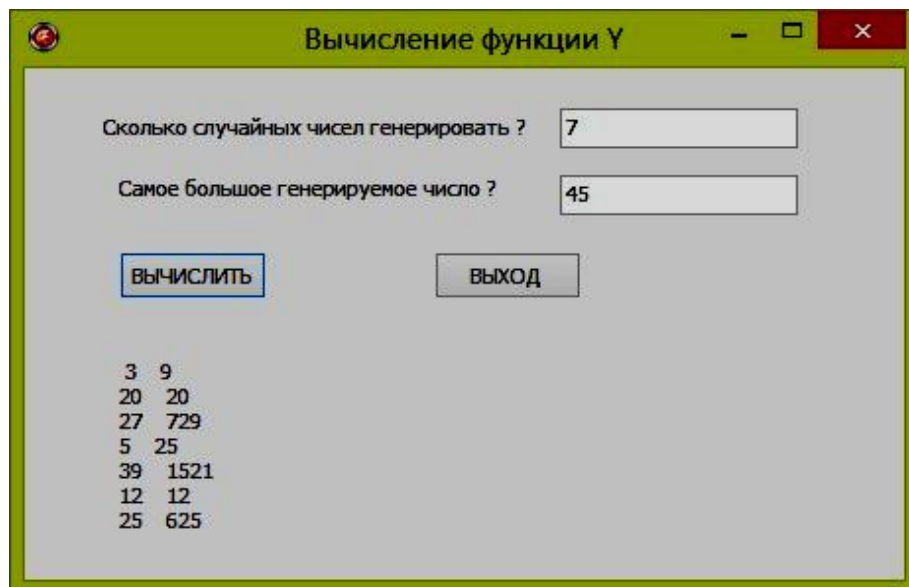
procedure TForm1.Button1Click(Sender: TObject);
  Var N, M, I, x: Integer;
      y: Real;
  { N – количество генерируемых случайных чисел
    M – самое большое генерируемое число. Правая граница
диапазона генерируемых случайных чисел
    I – порядковый номер генерируемого случайного числа
    x – сгенерированное случайное число
    y – значение функции y }
begin
  N:=StrToInt(Edit1.Text);
  M:=StrToInt(Edit2.Text);
  Label3.Caption:=' ';
  Randomize;
  for I := 1 to N do
  begin
    x:=Random(M);
    if Frac (x/2)=0 Then y:=x
      Else y:=Sqr(x);
    Label3.Caption:= Label3.Caption+ IntToStr(x)+ ' ' +
FloatToStrF(y, FFixed,10,0) + #13;
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close ;
end;

end.

```

## Результаты работы:



### **Задача 5b**

Составить программу для вычисления таблиц значений квадратов, кубов, квадратных и кубических корней,  $\pi n$ ,  $\pi/(4n^2)$ ,  $1/n$ .

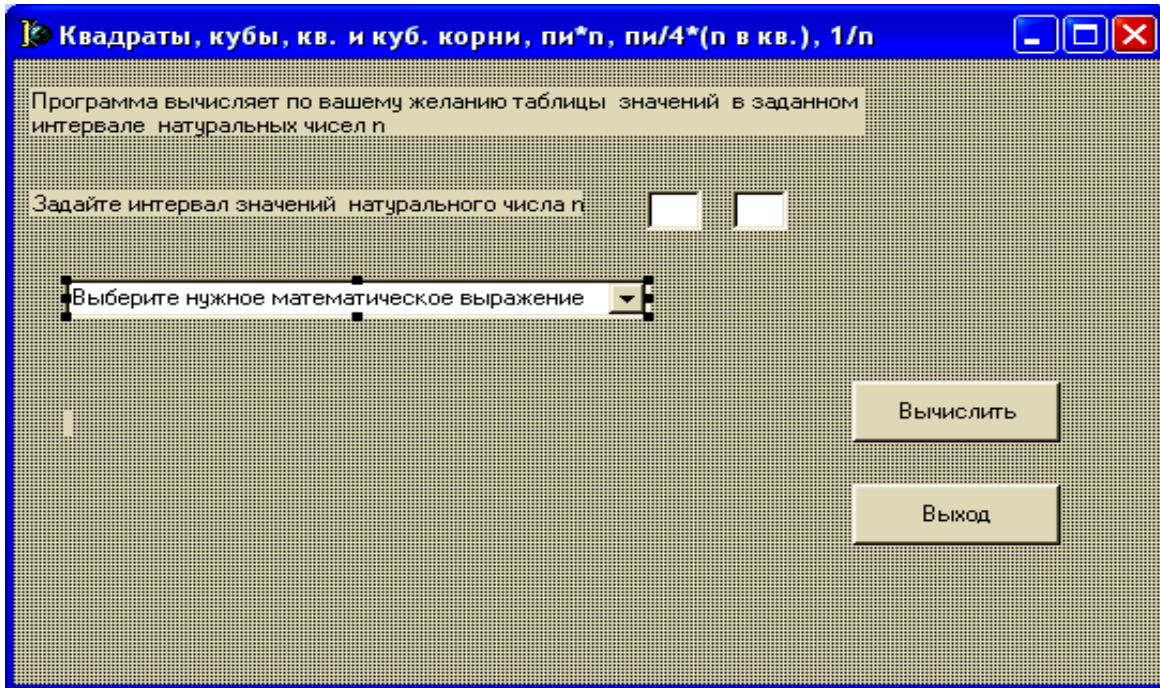
### Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Квадраты, кубы, кв. и куб. корни, $\pi n$ , $\pi/4(n \text{ в кв.})$ , $1/n$	Заголовок формы
Label1	Caption	Программа вычисляет по вашему желанию таблицы значений в заданном интервале натуральных чисел $n$	Справочная информация для пользователя программы
Label2	Caption	Задайте интервал значений натурального числа $n$	Подсказка пользователю
Label3	Caption		Поле для вывода результатов
Edit1	Text		Поле для ввода начального значения $n$
Edit2	Text		Поле для ввода конечного значения $n$
ComboBox1	Text Items	Выберите нужное математическое выражение Квадраты Кубы Квадратные корни Кубические корни $\pi n$ $\pi/[4(n \text{ в кв.})]$ $1/n$	Текст в заголовке списка Значения строк в списке
Button1	Caption	Вычислить	Кнопка для вычисления
Button2	Caption	Выход	Кнопка закрытия формы и выхода из программы

## Переменные

Обозначение в программе	Содержание	Тип
n	Текущее значение числа	целый
n1	Начальное значение интервала натуральных чисел	целый
n2	Конечное значение интервала натуральных чисел	целый

## Проект формы



## Текст модуля

```
unit Unit1;  
interface  
uses  
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
    Controls, Forms, Dialogs, StdCtrls, Math;  
// обязательно к стандартному списку подключить библиотеку  
// математических функций Math  
.....  
var  
    Form1: TForm1;  
    n1,n2:Integer; {начальное и конечное значения интервала  
    натуральных чисел}
```

```

    n:integer; // текущее значение числа
implementation
{$R *.dfm}
procedure TForm1.Button2Click(Sender: TObject);
begin
//процедура закрывает форму и осуществляет
//выход из программы
Form1.Close
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
//ввод начального и конечного значений интервала
//натуральных чисел
n1:=StrToInt(Edit1.Text);
n2:=StrToInt(Edit2.Text);
Case ComboBox1.ItemIndex of
0: begin
// очистка поля ввода от предыдущих значений
label3.Caption:=' ';
// если в поле со списком выбрано значение Квадраты
// в цикле от n1 до n2 с шагом 1 вычисляются значения
// квадратов чисел
    For n:=n1 To n2 Do
        Label3.Caption:= Label3.Caption+ #13+
            IntToStr(n)+' '+IntToStr(Sqr(n));
    end;

1:begin
// очистка поля ввода от предыдущих значений
label3.Caption:=' ';
// если в поле со списком выбрано значение Кубы
// в цикле от n1 до n2 с шагом 1 вычисляются значения
// кубов чисел
    For n:=n1 To n2 Do
        Label3.Caption:= Label3.Caption+ #13+
            IntToStr(n)+ ' ' + IntToStr(n*n*n);
    end;

2:begin label3.Caption:=' ';

```

```

// если в поле со списком выбрано значение Квадратные корни
// в цикле от n1 до n2 с шагом 1 вычисляются значения
// квадратных корней чисел
    For n:=n1 To n2 Do
        Label3.Caption:= Label3.Caption+ #13+
        IntToStr(n)+' ' + FloatToStrF(Sqrt(n),Ffixed,10,6);
    end;
3:begin label3.Caption:=' ';
//функция логарифм не определена при нуле,
//поэтому сдвигаем интервал значений на единицу
    If n1=0 Then begin label3.Caption:='0';n1:=n1+1; end;
// если в поле со списком выбрано значение Кубические корни
// в цикле от n1 до n2 с шагом 1 вычисляются значения
// кубических корней чисел
    For n:=n1 To n2 Do
        Label3.Caption:= Label3.Caption+ #13+
        IntToStr(n)+' ' + FloatToStrF(exp(1/3*ln(n)),Ffixed,10,6);
    end;
4:begin label3.Caption:=' ';
    For n:=n1 To n2 Do
// если в поле со списком выбрано значение пи*n
// в цикле от n1 до n2 с шагом 1 вычисляются значения
// пи*n чисел
        Label3.Caption:= Label3.Caption+ #13+
        IntToStr(n)+' ' + FloatToStrF((pi*n),Ffixed,10,6);
    end;
5:begin label3.Caption:=' ';
// если в поле со списком выбрано значение пи/[4*(n в кв.)]
// в цикле от n1 до n2 с шагом 1 вычисляются значения
// пи/[4*(n в кв.)]
If n1=0 Then begin
        Label3.Caption:='Бесконечность'; n1:=n1+1
    end;
For n:=n1 To n2 Do
        Label3.Caption:= Label3.Caption+ #13+
        IntToStr(n)+' ' + FloatToStrF((pi/(4*n*n)),Ffixed,10,6);
    end;
6:begin label3.Caption:=' ';

```

```

If n1=0 Then begin
    Label3.Caption:=' Бесконечность'; n1:=n1+1
end;
// если в поле со списком выбрано значение 1/n
// в цикле от n1 до n2 с шагом 1 вычисляются значения 1/n
For n:=n1 To n2 Do
    Label3.Caption:= Label3.Caption+ #13+
    IntToStr(n)+' '+ FloatToStrF(1/n,Ffixed,10,6);
end
Else Exit;
end;
end;
end.

```

### ***Задача 5с***

Составить программу для вычисления таблицы уклонов местности. Уклоны местности определяют при проектировании каналов, дорог, направления движения тракторных агрегатов на пашне, изучении эрозии почв. Уклоном  $i$  называют отношение превышения между точками  $h$  к горизонтальному проложению между ними  $S$ . Для вычислений воспользоваться следующими формулами:

$$i = h / s$$

$$i^{\circ} = \text{Arc tan}(i) / \pi * 180$$

$$i \text{ в гр.} = \text{целая часть}(i^{\circ})$$

$$i \text{ в мин.} = \text{округление} ((i^{\circ} - i \text{ в гр.}) * 60)$$

### **Компоненты**

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Таблица уклонов	Заголовок формы
Label1	Caption	При проектировании каналов, дорог, направления движения тракторных агрегатов на пашне, изучении эрозии почв определяют уклоны местности. Уклоном $i$ называют отношение превышения $h$ между точками к горизонтальному проложению $S$ между ними.	Справочная информация для пользователя программы

Label2	Caption	Интервал изменения превышений	Подсказка пользователю
Edit1	Text		Нижняя граница интервала превышений
Edit2	Text		Верхняя граница интервала превышений
Label3	Caption	Шаг изменения превышений	Подсказка пользователю
Edit3	Text		Поле для ввода шага изменения превышений
Label4	Caption	Интервал изменения горизонтальных проложений	Подсказка пользователю
Edit4	Text		Нижняя граница интервала горизонтальных проложений
Edit5	Text		Верхняя граница интервала горизонтальных проложений
Label5	Caption	Шаг изменения горизонтальных проложений	Подсказка пользователю
Edit6	Text		Поле для ввода шага изменения горизонтальных проложений
Label6	Caption		В это поле программа выводит шапку таблицы уклонов
Memo1	Lines  ScrollBars	.....  ssBoth	Компонент для вывода значений таблицы уклонов  Выводить обе полосы прокрутки (вертикальную и горизонтальную)
Button1	Caption	Вычислить	Кнопка для выполнения
Button2	Caption	Выход	Кнопка для завершения работы программы

## Переменные

Обозначение в программе	Содержание	Тип
h	Текущее значение превышения	Вещественный расширенный
h1	Нижняя граница интервала превышений	Вещественный расширенный
h2	Верхняя граница интервала превышений	Вещественный расширенный
s	Текущее значение горизонтального проложения	Вещественный расширенный
s1	Нижняя граница интервала горизонтальных проложений	Вещественный расширенный
s2	Верхняя граница интервала горизонтальных проложений	Вещественный расширенный
Dh	шаг изменения превышений	Вещественный расширенный
Ds	шаг изменения горизонтальных проложений	Вещественный расширенный
i	текущее значение уклона в тангенсах	Вещественный расширенный
ig	градусная часть уклона	целый
im	минутная часть уклона	целый
Ari	текущее значение уклона в градусах	Вещественный расширенный



## Проект формы

Таблица уклонов

При проектировании каналов, дорог, направления движения тракторных агрегатов на пашне, изучении эрозии почв определяют уклоны местности. Уклоном  $i$  называют отношение превышения  $h$  между точками к горизонтальному проложению  $S$  между ними.

Интервал изменения превышений:

Шаг изменения превышений:

Интервал изменения горизонтальных проложений:

Шаг изменения горизонтальных проложений:

ВЫЧИСЛИТЬ

ВЫХОД

## Текст модуля

```
unit Unit1;  
interface  
uses  
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
Forms,  
Dialogs, StdCtrls, Math;  
// библиотека математических функций Math  
// подключается программистом  
type  
TForm1 = class(TForm)  
Label1: TLabel;  
Label2: TLabel;  
Edit1: TEdit;  
Label3: TLabel;  
Edit2: TEdit;  
Edit3: TEdit;
```

```

Edit4: TEdit;
Label4: TLabel;
Edit5: TEdit;
Label5: TLabel;
Edit6: TEdit;
Label6: TLabel;
Button1: TButton;
Button2: TButton;
Memo1: TMemo;
procedure Button2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;
h1,h2: Extended; // границы интервала изменения превышений
Dh:Extended;// шаг изменения превышений
s1,s2:Extended;// интервал изменения горизонтальных проложений
Ds: Extended; // шаг изменения горизонтальных проложений
implementation
{$R *.dfm}
procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close
end;
procedure TForm1.Button1Click(Sender: TObject);
Var i:Extended;// текущее значение уклона в тангенсах
ig:Integer; // градусная часть уклона
im:Integer; // минутная часть уклона
h,s:Extended;// текущее значение превышения и
//горизонтального проложения
ARi:Extended;// текущее значение уклона в градусах
begin
// ввод исходных данных для программы

```

```

h1:=StrToFloat(Edit1.Text);
h2:=StrToFloat(Edit2.Text);
Dh:=StrToFloat(Edit3.Text);
s1:=StrToFloat(Edit4.Text);
s2:=StrToFloat(Edit5.Text);
Ds:=StrToFloat(Edit6.Text);
// очистка поля вывода от результатов предыдущих запусков
Memo1.line.Clear;
// вывод шапки таблицы уклонов
Label6.Caption:=
'Превышение Горизонтальное проложение Уклон в тангенсах '
+ ' Уклон в гр., мин.';
// текущее значение превышения принимает значение нижней
// границы
h:=h1;
{ цикл выполняется пока текущее значение превышения
не превысит значение верхней границы}
While h <= h2 Do
Begin
// текущее значение горизонтального проложения принимает
// значение нижней границы
s:=s1;
{ цикл выполняется пока текущее значение горизонтального
проложения не превысит значение верхней границы}
While s <= s2 Do
begin
// вычисление текущего значения уклона в тангенсах
i:=h/s;
// перевод уклона в градусную меру
ARi:=ArcTan(i)/pi*180;
// выделение целой части - градусная часть уклона
ig := Trunc(ARi);
// вычисление минутной части значения уклона
im:=Round((ARi-ig)*60);
// вывод строки таблицы уклонов
Memo1.Lines.Add(' '+ FloatToStrF(h,Fffixed,6,1) +
' | '
+ FloatToStrF(s,Fffixed,15,2) + ' | '

```

```

+ FloatToStrF(i,Ffixed,15,5) + ' | ' +
' ' + IntToStr(ig) + ' ' + IntToStr(im));
// увеличение горизонтального проложения на величину шага
s:=s+Ds;
end;
// увеличение превышения на величину шага
h:=h+Dh;
end;
end;
end.

```

### Один из возможных результатов работы программы

При проектировании каналов, дорог, направления движения тракторных агрегатов на пашне, изучении эрозии почв определяют уклоны местности. Уклоном  $i$  называют отношение превышения  $h$  между точками к горизонтальному проложению  $S$  между ними.

Интервал изменения превышений:

Шаг изменения превышений:

Интервал изменения горизонтальных проложений:

Шаг изменения горизонтальных проложений:

Пре́вышение | Горизонтальное проложение | Уклон в тангенсах | Уклон в гр., мин.

0,3	170,00	0,00176	0 6
0,3	180,00	0,00167	0 6
0,5	100,00	0,00500	0 17
0,5	110,00	0,00455	0 16
0,5	120,00	0,00417	0 14
0,5	130,00	0,00385	0 13
0,5	140,00	0,00357	0 12
0,5	150,00	0,00333	0 11

ВЫЧИСЛИТЬ

ВЫХОД

### **Задача 5d**

Составить программу для вычисления числа  $e$ .

Число  $e=2,71828$  играет важную роль в качестве основания натуральных логарифмов.

Известно, что сумма ряда  $1+1/1!+1/2!+...+1/n!$  стремится к числу  $e$ .

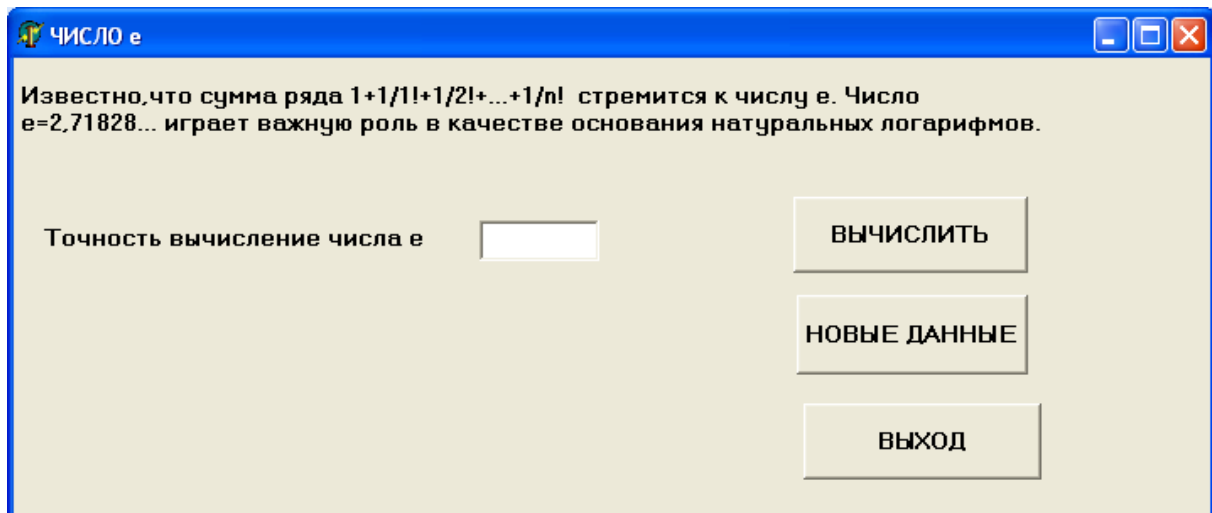
## Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Число e	Заголовок формы
Label1	Caption	Известно, что сумма ряда $1+1/1!+1/2!+\dots+1/n!$ стремится к числу e. Число $e=2,71828$ играет важную роль в качестве основания натуральных логарифмов.	Справочная информация для пользователя программы
Label2	Caption	Точность вычисления числа e	Подсказка пользователю
Edit1	Text		Поле для ввода точности вычисления числа e
Button1	Caption	Вычислить	Кнопка для вычисления
Button2	Caption	Новые данные	Кнопка для очистки введенного значения точности
Button3	Caption	Выход	Кнопка для завершения работы программы
Label3	Caption		Поле, в которое программа выводит вычисленное значение числа e

## Переменные

Обозначение в программе	Содержание	Тип
n	номер члена ряда -1	целый
e	число e	вещественный, расширенный
z	знаменатель члена ряда	Длинный целый
element	текущий элемент ряда	вещественный, расширенный
T	точность вычисления числа e	вещественный, расширенный

## Проект формы



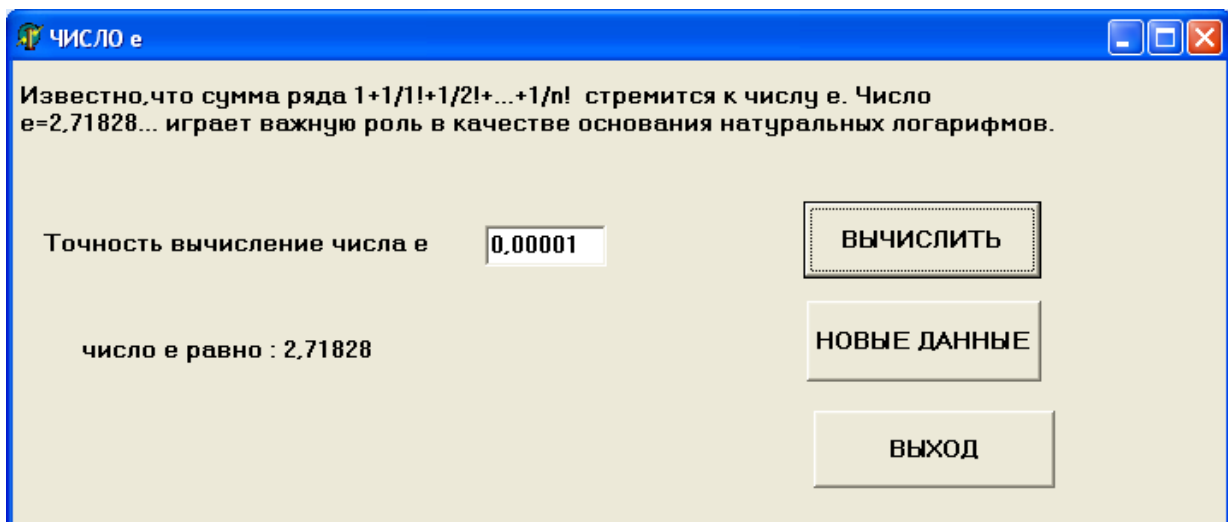
### Текст модуля

```
unit Unit1;  
.....  
var  
    Form1: TForm1;  
implementation  
{$R *.dfm}  
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    Edit1.Clear  
end;  
procedure TForm1.Button3Click(Sender: TObject);  
begin  
    Form1.Close  
end;  
procedure TForm1.Button1Click(Sender: TObject);  
Var e: Extended; // число e  
    element: Extended; // текущий элемент ряда  
    T: Extended; // точность вычисления числа e  
    n: Integer; // номер члена ряда -1  
    z: LongInt; // знаменатель члена ряда  
begin  
    // инициализируем числом 1 значения переменных e, z, n  
    e:=1;
```

```

z:=1;
n:=1;
// ввод точности вычисления числа e
T:=StrToFloat(Edit1.Text);
//цикл вычисления суммы ряда для числа e
Repeat
z:=Z*n;
element:=1/z ;
e:=e+element;
n:=n+1;
{ выход из цикла происходит, когда очередной член ряда
станет меньше или равен заданной точности вычисления T }
Until element <= T ;
// вывод вычисленного значения e
Label3.Caption:='число e равно : '+FloatToStrF(e,Ffixed,10,8)
end;
end.

```



### **Рекомендуемый Интернет-ресурс**

<http://www.youtube.com/watch?v=96N8dQJWEao&list=PLDDB9D4A0E37B4953&index=3>

### **Вопросы для самоконтроля**

1. Опишите алгоритм, реализующий суммирование ряда натуральных чисел в диапазоне значений от  $N_1$  до  $N_2$  с шагом один.
2. Опишите алгоритм произведения ряда натуральных чисел из диапазона значений  $K_1$  и  $K_2$  с шагом  $Step$ , не равным единице ( $Step \neq 1$ ).
3. Опишите алгоритм получения и вывода значений функции  $F(x)$  для ряда значений аргументов  $x$ .
4. Для чего нужны операторы цикла в программах?
5. Какие операторы цикла есть в Delphi?
6. В каких случаях в программе удобнее применить оператор цикла с параметром? Опишите его формат.
7. Назначение и формат оператора цикла с предусловием.
8. Назначение и формат оператора цикла с постусловием.
9. Приведите пример использования оператора цикла с параметром.
10. Приведите пример использования оператора цикла с предусловием.
11. Приведите пример использования оператора цикла с постусловием.
12. Назначение оператора `break`.
13. Назначение оператора `continue`.
15. Какой оператор цикла использован в программе «Вычисление функции  $Y$ » (**Задача 5a**)? Укажите фрагмент программы, где записан этот оператор.
16. Какой оператор цикла использован в программе «Квадраты, кубы...» (**Задача 5b**)? Укажите несколько фрагментов программы, где использован этот оператор.
17. Какой оператор цикла использован в программе «Таблица уклонов» (**Задача 5c**)? Укажите фрагменты программы, где использован этот оператор.
18. Какой оператор цикла использован в программе «Число  $e$ .» (**Задача 5c**)? Укажите фрагмент программы, где использован этот оператор.
19. Можно ли оператор цикла с параметром заменить оператором цикла с предусловием?



20. Можно ли оператор цикла с предусловием заменить оператором цикла с параметром?

### *Задачи для самостоятельного решения*

#### **Группа А**

№ 5.1 Написать программу для вычисления суммы  $S=S+1/i$ , где  $i=1,2,\dots,N$ .

№ 5.2 Написать программу для вычисления произведения первых  $N$  натуральных чисел.

№ 5.3 Написать программу для вычисления  $dx=h*h/(2*s)$ , при значениях:  $100 \leq S \leq 1000$  м, с шагом 100 м.

№ 5.4 Написать программу для вычисления суммы  $S$  из значений  $Y=X*X*X$ , где  $X$  задан в интервале от  $N1$  до  $N2$  ( $N1$  и  $N2$  – натуральные числа).

№ 5.5 Написать программу для вычисления выражения  $Y=N!$ .

№ 5.6 Для  $n$  случайных целых чисел  $x$  из интервала от  $-N1$  до  $+N2$  вычислить и показать на форме величины  $x$  и  $y$

$$y = \begin{cases} x, & \text{при } x \geq 0 \\ x^2, & x < 0 \end{cases}$$

№ 5.7 Вычислить и показать на форме величины функции  $y = \sqrt[3]{x}$  для значений  $N1 \leq x \leq N2$ , где  $x$ ,  $N1$ ,  $N2$  – натуральные числа, причем  $N2 > N1$ .

№ 5.8 Вычислить и показать на форме значения  $y=x^3$  для  $n$  значений случайных целых аргументов  $x$ , изменяющихся в пределах от 0 до 99.

№ 5.9 Вычислить и показать на форме  $n$  значений аргументов  $x$  и функции  $y=x^2$ . Здесь  $x$  случайные целые числа, изменяющиеся в интервале от 0 до  $M$ .

№ 5.10 Для  $n$  случайных целых чисел из интервала от 0 до  $M$  вычислить и вывести на форму аргументы  $x$  и значения функции

$$y = \begin{cases} x, & \text{для нечетных значений } x \\ x^3, & \text{для четных } x \end{cases}$$

#### **Группа Б**

№ 5.11 Написать программу для вычисления значений  $Y=Z^2$  при изменении  $Z$  от  $N1$  до  $N2$  с шагом  $K$ , где  $K$ ,  $N1$ ,  $N2$  – натуральные числа.

№ 5.12 Написать программу для вычисления суммы нечетных чисел натурального ряда в интервале от  $N_1$  до  $N_2$ , где  $N_1, N_2$  – натуральные числа.

№ 5.13 Написать программу для вычисления значений  $Q = \sqrt{Y}$  при изменении  $Y$  от  $N_1$  до  $N_2$  с шагом  $K$ , где  $K, N_1, N_2$  – натуральные числа.

№ 5.14 Написать программу для вычисления суммы обратных величин чисел от  $N_1$  до  $N_2$  с шагом  $K$ , где  $K, N_1, N_2$  – натуральные числа.

№ 5.15 Написать программу для вычисления выражения  $S = \pi * R * R$  при изменении  $R$  в интервале от 1 до  $N$  с шагом  $K$  где  $K, N$  – натуральные числа.

№ 5.16 Составить программу для вычисления суммы членов последовательности:

$$S_n = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \dots + \frac{1}{n \cdot (n+1)}, n - \text{натуральное число.}$$

Члены последовательности вычислять до тех пор, пока текущий член последовательности не станет меньше заданного числа  $\varepsilon$  ( $0 < \varepsilon < 0,0001$ ).

№ 5.17 Составить программу для вычисления суммы ряда:

$$\frac{1}{1^3} + \frac{1}{2^3} + \frac{1}{3^3} + \dots + \frac{1}{n^3}, n - \text{натуральное число.}$$

Сумму ряда вычислять до тех пор, пока текущий член последовательности не станет меньше заданного числа  $\varepsilon$  ( $0 < \varepsilon < 0,0001$ ).

№ 5.18 Составить программу для вычисления суммы ряда:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \dots$$

Сумму ряда вычислять до тех пор, пока текущий член последовательности не станет меньше заданного числа  $\varepsilon$  ( $0 < |\varepsilon| < 0,0001$ ).

№ 5.19 Вычислить произведение

$$P_n = \frac{1}{2} * \frac{1}{6} * \dots * \frac{1}{i * (i - 1)} * \dots * \frac{1}{n(n - 1)}$$

Вычисление прекратить, как только очередной сомножитель станет меньше введенного пользователем положительного числа  $\varepsilon$  ( $0 < \varepsilon \leq 0,1$ ).

№ 5.20 Вычислить произведение

$$Q_n = \frac{1}{\sqrt{2}} * \left(-\frac{1}{\sqrt{3}}\right) * \frac{1}{\sqrt{4}} \left(-\frac{1}{\sqrt{5}}\right) * \dots * \frac{1}{\sqrt{i}} * \dots * \left(-\frac{1}{\sqrt{n}}\right),$$

где четный сомножитель имеет отрицательный знак. Вычисление прекратить, как только очередной сомножитель по модулю станет меньше введенного пользователем числа  $\varepsilon$  ( $0 < |\varepsilon| \leq 0,1$ ).

## Приложение 1. Основные свойства базовых компонентов

Свойства компонента форма (объект типа TForm).

Свойство	Описание
Name	Имя (идентификатор) формы. Используется для доступа к форме, ее свойствам и методам, а также для доступа к компонентам формы
Caption	Текст заголовка
Width	Ширина формы
Height	Высота формы
Position	Положение окна в момент первого его появления на экране (poCenterScreen – в центре экрана; poOwnerFormCenter – в центре родительского окна; poDesigned – положение окна определяют значения свойств Top и Left)
Top	Расстояние от верхней границы формы до верхней границы экрана
Left	Расстояние от левой границы формы до левой границы экрана
BorderStyle	Вид границы. Граница может быть обычной (bsSizeable), тонкой (bsSingle) или вообще отсутствовать (bsNone). Если у окна обычная граница, то во время работы программы пользователь может с помощью 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 (значок в заголовке окна), biMinimize – кнопки Свернуть, biMaximize – кнопки Развернуть, biHelp – кнопки вывода справочной информации
Icon	Значок в заголовке диалогового окна, обозначающий кнопку вывода системного меню
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. 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)

Свойства компонента поле ввода–редактирования Edit (объект типа TEdit). Компонент находится на вкладке Standard.

Свойство	Описание
Name	Имя компонента. Используется для доступа к компоненту и его свойствам
Text	Текст, который находится в поле редактирования
Left	Расстояние от левой границы компонента до левой границы формы
Top	Расстояние от верхней границы компонента до верхней границы формы
Height	Высота компонента
Width	Ширина компонента
Font	Шрифт, используемый для отображения текста в поле компонента
ParentFont	Признак наследования шрифта от формы. Если значение свойства равно True, то для отображения текста в поле компонента используется шрифт формы
MaxLength	Количество символов, которое можно ввести в поле редактирования. Если значение свойства равно нулю, ограничения на количество символов нет
TabOrder	Определяет порядок перемещения фокуса (курсора) с одного элемента управления на другой в результате нажатия клавиши <Tab>
Visible	Устанавливает признак видимости компонента. По умолчанию установлено True. Компонент видим

Свойства компонента поле текста Label (объект типа TLabel). Компонент находится на вкладке Standard.

Свойство	Описание
Name	Имя компонента. Используется для доступа к компоненту
Caption	Отображаемый текст
Font	Шрифт, используемый для отображения текста
ParentFont	Признак наследования характеристик шрифта от объекта (формы), на котором компонент находится
AutoSize	Признак автоматического изменения размера компонента при изменении текста, отображаемого в поле компонента
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
WordWrap	Признак того, что слова, которые не помещаются в текущей строке, автоматически переносятся на следующую строку (значение свойства AutoSize должно быть False)
Visible	Устанавливает признак видимости компонента. По умолчанию установлено True. Компонент видим

Свойства компонента командная кнопка Button (объект типа TButton). Компонент находится на вкладке Standard.

Свойство	Описание
Name	Имя компонента. Используется для доступа к компоненту и его свойствам
Caption	Текст на кнопке
Enabled	Признак доступности кнопки. Кнопка доступна (программа реагирует на ее нажатие), если значение свойства равно True, и не доступна, если значение свойства равно False
Left	Расстояние от левой границы кнопки до левой границы формы
Top	Расстояние от верхней границы кнопки до верхней границы формы Расстояние от левой границы кнопки до левой границы формы Признак доступности кнопки. Кнопка доступна (программа реагирует на ее нажатие), если значение свойства равно True, и не доступна, если значение свойства равно False
Height	Высота кнопки
Width	Ширина кнопки
TabOrder	Определяет порядок перемещения фокуса (курсора) с одного элемента управления на другой в результате нажатия клавиши <Tab>
Visible	Устанавливает признак видимости компонента. По умолчанию установлено True. Компонент видим

Свойства компонента список ListBox (объекта типа TListBox). Компонент находится на вкладке Standard.

Свойство	Описание
Name	Имя компонента. В программе используется для доступа к компоненту и его свойствам
Items	Элементы списка – массив строк
Items.Count	Количество элементов списка
ItemIndex	Номер выбранного элемента (элементы списка нумеруются с нуля). Если в списке ни один из элементов не выбран, то значение свойства равно -1 (минус один)
Sorted	Признак необходимости автоматической сортировки (True) списка после добавления очередного элемента
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

Свойства компонента выпадающий список ComboBox (объекта типа TComboBox). Компонент находится на вкладке Standard.

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Text	Текст, находящийся в поле ввода/редактирования
Items	Элементы списка – массив строк
Count	Количество элементов списка
ItemIndex	Номер элемента, выбранного в списке. Если ни один из элементов списка не был выбран, то значение свойства равно –1 (минус один)
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

Свойства компонента редактор текста Memo (объект типа TMemo). Компонент находится на вкладке Standard.

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Text	Текст, находящийся в поле Memo (свойство доступно только во время работы программы)
Lines	Массив строк, соответствующий содержимому поля. Доступ к строке осуществляется по номеру. Строки нумеруются с нуля
Lines.Count	Количество строк текста в поле Memo (количество элементов в массиве Lines)
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

Свойства компонента сетка StringGrid (объект типа TStringGrid).  
Компонент находится на вкладке Additional.

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
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



Свойства компонента радио переключатель **RadioButton** (объект типа **TRadioButton**). Компонент находится на вкладке **Standard**

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 переключателя до левой границы формы
Top	Расстояние от верхней границы переключателя 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

Свойства компонента группа радио переключателей **RadioGroup** (объект типа **TRadioGroup**). Компонент находится на вкладке **Standard**

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Caption	Текст, который находится справа от кнопки
Items	Элементы группы переключателей– массив названий
ItemIndex	Номер переключателя, выбранного в группе. Если ни один из переключателей группы не был выбран, то значение свойства равно –1 (минус один)
Left	Расстояние от левой границы переключателя до левой границы формы
Top	Расстояние от верхней границы переключателя 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

Свойства компонента группа **GroupBox** (объект типа **TGroupBox**). Компонент находится на вкладке **Standard**

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента
Caption	Название группы компонентов
Left	Расстояние от левой границы группы до левой границы формы
Top	Расстояние от верхней границы группы до верхней границы формы
Height	Высота поля вывода содержимого группы
Width	Ширина поля вывода содержимого группы
Font	Шрифт, используемый для отображения содержимого группы
ParentFont	Признак наследования характеристик шрифта родительской формы

Свойства компонента диалог открытия файла `OpenDialog` (объект типа `TOpenDialog`). Компонент находится на вкладке `Dialogs`

Свойство	Описание
<code>Title</code>	Текст в заголовке окна. Если значение свойства не указано, то в заголовке отображается текст <code>Открыть</code>
<code>Filter</code>	Свойство задает список фильтров имен файлов. В списке файлов отображаются только те файлы, имена которых соответствуют выбранному (текущему) фильтру. Во время отображения диалога пользователь может выбрать фильтр в списке <code>Тип файлов</code> . Каждый фильтр задается строкой вида <code>описание маска</code> , например <code>Текст *.txt</code>
<code>FilterIndex</code>	Если в списке <code>Filter</code> несколько элементов (например, <code>Текст *.txt Все файлы *.*)</code> , то значение свойства задает фильтр, который используется в момент появления диалога на экране
<code>InitialDir</code>	Каталог, содержимое которого отображается при появлении диалога на экране. Если значение свойства не указано, то в окне диалога отображается содержимое папки <code>Мои документы</code>
<code>FileName</code>	Имя файла, выбранного пользователем

Свойства компонента диалог сохранения файла `SaveDialog` (объект типа `TSaveDialog`). Компонент находится на вкладке `Dialogs`

Свойство	Описание
<code>Title</code>	Текст в заголовке окна. Если значение свойства не указано, то в заголовке отображается текст <code>Сохранить как</code> <code>Filter</code> Свойство задает список фильтров имен файлов. В списке файлов отображаются только те файлы, имена которых соответствуют выбранному (текущему) фильтру. Во время отображения диалога пользователь может выбрать фильтр в списке <code>Тип файлов</code> . Каждый фильтр задается строкой вида <code>описание маска</code> , например <code>Текст *.txt</code>
<code>FilterIndex</code>	Если в списке <code>Filter</code> несколько элементов (например, <code>Текст *.txt Все файлы *.*)</code> , то значение свойства задает фильтр, который используется в момент появления диалога на экране
<code>InitialDir</code>	Каталог, содержимое которого отображается при появлении диалога на экране. Если значение свойства не указано, то в окне диалога отображается содержимое папки <code>Мои документы</code>
<code>FileName</code>	Имя файла, введенное пользователем в поле <code>Имя файла</code>
<code>DefaultExt</code>	Расширение, которое будет добавлено к имени файла, если в поле <code>Имя файла</code> пользователь не задаст расширение файла

Свойства компонента главное меню `MainMenu` (объект типа `TMainMenu`). Компонент находится на вкладке `Standard`

Свойство	Описание
<code>Name</code>	Имя компонента.
<code>Items.Name</code>	Имя пункта меню. Используется для доступа к пункту меню
<code>Items.Caption</code>	Текст, находящийся в поле пункта меню

## Приложение 2. События, которые можно связать с большинством объектов Delphi

Событие	Описание
Click	Щелчок кнопкой мыши
DblClick	Двойной щелчек кнопкой мыши
MouseDown	Нажатие кнопки мыши
MouseUp	Отпускание нажатой кнопки мыши
MouseMove	Перемещение указателя мыши
KeyPress	Нажатие клавиши клавиатуры
KeyDown	Нажатие клавиши клавиатуры. События KeyDown и KeyPress – это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие KeyUp)
KeyUp	Отпускание нажатой клавиши клавиатуры
Create	Создание объекта (формы, элемента управления). Процедура обработки этого события обычно используется для инициализации переменных, выполнения подготовительных действий
Paint	Событие происходит при появлении окна на экране в начале работы программы, после появления части окна, которая, например, была закрыта другим окном
Enter	Получение элементом управления фокуса
Exit	Потеря элементом управления фокуса

## Приложение 3. Сообщения компилятора Delphi об ошибках

Текст сообщения	Причина ошибки
Array type required	Требуется массив.
Assignment to FOR-Loop variable '<Имя>'	Переменная - параметра цикла FOR может иметь неопределённое значение после выполнения этого цикла.
Case label outside of range of case expression	Значения указанные в операторе CASE таковы, что селектор не может их принимать.
Compile terminated by user	Компиляция прервана пользователем клавишами Ctrl и Break.
Constant expression expected	Ожидалось выражение состоящее из одних констант.
Constant expression violates subrange bounds	Константное выражение выходит за пределы заданного интервала.
Could not create output file	Невозможно создать exe-файл.
Declaration of <Имя> differs from previous declaration	Текущее декларирование отличается от предыдущего.
Division by zero	Выполняется деление на нуль.
Duplicate case label	Значения в операторе CASE повторяются.

EXCEPT or FINALLY expected	Ожидается секция исключений оператора TRY.
Expression has no value	Выражение не возвращает после себя никакого значения.
File not found: '<Имя_файла>.dcu'	Файл внешнего модуля не был обнаружен в текущем каталоге.
File not found: <Имя_файла>	Не был найден указанный файл.
File type not allowed here	Файловый тип в данном месте не разрешен. Файловая переменная может передаваться в подпрограмму только как параметр - переменная.
FOR-Loop variable '<Имя>' cannot be passed as var parameter	Переменная - параметр цикла не может передаваться в подпрограмму как параметр. Она должна быть описана локально.
For loop control variable must be simple local variable	Переменная - параметр цикла FOR должна быть описана локально.
For loop control variable must have ordinal type	Переменная параметра цикла FOR может быть целого, символьного либо перечислимого типов.
FOR or WHILE loop executes zero times - deleted	Цикл FOR или WHILE не будет выполняться ни разу и поэтому удален.
Function needs result type	Была определена функция в которой не определено результирующее значение.
Identifier redeclared: '<Имя>'	Повторно был описан указанный идентификатор. Все имена в программе должны быть уникальны.
Illegal character in input file: '<Символ>' (\$)	В программе были обнаружены недопустимые символы.
Illegal type in Read/Readln statement	Несовместимость типов при вызове оператора ввода.
Illegal type in Write/Writeln statement	Несовместимость типов при вызове оператора вывода.
Inaccessible value	Значение данной переменной не доступно либо не определено.
Incompatible types	Несовместимость типов.
Incompatible types: '<Тип1>' and '<Тип2>'	Имеет место несовместимость указанных типов.
Internal error: <Код_ошибки>	Внутренняя ошибка.
Invalid function result type	Была определена функция, в которой выходное значение не совместимо по типу с указанным в заголовке.
Label already defined: '<Имя_метки>'	Указанная метка уже была определена.
Left side cannot be assigned to	Имеет место попытка изменить значение константы.
Line too long (more than 255 characters)	Строка слишком длинна.
Low bound exceeds high bound	Нижняя граница диапазона определена большей чем верхняя граница.
Method identifier expected	Требуется указать название метода.
Missing operator or semicolon	Пропущен оператор либо точка с запятой.
Missing parameter type	Формальные параметры подпрограммы даны без указания типов.

Not enough actual parameters	При вызове какой - либо процедуры или функции было указано недостаточное количество фактических параметров.
Object type required	Требуется объект.
Operator not applicable to this operand type	Оператор не предназначен для работы с операндами данного типа.
Ordinal type required	В данном месте требуется скалярный тип.
Pointer type required	Требуется указатель.
Procedure cannot have a result type	Процедура не может иметь результирующего значения. В таких случаях нужно использовать функцию.
PROCEDURE or FUNCTION expected	В данном месте ожидается процедура либо функция.
Record, object or class type required	Требуется запись, объект или класс.
Return value of function '<Имя_функции>' might be undefined	Выходное значение указанной функции может быть не определено.
Statement expected, but expression of type '<Тип>' found	Ожидался оператор, но было обнаружено выражение указанного типа.
String constant truncated to fit STRING[<Номер>]	Строковая константа укорочена.
Syntax error in real number	Ошибка при написании действительного числа.
Text after final 'END.' - ignored by compiler	Текст, написанный в программе после последнего END был игнорирован.
Too many actual parameters	При вызове какой - либо процедуры или функции было указано слишком большое количество фактических параметров.
Type of expression must be BOOLEAN	Тип выражения должен быть логическим. Имеет место несовместимость типов.
Type of expression must be INTEGER	Тип выражения должен быть целым. Имеет место несовместимость типов.
Types of actual and formal var parameters must be identical	Типы формальных и фактических параметров должны совпадать.
Incompatible types	Несовместимые типы. В инструкции присваивания тип выражения не соответствует или не может быть приведен к типу переменной, получающей значение выражения. Тип фактического параметра процедуры или функции не соответствует или не может быть приведен к типу формального параметра
Undeclared identifier: '<Имя>'	Неизвестный идентификатор, либо оператор написан с ошибкой.
Unexpected end of file in comment started on line <Номер>	Комментарий начатый в указанной строке не закрыт.
Unterminated string	Незавершенная строка. При записи строковой константы, например сообщения, не поставлена завершающая кавычка.
User break - compilation aborted	Компиляция прервана по желанию пользователя.
Unknown directive: '<Директива>'	Неизвестная директива либо написана с ошибкой.

Unsatisfied forward or external declaration: '<Имя_процедуры>'	Указанное декларирование не имеет реализации.
Value assigned to '<Имя>' never used	Значение определённое для указанной переменной не было ни разу использовано.
Variable '<Имя>' might not have been initialized	У указанной переменной не было определено начальное значение.
<Директива1> expected but <Директива2> found	Была обнаружена директива2, хотя ожидалась директива1.
'<Имя>' is not a type identifier	Указанное имя не является названием типа.
«;» not allowed before 'ELSE'	Точка с запятой не ставится перед ELSE.

## Литература

1. Культин Н.Б. Основы программирования в Delphi XE. – СПб.: БХВ–Петербург, 2011.– 416 с.
2. Культин Н.Б. Delphi в задачах и примерах.– СПб.: БХВ–Петербург, 2012.– 288 с.
3. Осипов Д.Л. Delphi XE2.– СПб.: БХВ–Петербург, 2012 – 912 с.
4. Практикум по информатике. Часть II. Delphi /Учеб. пособие под ред. Коробочкина М.И. – М.: ГУЗ, 2008. – 180 с.

# Оглавление

<b>Введение</b> .....	3
<b>Глава 1. Знакомство со средой программирования Delphi</b> .....	4
Среда программирования Delphi .....	5
Вопросы для самоконтроля .....	19
<b>Глава 2. Разработка линейных программ</b> .....	20
Теория .....	20
Алгоритм и программа .....	20
Этапы разработки программы .....	21
Типы данных языка Delphi .....	21
Оператор присваивания .....	25
Структура программы на языке Delphi .....	26
Некоторые функции преобразования типов .....	29
Структура проекта .....	30
Сохранение проекта .....	32
Структура модуля .....	32
Компиляция и выполнение проекта .....	33
Практика .....	34
Вопросы для самоконтроля .....	39
Задачи для самостоятельного решения .....	40
<b>Глава 3. Стандартные и библиотечные функции</b> .....	45
Теория .....	45
Ввод из окна ввода .....	50
Вывод в окно сообщения .....	50
Практика .....	52
Вопросы для самоконтроля .....	71
Задачи для самостоятельного решения .....	72
<b>Глава 4. Программы с разветвлениями</b> .....	78
Теория .....	78
Логические выражения .....	78
Условный оператор .....	80
Оператор выбора .....	81
Оператор перехода .....	83
Компонент ListBox .....	83
Компонент ComboBox .....	84
Некоторые приемы работы с отладчиком .....	85
Обработка исключительных ситуаций .....	86
Практика .....	89
Вопросы для самоконтроля .....	97
Задачи для самостоятельного решения .....	98
<b>Глава 5. Организация циклов</b> .....	105
Теория .....	105
Алгоритмы циклических вычислений .....	105
Операторы цикла .....	107
Оператор цикла с параметром .....	108



Оператор цикла с предусловием .....	109
Оператор цикла с постусловием .....	110
Операторы break и continue .....	110
Практика .....	113
Вопросы для самоконтроля .....	129
Задачи для самостоятельного решения .....	130
<b>Литература .....</b>	<b>143</b>