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

Кафедра информатики

Е.Е. Дмитриева
М.И. Коробочкин

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

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 2015

УДК 681.3
ББК 32.97
Д 53

Рекомендовано к печати кафедрой информатики Государственного университета по землеустройству (протокол №2 от 07 октября 2014 г.)

Утверждено к изданию ученым советом факультета «Кадастр недвижимости» Государственного университета по землеустройству (протокол № 2 от 27 октября 2014г.)

Рецензенты:
А.Н. Лимонов, доцент, к.т.н.
В.Н. Никонов, доцент, к.э.н.

Д 53 Практикум по информатике. Часть II. Программирование в среде Delphi: Учебное пособие / Е.Е. Дмитриева, М.И. Коробочкин − М.: ГУЗ, 2015. − 268 с.

ISBN 978-5-9215-0281-9

«Практикум по информатике. Часть II. Программирование в среде Delphi» предназначен для поддержки второй части курса «Информатика», посвященной изучению вопросов программирования.

«Практикум» состоит из введения, 9 глав и трех приложений.

В главах рассмотрены следующие темы: «Знакомство со средой программирования Delphi», «Разработка линейных программ», «Стандартные и библиотечные функции», «Программы с разветвлениями», «Организация циклов», «Использование одномерных массивов», «Работа с двумерными массивами», «Процедуры и функции, разрабатываемые программистом», «Работа с внешними файлами».

Каждая глава «Практикума» содержит необходимые сведения из теории, соответствующие типовые задачи, доведенные до работающих программных продуктов, вопросы для самоконтроля и задачи для самостоятельного решения.

В приложениях описаны основные свойства базовых компонентов, события, которые можно связывать с большинством объектов Delphi, сообщения компилятора Delphi об ошибках.

«Практикум» позволяет эффективно изучать приемы разработки современных программных приложений в среде Delphi.

Книга предназначена для студентов, преподавателей, слушателей факультетов повышения квалификации и широкого круга читателей, желающих освоить программирование.

ISBN 978-5-9215-0281-9

УДК 681.3
ББК 32.97

Введение

Цель данного «Практикума» – помочь читателю 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.).

Язык программирования Pascal, положенный в основу языка Delphi, был разработан в 1970 году профессором Цюрихской высшей технической школы, обладателем премии Тьюринга (высшей награды для специалистов в области информационных технологий) Никлаусом Виртом, для обучения студентов программированию, как системной дисциплине. Автор указывал в качестве цели его создания – построение эффективного языка, способствующего хорошему стилю программирования, использующему структурное программирование и структурированные данные. Популярности языка Pascal среди профессионалов–программистов способствовало создание компанией Borland в 1992 году среды разработки Borland Pascal with Objects 7.0 и особенно появление в 1995 году знаменитой Borland Delphi 1.0 для 16–разрядной ОС Microsoft Windows 3.

В настоящее время права на язык Delphi принадлежат компании Embarcadero Technologies. Новые владельцы внесли в среду раз-

работки многочисленные улучшения и усовершенствования. Так последние версии системы Delphi XE позволяют разрабатывать 32 и 64–разрядные приложения для Windows.

«Практикум» можно использовать при обучении основам программирования на компьютерах, где установлены версии системы Delphi 6, Delphi 7, Delphi XE. Проекты (программы), разработанные в ранних версиях Delphi, открываются и работают в более поздних версиях.

Учебное пособие «Практикум по информатике. Часть II. Программирование в среде Delphi» содержит девять глав и три приложения.

Каждая глава включает теоретическую часть и практику, где подробно рассматриваются соответствующие типовые задачи и реализующие их, реально работающие программы (проекты Delphi).

Каждая глава пособия сопровождается вопросами для самоконтроля и задачами для самостоятельного решения, а также ссылками на интернет-ресурсы, полезные для лучшего усвоения материала.

Первое издание «Практикума по информатике» вышло в 2002 году, второе издание в 2003 году, третье издание вышло в 2008 году в двух частях. В 2013 году опубликована первая часть четвертого издания. В текущем 2015 году публикуется вторая часть четвертого издания «Практикума».

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 осваиванию новых версий, которые будут появляться на рынке программных продуктов.

Среда программирования Delphi

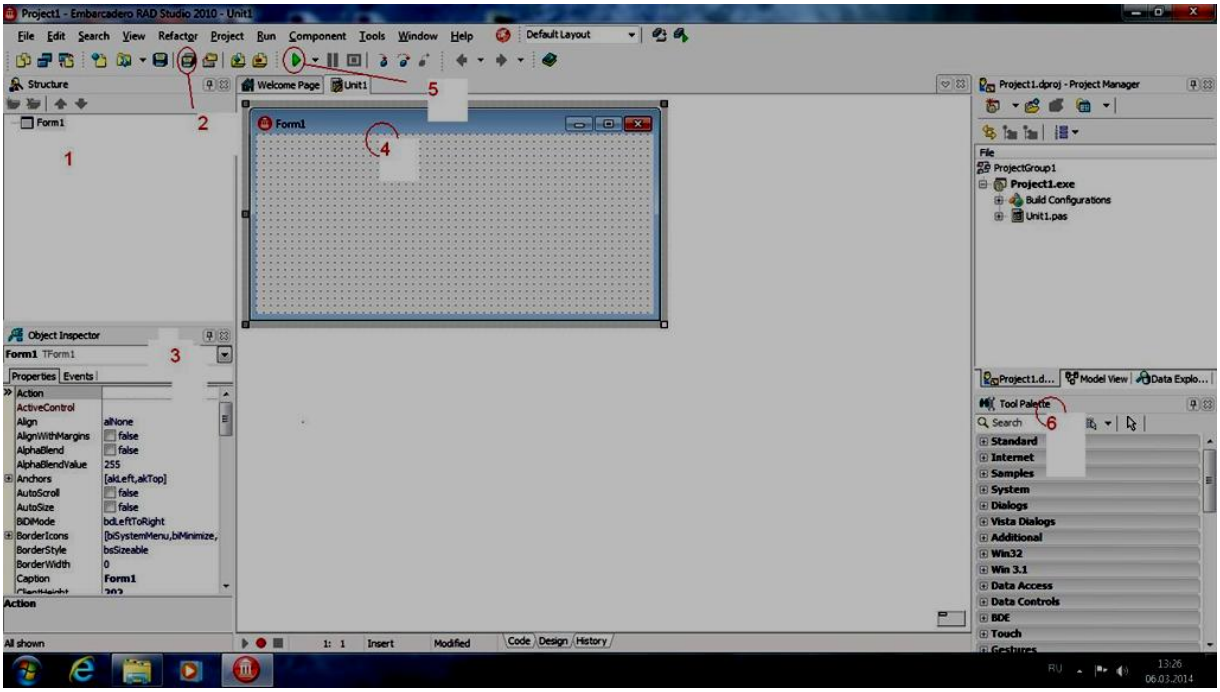


Рис. 1–1. Окно среды Delphi XE

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 Disign.

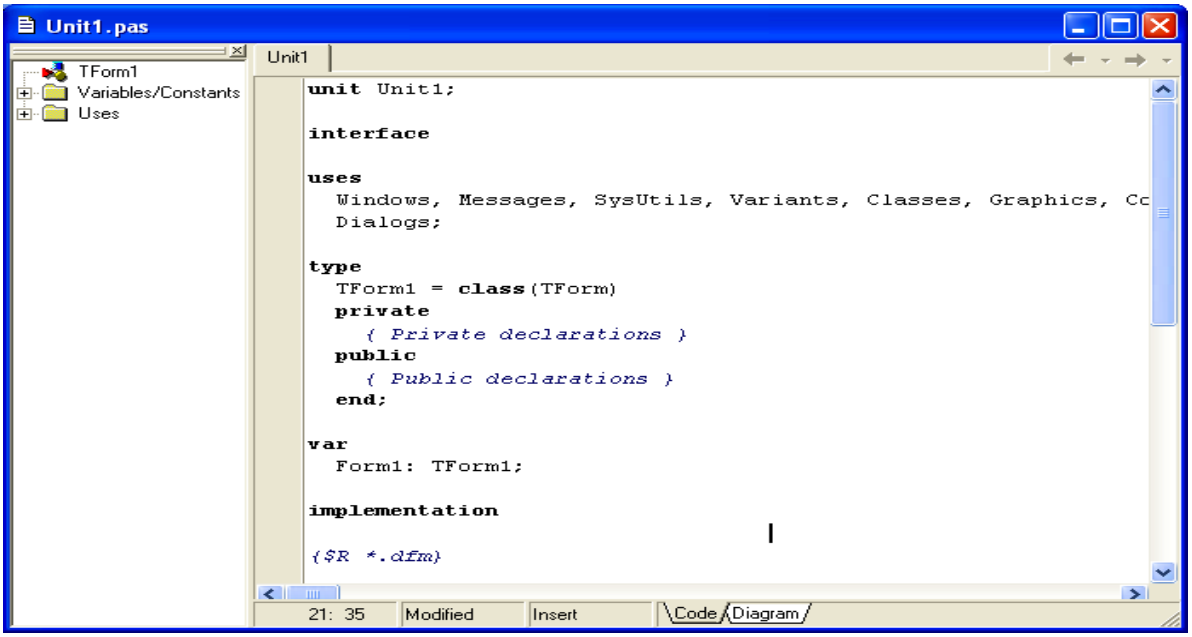


Рис. 1–3. Окно Редактора Кода

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 OK, 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.

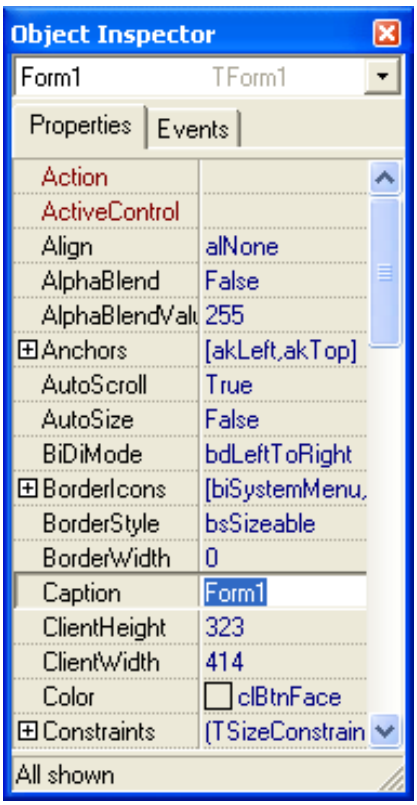


Рис. 1–5. Инспектор объектов

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.

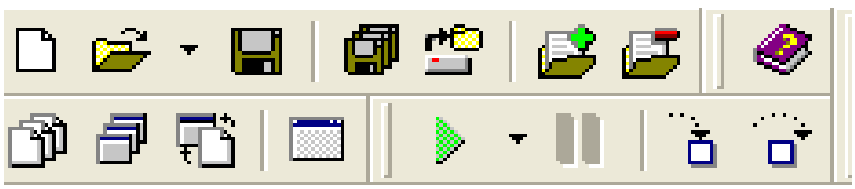


Рис. 1–6: Панель с кнопками для быстрого доступа (SpeedBar) для Delphi 6, 7

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

Sга. = Sкв.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 Widows. 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).

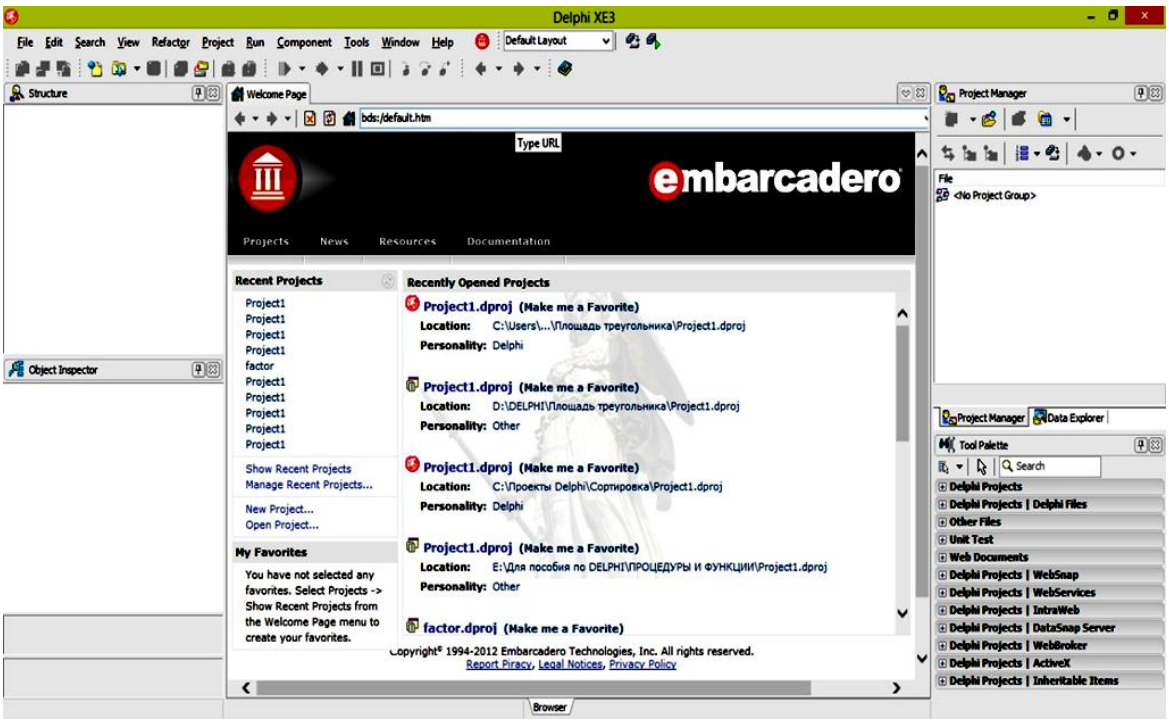


Рис. 1–7. Стартовое окно Delphi XE

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

1. Закрыть 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).

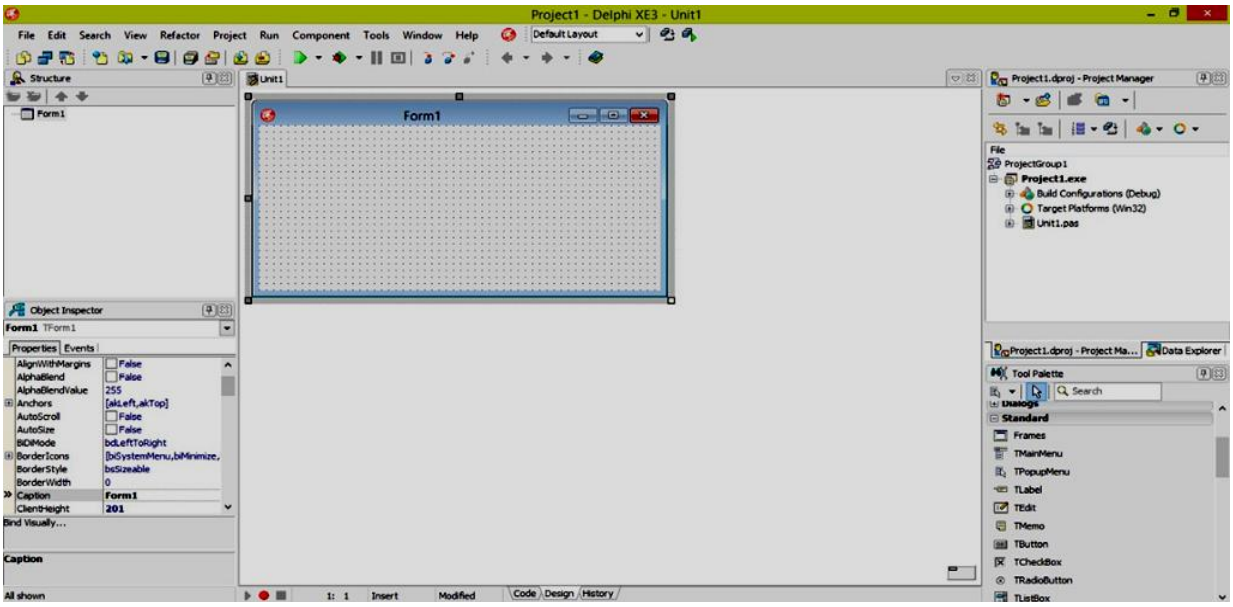


Рис. 1–8. Окно стартовой формы

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

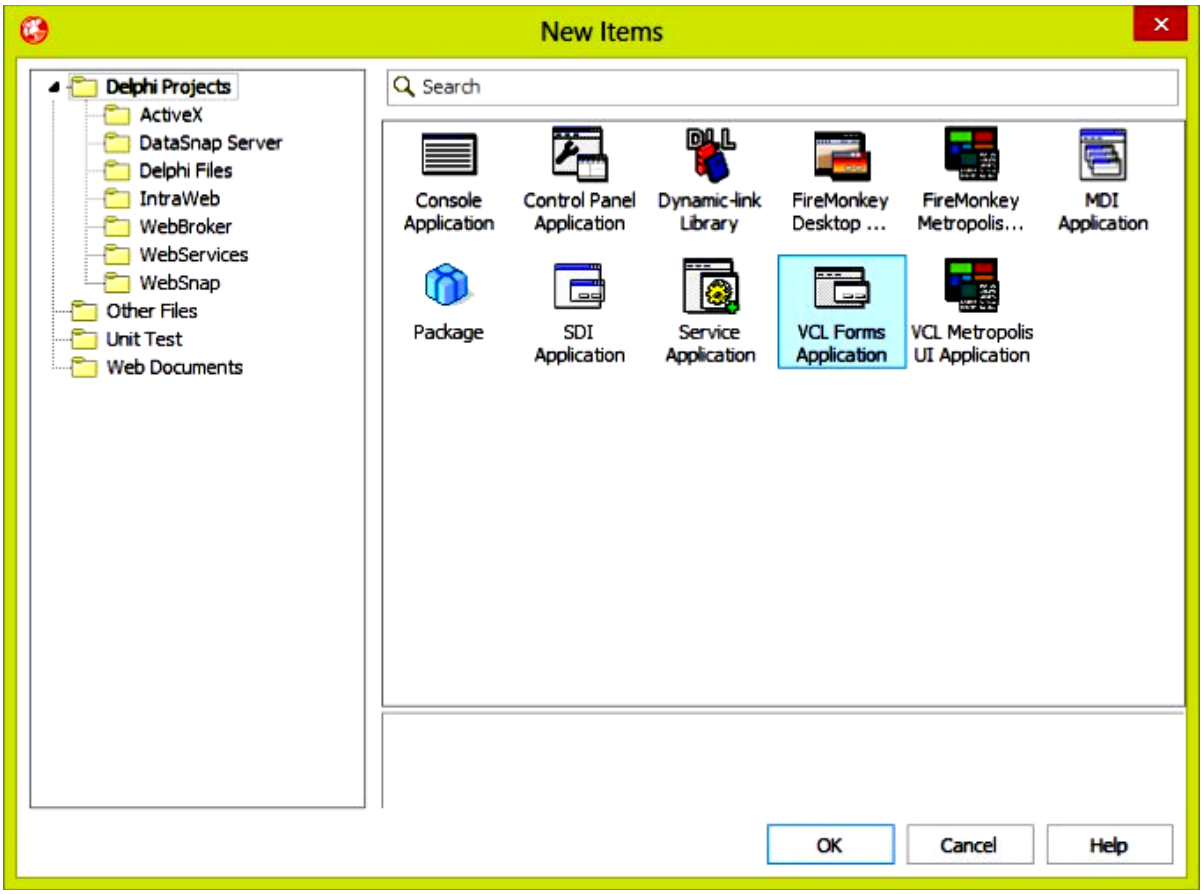


Рис. 1–9. Окно, где нужно выбрать вид создаваемого приложения

Мы создаем VCL Forms Applicaton (приложение на основе библиотеки визуальных компонентов). Выбираем соответствующий значок и нажимаем кнопку OK. 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 вводить 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.

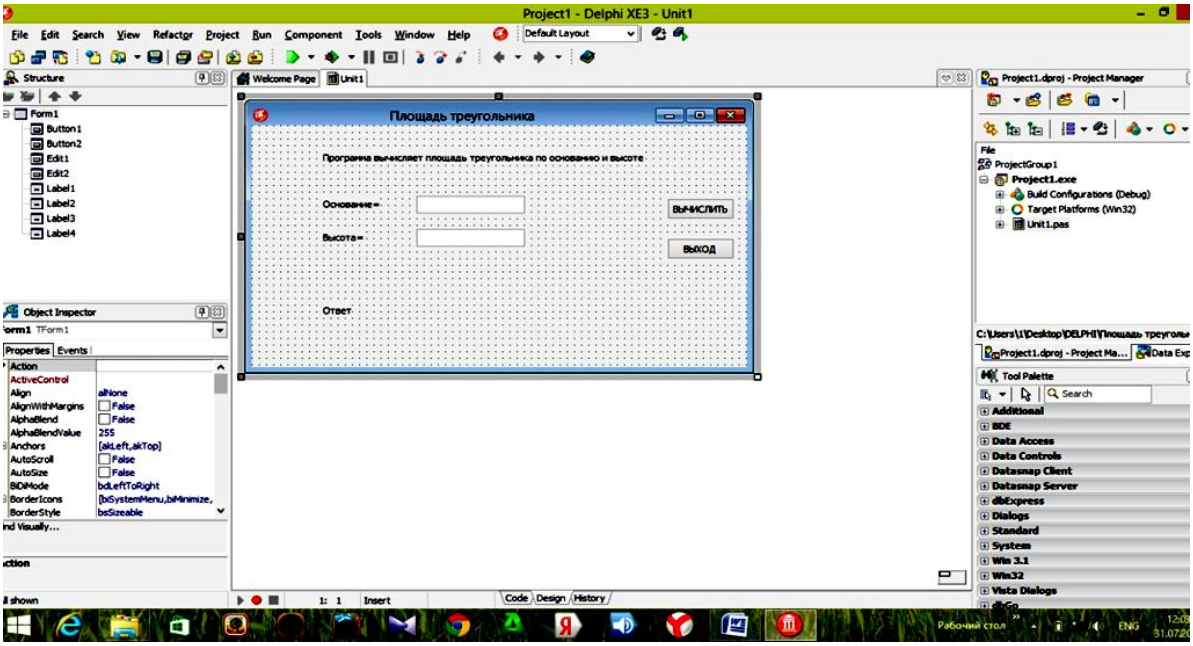


Рис. 1-10. Окно проекта «Площадь треугольника»

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 делам 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 находится 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).

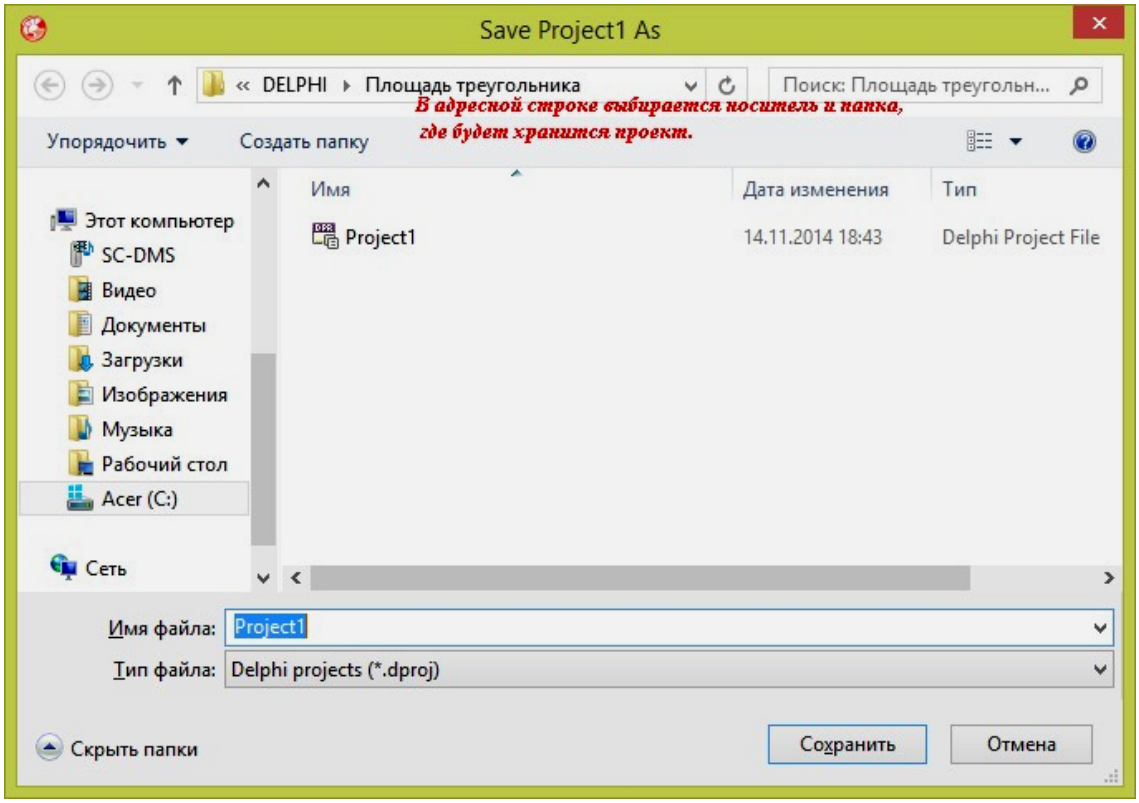


Рис.1−11. Окно сохранения проекта

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.

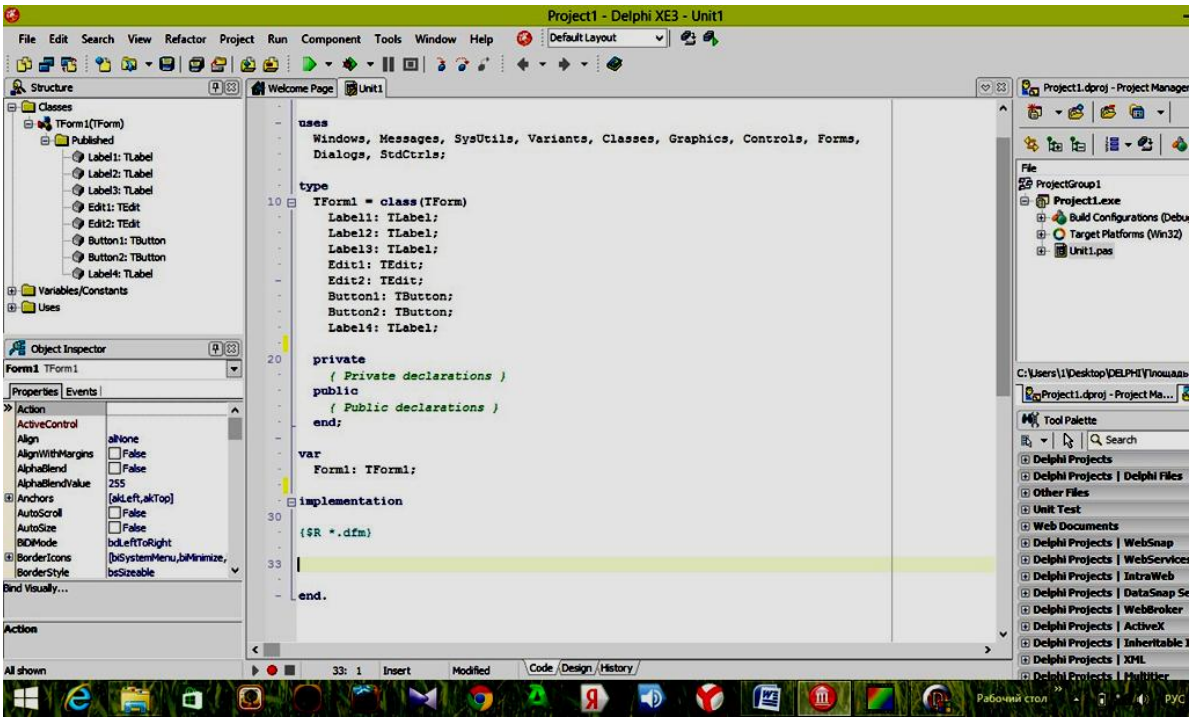


Рис. 1–12. Код будущего проекта «Площадь треугольника» после визуального проектирования окна интерфейса

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);
```

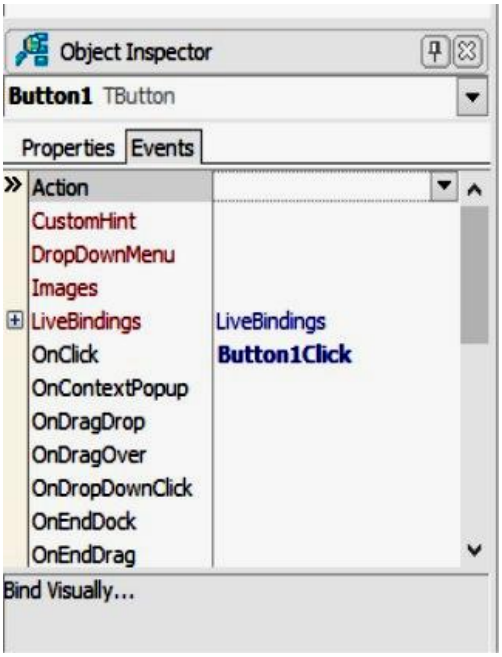


Рис. 1–13. Вкладка событий в инспекторе объектов

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 клик 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).

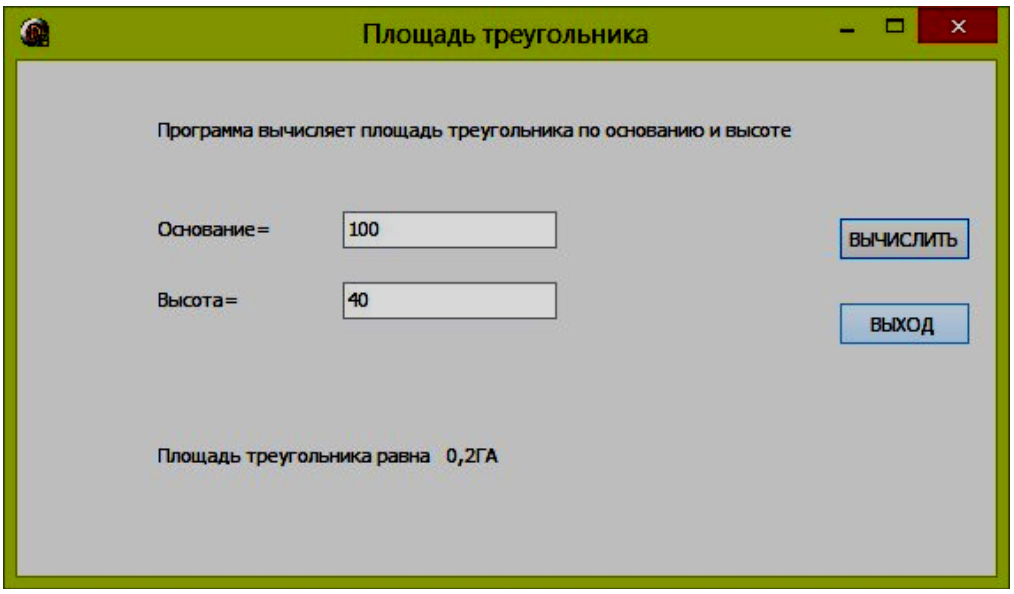


Рис. 1−14. Результат работы программы

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

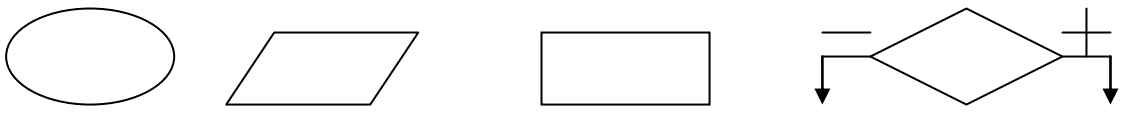
Дискретность – предполагает, что алгоритм состоит из отдельных операций. Работа операций, которые зависят от других операций, должна выполняться только после завершения операций, от которых они зависят. Независимые операции могут выполняться в любом порядке, в том числе параллельно.

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:



Начало/конец Ввод/вывод Преобразование Проверка данных (обработка) условия

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 ошибок (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, Intl6	-32 768..32 767	2 байта, со знаком
LongInt, Integer, Int32	-2 147 483 648.. 2147 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 пе-

ременные 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:

```
45
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 metkal, 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 stl = '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> = <0писание типа>;

Пример. Описание типов.

Type char2 = ('a' .. 'z') ;

NumberArray = array[l .. 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;
nl,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>;
<0ператорN>;
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.

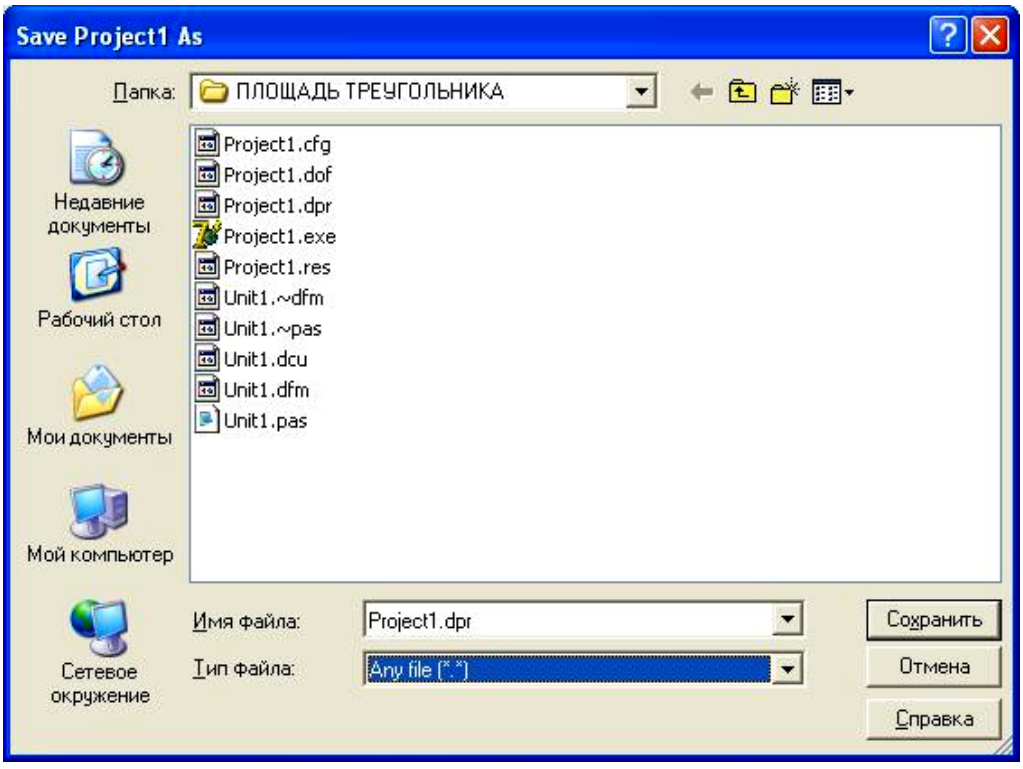


Рис. 2−1. Содержание проекта «Площадь треугольника»

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, поэтому 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 Projectl.

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 под-

сказки 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.

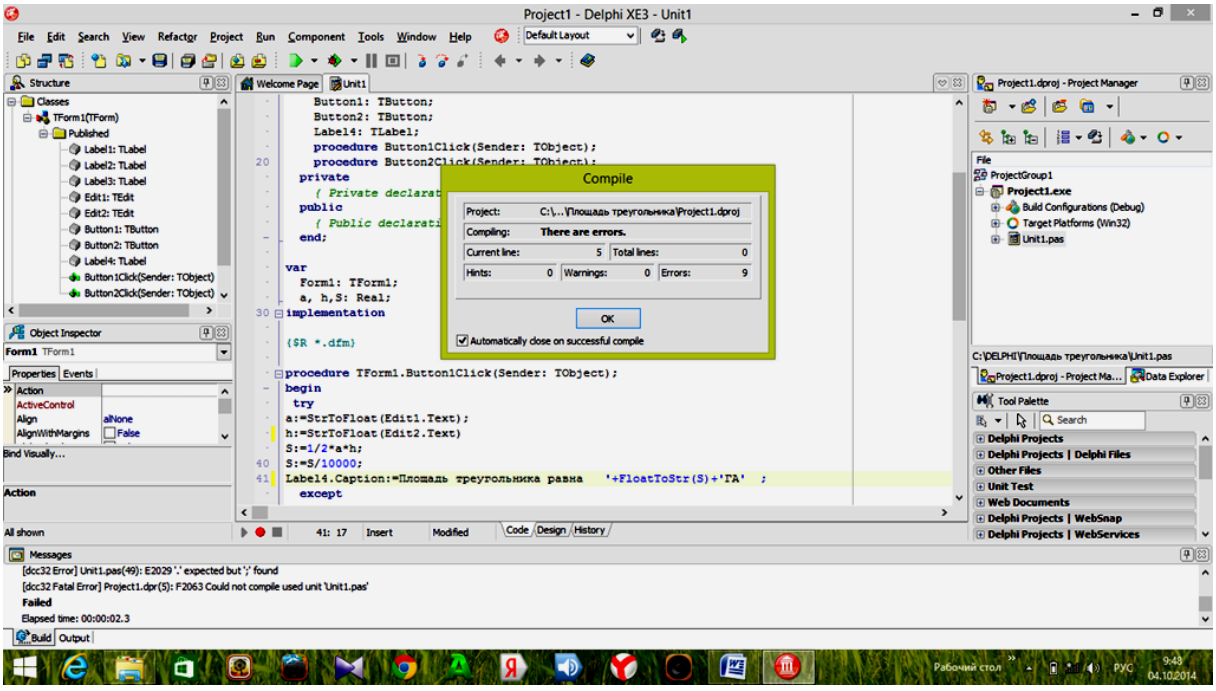


Рис. 2–2. Окно проекта Delphi после компиляции. Найдены ошибки

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.

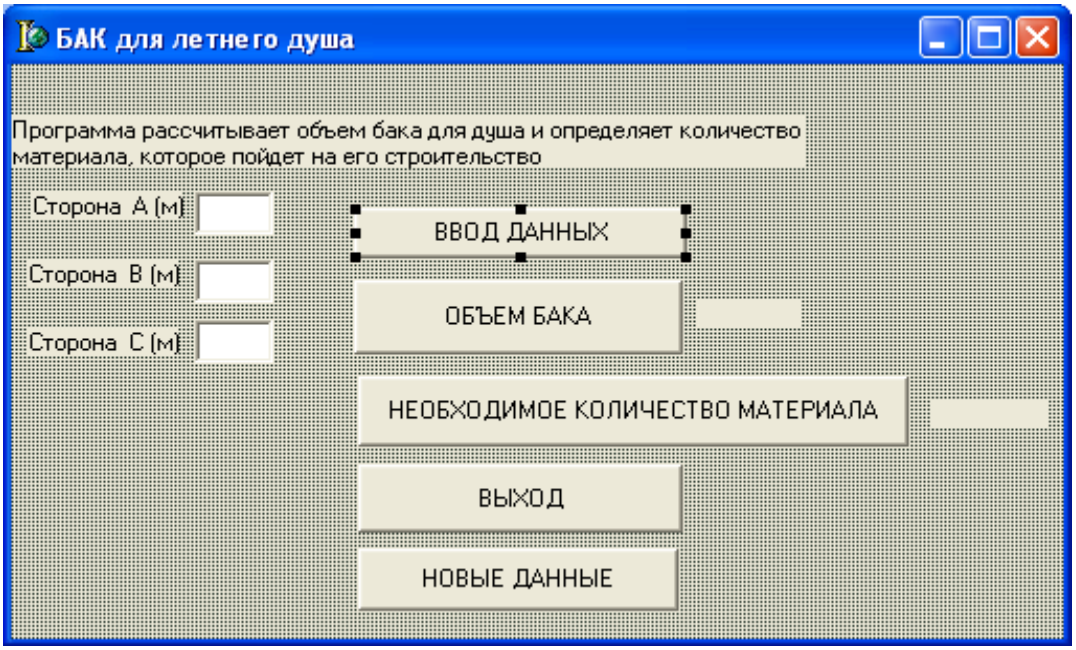
Компоненты:

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 A (м)	Подсказка пользователю
Label3	Caption	Сторона B (м)	Подсказка пользователю
Label4	Caption	Сторона C (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 A
Edit2	Text	Должно быть очищено.	Поле для ввода стороны B
Edit3	Text	Должно быть очищено.	Поле для ввода стороны C
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 A	вещественный
b	сторона B	вещественный
c	сторона C	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.

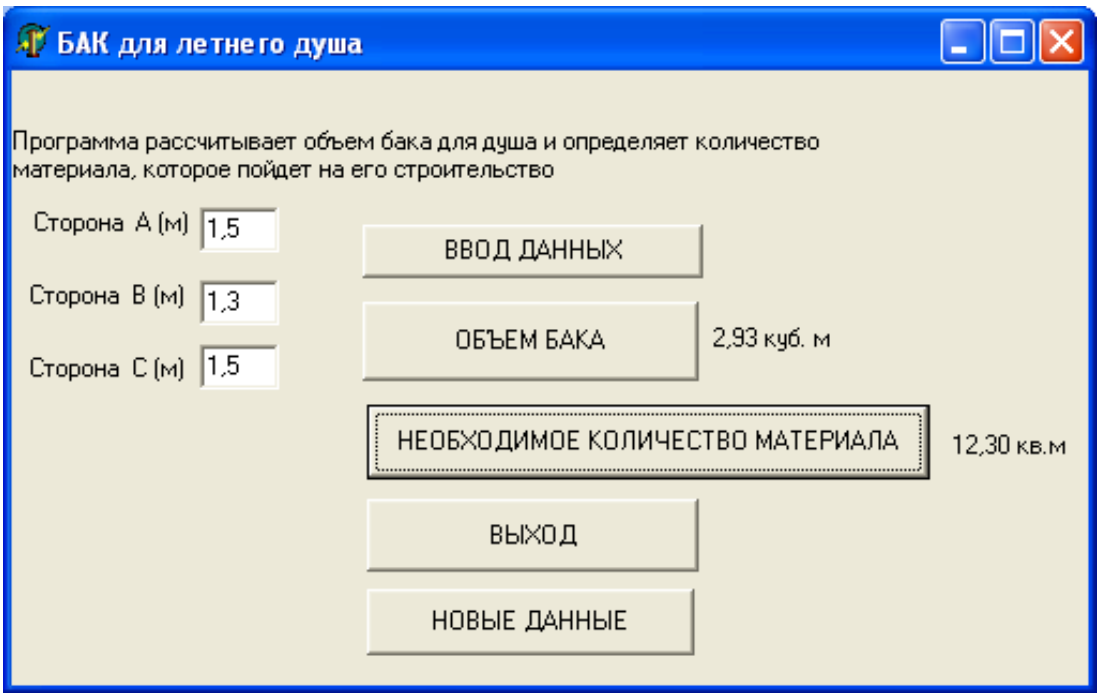
Текст модуля

```
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,Fffixed,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,Fffixed,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



Рекомендуемый интернет–ресурс

http://www.youtube.com/watch?v=TL2obDWmF7o&index=2&list=PLDDB9D4A0E37B4953&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:

$$M = 2\pi Rh,$$

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:

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

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 φ (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 м^3. Формула для вычисления объема:

$$V = H/3(S1 + \sqrt{S1S2} + S2),$$

где

$$S1 = \pi R_1^2, S2 = \pi R_2^2.$$

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 *a1, b1* – 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 (ΔS) 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≈330км).

R – радиус Земли. *R*≈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<=y<=330.

№ 2.12 Составить программу для вычисления поправки ΔP в площадь за переход с поверхности шара на плоскость в проекции Гаусса:

$$\Delta P = P\frac{y2}{R^2}.$$

P – площадь участка на поверхности шара (га).

R – радиус Земли. *R*≈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 $\boldsymbol{a_1}$ арифметической прогрессии, $\boldsymbol{d}$ – разность арифметической прогрессии. Составить программу для вычисления $\boldsymbol{n}$–го члена прогрессии

$$\boldsymbol{a_n = a_1 d(n-1)}$$

и суммы n–членов прогрессии:

$$\boldsymbol{S_n = (a_1 + a_n)n/2.}$$

№ 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:

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

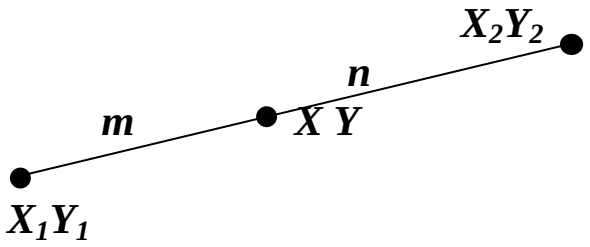


Рис. 2–3

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≈6371,1 км); $\boldsymbol{S_i}$ – 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:

$$Z = \frac{2260}{P} + 1{,}48 K_1 K_2 \sqrt{P} + 67{,}4,$$

где *K1* – коэффициент, характеризующий конфигурацию земельной площади и положение хозяйственного центра;

K2 – 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* Р/ГА.

№ 2.20 Составить программу для вычисления площади сектора S_1 и площади сегмента S_2 если известны r – радиус круга, α – 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^2.

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 π
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**

Функция	Возвращаемое значение
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**

Функция	Описание
Concat(sl, s2, s3)	Возвращает последовательное соединение строк. Эквивалентна оператору sl+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(sourse, 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)	Арксинус 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 о 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;

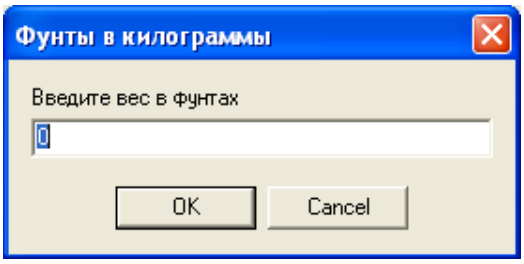
Заголовок –текст, выводимый в заголовок окна;

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);
```



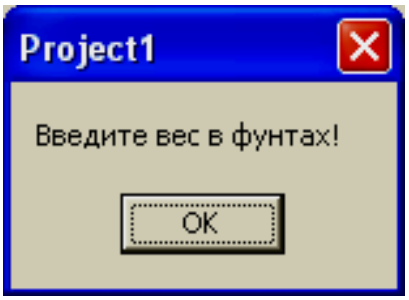
Если во время работы программы пользователь введет строку и щелкнет на кнопке OK, то значением функции InputBox будет введенная строка, если – на кнопке Cancel, то значением функции будет строка, переданная функции в качестве параметра Значение (в примере 0).

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 функцииMessageDlg следующий:

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

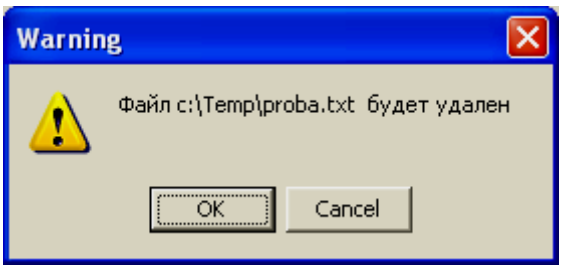
Где Сообщение – текст сообщения; Тип – тип сообщения. Тип сообщения задается именованной константой:

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(Угол в радианах)

4. Вывод катета в окно интерфейса.

Компоненты

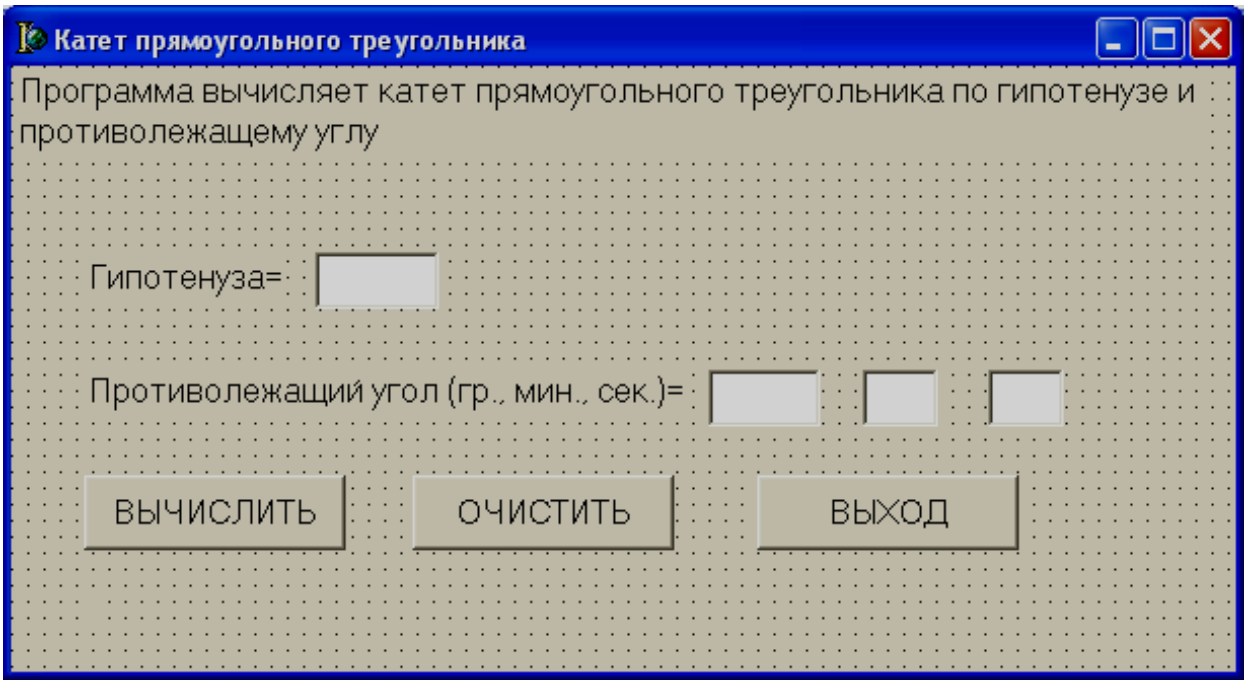
Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Катет прямоугольного треугольника	Заголовок формы
Label1	Caption WordWrap	Программа вычисляет катет прямоугольного треугольника по гипотенузе и противолежащему углу True	Справочная информация для пользователя программы Перенос не уместившихся 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

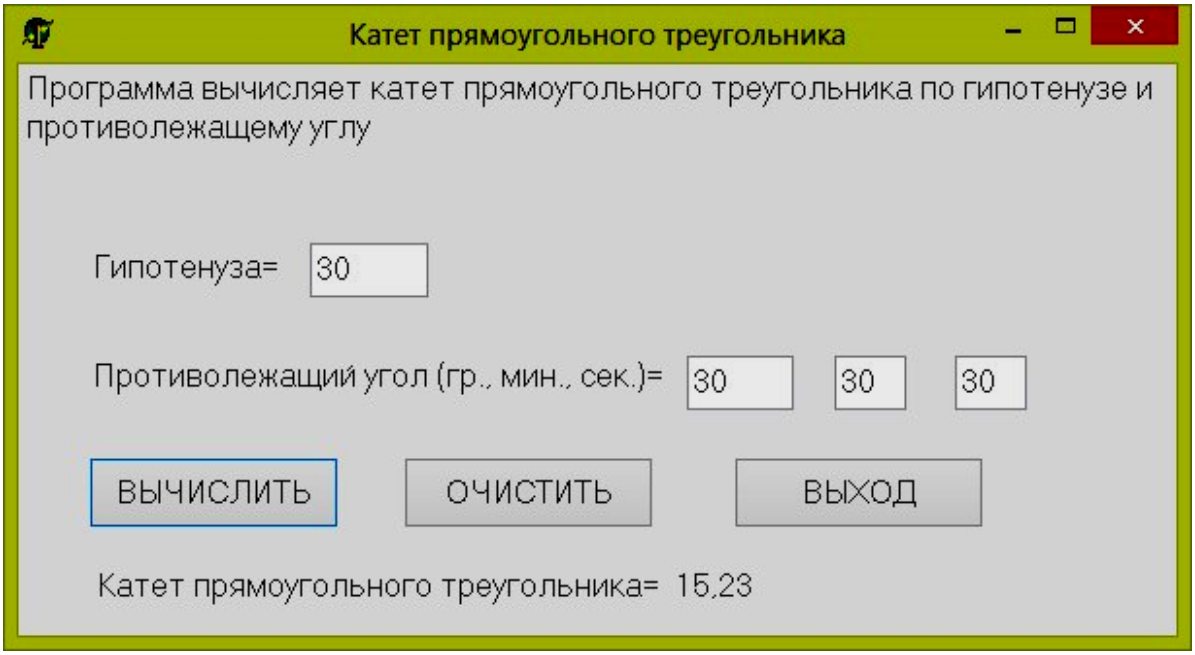


Текст модуля

```
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, Fffixed,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.

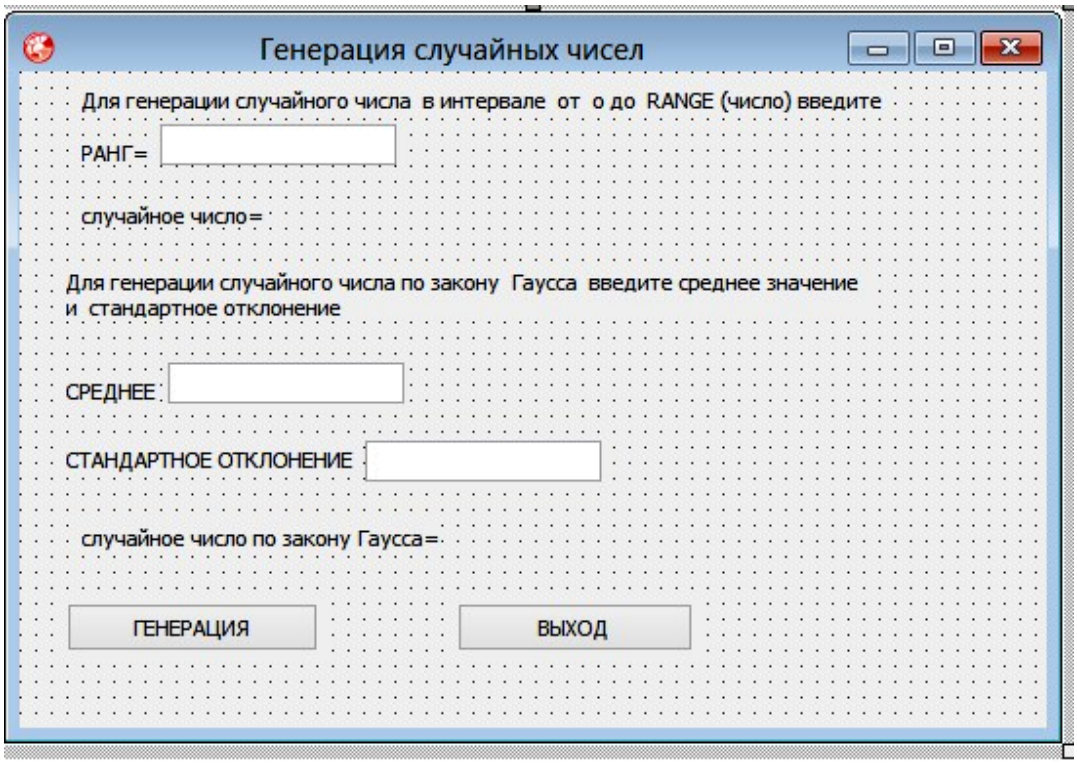
Перед использованием генератора случайных чисел нужно его проинициализировать с помощью функции Randomize.

Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Генерация случайных чисел	Заголовок формы
Label1	Caption	Для генерации случайного числа в интервале от о 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

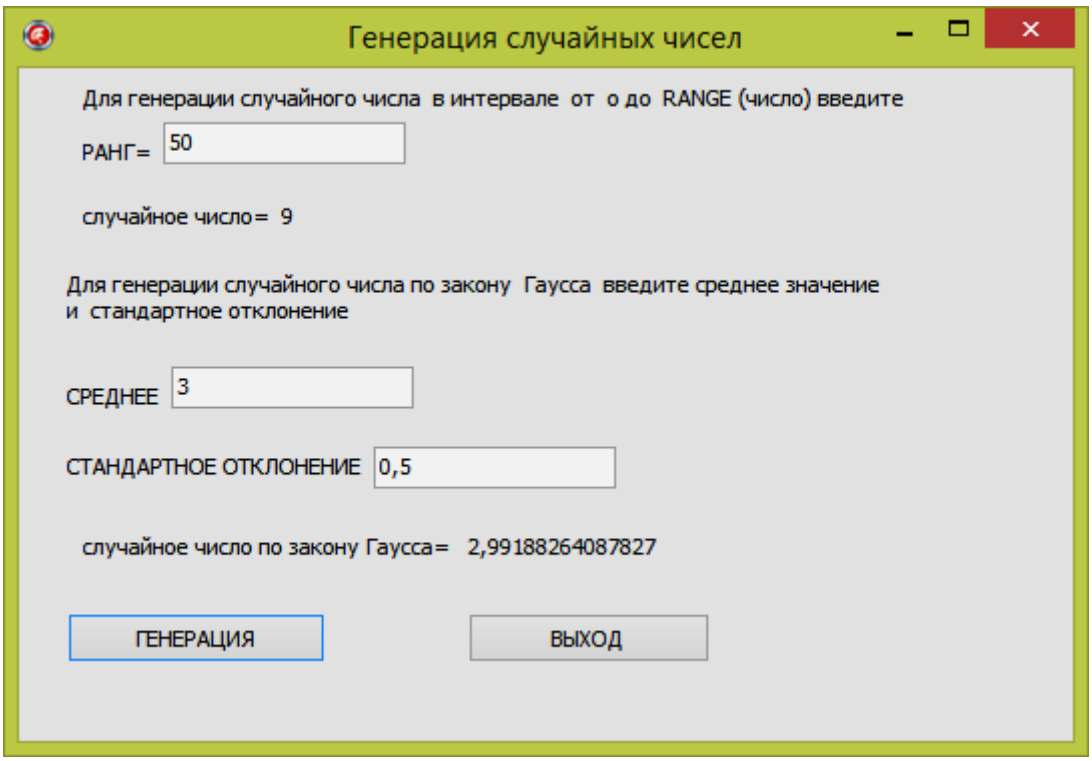


Текст модуля

```
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;
```

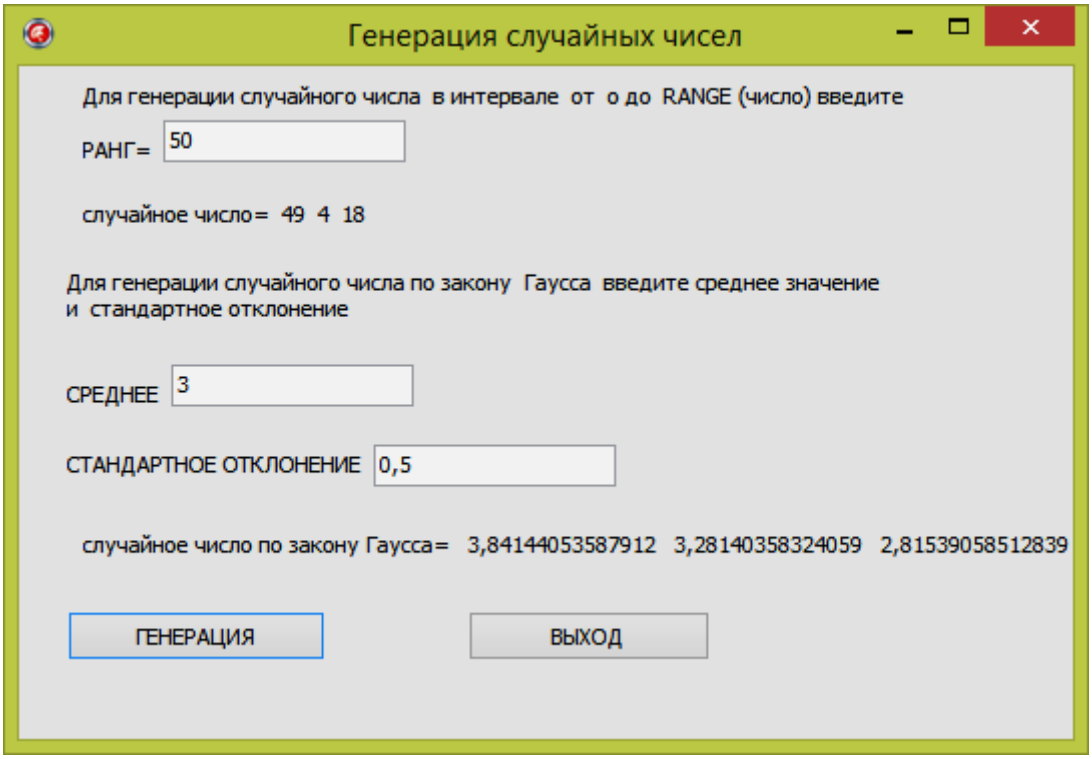
```
    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

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+ '    '+
FloatToStr(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 трехразовом нажатии на кнопку ГЕНЕРАЦИЯ мы получили следующий результат



Задача 3с

Составить программу, определяющую сколько дней прожил пользователь программы.

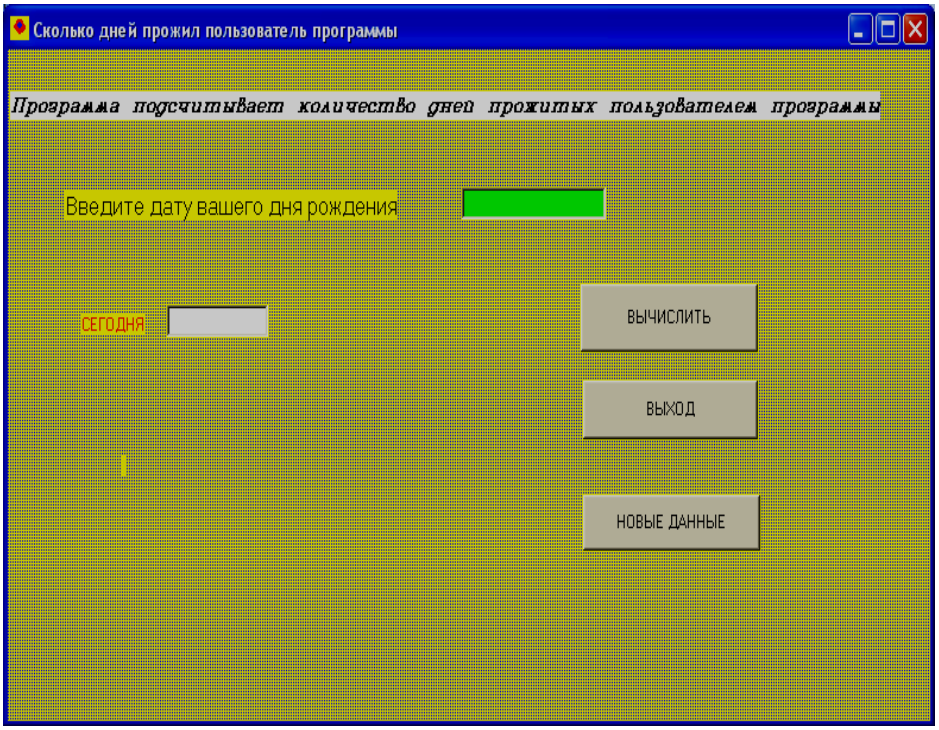
Эта задача на использовании функций работы с датами. Не забудьте в раздел Uses добавить библиотечный модуль **DateUtils**.

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

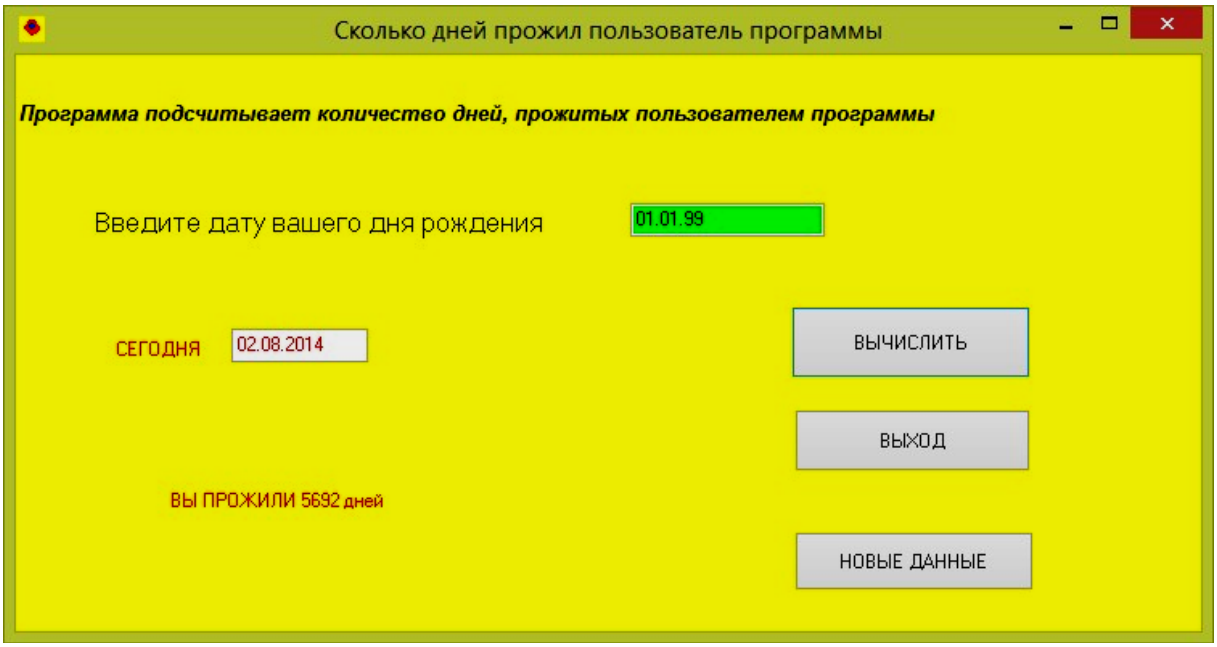


Текст модуля

```
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 A (X_A и Y_A) линии AB, по дирекционному углу этой линии α_{AB} и ее горизонтальному проложению S_{AB} вычисляют координаты другого конца B 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 AB вычисляют дирекционный угол и горизонтальное проложение этой линии. То есть известны: X_A, Y_A, X_B, Y_B. Надо вычислить α_{AB} и S_{AB}.

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

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

Компоненты

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

Edit2	Text Visible	Должно быть очищено False	Поле для ввода координаты YA невидимый
Edit3	Text Visible	Должно быть очищено False	Поле для ввода координаты XB невидимый
Edit4	Text Visible	Должно быть очищено False	Поле для ввода координаты YB 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 Visible	Вычислить False	Кнопка вычисления обратной геодезической задачи 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 лини 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.

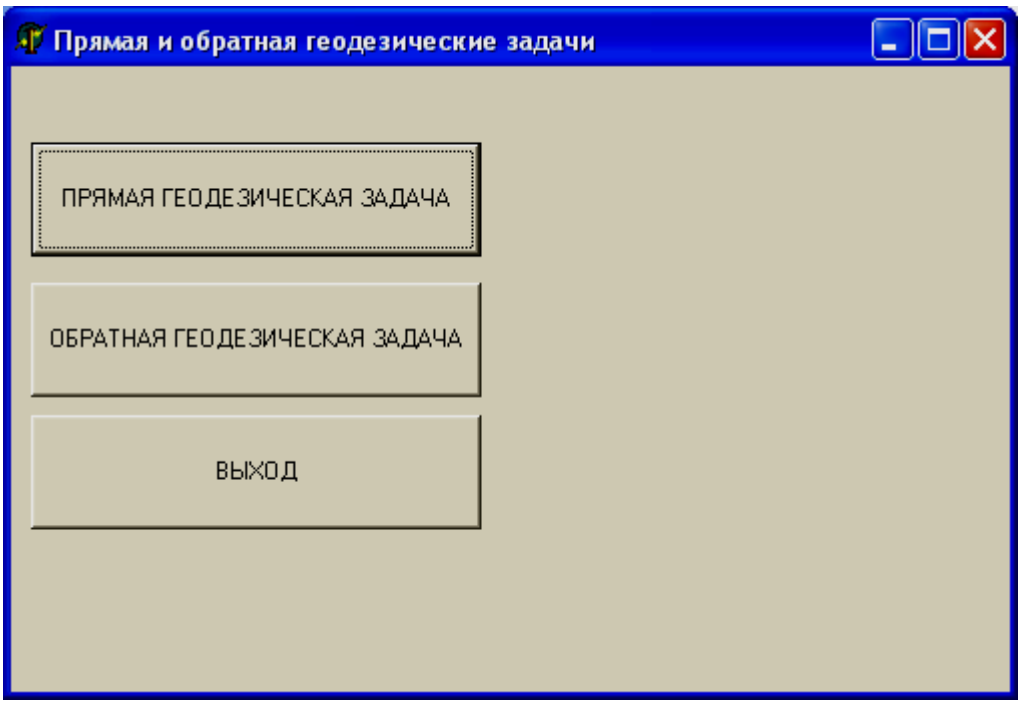
Текст модуля

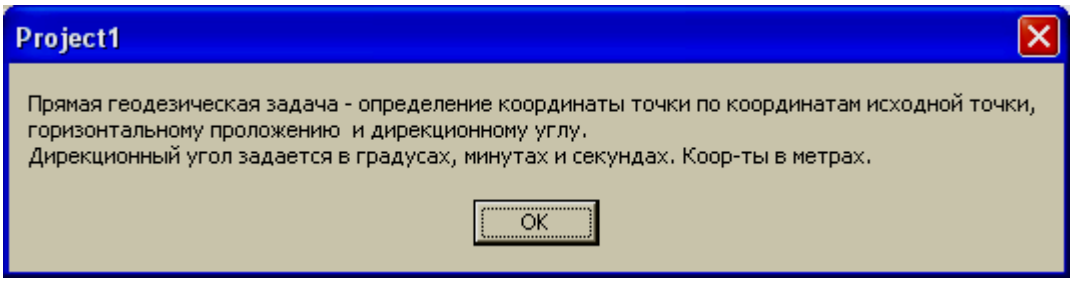
```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
Dialogs, StdCtrls, Math;
      { библиотечный модуль Math нужно добавить в стандартный
набор модулей для подключения математической библиотеки}
……..
var
Form1: TForm1;
XA,YA:Extended; //координаты точки A
XB,YB:Extended; // координаты точки B
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 AB
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 AB из окон ввода
XA:=StrToFloat(InputBox('Прямая геодезическая задача',
'Введите координату X точки A','0'));
YA:=StrToFloat(InputBox('Прямая геодезическая задача',
'Введите координату Y точки A','0'));
G:=StrToInt(InputBox('Прямая геодезическая задача',
'Введите часть дирекционного угла линии AB в градусах' +#13
+'Значение < 360','0'));
M:=StrToInt(InputBox('Прямая геодезическая задача',
'Введите часть дирекционного угла линии AB в минутах'+
#13+'Значение <60','0'));
S:=StrToFloat(InputBox('Прямая геодезическая задача',
```

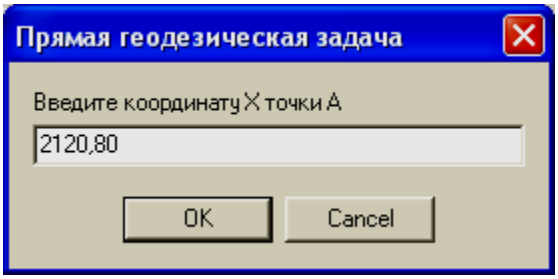
```
        'Введите часть дирекционный угла линии АВ в секундах'
+#13+'Значение <60','0'));
        L:=StrToFloat(InputBox('Прямая геодезическая задача',
        'Введите горизонтальное проложение линии АВ в метрах','0'));
        // вычисление координат точки В
        XB:=XA+L*cos((g+m/60+s/3600)/180*pi);
        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,Fffixed,10,2)+ #13
         + 'YB='+FloatToStrF(YB,Fffixed,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;
```

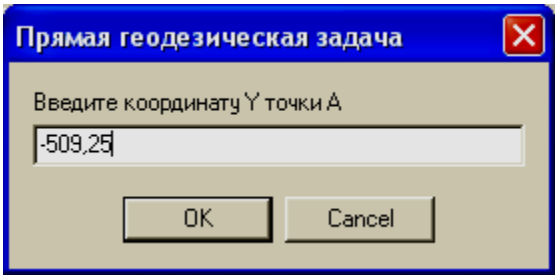
```
// перенести фокус в первое поле ввода на форме
Edit1.SetFocus;
Button4.Visible:=True;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
// ввод координат точек A и B из полей ввода на форме
XA:=StrToFloat(Edit1.Text);
YA:=StrToFloat(Edit2.Text);
XB:=StrToFloat(Edit3.Text);
YB:=StrToFloat(Edit4.Text);
DX:=XB-XA;
DY:=YB-YA;
// вычисление горизонтального проложения линии AB
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 квандарте 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,Fffixed,10,2);
Label9.Caption:=IntToStr(G);
Label10.Caption:=IntToStr(M);
Label11.Caption:=FloatToStrF(S,Fffixed,5,1);
end;
end.
```

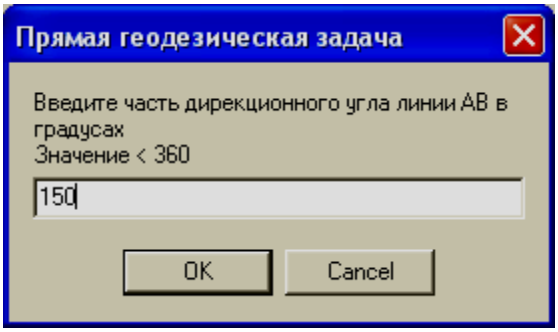
Такие окна будут появляться на экране при выполнении проекта:

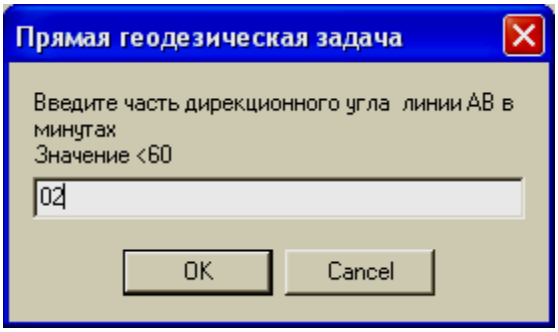

Прямая и обратная геодезические задачи
ПРЯМАЯ ГЕОДЕЗИЧЕСКАЯ ЗАДАЧА
ОБРАТНАЯ ГЕОДЕЗИЧЕСКАЯ ЗАДАЧА
ВЫХОД

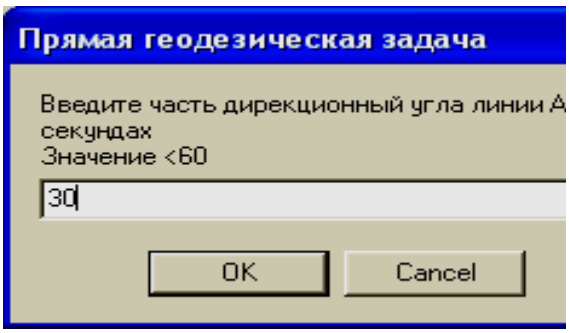

Project1
Прямая геодезическая задача - определение координаты точки по координатам исходной точки, горизонтальному проложению и дирекционному углу.
Дирекционный угол задается в градусах, минутах и секундах. Коор-ты в метрах.
OK

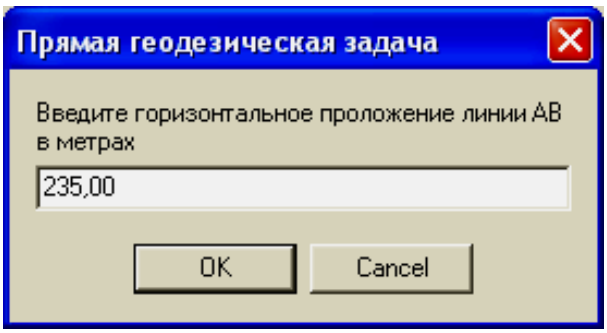

Прямая геодезическая задача
Введите координату X точки A
2120,80
OK
Cancel

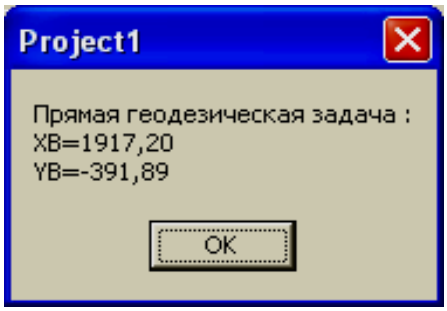

Прямая геодезическая задача
Введите координату Y точки A
-509,25
OK
Cancel

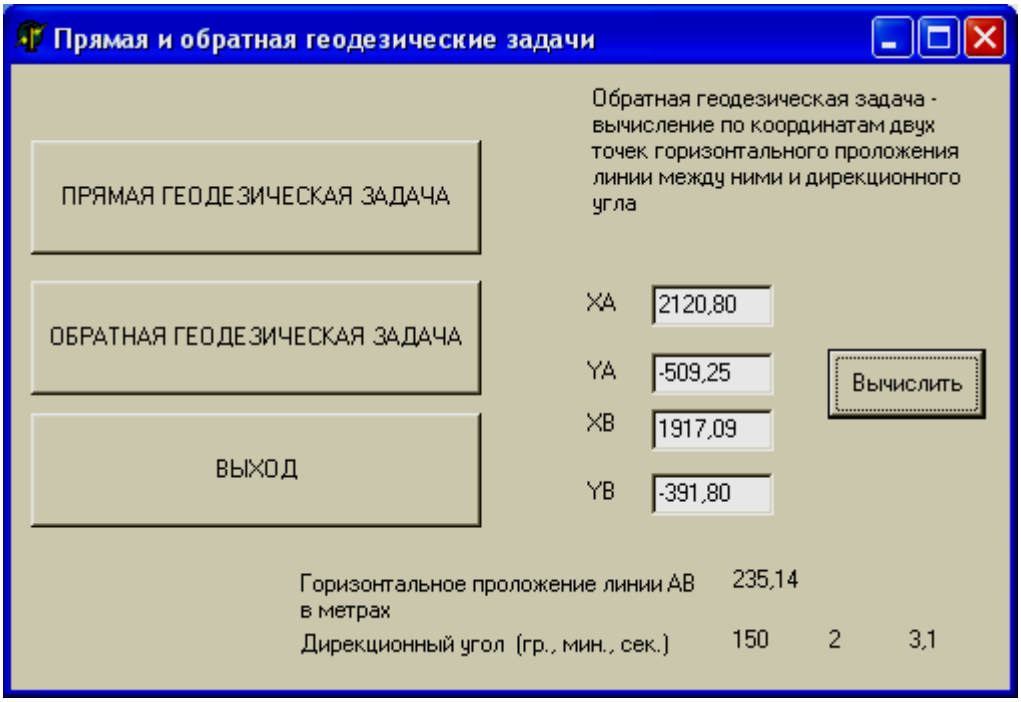

Прямая геодезическая задача
Введите часть дирекционного угла линии AB в градусах
Значение < 360
150
OK
Cancel


Прямая геодезическая задача
Введите часть дирекционного угла линии AB в минутах
Значение <60
02
OK
Cancel









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 происходить 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:

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

№ 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:

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

№ 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:

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

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

№ 3.8 Даны действительные числа:δ и σ. Найти p(δ, σ) 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 A (X_A, Y_A), горизонтальному проложению (L в м) от точки A до точки B и дирекционному углу (α) этого проложения найти координаты точки B:

$$X_B = X_A + L \cdot \cos\alpha$$
$$Y_B = Y_A + L \cdot \sin\alpha$$

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

В программе предусмотреть следующие запросы при вводе исходных данных:

XA =, YA =

ПРОЛОЖЕНИЕ AB =

ДИР. УГОЛ AB: G, M, S =

Форма вывода результата:

XB = значение XB YB = значение YB.

№ 3.12 Написать программу, которая по заданному значению вертикального угла ν и расстоянию S (в м) между точками A и B определяет превышение точки B над точкой A:

$$\mathrm{R} = \mathrm{S} \cdot \sin\nu$$

и горизонтальное проложение между A и B:

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

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 α, 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 δ'_x
DY1 = вводить δ'_y
DX2 = вводить δ''_x
DY2 = вводить δ''_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 β и γ. 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 (ρ,Θ) 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 ρ округлять до 0,1 м. Значение Θ 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 M из полярной системы в декартову, считая, что полюс совпадает с началом декартовых координат, а полярная ось совпадает с осью OX:

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

Здесь ρ и Θ – координаты точки M в полярной системе, где Θ 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 α, образуемого осью лесной полосы и направлением ветра:

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

Значение α 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 A и прилежащих к ней углов β и γ. Значения углов вводить в градусах, минутах и секундах, длину стороны A – 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 α определять в градусах, минутах и секундах, округлив секунды до 0,1 сек.

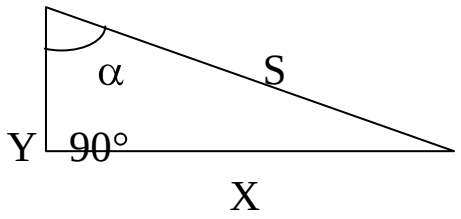


Рис. 3-1

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 α (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 >= 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) and (x < b) or (y<c)

Допустим значения переменных следующие: x=5; a=3; b=8; y=5; c=4. Значения простых логических выражений: 5>3= 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 <0ператор1> [else <0ператор2>];

Условие представляет собой выражение логического типа. Оператор работает следующим образом: если условие истинно (имеет значение True), то выполняется Оператор1, в другом случае выполняется 0ператор2. Оба оператора могут быть составными. Условный оператор может быть записан в сокращенной форме, когда слово else и оператор после него отсутствуют.

if <Условие> then <0ператор1>;

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 стоят только один оператор. В таких случаях операторы, которые нужно выполнить оформляются, как составной оператор (заключаются 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 отличии 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 OK.

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).

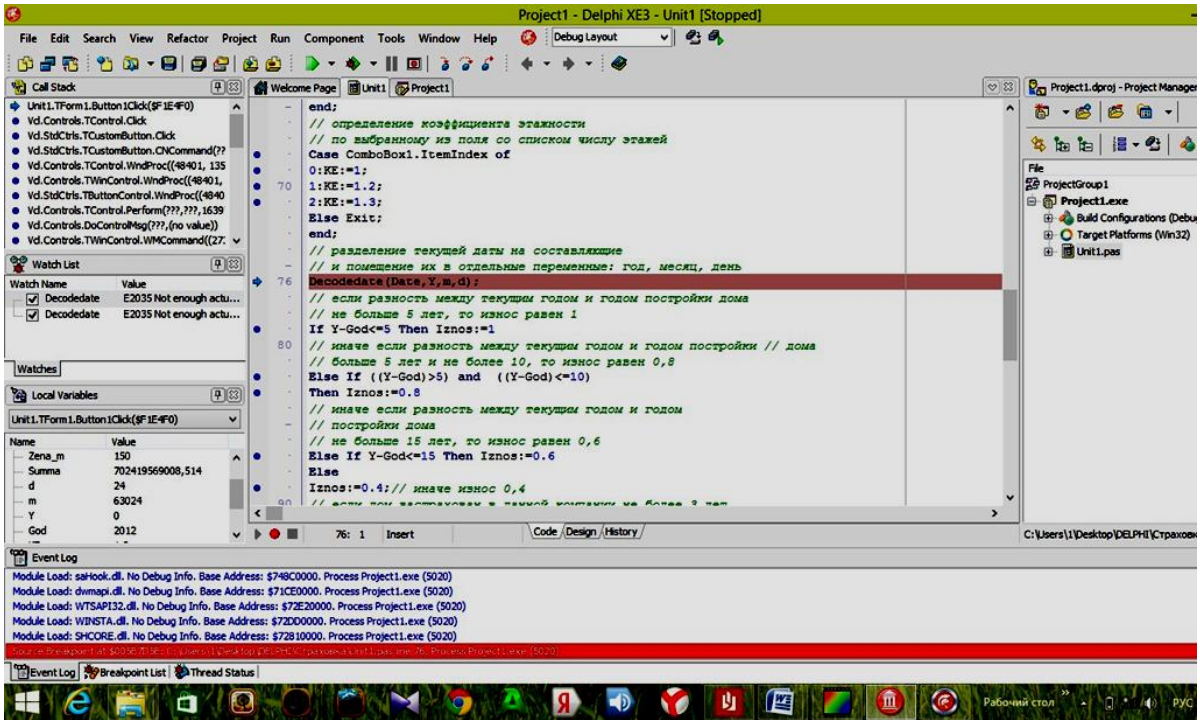


Рис. 4.1 Окно Delphi XE при использовании команд отладчика

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, сбои 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 случае 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:

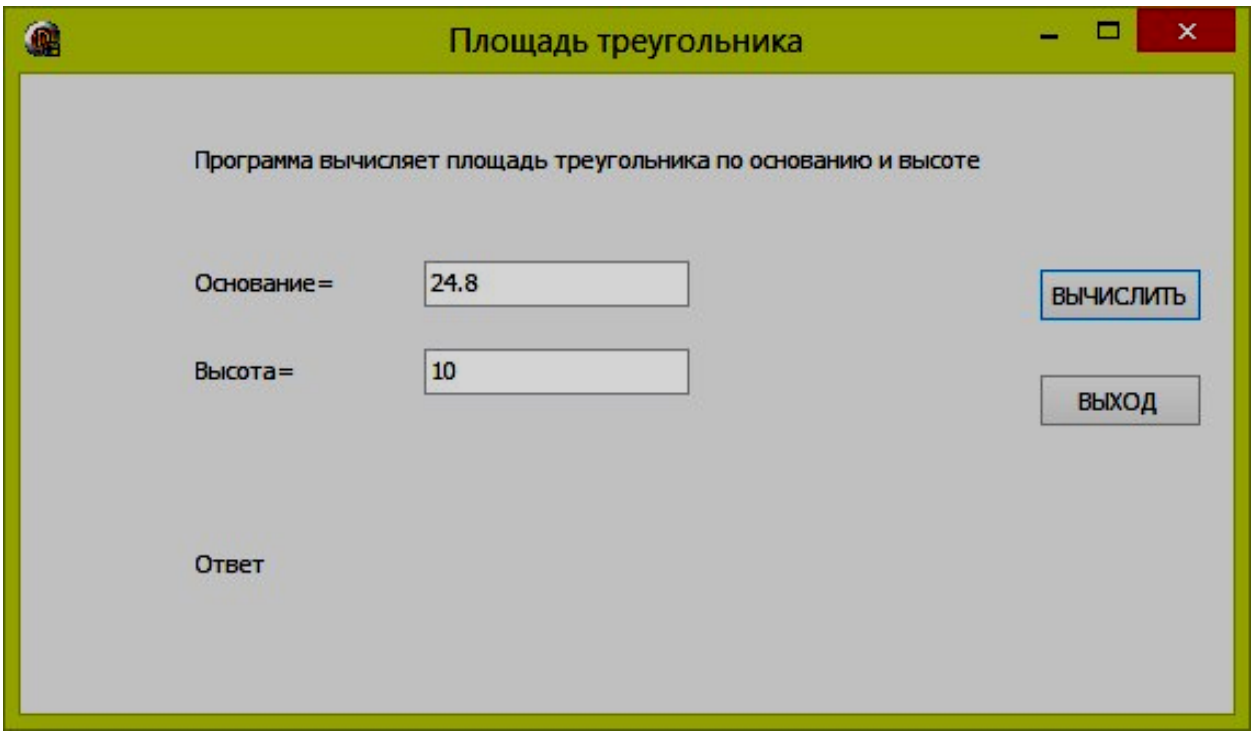
```
procedure TForm1.Button1Click(Sender: TObject);
begin
try
a:=StrToFloat(Edit1.Text);
h:=StrToFloat(Edit2.Text);
S:=1/2*a*h;
S:=S/10000;
Label4.Caption:='Площадь треугольника равна
'+FloatToStr(S)+'ГА';
except
if (Length(Edit1.Text) = 0) or (Length(Edit2.Text)=0)
```

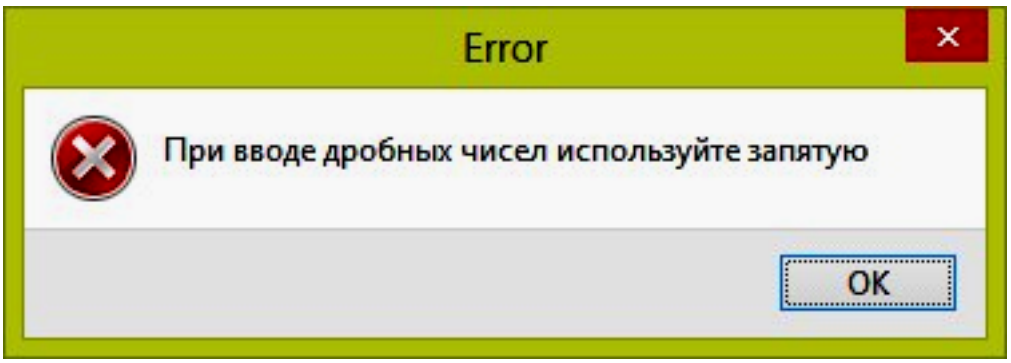
then MessageDlg('Надо ввести данные в оба поля', mtWarning,[mbOk],0)

else MessageDlg('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 except 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 OK 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.

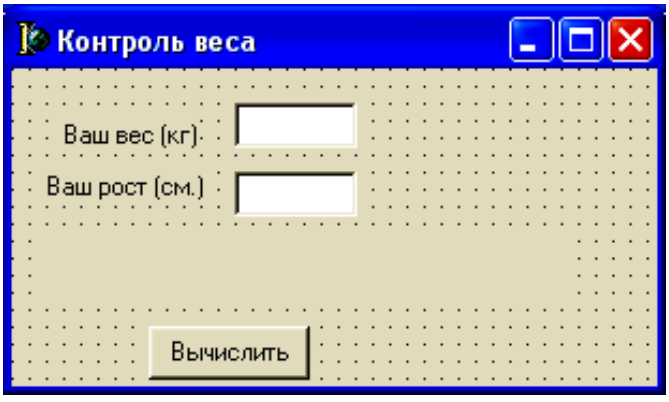
Компоненты

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

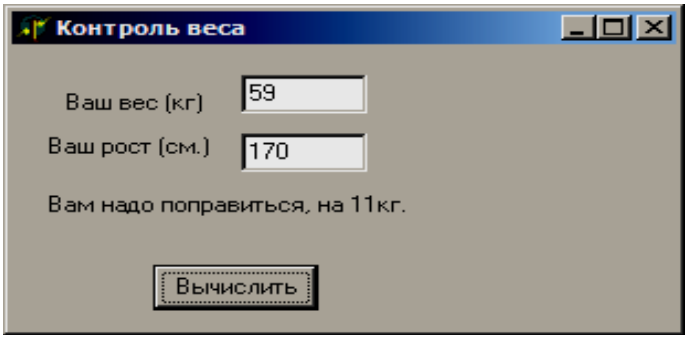
Проект формы



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.
```

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 до15 лет – 0,6

от 6 до 10лет – 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

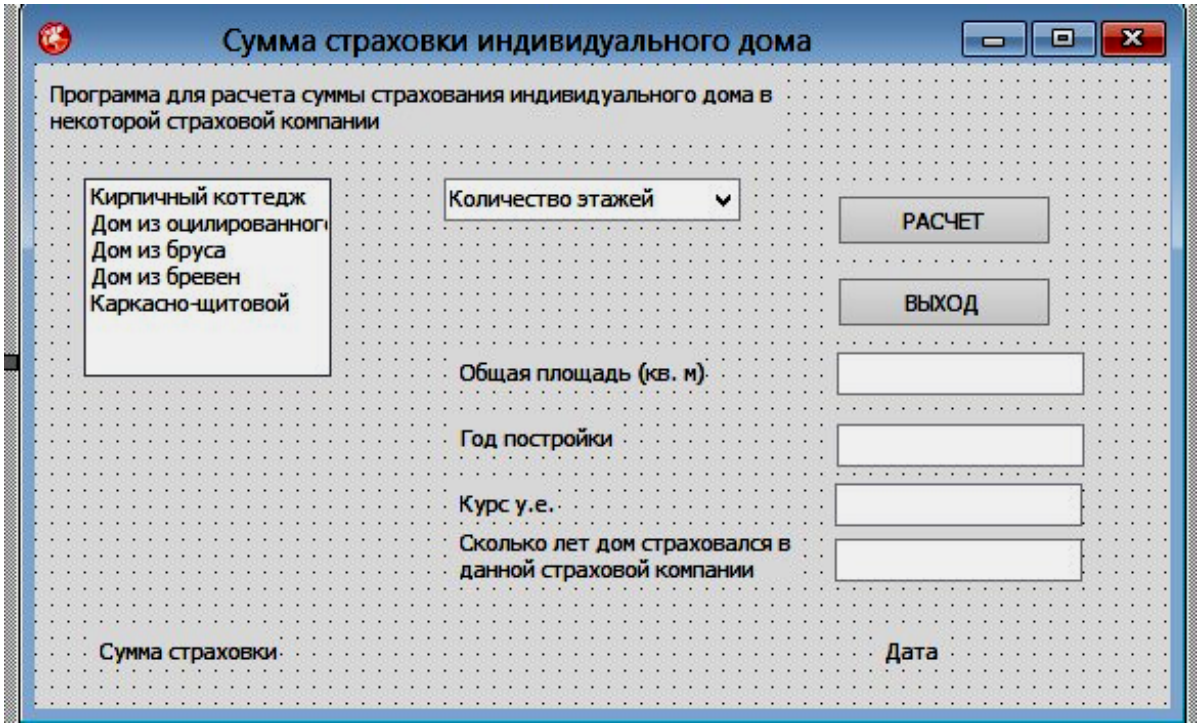
Имя компонента	Свойства компонентов	Значение	Назначение
Form1	Caption	Сумма страховки индивидуального дома	Заголовок формы
Label1	Caption WordWrap	Программа для расчета суммы страхования индивидуального дома в некоторой страховой компании True	Справочная информация для пользователя программы
Label2	Caption	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 оцилированного 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



Текст модуля

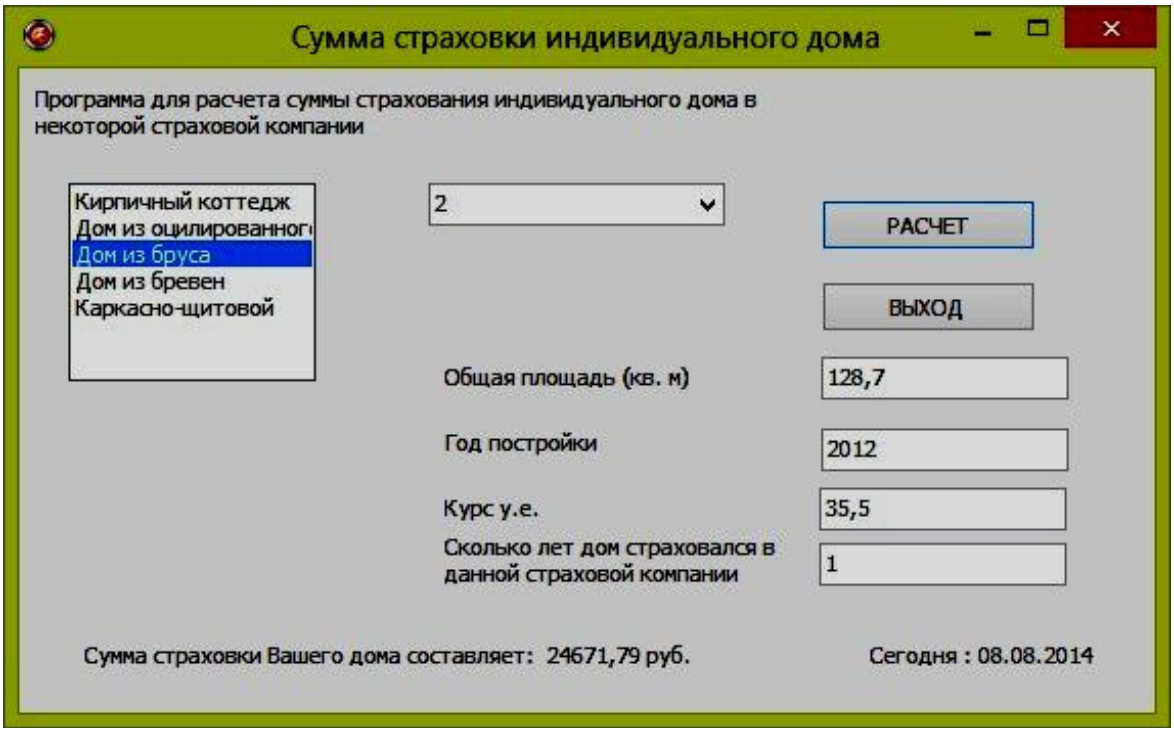
```
unit Unit1;

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,Fffixed,10,2)+' руб.'

end;

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

end.
```

Результаты работы программы



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

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

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

№ 4.3 Даны действительные числа x, y (x≠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

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

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

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

№ 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)

<= 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 A, B и C. 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 A, B и C. Треугольник со сторонами A, B и C возможен лишь в том случае, если одновременно выполняются неравенства $A+B \succ C, A+C \succ B, B+C \succ A$. 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 O трапециевидного участка (рис. 4–1) измерены расстояния A, C и B до трех остальных вершин и угол между сторонами A и B. Написать программу для вычисления площади этого участка по формуле:

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

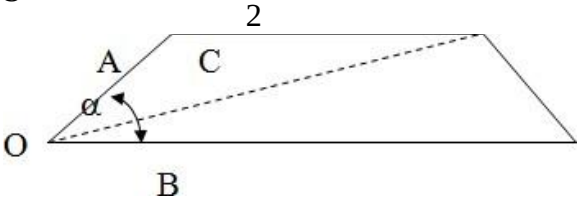


Рис. 4–1

Расстояния A, B и C измерены в метрах, угол α 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 $\alpha \geq 180°$

$C \prec A * \sin\alpha$

$0 \prec \alpha \leq 90°$ 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:

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

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

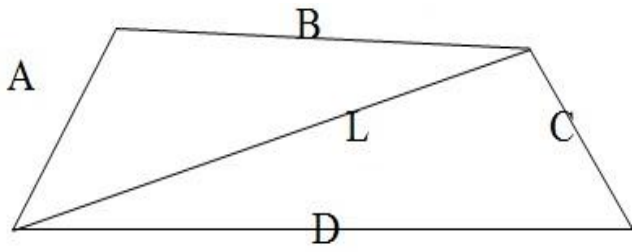


Рис. 4–2

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<=3гр.
В противном случае выводить сообщение:
ИЗМЕРЕНИЕ ПЛОЩАДЕЙ ПО ФОТОСХЕМЕ НЕДОПУСТИМО: *I*>3гр.
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, рассекающие земную поверхность.

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

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

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 $R \le P/2$.

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

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

где $\Delta X = X1 - X0$, $\Delta Y = Y1 - Y0$.

В программе предусмотреть выдачу запросов на ввод исходных данных:

X0, Y0=

R=

X1, Y1=

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

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

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

где в программе предусмотреть выдачу запросов на ввод исходных данных:

X0, Y0, Z0=

R=

X1,Y1, Z1=

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 *X1, Y1* 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=

X1, Y1=

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≠0, C=0 – прямая совпадает с осью *X*,

A≠0, B=0, C=0 – прямая совпадает с осью *Y*,

A=0, B≠0, C≠0 – прямая проходит через точку *Y= –C/B* параллельно оси *X*,

A≠0, B=0, C≠0 – прямая проходит через точку *Y= –C/A* параллельно оси *Y*,

A≠0, B≠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≠0, B≠0, C≠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≠0, B≠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 X1, Y1=

КООРДИНАТЫ X2, Y2=

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 приступать 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 <=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≠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:= K1.

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 K2. Если K меньше или равно K2, то вернемся к пункту 4 данного алгоритма. Иначе перейдем к следующему пункту алгоритма.

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

C. Составим алгоритм получения и вывода значений функции F(x) для значений аргументов, меняющихся в диапазоне от X1 до X2 с шагом Step.

1. Введем исходные данные X1,X2, Step.

2. Заведем переменную X для хранения текущего значения аргумента и присвоим ей значение крайней левой границы изменения аргументов. X:=X1;

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 X2 (X<=X2), то возвращаемся к пункту 3. В противном случае программа заканчивает свою работу.

D. Разберем алгоритм для создания цикла, выход из которого произойдет при достижении заданной точности вычислений **ε.**

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

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

(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 **ε**, 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 к := 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 выполнятся 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,FFfixed,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, FFfixed,10,2) +'   '+ 'F(X)=' +
FloatToStrF(F, FFfixed,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

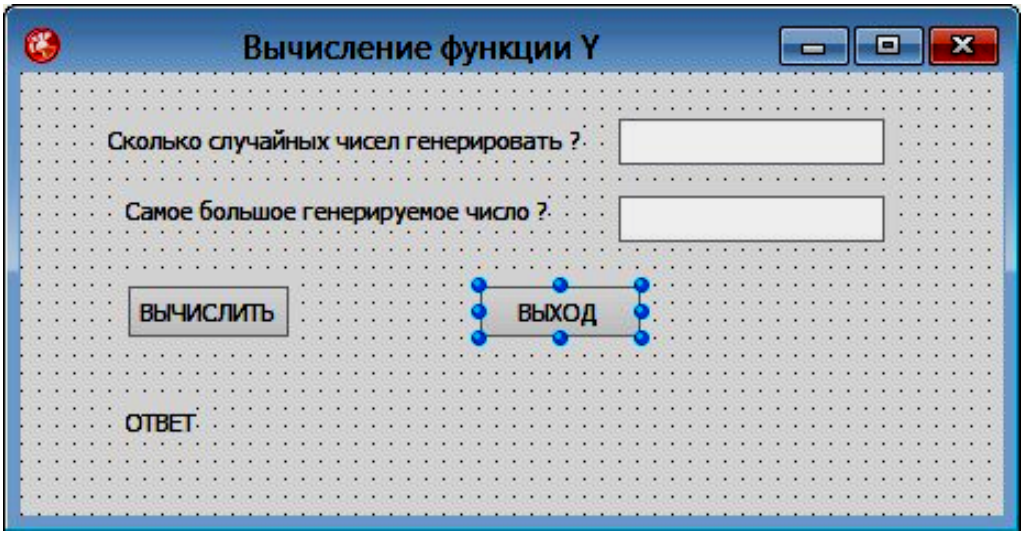
Задача 5а

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

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, System.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, FFfixed,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, πn, $\pi/(4n^2)$,$1/n$.

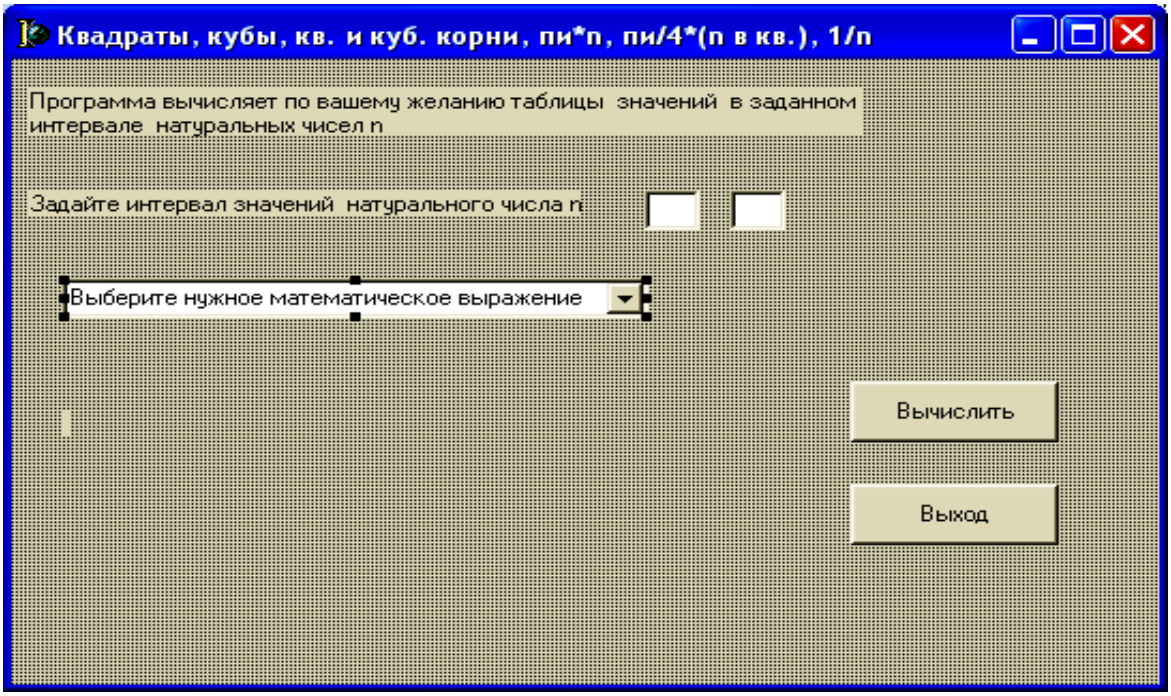
Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Квадраты, кубы, кв. и куб. корни, пи*n, пи/4*(n 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 Пи*n Пи/[4*(n 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),Fffixed,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)),Fffixed,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),Fffixed,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)),Fffixed,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,Fffixed,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:

$i = h / s$

$i^o = Arc\tan(i) / \pi * 180$

$i \ в \ гр. = целая \ часть(i^o)$

$i \ в \ мин. = округление \ ((i^o - i \ в \ гр.) * 60)$

Компоненты

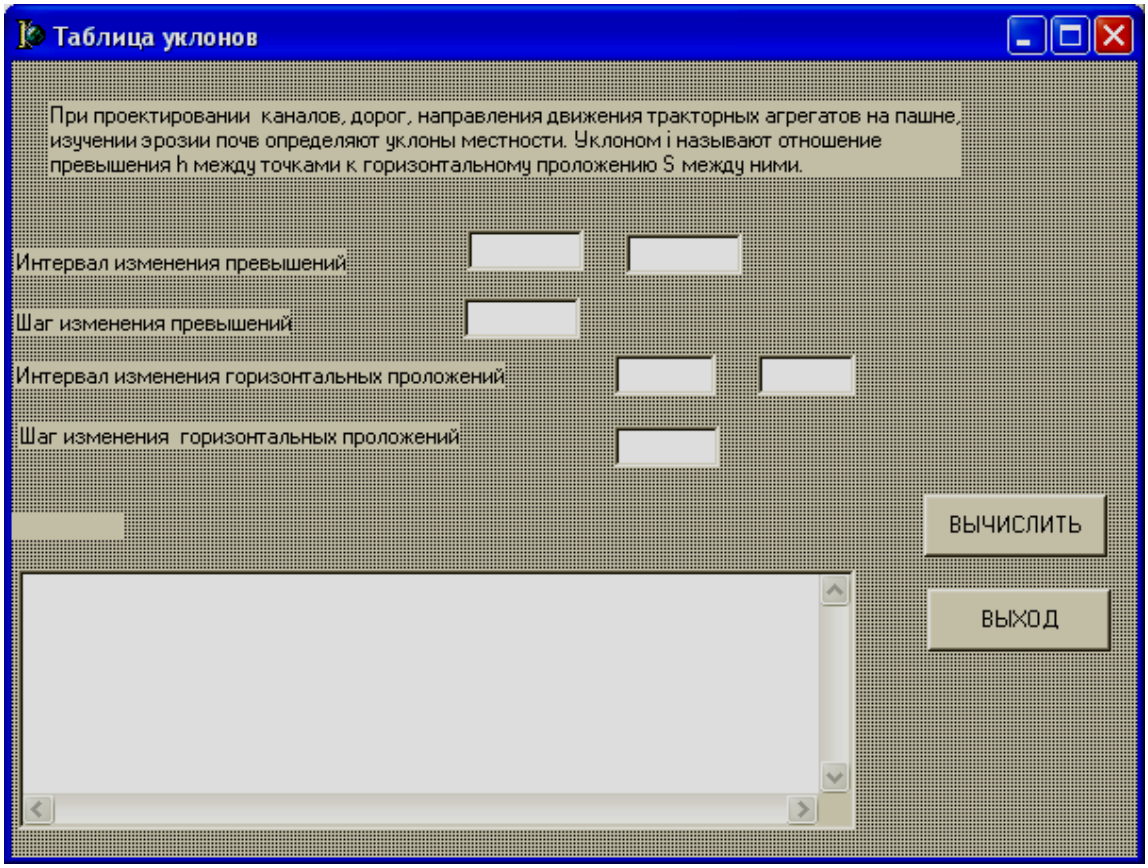
Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Таблица уклонов	Заголовок формы
Label1	Caption	При проектировании каналов, дорог, направления движения тракторных агрегатов на пашне, изучении эрозии почв определяют уклоны местности. Уклоном i называют отношение превышения h между точками 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



Текст модуля

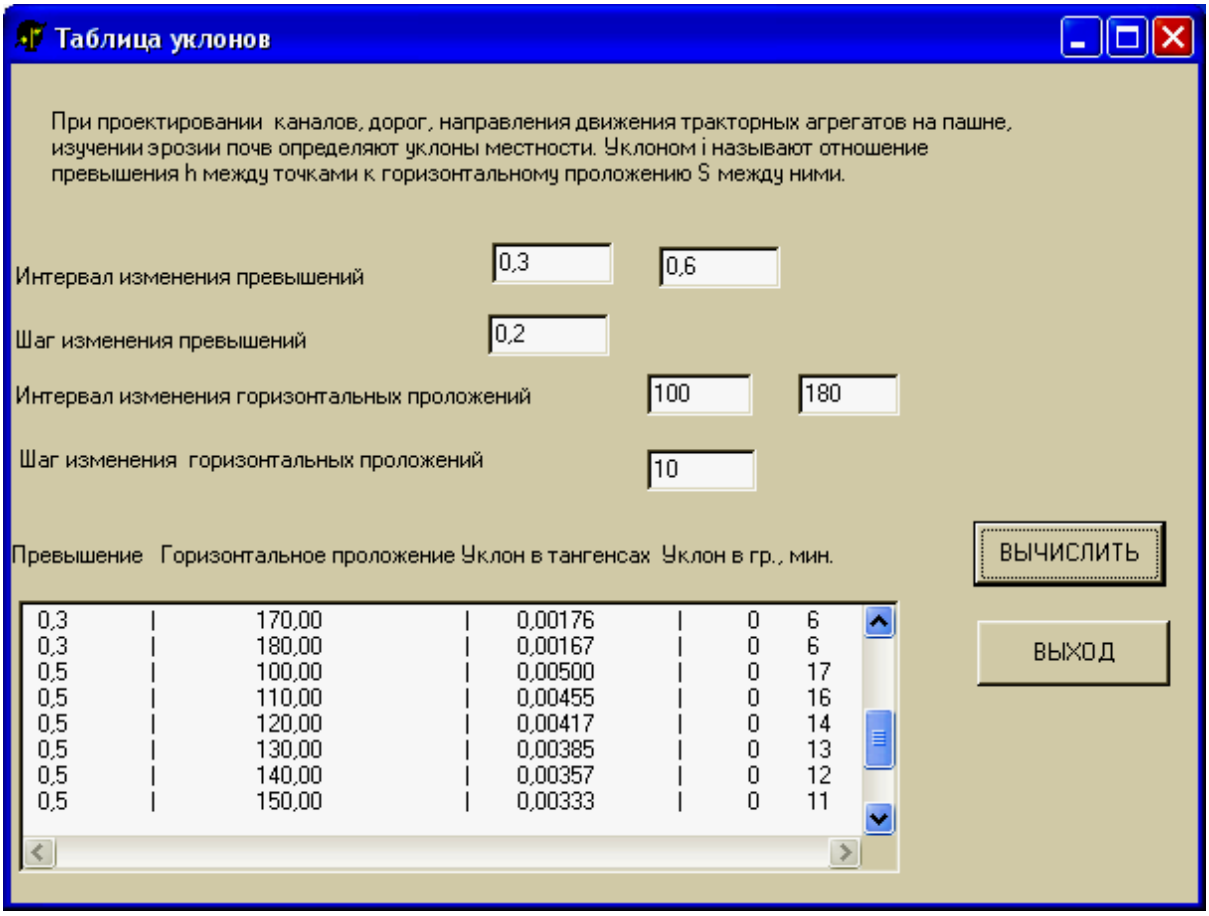
```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
Dialogs, StdCtrls, Math;
// библиотека математических функций Math
// подключается программистом
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Label3: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



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.

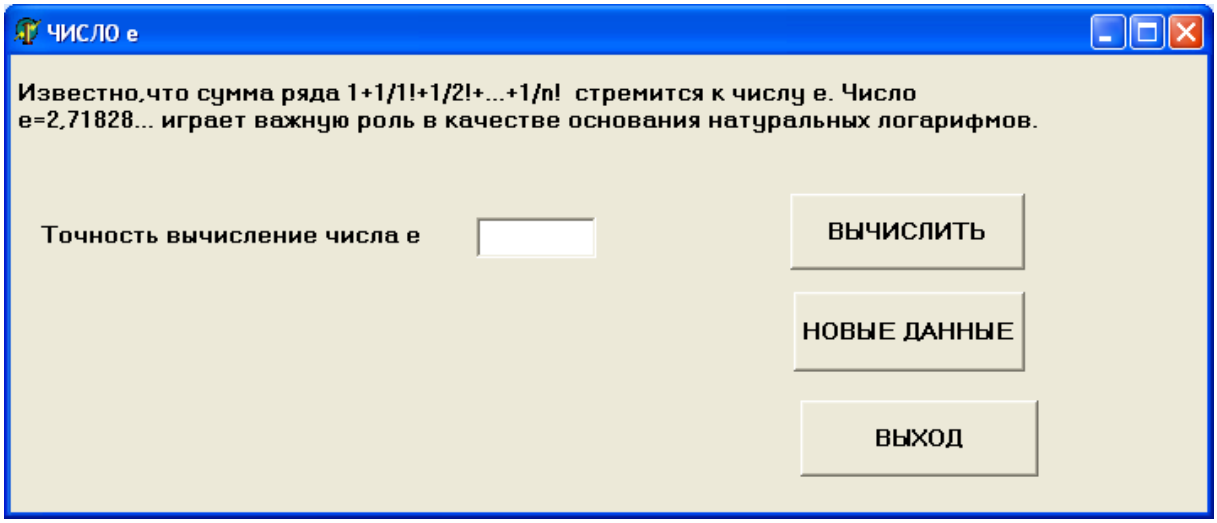
Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Число е	Заголовок формы
Label1	Caption	Известно, что сумма ряда 1+1/1!+1/2!+...+1/n! стремится к числу е. Число е=2,71828 играет важную роль в качестве основания натуральных логарифмов.	Справочная информация для пользователя программы
Label2	Caption	Точность вычисления числа е	Подсказка пользователю
Edit1	Text		Поле для ввода точности вычисления числа е
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 е

Переменные

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

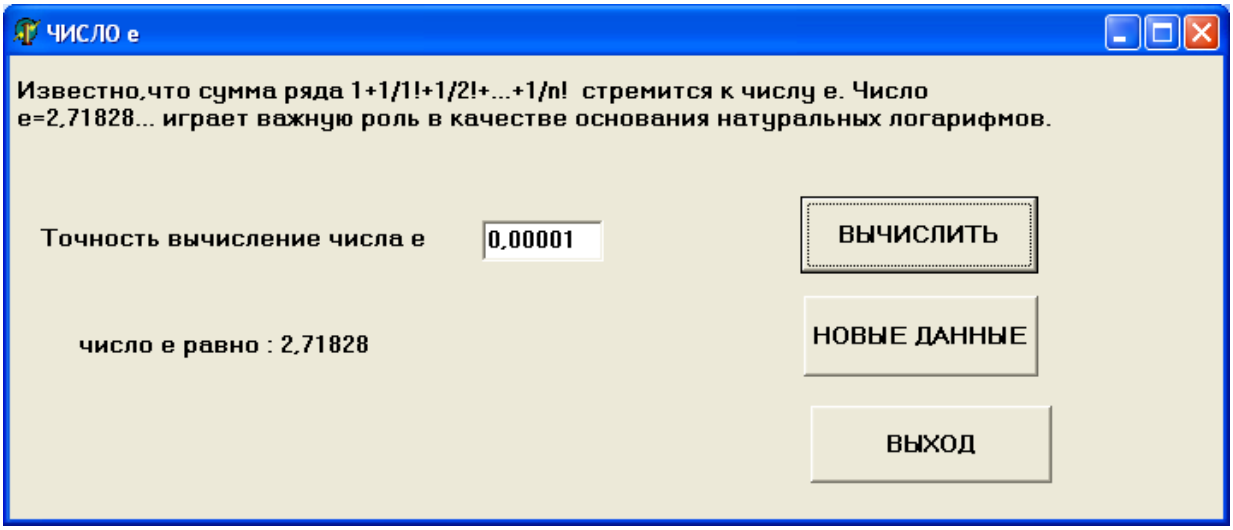
Проект формы



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,Fffixed,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 N1 до N2 с шагом один.

2. Опишите алгоритм произведения ряда натуральных чисел из диапазона значений K1 и K2 с шагом Step, не равным единице (Step≠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,...,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≤ S ≤ 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

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

№ 5.7 Вычислить и показать на форме величины функции $y = \sqrt[3]{x}$ для значений N1≤ *x* ≤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

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

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 N1 до N2, , где N1, N2 – натуральные числа.

№ 5.13 Написать программу для вычисления значений $Q = \sqrt{Y}$ при изменении Y от N1 до N2 с шагом K, где K, N1, N2 – натуральные числа.

№ 5.14 Написать программу для вычисления суммы обратных величин чисел от N1 до N2 с шагом K, где K, N1, N2 – натуральные числа.

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

№ 5.16 Составить программу для вычисления суммы членов последовательности:

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

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

№ 5.17 Составить программу для вычисления суммы ряда:

$\frac{1}{1^3} + \frac{1}{2^3} + \frac{1}{3^3} + ... + \frac{1}{n^3}$, n – натуральное число.

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

№ 5.18 Составить программу для вычисления суммы ряда:

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

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

№ 5.19 Вычислить произведение

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

Вычисление прекратить, как только очередной сомножитель станет меньше введенного пользователем положительного числа ε (0<ε≤0,1).

№ 5.20 Вычислить произведение

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

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

Глава 6. Использование одномерных массивов

В данной главе рассматриваются следующие вопросы:
массивы, компонент Memo для ввода и вывода массивов

Теория

Массивы

Массивом называется упорядоченная индексированная совокупность однотипных элементов, имеющих общее имя. Элементами массива могут быть данные различных типов, включая структурированные. Как представить массив? Студенческая группа представляет собой массив. Она имеет общее имя, например, 12Б. Каждый студент является элементом массива со своим порядковым номером по журналу. Каждый элемент массива однозначно определяется *именем* массива и *индексом* (номером этого элемента в массиве) или несколькими индексами, если массив многомерный. Для обращения к отдельному элементу массива указываются имя массива и номер или номера элемента, заключенные в квадратные скобки, например,

a[7] или arr[3, 35].

Количество индексных позиций определяет мерность массива (одномерный, двумерный и т. д.), при этом мерность массива не ограничивается. В математике аналогом одномерного массива является вектор, а двумерного массива – матрица. Индексы элементов массива должны принадлежать порядковому типу. Наиболее часто типом индекса является целочисленный тип.

Различают **статические** и **динамические** массивы.

Статические массивы представляют собой массивы, границы индексов и соответственно размеры которых задаются при объявлении, т. е. известны до компиляции программы. В процессе

выполнения программы структура статического массива не изменяется. У динамических массивов можно менять размеры во время выполнения программы, поэтому память для данных выделяется динамически.

Пример. Объявление статических массивов.

```
a1: Array[1 .. 100] of integer;
a2: Array[1..26] of char;
Type tm = Array[l .. 10, 1 .. 100] of real;
Var a3, a4: tm;
```

Переменные a1 и a2 являются одномерными массивами, соответственно на 100 целых чисел и 26 символов.

Переменные a3 и a4 являются двумерными массивами по 1000 элементов (10 строк и 100 столбцов). Каждый элемент этих массивов представляет собой число типа real. Для объявления массивов a3 и a4 введен специальный тип tm.

Доступ к каждому элементу массива осуществляется по индексу. Например, a1[34] –тридцать четвертый по порядку элемент массива a1; a2[12] – двенадцатый элемент массива a2 (находящийся на 12 месте); a3[5,15] –элемент двухмерного массива a3, находящийся в пятой строке и пятнадцатом столбце; a4[9,6] – элемент двухмерного массива a4, находящийся в девятой строке и шестом столбце.

Компонент Memo для ввода и вывода массивов

Memo — многострочный текстовый редактор (иная форма Edit). Memo находится на вкладке Standard. Подразумевает работу с большими текстами. Memo может выполнять основные функции редактора (ввод и редактирование текста). Memo имеет ограничения на объем текста в 32Кб, это составляет 10-20 страниц текста.

Главные свойства компонента Memo

Свойство	Назначение
Name	Имя компонента
Text	Текст, находящийся в поле Memo. Рассматривается как единое целое
Lines	Текст, находящийся в поле Memo. Рассматривается как совокупность строк. Доступ к строке осуществляется по номеру
Lines.Count	Количество строк в поле Memo

Присвоить переменной ST весь текст, находящийся в компоненте Memo, можно так:

```
ST:=Memo1.Text;
```

Переменной можно присвоить содержимое одной строки Memo:

```
a[i]:= Memo1.Lines[i];
```

Добавить строку в Memo можно так:

```
Memo1.Lines.Add (′Еще одна строка′);
Memo1.Lines.Add(a[i])
```

При использовании компонента Memo для ввода массива значение каждого элемента массива следует вводить в отдельной строке и после ввода каждого элемента массива нажимать клавишу ENTER.

Получить доступ к находящейся в поле Memo строке текста можно при помощи свойства Lines, указав в квадратных скобках номер нужной строки (строки нумеруются с 0).

***Замечание**. В нулевой строке поля Memo по умолчанию выводится имя самого компонента Memo. Чтобы номер элемента вашего массива и номер строки Memo совпадали, не надо стирать имя Memo. Тогда в первой строке будет находиться первый элемент массива, во второй строке – второй элемент и т. д.*

Чтение элементов символьного массива из Memo можно написать, например, так:

```
For i:=1 to Size do
    a2 [i]:=Memo1.Lines[i];
```

Чтение элементов целочисленного массива из Memo запишется так:

```
For i:=1 to N do
    a1[i]:=StrToInt(Memo1.Lines[i]);
```

Чтение элементов массива действительных чисел из Memo запишется так:

```
For i:=1 to N do
    a3[i]:=StrToFloat(Memo1.Lines[i]);
```

Запись элементов символьного массива в Memo можно написать, например, так:

```
For i:=1 to Size do
    Memo1.Lines.add(a2 [i]);
```

Запись элементов целочисленного массива в Memo запишется так:

```
For i:=1 to N do
    Memo1.Lines.add(IntToStr(a1[i]));
```

Запись элементов массива действительных чисел в Memo запишется так:

```
For i:=1 to N do
    Memo1.Lines.add(FloatToStr(a3[i]));
```

или

```
For i:=1 to N do
    Memo1.Lines.add(FloatToStrF(a4[i],fffixed,8,2));
```

Практика

Рассмотрим алгоритмы нескольких типовых задач работы с массивами.

Задача 6а

Поиск минимального элемента в массиве.

Алгоритм решения.

1. Вводим количество элементов N в массиве Massiv(длину массива).

2. Вводим N элементов массива Massiv (сами элементы).

3. Переменной MIN (в ней будет храниться минимальный элемент) присвоим значение первого элемента массива:

MIN:=MASSIV[1];

4. Индексу элемента массива I присвоим значение 2. Поиск начинаем со второго элемента.

5. Сравним MIN и текущий элемент массива MASSIV[I]. Если текущий элемент Massiv[I] меньше чем MIN, то в переменную MIN поместим текущее значение элемента массива: MIN:=MASSIV[1]; (получили новое значение минимума и переходим к следующему пункту алгоритма (пункт 6)).

Иначе сразу переходим к следующему пункту алгоритма (пункт 6).

6. Увеличим индекс элемента массива на единицу I:=I+1;

7. Сравним текущий индекс элемента массива с количеством элементов в массиве. Если массив не закончился (I<=N),то воз-

вращаемся к пункту 5. Иначе переменная MIN содержит минимальный элемент массива (переходим к пункту 8).

8. Выводим минимальный элемент массива MIN.

Приведем фрагмент программы, который реализует поиск минимального элемента в массиве согласно обсуждаемому здесь алгоритму.

```
MIN:=MASSIV[1];
For I:=2 To N Do
IF MASSIV[I]<MIN
                Then MIN:=MASSI[I];
Label3.Caption:= 'Минимальный элемент массива равен '+
FloatToStrF(MIN, Fffixed,10,2);
```

Задача 6b

Дан массив, состоящий из N вещественных чисел. Подсчитать количество отрицательных, положительных и нулевых элементов в массиве.

Опишем алгоритм для решения этой задачи.

1. Введем количество чисел в анализируемом массиве N.

2. Обнулим переменные−счетчики чисел SO − счетчик отрицательных чисел, SP − счетчик положительных чисел, S − счетчик нулей.

SO:=0; SP:=0; S:=0;

3. Начнем ввод и анализ массива с первого элемента. Индексу массива присвоим значение 1. I:=1;

4. Введем I−ый элемент массива A[I].

5. Если I−ый элемент массива A[I] меньше нуля, то счетчик отрицательных чисел увеличим на единицу: SO:=SO+1 и перейдем на пункт 7. В противном случае перейдем к пункту 6.

6. Если I−ый элемент массива A[I] больше нуля, то счетчик положительных чисел увеличим на единицу: SP:=SP+1 и перейдем к пункту 7. В противном случае, анализируемый элемент массива является нулем. Увеличим счетчик нулей на единицу S=S+1 и перейдем к пункту 7.

7. Увеличим индекс элемента массива на единицу I:=I+1;

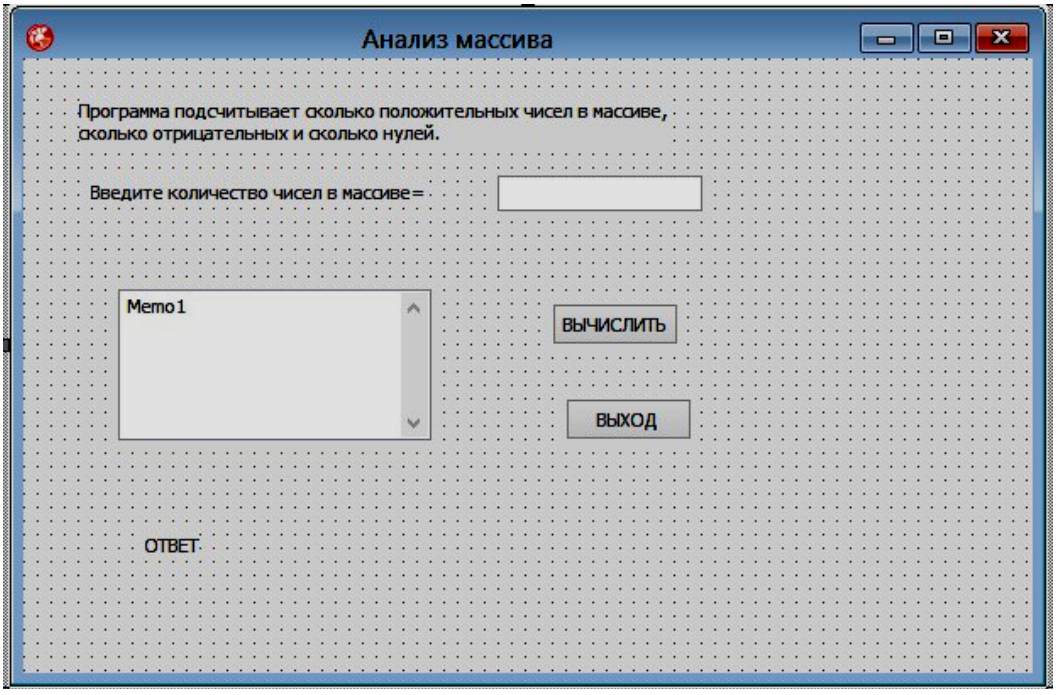
8. Сравним текущий индекс элемента массива с количеством чисел в массиве. Если I<=N, то перейдем к пункту 4. В противном случае выполним следующий пункт алгоритма.

9. Выводим полученные значения счетчиков отрицательных чисел, положительных чисел и нулей: SO, SP, S.

Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Анализ массива	Заголовок формы
Label1	Caption	Программа подсчитывает сколько положительных чисел в массиве, сколько отрицательных и сколько нулей.	Справочная информация для пользователя программы
Label2	Caption	Введите количество чисел в массиве	Подсказка пользователю
Edit1	Text	…..	Поле для ввода количества элементов в массиве
Memo1	Lines ScrollBars	Memo1 SSVertica	Компонент для ввода элементов массива в столбик (в одной строке – один элемент) Вертикальная полоса прокрутки
Label3	Caption	ОТВЕТ	Место, где будет выведен результат работы программы
Button1	Caption	ВЫЧИСЛИТЬ	Кнопка для запуска анализа массива
Button2	Caption	ВЫХОД	Кнопка для завершения работы программы

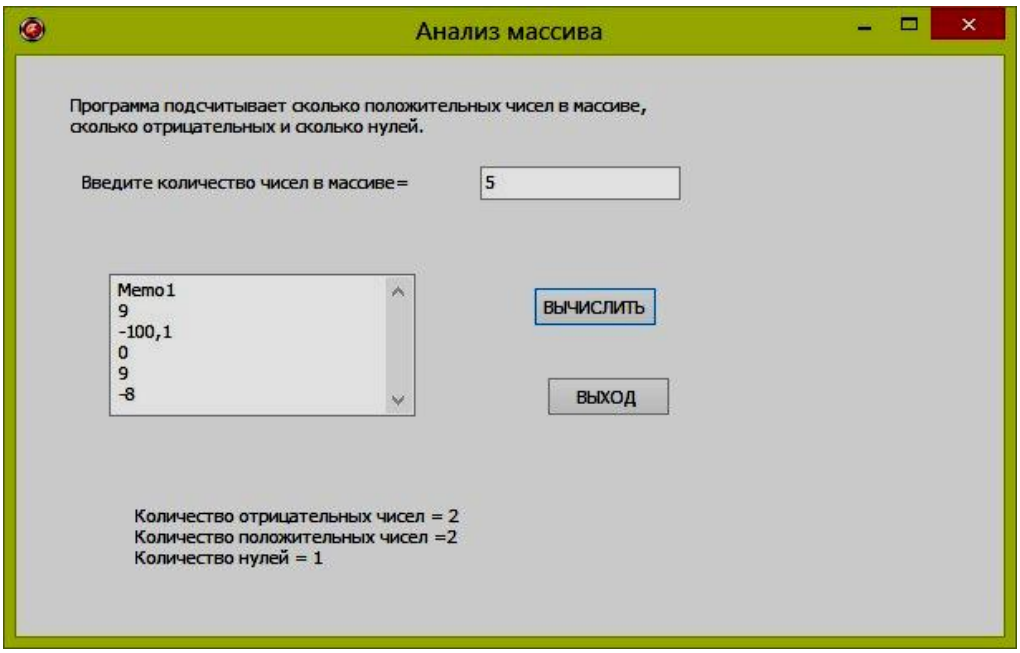
Проект формы



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;
  Label2: TLabel;
  Edit1: TEdit;
  Memo1: TMemo;
  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 A : array [1..1000] of Extended;// массив вводимых вещественных чисел
  N : Integer; // количество чисел в массиве
  I : Integer;// индекс элемента массива
  SO: Integer;// количество отрицательных чисел
```

```
 SP: Integer;// количество положительных чисел
 S : Integer;// количество нулей
begin
N:=StrToInt(Edit1.Text);
SO:=0; SP:=0; S:=0;
for I := 1 to N do
begin
 A[I]:=StrToFloat(Memo1.Lines[I]);
 if A[I]<0 then  SO:=SO+1
     else  if A[I]>0 then SP:=SP+1
           else S:=S+1;
end;
Label3.Caption:='Количество отрицательных чисел = '+
IntToStr(SO) + #13+
        'Количество положительных чисел =' + IntToStr(SP)
+#13+
        'Количество нулей = '+ IntToStr(S)
end;

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

end.
```



Задача 6с

Составить программу для сортировки массива действительных чисел по желанию пользователя по возрастанию или по убыванию. Результаты сортировки вывести в окно сообщения.

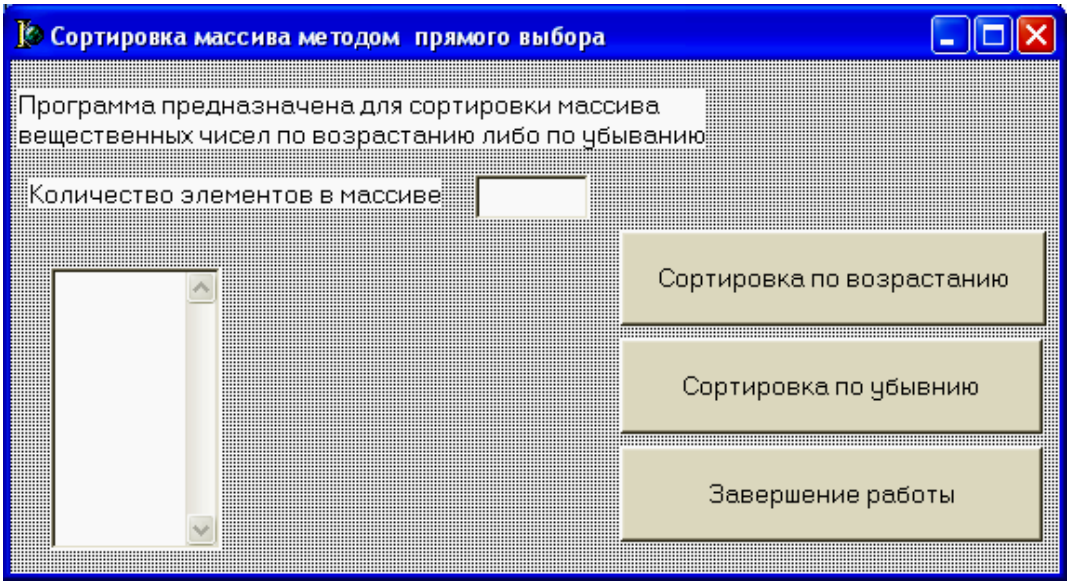
Компоненты

Имя компонента	Свойства компонент	Значение	Назначение
Form1	Caption	Сортировка массива методом прямого выбора	Заголовок формы
Label1	Caption	Программа предназначена для сортировки массива действительных чисел по возрастанию либо по убыванию	Справочная информация для пользователя программы
Label2	Caption	Количество элементов в массиве	Подсказка пользователю
Edit1	Text	…..	Поле для ввода количества элементов в массиве
Memo1	Lines ScrollBars	Memo1 SSVertical	Компонент для ввода элементов массива в столбик (в одной строке – один элемент) Вертикальная полоса прокрутки
Button1	Caption	Сортировка по возрастанию	Кнопка для выполнения сортировки по возрастанию
Button2	Caption	Сортировка по убыванию	Кнопка для выполнения сортировки по убыванию
Button3	Caption	Завершение работы	Кнопка для завершения работы программы

Переменные и массивы

Обозначение в программе	Содержание	Тип
N	Количество элементов в массиве действительных чисел	целый
Massiv	Массив действительных чисел	вещественный, расширенный
I	Текущий индекс элемента массива	целый
J	индекс элемента массива, с которого начинается поиск минимального либо максимального элемента	целый
buf	буфер для обмена местами элементов массива	вещественный, расширенный
st	строковое сообщение, содержащее результат	строковый

Проект формы



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;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button3Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
  var
```

```
    Form1: TForm1;
    Massiv: array [1..100] of Extended;// массив действительных
чисел
    N : Integer;
  implementation
  {$R *.dfm}
  procedure TForm1.Button3Click(Sender: TObject);
  begin
  Form1.Close
  end;
  procedure TForm1.Button1Click(Sender: TObject);
    Var I:Integer;// индекс элемента исходного массива
        j : Integer; //индекс элемента массива
        //с которого начинается
        // поиск минимального элемента
        buf: Extended; // буфер для обмена местами
        // элементов массива
        St:String; // строковое сообщение, содержащее результат
  begin
  //ввод количества элементов в массиве
  N:=StrToInt(Edit1.Text);
  //ввод массива действительных чисел
  For I:=1 To N do
  Massiv[I]:=StrToFloat(Memo1.Lines[I]);
  // поиск минимального элемента массива методом прямого
выбора
  {Алгоритм метода прямого выбора можно описать так. Сначала в
качестве минимального элемента берется первый элемент и с ним
последовательно сравниваются все следующие элементы массива,
если попадается элемент меньше принятого за минимальный, то они
меняются местами. В результате на первом месте оказывается самый
маленький элемент массива. Затем берется второй или следующий
элемент массива и просматривается с перестановкой оставшаяся
часть массива. Эта последовательность шагов повторяется пока но-
мер очередного элемента не достигнет N-1}
  For I:=1 to N-1 do
     For j:=I+1 to N do
       if Massiv[j] < Massiv[I]
```

```
          then
        begin
          buf:=Massiv[I];
          Massiv[I]:=Massiv[j];
          Massiv[j]:=buf;
          end;
        end;
    //вывод результатов сортировки
    St := 'Отсортированный по возрастанию массив'+#13;
    For I:=1 to N do
     st:=st+IntToStr(i)+ ' '+FloatToStrF(Massiv[I],Fffixed,10,2)+#13;
     ShowMessage(st);
    end;
    procedure TForm1.Button2Click(Sender: TObject);
    Var I:Integer;// индекс элемента массива
         j : Integer; //индекс элемента массива, с которого
         // начинается поиск максимального элемента
         buf: Extended; // буфер для обмена местами элементов
         // массива
         St:String;// строковое сообщение, содержащее результат
    begin
    //ввод количества элементов в массиве
    N:=StrToInt(Edit1.Text);
    //ввод массива действительных чисел
    For I:=1 To N do
    Massiv[I]:=StrToFloat(Memo1.Lines[I]);
    // поиск максимального элемента массива методом прямого
выбора
    // алгоритм аналогичен поиску минимального элемента.
    // отличие состоит в знаке сравнения
    For I:=1 to N-1 do
        For j:=I+1 to N do
            if Massiv[j] > Massiv[i] then
            begin
            buf:=Massiv[I];
        Massiv[I]:=Massiv[j];
        Massiv[j]:=buf
          end;
```

```
  end;
//вывод результатов сортировки
st:= 'Отсортированный по убыванию массив'+#13;
For I:=1 to N do
st:= st+IntToStr(i)+ ' '+FloatToStrF(Massiv[I],Fffixed,10,2)+#13;
ShowMessage(st);
end;
end.
```

Рекомендуемый интернет-ресурс

http://www.youtube.com/watch?v=QVtwWA283to&list=PLDDB9D4A0E37B4953&index=6

Вопросы для самоконтроля

1. Дайте определение массива.
2. Что хранит индекс элемента массива?
3. Как объявить массив в программе?
4. Как использовать компонент Memo для чтения из него элементов массива?
5. Как использовать компонент Memo для записи в него элементов массива?
6. Как одновременно найти и минимальный и максимальный элементы массива (Задача 6а)?
7. Опишите алгоритм программы «Анализ массива»(Задача 6в).
8 Какая переменная хранит количество нулей в программе «Анализ массива» (Задача 6в).?
9. Опишите алгоритм сортировки методом прямого выбора (Задача 6с)..
10. Для чего используется переменная buf в программе «Сортировка массива»(Задача 6с)?

Задачи для самостоятельного решения

Группа А

№ 6.1 Дан массив действительных чисел. Найти сумму и произведение элементов массива.

№ 6.2 Дан массив действительных чисел. Найти сумму квадратов отрицательных элементов массива.

№ 6.3 Дан массив действительных чисел. Найти модуль суммы и сумму модулей элементов массива.

№ 6.4 Дан массив действительных чисел. Найти наибольший элемент массива.

№ 6.5 Дан массив действительных чисел. Найти наименьший из элементов массива с нечетными индексами.

№ 6.6 Дан массив действительных чисел. Найти среднее арифметическое из элементов массива. Сколько чисел в массиве меньше среднего арифметического? Сколько нулей в массиве?

№ 6.7 Дан массив действительных чисел. Найти среднее арифметическое из элементов массива. Сколько чисел в массиве, больше среднего арифметического? Сколько отрицательных чисел в массиве?

№ 6.8 Дан массив действительных чисел. Найти наибольший элемент массива и его индекс.

№ 6.9 Написать программу для вычисления среднего гармонического

Mn = n / (1/a1+ 1/a2+...+ 1/an), где ai≠0.

№ 6.10 Дан массив целых чисел. Подсчитать количество четных и нечетных чисел в массиве.

Группа Б

№ 6.11 Написать программу для обработки результатов N–кратного измерения величины L. Программа должна вычислять

среднее:

$$\overline{L} = \frac{1}{N}\sum_{i=1}^{N} L_i,$$

где L_i – результат i–го измерения.

Среднюю квадратическую погрешность отдельного измерения:

$$m = \sqrt{\frac{N(\overline{L^2} - (\overline{L})^2)}{N-1}},$$

где

$$\overline{L^2} = \frac{1}{N}\sum_{i=1}^{N} L_i^2.$$

Среднюю квадратическую погрешность арифметической середины

$$M = m / \sqrt{N}.$$

Предельную абсолютную погрешность измерений $\Delta L = 3M$.

Количество измерений произвольно в пределах от 5 до 15.

В программе предусмотреть также проверку условия N>=5. Если это условие не выполняется, выдавать сообщение УВЕЛИЧИТЬ КОЛИЧЕСТВО ИЗМЕРЕНИЙ и осуществить переход к оператору ввода N.

Форма выдачи результатов:

УСРЕДНЕННЫЙ РЕЗУЛЬТАТ= значение

СКП ОТДЕЛЬНОГО ИЗМЕРЕНИЯ= значение

СКП СРЕДНЕГО= значение

ПРЕДЕЛЬНАЯ АБСОЛЮТНАЯ ПОГРЕШНОСТЬ= значение

№ 6.12 Написать программу для определения суммарной площади земельных участков:

$$S = S_1 + S_2 + ... + S_N$$

и средней квадратической погрешности определения этой суммы:

$$M = \sqrt{M_1^2 + M_2^2 + ... + M_N^2}.$$

Здесь S_i – площадь i–го участка; M_i – средняя квадратическая погрешность определения этой площади. Общее количество участков произвольно, но не более 255.

Форма вывода результатов:

СУММАРНАЯ ПЛОЩАДЬ= значение

СКП= значение

№ 6.13 Написать программу для вычисления среднего арифметического:

$$P = \frac{X_1 + X_2 + ... + X_N}{N}$$

Среднего геометрического:

$$G = \sqrt[N]{| X_1 \cdot X_2 \cdot ... \cdot X_N |}$$

Среднего гармонического:

$$H = \frac{N}{1/X_1 + 1/X_2 + ... + 1/X_N}$$

И среднего квадратичного:

$$B = \sqrt{X_1^2 + X_2^2 + ... + X_N^2}$$

для N – чисел. N – произвольное число, но не более 100. (Указание $\sqrt[N]{A} = A^{1/N}$).

Форма вывода результатов:

СРЕДНЕЕ АРИФМЕТИЧЕСКОЕ= значение

СРЕДНЕЕ ГЕОМЕТРИЧЕСКОЕ= значение

СРЕДНЕЕ ГАРМОНИЧЕСКОЕ= значение

СРЕДНЕЕ КВАДРАТИЧЕСКОЕ= значение

№ 6.14 Написать программу для вычисления средних арифметических:

$$P_1 = \frac{1}{N}\sum_{i=1}^{N} X_{1i,} P_2 = \frac{1}{N}\sum_{i=1}^{N} X_{2i,}$$

и средних квадратических отклонений

$$\sigma_1 = \sqrt{X_1^2 - P_1^2}, \sigma_2 = \sqrt{X_2^2 - P_2^2}$$

здесь

$$\overline{X_1^2} = \frac{1}{N}\sum_{i=1}^{N} X_{1i}^2, \overline{X_2^2} = \frac{1}{N}\sum_{i=1}^{N} X_{2i}^2$$

для двух массивов чисел, а также коэффициента корреляции между этими массивами:

$$r = \frac{\overline{X_1 X_2} - P_1 P_2}{\sigma_1 \sigma_2}$$

здесь

$$\overline{X_1 X_2} = \frac{1}{N}\sum_{i=1}^{N} X_{1i} X_{2i}.$$

№ 6.15 Дана информация о среднесуточной температуре каждого дня месяца. Подсчитать среднемесячную температуру. Сколько раз столбик термометра опускался ниже нуля? Сколько раз среднесуточная температура держалась выше 10 градусов.

№ 6.16 Написать программу для определения суммы горизонтальных проложений между точками с указанными координатами:

$$S = \sum_{i=1}^{N-1} \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2}$$

Общее количество точек произвольно, но не более 20.

Форма вывода результата:

СУММА ГОРИЗОНТАЛЬНЫХ ПРОЛОЖЕНИЙ= значение S.

Найти самое большое и самое маленькое горизонтальное проложение.

№ 6.17 Написать программу для определения суммарного расстояния между точками с заданными координатами и значениями вертикального угла, под которым из данной точки видна следующая точка:

$$S = \sum_{i=1}^{N-1} \frac{\sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2}}{\cos\alpha_i}$$

Общее количество точек произвольно, но не более 25.

Форма вывода результата:

СУММАРНОЕ РАССТОЯНИЕ = значение S.

№ 6.18 Дана информация о багаже нескольких пассажиров. Багаж каждого пассажира характеризуется двумя числами: количеством вещей и общим весом вещей. Выяснить, имеется ли пассажир, багаж которого превышает багаж каждого из остальных пассажиров и по числу вещей, и по весу.

№ 6.19 Дана информация о багаже нескольких пассажиров. Багаж каждого пассажира характеризуется двумя числами: количеством вещей и общим весом вещей. Определить, имеются ли два пассажира, багажи которых совпадают по числу вещей и различаются по весу не более чем на 0,5 кг.

№ 6.20 Дана информация о багаже нескольких пассажиров. Багаж каждого пассажира характеризуется двумя числами: количеством вещей и общим весом вещей. Выяснить фамилии пассажиров, багаж которых состоит из одной вещи весом не менее 30 кг.

Глава 7. Работа с двухмерными массивами

В данной главе рассматриваются следующие вопросы: **компонент StringGrid, ввод и вывод двумерных массивов с помощью компонента StringGrid, понятие о классах в Delphi,технология разработки многооконных проектов.**

Теория

Определение двумерного массива было дано в предыдущем параграфе.

Компонент StringGrid

Находится на вкладке Additional.

StringGrid – служит для представления текстовых данных в виде таблицы. Очень удобен для ввода и вывода массивов (особенно двухмерных). Доступ к каждому элементу таблицы происходит через свойство Cells. Наиболее важными для нас являются следующие свойства компонента StringGrid.

Свойство	Назначение
Name	Имя компонента
ColCount	Количество колонок таблицы
RowCount	Количество строк таблицы
Cells	Соответствующий таблице двумерный массив. Ячейка таблицы, находящаяся на пересечении столбца номер col и строки номер row определяется элементом cells[col, row]
FixedCols	Количество зафиксированных слева колонок таблицы. Зафиксированные колонки выделяются цветом и при горизонтальной прокрутке остаются на месте
FixedRows	Количество зафиксированных сверху строк таблицы. Зафиксированные строки выделяются цветом и при вертикальной прокрутке таблицы остаются на месте
Options.goEditing	Признак допустимости редактирования содержимого ячеек таблицы.True – редактирование разрешено. False – запрещено
Options.goTabs	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

Внимание! **У элементов таблицы StringGrid первый индекс это номер столбца, второй индекс это номер строки (cells[col, row]).**

Ввод и вывод двухмерных массивов с помощью компонента StringGrid

После создания на форме строковой таблицы (компонент StringGrid) и задания с помощью Инспектора Объектов ее свойств, в программе нужно записать группу операторов для подписи строк и столбцов таблицы, чтобы пользователю программы было понятно, как вводить данные. Это можно сделать, как и раньше, добавив на форму компонент ButtonN и создав процедуру*ButtonNClick.* Но можно записать и в другой процедуре, например, в процедуре, связанной с событием создания формы (FormCreate), или в процедуре связанной с событием выход (Exit) из компонента Edit, где вводится размерность массива.

Пример

```
{Процедура подготовки на форме таблицы для ввода квадратной матрицы. Процедура выполняется автоматически при выходе из поля ввода порядка матрицы.}
    procedure TForm1.Edit1Exit(Sender: TObject);
    Var i,j:Integer;// номера строки и столбца квадратной матрицы
    begin
    // порядок матрицы получает свое значение из поля ввода
    N:=StrToInt(Edit1.Text);
// свойство количество строк компонента StringGrid
// получает значение: порядок +1
    StringGrid1.RowCount:=N+1;
// свойство количество столбцов компонента StringGrid
// получает значение порядок: +1
    StringGrid1.ColCount:=N+1;
    //подпись заголовка таблицы
    For j:=1 to StringGrid1.RowCount do
    StringGrid1.Cells[j,0]:= IntToStr(j);
    //подпись строк таблицы
    For i:=1 to StringGrid1.ColCount do
     StringGrid1.Cells[0,i]:= IntToStr(i);
     // добавление в свойства таблицы признака
```

```
//допустимости редактирования содержимого ячеек таблицы
StringGrid1.Options:= StringGrid1.Options +[goEditing];
end;
```

Замечание

Для создания процедуры, связанной с каким-то событием, происходящим с компонентом, можно воспользоваться следующими способами.

1 способ

В инспекторе объектов выделить объект, например, Edit1. Раскрыть вкладку Events (события). Выделить строку с нужным событием, например, OnExit (выход из компонента) и два раза щелкнуть в свободном поле справа. Там появится событие Edit1Exit (Рис. 7−1).

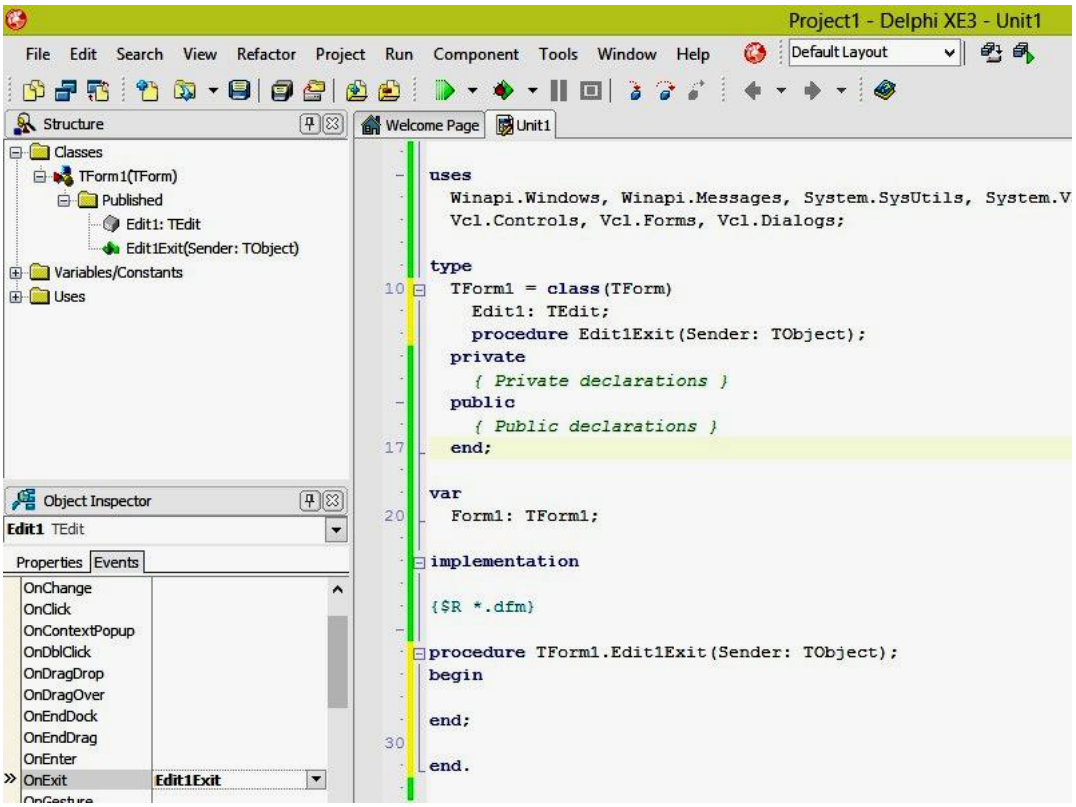


Рис. 7−1. Создание события Exit.

2 способ

Если два раза щелкнуть по компоненту Edit1 на форме, то в проекте появится заготовка процедуры, связанной с событием изменения содержимого(Change) поля ввода Edit1 (см. рис. 7−2).

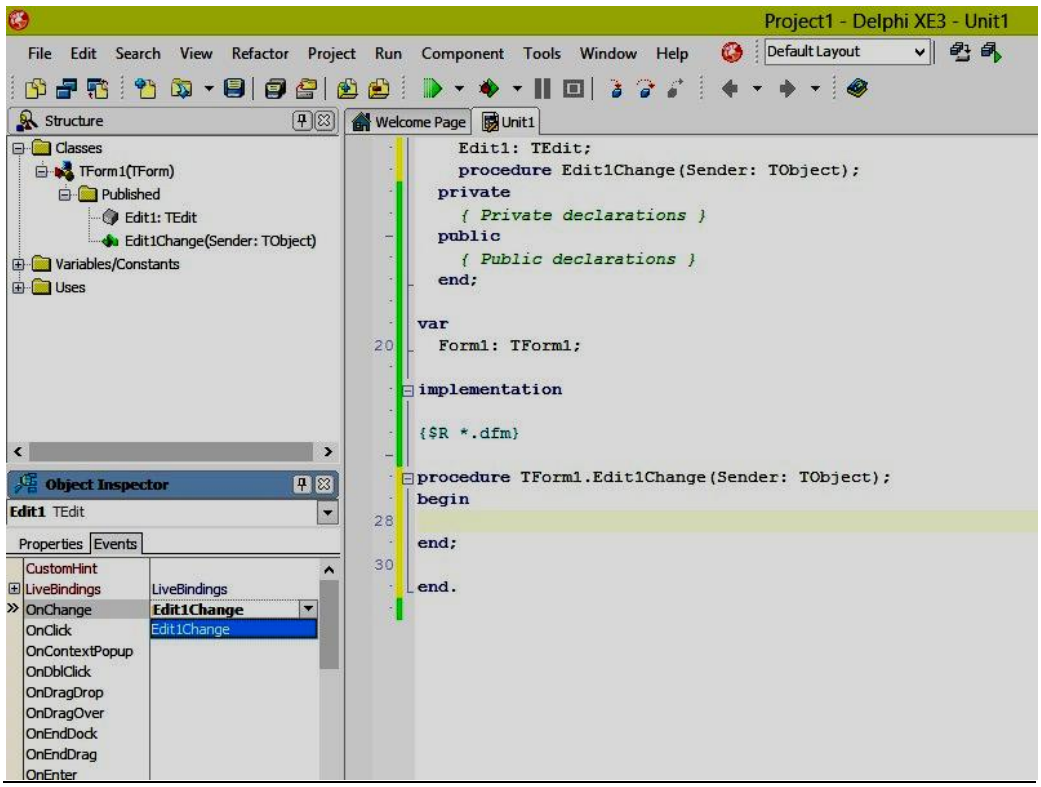


Рис. 7–2. Созданное по умолчанию событие Change

Допустим нам нужно создать событие OnEnter – нажатие клавиши Enter в поле ввода Edit1. В инспекторе объектов на вкладке Events (события) стираем Edit1Change напротив OnChange. Напротив OnEnter пишем Edit1Enter (см. рис. 7–3).

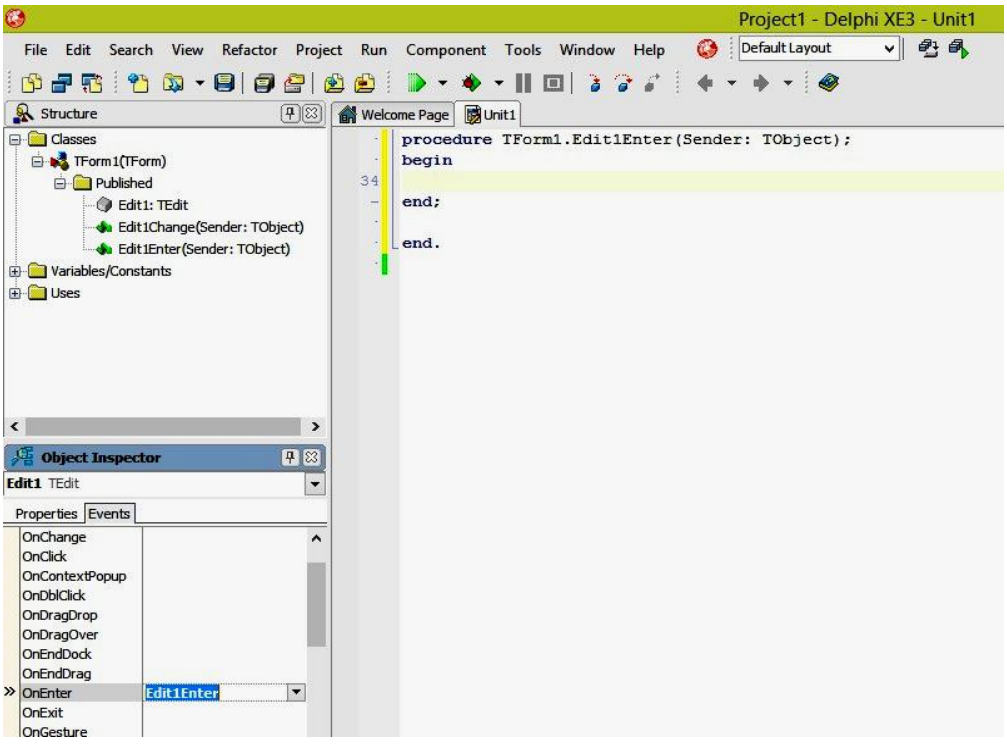


Рис. 7–3. Создание события Enter

Для чтения двухмерного массива действительных чисел из StrigGrid необходимо записать операторы, подобные этим:

```
For i:=1 to StringGrid1.RowCount-1 do
    For j:=1 to StringGrid1.ColCount-1 do
        Massiv[i,j]:=StrToFloat(StringGrid1.Cells[j,i]);
```

Для записи двухмерного массива действительных чисел в StrigGrid необходимо записать операторы, подобные этим:

```
For i:=1 to StringGrid1.ColCount-1 do
    For j:=1 to StringGrid1.RowCount-1 do
    StringGrid1.Cells[j,i]:=FloatToStr(Massiv[i,j]);
```

Понятие о классах в Delphi

В языке объектно-ориентированного программирования Delphi дается возможность определять так называемые *классы*. Класс – это сложная структура, включающая помимо описания данных, описание процедур и функций, которые могут быть выполнены над представителем класса – *объектом*.

Система Delphi всегда сама генерирует свои классы во время проектирования приложения. Описание сформированного класса находится в разделе **type** интерфейсной части модуля. Например, в программе про бак для летнего душа из главы 2 описание класса выглядит так:

```
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
{ Publicdeclarations }
end;
```

Как видим, в класс включены компоненты (данные) и методы их обработки (процедуры). В секции **private** (закрытые) объявляются элементы класса, которые должны быть недоступны в других модулях. В секции **public**(открытые) объявляются элементы, доступные в других модулях.

Технология создания многооконных проектов

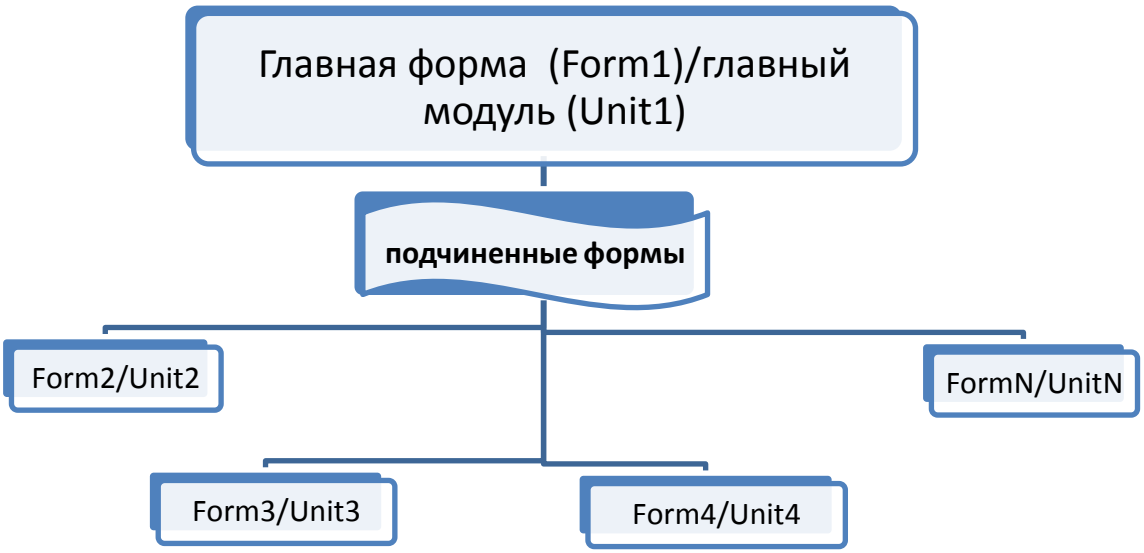
Решение многих задач требует создания многооконных проектов. Одной из причин, вызывающей необходимость создания многооконных проектов, может быть разделение ввода исходных данных в программу и вывода результатов счета.

После загрузки самой среды Delphi на экране уже появляется заготовка проекта, состоящего из одной формы и одного модуля. Для добавления новой формы в проект можно воспользоваться кнопкой на панели кнопок быстрого доступа или командой из главного меню

	Добавление новой формы в проект. Можно с помощью команды File→New→NewForm
	Вывод списка форм проекта
	Вывод списка модулей проекта

Вместе с формой в проект добавляется связанный с ней модуль. Например, для формы Form2 добавится модуль Unit2.

Структуру многооконного проекта можно представить в виде схемы



Основная программа обычно записывается в главном модуле, связанном с главной формой. К главному модулю нужно подключить модули дочерних форм, т. е. в раздел Uses дописать названия дочерних модулей. Например,

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics,

Controls, Forms, Dialogs, StdCtrls, Grids,***Unit2,Unit3***;

Для вывода на экран дочерних форм используется процедура **Show (показать)**. Например,

Form2.Show;

В раздел public дочерних модулей целесообразно включить описание переменных и массивов, передающихся из родительского модуля.

Если в родительском модуле используются компоненты из дочерних форм, перед их названием пишется имя формы, где они находятся. Например, Form2.StringGrid1.

Практика

Задача7а

Дана целочисленная матрица размером M строк на N столбцов. Подсчитать количество четных чисел в матрице.

Справка

Чётное число – целое число, которое делится на 2: …, −4, −2, 0, 2, 4, 6, 8, …

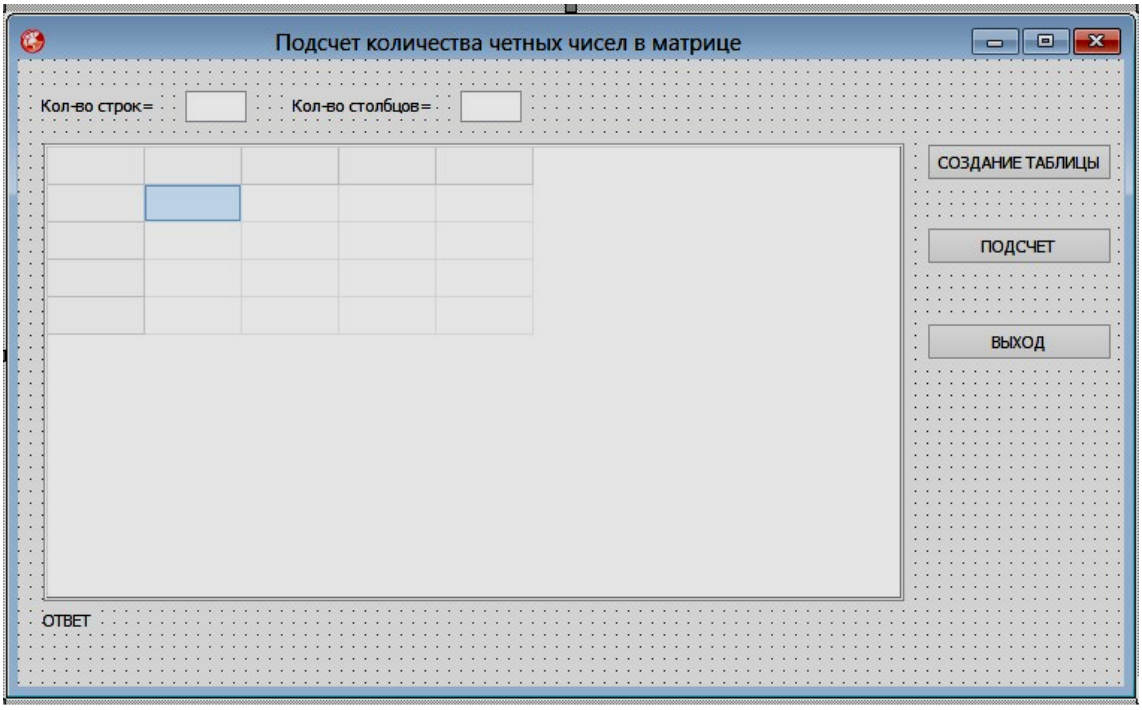
Нечётное число – целое число, которое *не делится* на 2: …, −3, −1, 1, 3, 5, 7, 9, …

Если *m* чётно, то оно представимо в виде $m = 2k$, а если нечётно, то в виде $m = 2k + 1$, где *к* целое.

Компоненты

Имя Компонента	Свойства компонента	Значение	Назначение
Form1	Caption	Подсчет количества четных чисел в матрице	Заголовок формы
Label1	Caption	Кол−во строк в матрице-	Подсказка пользователю
Edit1	Text		Поле для ввода кол-ва строк в матрице
Label1	Caption	Кол-во столбцов в матрице	Подсказка пользователю
Edit2	Tex		Поле для ввода кол-ва столбцов в матрице
StringGrid1	Options: goEditing goTabs goAlwaysShowEditor ScrollBars	 True True True ssBoth	Опции: Признак допустимости редактирования содержимого ячеек таблицы.True — редактирование разрешено. False — запрещено Разрешает использование клавиши Tab для перемещения в другую ячейку Признак нахождения компонента в режиме редактирования Выводятся обе полосы прокрутки
Label3	Caption	ответ	Поле для вывода результатов работы программы
Button1	Caption	ПОДСЧЕТ	Кнопка для запуска подсчета кол-ва четных чисел
Button2	Caption	ВЫХОД	Кнопка закрывает окно программы

Проект формы



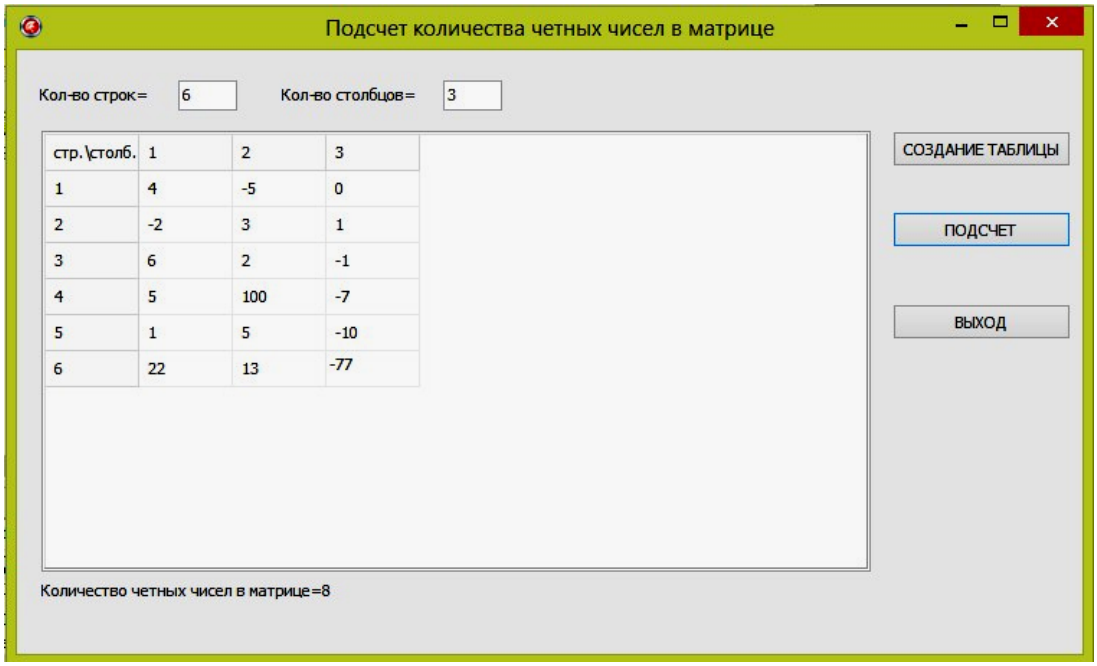
Текст модуля

```
unit Unit1;

interface

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

type
 TForm1 = class(TForm)
  Label1: TLabel;
  Edit1: TEdit;
  Label2: TLabel;
  Edit2: TEdit;
  StringGrid1: TStringGrid;

  Label3: 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;
  Matrix :array [1..100,1..200]of integer; //целочисленная не
//квадратная матрица
    M, N  : Integer; // M – кол-во строк, N – кол-во столбцов
    I, J  : Integer; // I – номер строки, J – номер столбца
implementation

{$R *.dfm}
{ Процедура подписывает названия столбцов и строк табли-
цы, предназначенной для ввода матрицы}
procedure TForm1.Button1Click(Sender: TObject);
begin
  StringGrid1.Cells[0,0]:='стр.\столб.';
  M:=StrToInt(Edit1.Text);
  N:=StrToInt(Edit2.Text);
  StringGrid1.RowCount:=M+1;
  StringGrid1.ColCount:=N+1;
  for I:=1 to M+1 do
  StringGrid1.Cells[0,I]:=IntToStr(I);
  for J := 1 to N+1 do
  StringGrid1.Cells[J,0]:=IntToStr(J)
end;
  {процедура читает матрицу из таблицы и подсчитывает кол-
во четных чисел}
procedure TForm1.Button2Click(Sender: TObject);
Var
  KOLICH : Integer; // кол-во четных чисел в матрице
begin
```

```
KOLICH:=0;
for I:=1 to M do
  for J:=1 to N do
  begin
  Matrix[I,J]:=StrToInt(StringGrid1.Cells[J,I]);
  if NOT(ODD(Matrix[I,J])) then KOLICH:=KOLICH+1;
  end;
Label3.Caption:='Количество четных чисел в матрице=' +
IntToStr(KOLICH)
end;
 {процедура закрывает окно программы}
procedure TForm1.Button3Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```

Тестовый пример



Задача7b

Составить программу для транспонирования квадратной матрицы порядка N и определения следа матрицы.

Транспонированной матрицей к исходной является такая матрица, у которой элементы строк исходной матрицы расставлены в столбцы, а элементы столбцов исходной матрицы в строки.

Следом матрицы называется сумма элементов главной диагонали.

Компоненты

Имя компонента	Свойства компонента	Значение	Назначение
Form1	Caption	Матрица	Заголовок главной формы
Label1	Caption	Программа предназначена для преобразования квадратной матрицы (количество строк равно количеству столбцов)	Справочная информация для пользователя программы
Label2	Caption	Размерность матрицы	Подсказка пользователю
Edit1	Text	…..	Поле для ввода размерности матрицы
Form1.StringGrid1	ColCount	5	Количество столбцов по умолчанию
	FixedCol	1	Количество фиксированных слева столбцов
	FixedRow	1	Количество фиксированных сверху строк
	Options:		Опции:
	goFixedVertLine	True	Разметка зафиксированных вертикальных линий
	goFixedHorzLine	True	Разметка зафиксированных горизонтальных линий
	goVertLine	True	Разметка вертикальных линий внутри таблицы
	goHorzLine	True	Разметка горизонтальных линий внутри таблицы
	goTabs	True	Разрешает использование клавиши Tab для перемещения в другую ячейку
	goAlwaysShowEditor	True	Признак нахождения компонента в режиме редактирования

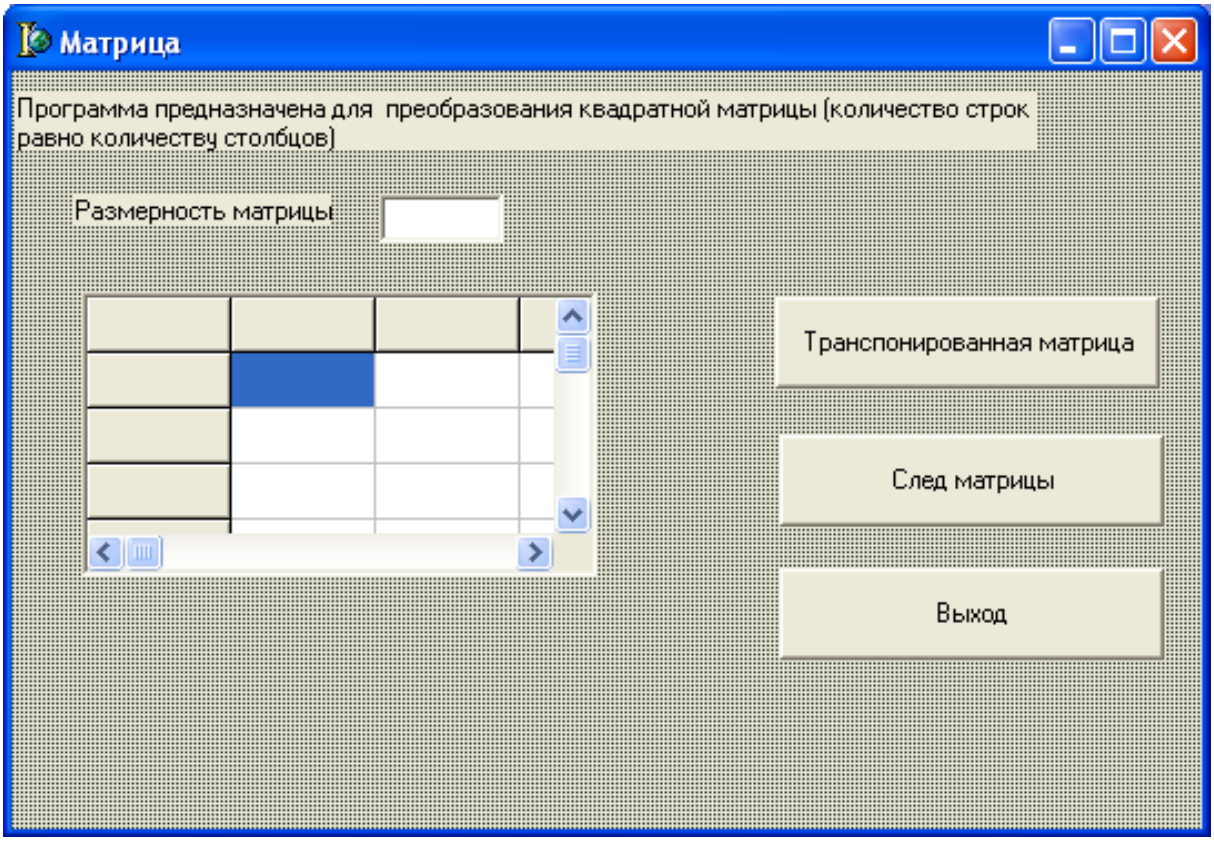
Form1.Button1	Caption	Транспонированная матрица	Кнопка для выполнения транспонирования и вывода в новое окно
Form1.Button2	Caption	След матрицы	Кнопка для вычисления следа матрицы и вывода в новое окно
Form1.Button3	Caption	Выход	Кнопка для завершения работы программы
Form2	Caption	Транспонированная матрица	Заголовок формы
Form2.StringGrid1	ColCount	5	Количество столбцов по умолчанию
	FixedCol	1	Количество фиксированных слева столбцов
	FixedRow	1	Количество фиксированных сверху строк
	Options: goFixedVertLine	 True	Опции: Разметка зафиксированных вертикальных линий
	goFixedHorzLine	True	Разметка зафиксированных горизонтальных линий
	goVertLine	True	Разметка вертикальных линий внутри таблицы
	goHorzLine	True	Разметка горизонтальных линий внутри таблицы
	goTabs	True	Разрешает использование клавиши Tab для перемещения в другую ячейку
	goAlwaysShowEditor	True	Признак нахождения компонента в режиме редактирования
Form2.Button1	Caption	Выход	Кнопка для закрытия окна “Транспонированная матрица”
Form3	Caption	След матрицы	Заголовок формы
Label1	Caption	Сумма главных диагональных элементов матрицы называется следом матрицы	Справочная информация для пользователя программы
Label2	Caption		Поле для вывода, рассчитанного программой значения следа матрицы
Form3.Button1	Caption	Выход	Кнопка для закрытия окна “След матрицы”

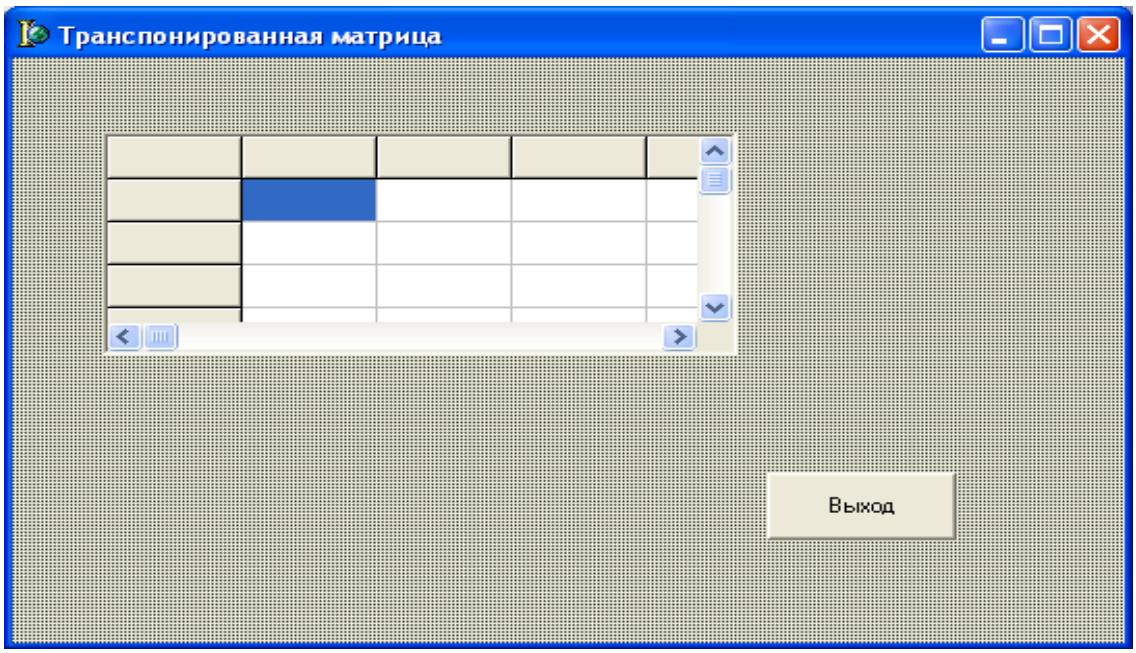
Переменные и массивы

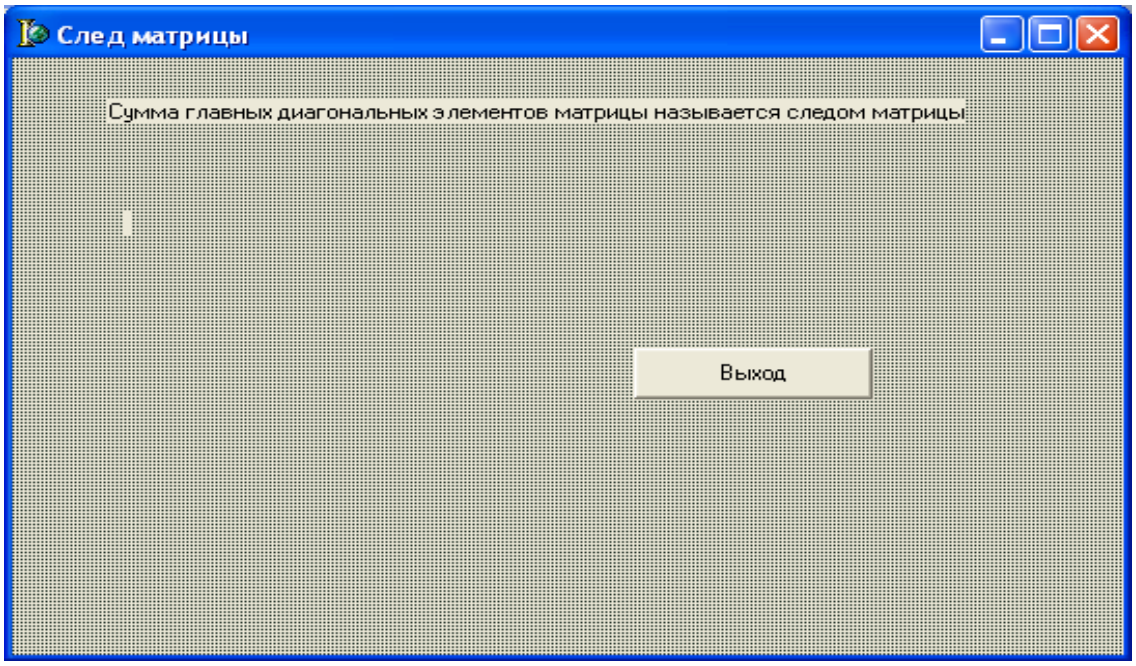
Обозначение в программе	Содержание	Тип
N	Порядок квадратной матрицы	Целый
M	Квадратная матрица действительных чисел	Вещественный, расширенный
I	Текущий номер строки матрицы или столбца в StringGrid	Целый
J	Текущий номер столбца матрицы или строки в StringGrid	Целый
Sled	След матрицы	Вещественный, расширенный

Проекты форм

Для решения данной задачи создадим три окна.







Текст модуля

unit Unit1;
interface
uses
*Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Grids,***Unit2,Unit3**;
// подключаются модули, связанные со второй и третьей формами
type

```
 TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    StringGrid1: TStringGrid;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Edit1Exit(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  A :array [1..100,1..100] of Extended; //квадратная матрица
  N:Integer;// порядок матрицы
  Sled: Extended; //след матрицы
  end;
  var
    Form1: TForm1;
  implementation
  {$R *.dfm}
{ Процедура подготовки на первой(главной) форме таблицы для
ввода квадратной матрицы. Процедура выполняется при выходе
из поля ввода размерности матрицы.}
  procedure TForm1.Edit1Exit(Sender: TObject);
  Var i,j:Integer;//номера строки и столбца квадратной матрицы
  begin
  // порядок матрицы получает свое значение из поля ввода
  N:=StrToInt(Edit1.Text);
// свойство количество строк компонента StringGrid
// получает значение: порядок +1
  StringGrid1.RowCount:=N+1;
// свойство количество столбцов компонента StringGrid
// получает значение порядок: +1
  StringGrid1.ColCount:=N+1;
```

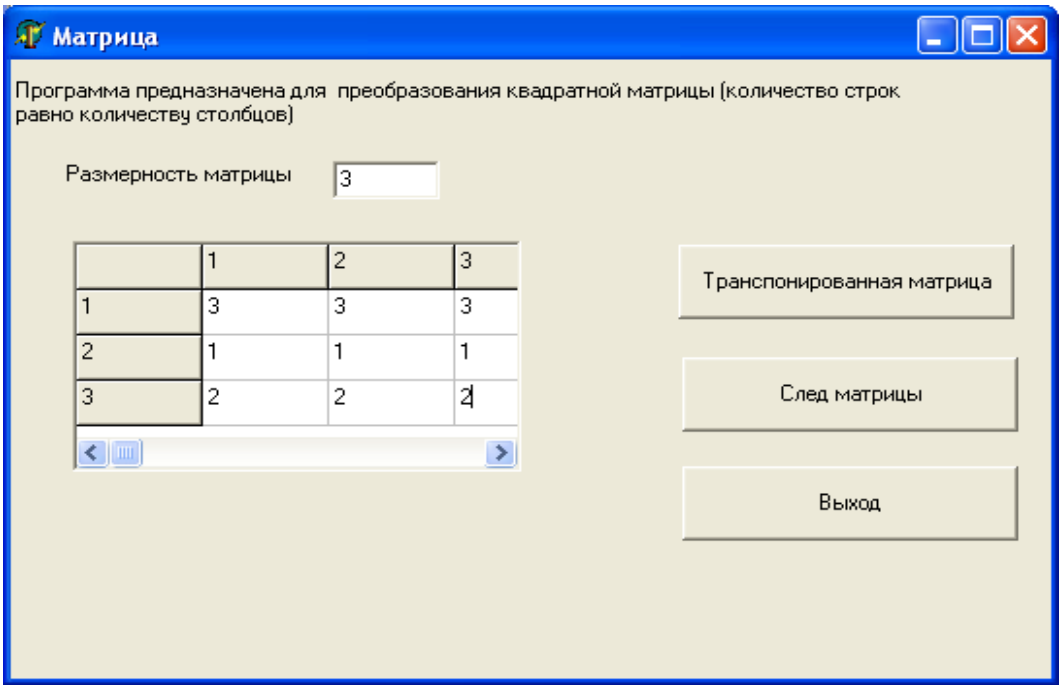
```
    //подпись столбцов таблицы
    For j:=1 to StringGrid1.ColCount do
    StringGrid1.Cells[j,0]:= IntToStr(j);
    //подпись строк таблицы
    For i:=1 to StringGrid1.RowCount do
    StringGrid1.Cells[0,i]:= IntToStr(i);
    // добавление в свойства таблицы признака
    // допустимости редактирования содержимого ячеек таблицы
    StringGrid1.Options:= StringGrid1.Options +[goEditing];
    end;
    procedure TForm1.Button1Click(Sender: TObject);
    begin
    Form1.Close
    end;

// Процедура подготовки на второй форме таблицы для вывода
//транспонированной матрицы, транспонирование и вывод матрицы
    procedure TForm1.Button2Click(Sender: TObject);
    Var i:Integer; //номер строки матрицы/номер столбца StringGrid
        j:Integer; // номер столбца матрицы/номер строки StringGrid
// количество строк и столбцов компонента StringGrid на второй форме
    // присваивается значение порядок матрицы +1
    Form2.StringGrid1.RowCount:=N+1;
    Form2.StringGrid1.ColCount:=N+1;
    // подписываются заголовок и номера строк таблицы для
    // транспонированной матрицы
    For i:=1 to Form2.StringGrid1.RowCount do
      Form2.StringGrid1.Cells[i,0]:= IntToStr(i);
    For j:=1 to Form2.StringGrid1.ColCount do
      Form2.StringGrid1.Cells[0,j]:= IntToStr(j);
    { вывод транспонированной матрицы в компонент StringGrid
на второй форме }
    { Транспонированная матрица не создается. Транспонирование
осуществляется путем вывода столбца исходной матрицы в строку
компонента StringGrid. В StringGrid номер столбца i, номер строки j.}
    For i:=1 to Form2.StringGrid1.ColCount-1 do
        For j:=1 to Form2.StringGrid1.RowCount-1 do
            Form2.StringGrid1.Cells[i,j]:=Form1.StringGrid1.Cells[j,i];
```

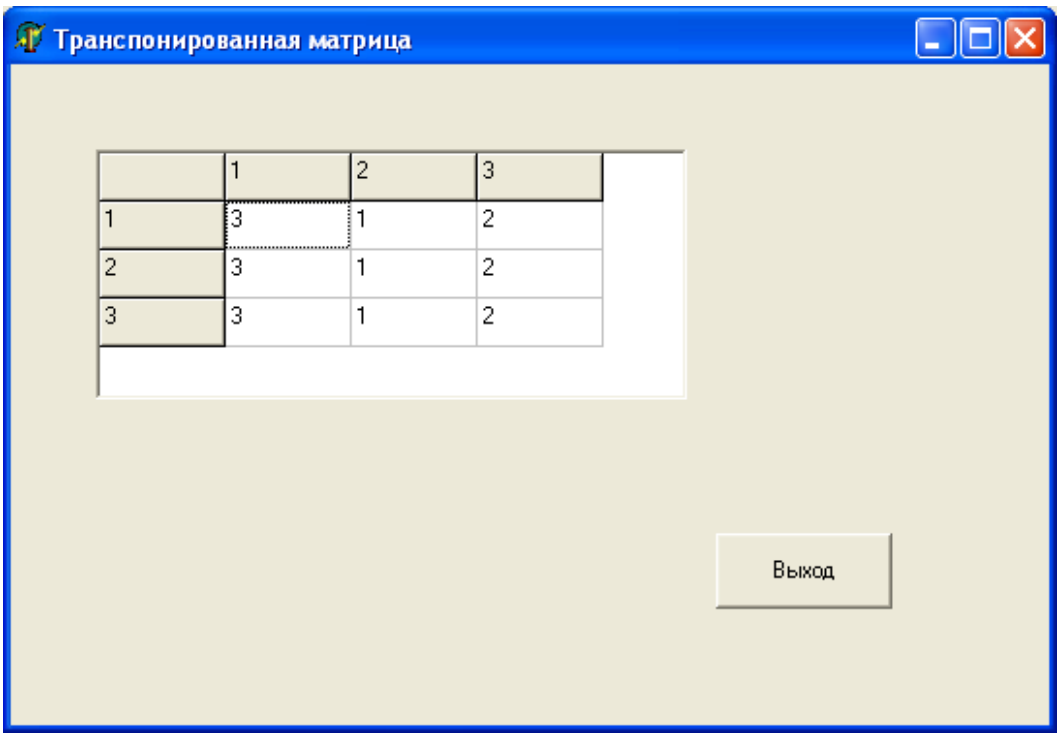
```
    end;
    // открывается вторая форма
    Form2.Show;

// Процедура чтения из таблицы на главной форме квадратной
//матрицы, создания соответствующего двухиндексного массива
//действительных чисел, вычисления следа матрицы и вывода
//результата на третью форму
    procedure TForm1.Button3Click(Sender: TObject);
    Var i: Integer;
    // чтение значений элементов квадратной матрицы и создание
    // массива действительных чисел
    begin
    For i:=1 to StringGrid1.RowCount-1 do
        For j:=1 to StringGrid1.ColCount-1 do
            A[i,j]:=StrToFloat(StringGrid1.Cells[j,i]);
    //Вычисление следа матрицы
    Sled:=0;
    For i:=1 to Form1.StringGrid1.ColCount-1 do
    Sled:= Sled + A[i,i];
    Form3.Label2.Caption:=' След матрицы равен ' +
FloatToStr(Sled)
    end;
    Form3.Show;

    end.

{ Модуль, связанный со второй формой}
    unit Unit2;
    interface
    uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, Grids;
    type
    TForm2 = class(TForm)
      StringGrid1: TStringGrid;
      Button1: TButton;
```

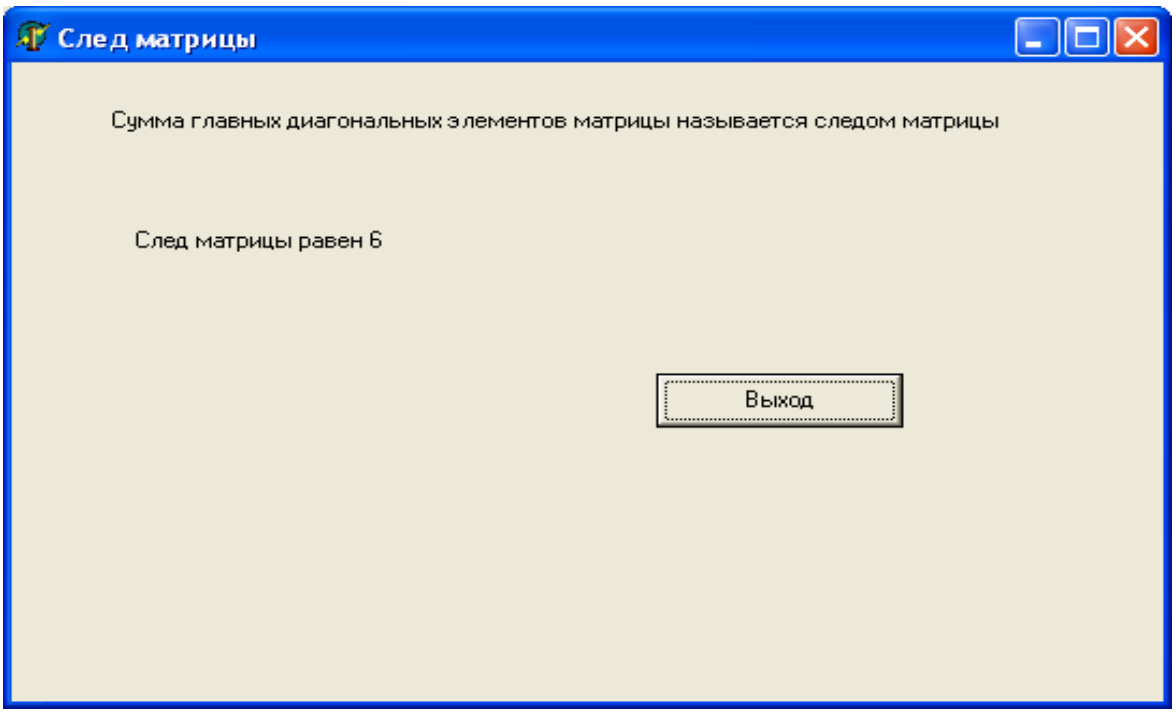
```
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

  end;
  var
    Form2: TForm2;
  implementation
  {$R *.dfm}
  procedure TForm2.Button1Click(Sender: TObject);
  begin
  Form2.Close
  end;
  end.
{ Модуль, связанный с третьей формой.}
unit Unit3;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
Dialogs, StdCtrls;
type
TForm3 = class(TForm)
  Label1: TLabel;
  Label2: TLabel;
  Button1: TButton;
  procedure Button1Click(Sender: TObject);
private
   { Private declarations }
public
   { Public declarations }

end; var
  Form3: TForm3;
implementation
{$R *.dfm}
```

```
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Close
end;
end.
```

Тестовый пример







Задачи для самостоятельного решения

Группа А

№ 7.1 Дана действительная матрица размера N на M элементов:

$$\begin{vmatrix} a_{11}a_{12}a_{13} \ldots a_{1m} \\ a_{21}a_{22}a_{23} \ldots a_{2m} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}a_{n2}a_{n3} \ldots a_{nm} \end{vmatrix}$$

Найти сумму положительных элементов матрицы.

№ 7.2 Дана целочисленная квадратная матрица порядка N. Найти номера строк: а) все элементы которых – нули; б) все элементы которых четны.

№ 7.3 Дана действительная матрица размера N на M элементов, элементы которой различны. Найти наибольший и наименьший элементы матрицы.

№ 7.4 Дана целочисленная квадратная матрица порядка N. Подсчитать: а) количество нулей, стоящих на главной диагонали; б) количество единиц в матрице.

№ 7.5 Дана действительная матрица размера N на M элементов, все элементы которой различны. Указать номер строки и столбца, на пересечении которых находится минимальный элемент матрицы. Подсчитать количество отрицательных элементов в матрице.

№ 7.6 Дана целочисленная квадратная матрица порядка N. Найти номера строк: а) все элементы которых положительны; б) все элементы которых нечетны.

№ 7.7 Дана целочисленная квадратная матрица порядка N. Найти номера столбцов: а) все элементы которых отрицательны; б) все элементы которых равны нулю.

№ 7.8 Дана действительная матрица размера N на M элементов:

$$\begin{vmatrix} a_{11}a_{12}\ a_{13} \ldots a_{1m} \\ a_{21}\ \ a_{22}\ a_{23} \ldots a_{2m} \\ \ldots\ldots\ldots\ldots\ldots\ \ldots \\ a_{n1}a_{n2}a_{n3} \ldots a_{nm} \end{vmatrix}$$

Найти номера строк матрицы, где все элементы различны.

№ 7.9 Дана целочисленная квадратная матрица порядка N. Подсчитать: а) произведение элементов главной диагонали; б) количество нулей в матрице.

№ 7.10 Дана действительная матрица размера N на M элементов, все элементы которой различны. Вывести элементы матрицы, которые меньше среднего арифметического из ее элементов.

Группа Б

№ 7.11 Дана матрица **A** размера N на M элементов, содержащая вещественные числа. Сформировать из нее массив B, состоящий из сумм положительных элементов каждой строки; если таких в строке нет, в массив B заносится 0.

№ 7.12 Дана матрица **A** размера N на M элементов, содержащая вещественные числа,. Сформировать из нее массив B, состоящий из произведений положительных элементов каждой строки; если таких в строке нет, в массив B заносится 0.

№ 7.13 Дана матрица **A** размера N на M элементов, содержащая вещественные числа. Сформировать из нее массив B, состоящий из сумм отрицательных элементов каждой строки; если таких в строке нет, результат должен быть равен 0.

№ 7.14 Дана матрица **A** размера N на M элементов, содержащая вещественные числа. Сформировать из нее массив B, состоящий из количества отрицательных элементов в каждой строке. Если в строке нет отрицательных элементов, то в массив B за-

носится -1. Сколько строк в исходной матрице, где нет отрицательных элементов.

№ 7.15 Дана действительная матрица размера N на M элементов:

$$\begin{matrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{matrix}$$

Получить последовательность b1,..., bn, где bi – произведение квадратов тех элементов i-й строки, модули которых принадлежат отрезку [1,1.5].

№ 7.16 Дана целочисленная квадратная матрица порядка n. Получить последовательность b1,...,bn, где bi – сумма элементов, расположенных за первым отрицательным элементом в i-й строке (если все элементы строки неотрицательны, то принять bi= –1).

№ 7.17 Дана действительная матрица размера N на M, в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы на ее наибольший по модулю элемент.

№ 7.18 Даны натуральное число N, действительное число x, действительная матрица размера N на 2N. Получить последовательность b1,...,bn из нулей и единиц, где bi=1, если элементы i-й строки матрицы не превосходят x, и bi=0 в противном случае.

№ 7.19 Дана целочисленная квадратная матрица A порядка M. Строку с номером i матрицы назовем отмеченной, если все ее элементы положительны и неотмеченной в противном случае.

а) Нужно все элементы, расположенные в неотмеченных строках матрицы, преобразовать по правилу: отрицательные элементы заменить на –1, положительные – на 1, а нулевые оставить без изменения.

б) Подсчитать число отрицательных элементов матрицы, расположенных в неотмеченных строках.

№ 7.20 Дана действительная квадратная матрица порядка N. Построить массив x по правилу: x_1 содержит наибольшее число первой строки, x_2 наибольшее число второй строки, x_n наибольшее число n−ной строки. Подсчитать значение суммы произведений $x_1x_n + x_2x_{n-1} + \dots + x_nx_1$.

Глава 8. Процедуры и функции, разрабатываемые программистом

В данной главе рассматриваются следующие вопросы: понятия подпрограммы, процедуры, функции и технология их разработки.

Теория

Подпрограмма представляет собой группу операторов, логически законченную и специальным образом оформленную. Подпрограмму можно вызывать неограниченное число раз из различных частей программы. Использование подпрограмм позволяет улучшить структурированность программы и сократить ее размер.

По структуре подпрограмма почти полностью аналогична программе и содержит заголовок и раздел операторов (блок), однако в блоке подпрограммы отсутствует раздел подключения модулей. Кроме того, заголовок подпрограммы по своему оформлению отличается от заголовка программы.

Работа с подпрограммой имеет два этапа:

– описание подпрограммы;

– вызов подпрограммы.

Любая подпрограмма должна быть предварительно описана, после чего допускается ее вызов. При описании подпрограммы указывается ее имя, список параметров (формальных параметров) и выполняемые подпрограммой действия. При вызове подпрограммы указываются имя подпрограммы и список аргументов (фактических параметров), передаваемых подпрограмме для работы.

В различных модулях Delphi имеется большое число стандартных подпрограмм, которые можно вызывать без предварительного описания. Кроме того, программист может создавать свои подпрограммы, которые также называют *пользовательскими.* Подпрограммы делятся на ***процедуры*** и ***функции,*** которые имеют между собой много общего. Основное различие между ними заключается в том, что функция может возвращать под своим именем в качестве результата значение и соответственно может использоваться в качестве операнда выражения.

С подпрограммой взаимодействие осуществляется по управлению и по данным. Взаимодействие *по управлению* заключается в передаче управления из программы в подпрограмму и организации возврата в программу. Взаимодействие *по данным* заключается в передаче подпрограмме данных, над которыми она выполняет определенные действия. Этот вид взаимодействия может осуществляться следующими основными способами:

– с использованием файлов;

– с помощью глобальных переменных;

– с помощью параметров.

Наиболее часто используется последний способ. При этом различают параметры и аргументы. *Параметры* (формальные параметры) являются элементами подпрограммы и используются при описании алгоритма, выполняемого подпрограммой. *Аргументы* (фактические параметры) являются элементами вызывающей программы. Они замещают параметры при вызове подпрограммы. При этом осуществляется проверка на соответствие типов и количества параметров и аргументов. Имена параметров и аргументов могут различаться, однако их количество и порядок следования должны совпадать, а типы параметров и соответствующих им аргументов должны быть совместимыми. Для прекращения работы подпрограммы можно использовать процедуру **Exit**, которая прерывает выполнение операторов подпрограммы и возвращает управление вызывающей программе. Подпрограммы можно вызывать не только из программы, но и из других подпрограмм.

Процедуры

Описание *процедуры* включает в себя заголовок и блок, который за исключением раздела подключения модулей не отличается от блока программы.

Заголовок состоит из ключевого слова procedure, имени процедуры и необязательного списка параметров в круглых скобках с указанием типа каждого параметра. Заголовок имеет формат:

Procedure <Имя> [(формальные параметры)];

Для обращения к процедуре используется *оператор вызова* процедуры. Он включает имя процедуры и список фактических параметров, заключенный в круглые скобки. Например, Test(a,b,c), –

имя вызываемой процедуры, a,b,c – фактические параметры. Процедура Test должна быть описана с помощью подобного заголовка:

Procedure Test(a:integer;b:Real; var c:Real);

где a, b – исходные данные передаваемые процедуре из вызывающей процедуры;

c – результат работы процедуры, возвращаемый в вызывающую процедуру.

Функции

Описание *функции* состоит из заголовка и блока. *Заголовок* включает ключевое слово Function, имя функции, необязательный список формальных параметров, заключенный в круглые скобки, и тип возвращаемого функцией значения. Заголовок имеет формат:

Function <Имя> [(Формальные параметр)] : <Тип результата>

Возвращаемое значение может иметь любой тип, кроме файлового. *Блок* функции представляет собой локальный блок, по структуре аналогичный блоку процедуры. В теле функции должен быть хотя бы один оператор присваивания, в левой части которого стоит имя функции. Именно он и определяет значение, возвращаемое функцией. Если таких операторов несколько, то результатом функции будет значение последнего выполненного оператора присваивания. В этих операторах вместо имени функции можно указывать переменную Result, которая создается в качестве синонима для имени функции. В отличие от имени функции, переменную Result можно использовать в выражениях блока функции. С помощью переменной Result можно в любой момент получить внутри блока доступ к текущему значению функции.

Замечание

Имя функции можно использовать в выражениях блока функции, однако это приводит к рекурсивному вызову функцией самой себя.

Вызов функции осуществляется по ее имени с указанием в круглых скобках списка аргументов, которого может и не быть. При этом аргументы должны попарно соответствовать параметрам, указанным в заголовке функции, и иметь те же типы. В отличие от процедуры, имя функции может входить как операнд в выражения. Например,

T:=Factor(n)/(n-k); где Factor(n) вызов функции для вычисления факториала. Функция Factor(n) должна быть описана примерно таким заголовком:

Function Factor(i:Integer):Int64;

Практика

Задача 8а

Известны стороны косоугольного треугольника: a, b, c. Составить программу для вычисления углов треугольника. Углы определять в градусах, минутах и секундах.

Для вычисления угла косоугольного треугольника нужно воспользоваться формулой

$$tg\frac{\gamma}{2} = \sqrt{\frac{(p-a)(p-b)}{p(p-c)}},$$

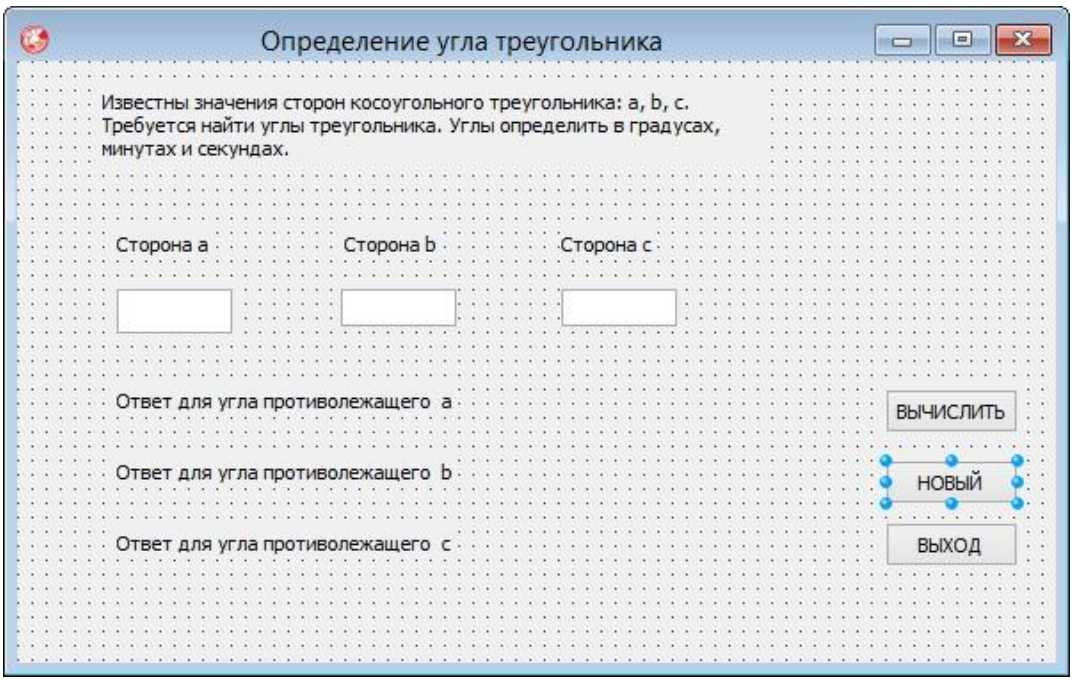
где γ − угол противолежащий стороне c,

p − полупериметр треугольника:

$$p = \frac{a+b+c}{2}$$

Для вычисления тангенса половины угла разработать функцию. Для перевода угла в градусы, минуты и секунды предусмотреть процедуру.

Проект формы



Компоненты

Имя компонента	Свойства компонентов	Значение	Назначение
Form1	Caption	Определение угла треугольника	Заголовок формы
Label1	Caption WordWrap	Известны значения сторон косоугольного треугольника: a, b, c. Требуется найти углы треугольника. Углы определить в градусах, минутах и секундах. TRUE	Справочная информация для пользователя программы Перенос не уместившихся слов на новую строку
Label2	Caption	Сторона a	Подсказка пользователю
Label3	Caption	Сторона b	Подсказка пользователю
Label4	Caption	Сторона c	Подсказка пользователю
Label5	Caption	Ответ для угла противолежащего a	Место для ответа
Label6	Caption	Ответ для угла противолежащего b	Место для ответа
Label7	Caption	Ответ для угла противолежащего c	Место для ответа
Edit1	Text	Должно быть очищено от значения, заданного по умолчанию	Поле для ввода стороны a
Edit2	Text	Должно быть очищено от значения, заданного по умолчанию	Поле для ввода стороны b
Edit3	Text	Должно быть очищено от значения, заданного по умолчанию	Поле для ввода стороны c
Button1	Caption	ВЫЧИСЛИТЬ	Кнопка для вычислений
Button2	Caption	НОВЫЙ	Кнопка очищающая поля ввода для новых данных
Button3	Caption	ВЫХОД	Кнопка для закрытия формы

Краткое описание программы

В модуле Unit1 разработана функция UG_Grad, которая получает из вызывающей программы стороны треугольника через формальные параметры d, e, f вещественного типа. Причем f − сторона лежащая напротив определяемого угла. Функция UG_Grad передает вещественный результат через свое имя. Локальными переменными функции являют p − полупериметр, Tan_Pol_Ugla − тангенс половины определяемого угла, Ugol_Rad − определяемый угол в радианах. В теле функции: 1) вычисляется полупериметр; 2) вы-

числяется тангенс половины определяемого угла; 3) вычисляется угол в радианах; 4) угол переводится в градусы и присваивается имени функции.

Процедура GMS получает из вызывающей программы угол в радианах через формальный параметр Ugol вещественного типа и возвращает градусы G, минуты M, секунды S. G, M − переменные целого типа, S − вещественного типа. Градусы вычисляются выделением целой части из угла в градусах. Минуты вычисляются выделением целой части из выражения (Ugol-G)*60. Секунды определяются по формуле ((Ugol-G)*60-M)*60.

В процедуре, связанной с событием клик по кнопке ВЫЧИСЛИТЬ: 1) вводятся стороны треугольника: a, b, c (переменные вещественного типа); 2) проверяется условие существования треугольника (a+b>c) и (a+c>b) и (b+c>a); 3) если треугольник существует, для каждого угла с помощью вызова функции UG_Grad вычисляются углы в градусах: Ugol_a_Grad, Ugol_b_Grad, Ugol_c_Grad (вещественный тип), процедура GMS выделяет из углов целые градусы, минуты и секунды; 4) если условие существования треугольника не выполняется, то выдается сообщение об этом.

Процедура, связанная с кнопкой НОВЫЙ, очищает поля ввода.

Процедура, связанная с кнопкой ВЫХОД, закрывает форму.

Текст модуля

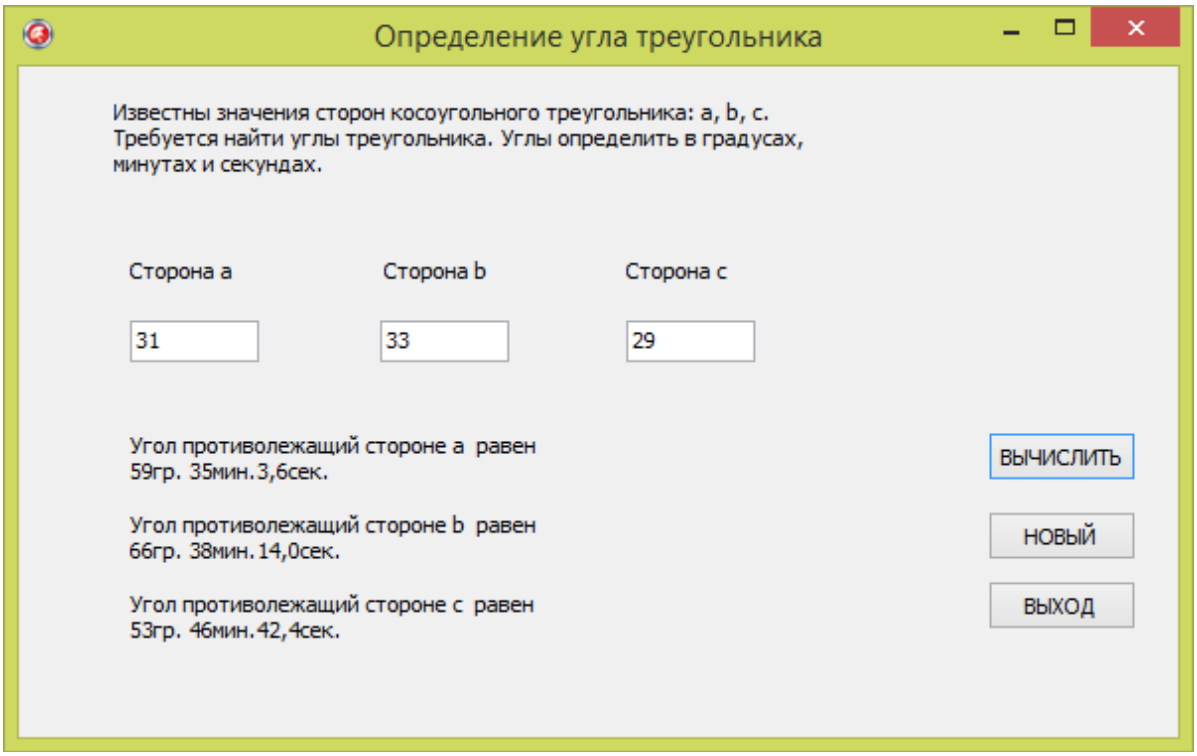
```
unit Unit1;

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;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
```

```
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  Label5: TLabel;
  Label6: TLabel;
  Label7: TLabel;
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  procedure Button3Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);

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

var
 Form1: TForm1;

implementation

{$R *.dfm}
{функция вычисляет угол в градусах}
function UG_Grad(d,e,f:Real):Real;
// f – сторона, лежащая против определяемого угла
Var p:Real;
 Tan_Pol_Ugla, Ugol_Rad:Real;
begin
  p:=(d+e+f)/2;
   Tan_Pol_Ugla:=Sqrt((p-d)*(p-e)/(p*(p-f)));
   Ugol_Rad:=Arctan( Tan_Pol_Ugla)*2;
   UG_Grad:= Ugol_Rad/pi*180;
end;
 {процедура выделяет из угла в градусах целые градусы, целые минуты и секунды}
```

```
procedure GMS(Ugol:Real;var G,M:Integer;var S:Real);
begin
 G:=Trunc(Ugol);
 M:=Trunc((Ugol-G)*60);
 S:=((Ugol-G)*60-M)*60;
end;


procedure TForm1.Button1Click(Sender: TObject);
 Var
 a,b,c: Real;
 G,M: Integer;
 S:Real;
 Ugol_a_Grad,Ugol_b_Grad,Ugol_c_Grad : Real;

begin
 a:= StrToFloat(Edit1.Text);
  b:= StrToFloat(Edit2.Text);
  c:= StrToFloat(Edit3.Text);
  If(a+b>c) and (a+c>b)and (b+c>a) Then
  Begin
   Ugol_a_Grad:= UG_Grad(b,c,a);
   GMS( Ugol_a_Grad,G,M,S);
   Label5.Caption:='Угол противолежащий стороне a равен '
+#13
   +IntToStr(G)+'гр. '
   +IntToStr(M)+'мин.'+ FloatToStrF(S,FFfixed,5,1)+'сек.' ;
   Ugol_b_Grad:= UG_Grad(a,c,b);
   GMS( Ugol_b_Grad,G,M,S);
   Label6.Caption:='Угол противолежащий стороне b равен '
+#13
   +IntToStr(G)+'гр. '
   +IntToStr(M)+'мин.'+ FloatToStrF(S,FFfixed,5,1)+'сек.' ;
   Ugol_c_Grad:= UG_Grad(a,b,c);
   GMS( Ugol_c_Grad,G,M,S);
   Label7.Caption:='Угол противолежащий стороне c равен '
+#13
   +IntToStr(G)+'гр. '
```

```
  +IntToStr(M)+'мин.'+ FloatToStrF(S,FFfixed,5,1)+'сек.'
 end
 Else  Label5.Caption:='ТРЕУГОЛЬНИК НЕ СУЩЕСТВУЕТ';

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Edit1.Text:=' ';
Edit2.Text:=' ';
Edit3.Text:=' ';
end;

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

end.
```

Возможный результат решения:



Задача 8b

Составить программу, вычисляющую факториал числа n. Факториал числа n − это произведение целых чисел от 1 до n, т.е.

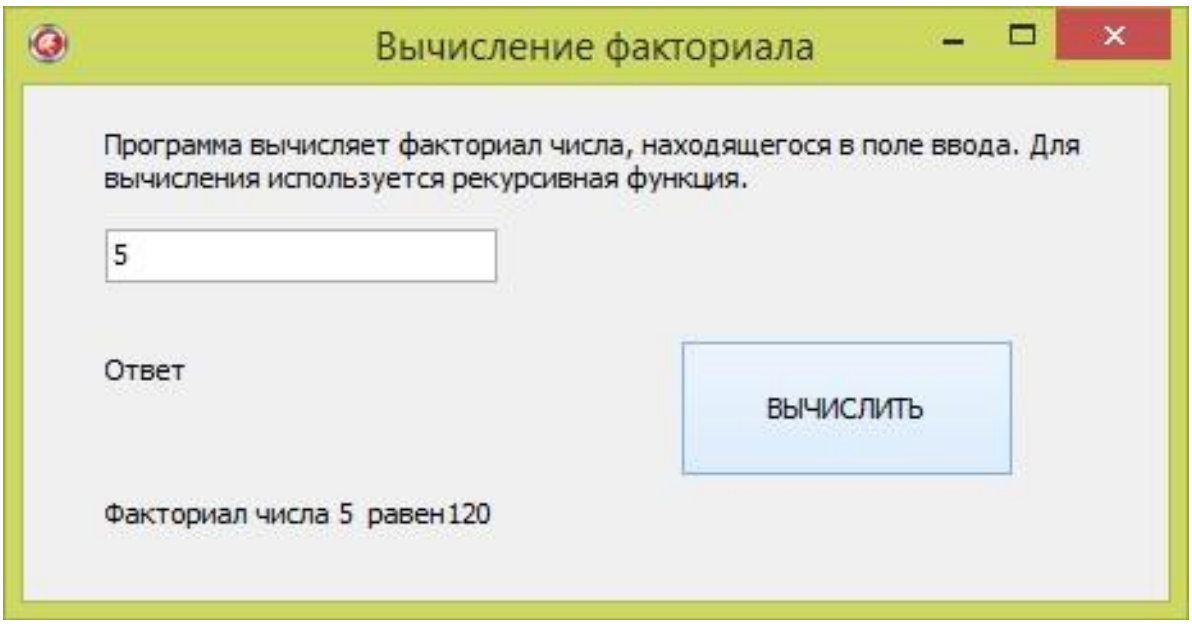
n!=1*2*3*...*(n-1)*n.

Факториал числа n равен произведению числа n на факториал числа (n-1). А факториал числа (n-1) равен произведению (n-1) на факториал (n-2) и т.д.

Таким образом, вычисление факториала числа n можно реализовать, как функцию, внутри которой будет оператор вызова функции вычисления факториала (n-1), т.е. самой себя. Такой способ вызова называется рекурсией, а функция, обращающаяся сама к себе называется рекурсивной функцией.


```
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;
```

```
    Edit1: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
  // рекурсивная функция, вычисляющая факториал числа
  function factorial(n:integer):integer;
  begin
  { функция вызывает сама себя только в том случае, если
  значение полученного параметра n не равно единице}
    if n>1 then factorial:=n*factorial(n-1)
      else factorial:=1;// факториал 1 равен 1
  end;
procedure TForm1.Button1Click(Sender: TObject);
Var
k:integer;//число, факториал которого надо вычислить
f: integer;//значение факториала числа k
begin
k:=StrToInt(Edit1.Text);
f:=factorial(k);
label3.Caption:='Факториал числа '+ Edit1.Text+' равен'+
IntToStr(f);
end;
end.
```



Задача 8с

Составить программу для вычисления значений выражений P=(x1+y1)/2 и Q=(x2+y2)/2, где x1 и x2– корни квадратного уравнения с коэффициентами A, B, C, а y1 и y2 – корни квадратного уравнения с коэффициентами D, F, G. Если хотя бы одно из уравнений не имеет действительных корней выдать на экран сообщение об этом и прекратить выполнение программы.

В программе разработать функцию для проверки введенных данных в поле ввода. Вводить в поле ввода можно только числа. Десятичным разделителем является запятая. Для вычисления корней квадратного уравнения разработать процедуру.

Компоненты

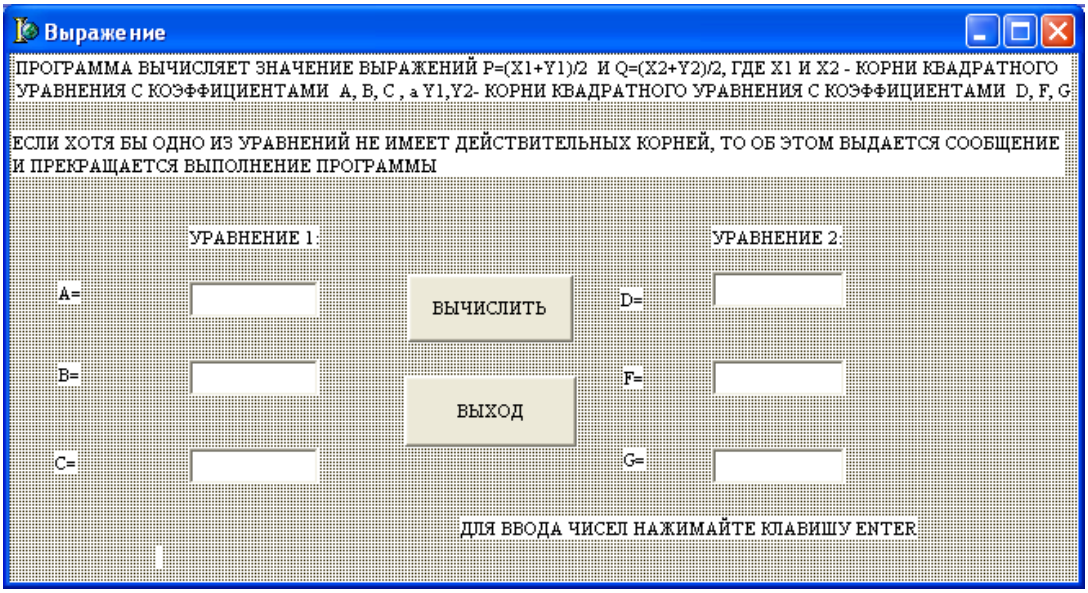
Имя компонента	Свойства компонентов	Значение	Назначение
Form1	Caption	Выражение	Заголовок формы
Label1	Caption	Программа вычисляет значение выражений: P=(X1+Y1)/2 и Q=(X2+Y2)/2, где X1 и X2 – корни квадратного уравнения с коэффициентами A, B, C , а Y1,Y2 – корни квадратного уравнения с коэффициентами D, F, G	Справочная информация для пользователя программы
	WordWrap	True	Перенос не уместившихся слов на новую строку

Label2	Caption	Если хотя бы одно из уравнений не имеет действительных корней, то об этом выдается сообщение и прекращается выполнение программы	Подсказка пользователю
	WordWrap	True	Перенос не уместившихся слов на новую строку
Label3	Caption	Уравнение 1	Подсказка пользователю
Label4	Caption	Уравнение 2	Подсказка пользователю
Label5	Caption	A=	Подсказка пользователю
Label6	Caption	B=	Подсказка пользователю
Label7	Caption	C=	Подсказка пользователю
Label8	Caption	D=	Подсказка пользователю
Label9	Caption	F=	Подсказка пользователю
Label10	Caption	G=	Подсказка пользователю
Label11	Caption		Поле для вывода результата
Label12	Caption	Для ввода чисел нажимайте клавишу ENTER	Подсказка пользователю
Edit1	Text	…..	Поле для ввода коэффициента A
Edit2	Text	…..	Поле для ввода коэффициента B
Edit3	Text	…..	Поле для ввода коэффициента C
Edit4	Text	…..	Поле для ввода коэффициента D
Edit5	Text	…..	Поле для ввода коэффициента F
Edit6	Text	…..	Поле для ввода коэффициента G
Button1	Caption	ВЫЧИСЛИТЬ	Кнопка для вычисления
Button2	Caption	ВЫХОД	Кнопка закрытия формы и выхода из программы

Переменные

Обозначение в программе	Содержание	Тип
A	коэффициент первого квадратного уравнения	Вещественный
B	коэффициент первого квадратного уравнения	Вещественный
C	коэффициент первого квадратного уравнения	Вещественный
D	коэффициент второго квадратного уравнения	Вещественный
F	коэффициент второго квадратного уравнения	Вещественный
G	коэффициент второго квадратного уравнения	Вещественный
S	строка символов, передаваемая из поля ввода	Строковый
i	номер по порядку символа в строке символов	Локальная переменная. Целый
Z	счетчик запятых	Локальная переменная. Целый
DT	дискриминант	Локальная переменная. Вещественный
R	Признак решения уравнения	Локальная переменная. Логический
R1	Признак решения уравнения 1	Логический
R2	Признак решения уравнения 2	Логический
Key	Ключ – какая клавиша нажата	Слово, целый беззнаковый
X1	1 корень уравнения 1	Вещественный
X2	2 корень уравнения 1	Вещественный
Y1	1 корень уравнения 2	Вещественный
Y2	2 корень уравнения 2	Вещественный
P	P=(X1+Y1)/2	Вещественный
Q	Q=(X2+Y2)/2	Вещественный

Проект формы



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;
  Label5: TLabel;
  Edit1: TEdit;
  Label6: TLabel;
  Edit2: TEdit;
  Label7: TLabel;
  Edit3: TEdit;
  Label8: TLabel;
  Label9: TLabel;
  Label10: TLabel;
  Edit4: TEdit;
  Edit5: TEdit;
  Edit6: TEdit;
  Label11: TLabel;
  Button1: TButton;
  Button2: TButton;
  Label12: TLabel;
  procedure Button2Click(Sender: TObject);
  procedure Edit1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
  procedure Edit2KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
  procedure Edit3KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
  procedure Edit4KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
```

```
    procedure Edit5KeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
    procedure Edit6KeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
    procedure Button1Click(Sender: TObject);
  private
     { Private declarations }
  public
     { Public declarations }
  end;
  var
    Form1: TForm1;
    A,B,C:Real;//коэффициенты первого квадратного уравнения
    D,F,G:Real;// коэффициенты второго квадратного уравнения
  implementation
  {$R *.dfm}
  {Функция проверяет – является ли введенное значение числом.
  Допустимыми значениями в числе являются цифры от 0 до 9,
  минус в качестве первого символа, одна десятичная запятая.
  Функция принимает значение Истина, если в строке встретил-
  ся запрещенный символ.}
   function CheckEdit(s:string):boolean;
   // s строка символов, передаваемая из поля ввода
   Var i:integer;// номер по порядку символа в строке символов
       Z:integer;// счетчик запятых
   begin
   CheckEdit:=False;// значению функции присваивается ЛОЖЬ
   Z:=0;// обнуление счетчика десятичных запятых
   //цикл выполняется столько раз, сколько символов
   // во введенной строке
   for i:=1 to Length(S) do
   Begin
//если в строке встречается запятая, счетчик увеличивается на 1
   if s[i]=',' Then Z:=Z+1;
   //если символ не относится к числу допустимых значение
   // функции принимает значение ИСТИНА
   if not(s[i] in ['0'..'9','-',',])Then CheckEdit:=True
   //иначе если не первый символ равен минусу или счетчик
```

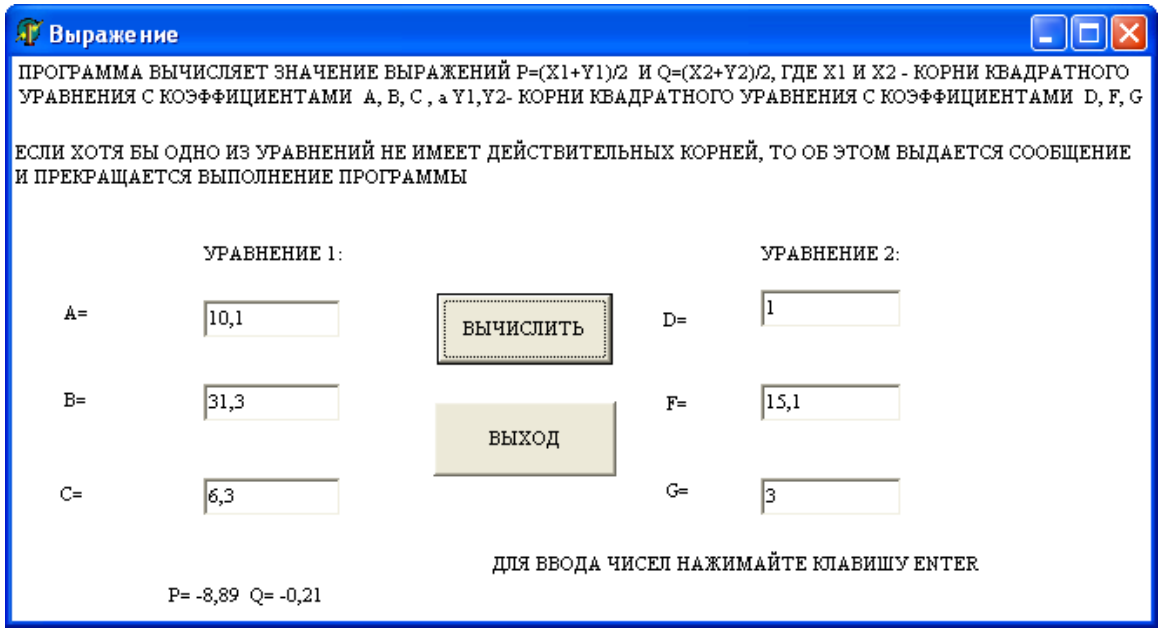
```
    // запятых больше или равен 2,
    // значение функции принимает значение ИСТИНА
    else if ((i<>1) and (S[i]='-')) or (Z>=2) Then CheckEdit:=True
    end;
    end;
    //процедура вычисления корней квадратного уравнения
    procedure Root(A,B,C:Real;Var R:Boolean;Var X1,X2:Real);
    Var DT:Real;//дискриминант
    begin
    //признак действительных решений у уравнения
    // принимает значение ИСТИНА
    R:=True;
    DT:=sqr(B)-4*A*C;
    if DT>0 Then begin
                    X1:=(-B-sqrt(DT))/(2*A);
                    X2:=(-B+sqrt(DT))/(2*A);
                    end
    else
    // признак действительных решений у уравнения
    // принимает значение ЛОЖЬ
                    R:=False;
    end;
    procedure TForm1.Button2Click(Sender: TObject);
    begin
    Form1.Close
    end;
    //процедура обработки события – переход вниз из поля ввода
    procedure TForm1.Edit1KeyDown(Sender: TObject; var Key:
Word; Shift: TShiftState);
    begin
//если была нажата клавиша ENTER,то если значение вызванной
//функции проверки допустимости символов в строке, которой
//в качестве фактического параметра передается содержимое поля
 //ввода, равно истине (символ недопустим), то поле ввода
// очищается и фокус остается в нем
    if Key=VK_RETURN
        then  if CheckEdit(Edit1.Text) Then begin
                                        Edit1.Clear;
```

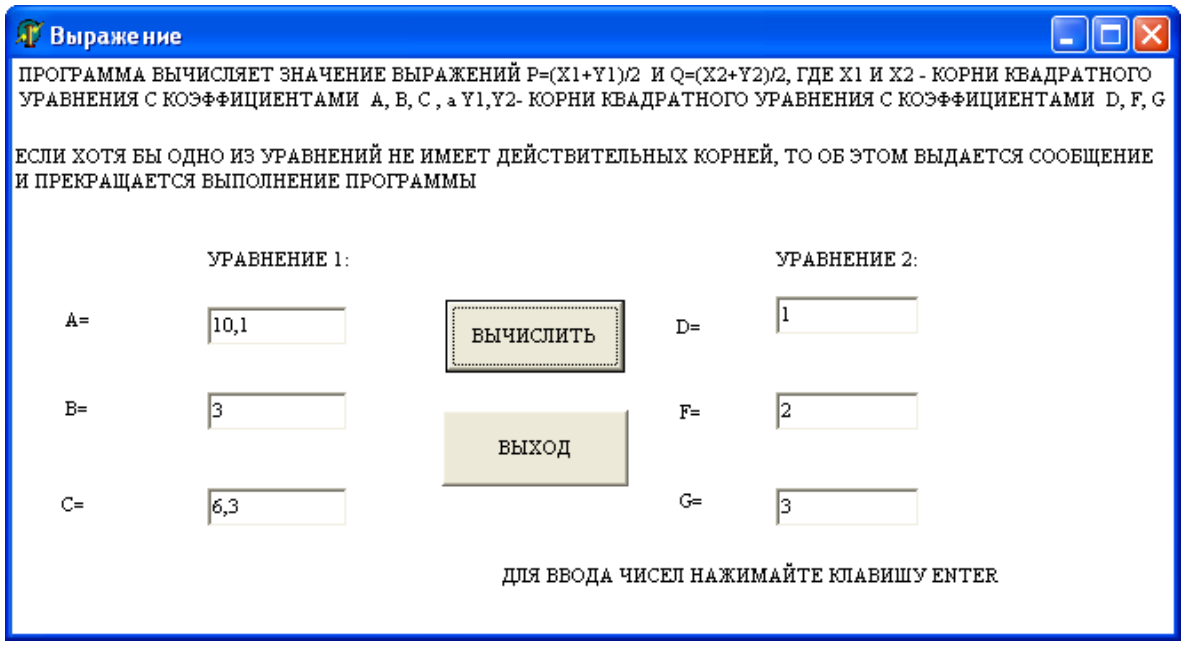
```
                                        Edit1.SetFocus
                                        end
    //иначе фокус переносится в следующее поле ввода
                                        Else
                                        Edit2.SetFocus;
    end;
//процедура аналогична процедуре
// procedure TForm1.Edit1KeyDown
procedure TForm1.Edit2KeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    begin
    if Key=VK_RETURN
            then  if CheckEdit(Edit2.Text) Then begin
                                                Edit2.Clear;
                                                Edit2.SetFocus
                                                end
                                                Else
                                                Edit3.SetFocus;
                                                end;
    //процедура аналогична procedure TForm1.Edit1KeyDown
procedure TForm1.Edit3KeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    begin
    If Key=VK_RETURN
            then  if CheckEdit(Edit3.Text) Then begin
                                                Edit3.Clear;
                                                Edit3.SetFocus
                                                end
                                                Else
                                                Edit4.SetFocus;
                                                end;
    //процедура аналогична procedure TForm1.Edit1KeyDown
procedure TForm1.Edit4KeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    begin
    If Key=VK_RETURN
            then  if CheckEdit(Edit4.Text) Then begin
                                                Edit4.Clear;
```

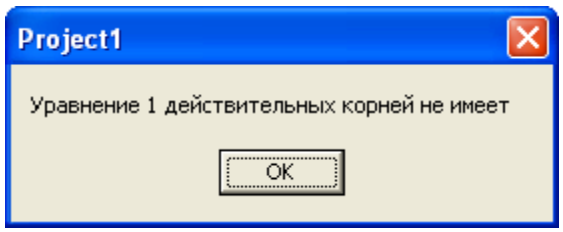
```
                                        Edit4.SetFocus
                                        end
                                        Else
                                        Edit5.SetFocus;
                                        end;
    //процедура аналогична procedure TForm1.Edit1KeyDown
procedure TForm1.Edit5KeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    begin
    If Key=VK_RETURN
            then  if CheckEdit(Edit5.Text) Then begin
                                        Edit5.Clear;
                                        Edit5.SetFocus
                                        end
                                        Else
                                        Edit6.SetFocus;
                                        end;
    //процедура аналогична procedure TForm1.Edit1KeyDown
procedure TForm1.Edit6KeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
    begin
    If Key=VK_RETURN
            then  if CheckEdit(Edit6.Text) Then begin
                                        Edit6.Clear;
                                        Edit6.SetFocus
                                        end
                                        Else
                                        Button1.SetFocus;
                                        end;
    //процедура связанная с кнопкой вычислить
    procedure TForm1.Button1Click(Sender: TObject);
    Var R1,R2:Boolean;{признаки решения первого и второго
квадратных уравнений }
            P,Q:Real;//значения вычисляемых выражений
            X1,X2:Real;//корни первого уравнения
            Y1,Y2: Real;// корни второго уравнения
    begin
    //инициализация переменных
```

```
 X1:=0;X2:=0;
 Y1:=0;Y2:=0;
 //ввод из полей ввода коэффициентов уравнений
 A:=StrToFloat(Edit1.Text);
 B:=StrToFloat(Edit2.Text);
 C:=StrToFloat(Edit3.Text);
 D:=StrToFloat(Edit4.Text);
 F:=StrToFloat(Edit5.Text);
 G:=StrToFloat(Edit6.Text);
 //вызов процедуры вычисления корней квадратного уравнения
 //(для двух уравнений), если уравнение не имеет
 //действительных корней, признак решения
 // принимает значение ложь
 Root(A,B,C,R1,X1,X2);
 Root(D,F,G,R2,Y1,Y2);
 if not(R1) Then begin
 ShowMessage('Уравнение 1 действительных корней не имеет');
 Exit //прекращение выполнения программы
 end
 else
 if not(R2) Then begin
ShowMessage('Уравнение 2 действительных корней не имеет ');
Exit
                    end
 //если оба уравнения имеют действительные решения,
 //то вычисляются значения P и Q, их значения выводятся
 // на форму в поля вывода
 else
     begin
     P:=(X1+Y1)/2;
     Q:=(X2+Y2)/2;
     Label11.Caption:='P= '+FloatToStrF(P,Fffixed,10,2)+
     ' Q= '+FloatToStrF(Q,Fffixed,10,2);
     end;
 end;
 end.
```

Результаты работы программы







Вопросы для самоконтроля

1. Дайте определение подпрограммы.
2. Назначение процедуры в Delphi.

3. Правила описания процедуры.

4. Вызов процедуры.

5. Назначение функции в Delphi.

6. Правила описания функции.

7. Вызов функции.

8. Назовите формальные параметры процедуры GMS в программе «Определение угла треугольника» (Задача 8a).

9. Назовите фактические параметры процедуры GMS в программе «Определение угла треугольника» (Задача 8a).

10. Покажите место в программе «Вычисление факториала», где описана функция вычисления факториала (Задача 8b).

11. Покажите место в программе «Вычисление факториала», где вызывается на выполнение функция вычисления факториала (Задача 8b).

12. Покажите место в программе «Выражение», где описана процедура вычисления корней квадратного уравнения (Задача 8c).

13. Покажите место в программе «Выражение», где вызывается на выполнение процедура вычисления корней квадратного уравнения (Задача 8c).

14. Покажите в программе «Выражение» хотя бы одну из процедур, связанных с событием «переход вниз из поля ввода» (Задача 8c).

15. Чем процедура и функция отличаются друг от друга?

Задачи для самостоятельного решения

№ 8.1 Составить программу для вычисления u= min (a, b), v=min(min(a/b, b/a), min (u, 3.14)), где a и b – действительные числа. Нахождения минимума оформить как функцию.

№ 8.2 Даны действительные числа s, t. Получить

h (s, t) + max (h^2 (s, s*t), h^2(s-t, s+t)), где

$$h(a,b) = \frac{a}{1+b^2} + \frac{b}{1+a^2} - (a-b)^3.$$

Вычисление h(a,b) и max оформить в виде процедур.

№ 8.3 Даны действительные числа s, t.

Получить f(t, -2*s, 1.17) + f(2.2, t, s-t), где

$$f(a,b,c)=\frac{2a-b-\sin c}{5+|c|}.$$

Для вычисления f(a,b,c) в программе предусмотреть процедуру.

№ 8.4 Даны действительные числа s, t. Получить

g(1.2, s) + g(t, s)-g(2*s-1,s*t),

где g(a, b) = $\frac{a^2+b^2}{a^2+2ab+3b^2+4}$.

Вычисление g в программе оформить в виде процедуры.

№ 8.5 Составить программу для вычисления суммы ряда

$$\sum_{n=0}^{m}(-1)^n\frac{x^{2n}}{(n!)^2 2^{2n}},$$

где x – вещественное число;

n, m – натуральные числа.

Для возведения в степень и вычисления факториала разработать функции.

№ 8.6 Даны натуральные числа m и n. Получить

$$\frac{m!+n!}{(m+n)!}$$

Для вычисления факториала в программе предусмотреть функцию.

№ 8.7 Даны коэффициенты двух квадратных уравнений $a_1x^2+b_1x+c_1=0$ и $a_2x^2+b_2x+c_2=0$. Составить программу для вычисления значений:

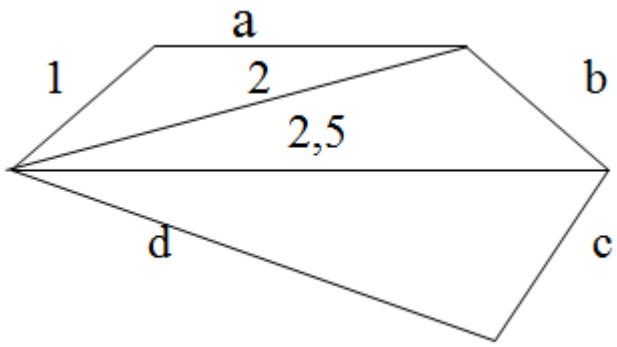
$$Z_1=\frac{x_1+y_1}{x_2-y_2};\ Z_2=\frac{x_2+y_2}{x_1-y_1},$$

где x_1 и y_1 – корни 1–го уравнения;

x_2 и y_2 – корни 2–го уравнения.

Для решения квадратного уравнения предусмотреть в программе процедуру.

№ 8.8 Даны действительные числа a, b, c, d. Найти площадь пятиугольника, изображенного на рисунке ниже. Определить процедуру вычисления площади треугольника по трем сторонам.



№ 8.9 Составить программу для вычисления

$$Q=\frac{1}{2c(n-1)x^{n-1}},$$

где ($n \geq 1$).

Возведение в степень оформить в виде функции.

Группа Б

№ 8.10. Написать программу для определения взаимного расположения точек с координатами X_1, Y_1 и X_2, Y_2 по отношению к прямой, которая задается уравнением AX+BY+C=0. Для этого в программе необходимо сравнить знаки выражений, вычисляемых по формулам AX_1+BY_1+C и AX_2+BY_2+C. При этом возможны следующие ситуации:

– если знаки обоих выражений совпадают, то обе точки лежат по одну сторону от прямой,

– если знаки у выражений различны, то точки лежат по разные стороны от прямой,

– если одно или оба выражения равны нулю, то соответственно одна или обе точки лежат на прямой.

Для вычисления выражений вида AX+BY+C разработать функцию.

В программе предусмотреть следующую форму запросов при вводе данных:

A, B, C=

X1, Y1=

X2, Y2=

В качестве результата выводить одно из следующих сообщений «ПО РАЗНЫЕ СТОРОНЫ» или «ПО ОДНУ СТОРОНУ» или «ТОЧКА номер точки ЛЕЖИТ НА ПРЯМОЙ».

№ 8.11. Написать программу для определения углов треугольника α, β, γ по заданным значениям сторон A, B, C, используя теорему косинусов:

$$\cos\alpha = \frac{B^2 + C^2 - A^2}{2BC}, \cos\beta = \frac{A^2 + C^2 - B^2}{2AC},$$

$$\cos\gamma = \frac{A^2 + B^2 - C^2}{2AB}.$$

В программе предусмотреть проверку существования треугольника с заданными сторонами. Условие существования:

$$(A+B>C)\wedge(A+C>B)\wedge(B+C>A).$$

Если указанное условие не выполняется, выдавать сообщение «ИЗ ВВЕДЕННЫХ СТОРОН ТРЕУГОЛЬНИК ПОСТРОИТЬ НЕЛЬЗЯ».

Для определения угла использовать преобразование

$$tg\alpha = \frac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}.$$

В этом случае:

$$\alpha = \begin{cases} arctg\dfrac{\sqrt{1-\cos^2\alpha}}{\cos\alpha}, если \cos\alpha > 0 \\ arctg\dfrac{\sqrt{1-\cos^2\alpha}}{\cos\alpha} + 180, если \cos\alpha < 0 \\ 90, если \cos\alpha = 0 \end{cases}$$

Аналогично для β и γ .

Значения углов выводить в градусах, минутах и секундах, округляя до 1 сек.

Для определения и вывода значения углов в гр., мин., сек. использовать процедуру.

В программе предусмотреть следующую форму запросов при вводе исходных данных:

СТОРОНЫ A, B, C =

Форма вывода результатов:

УГОЛ АЛЬФА = значение α в гр., мин., сек.

УГОЛ БЕТА = значение β в гр., мин., сек.

УГОЛ ГАММА = значение γ в гр., мин., сек.

№ 8.12. Написать программу вычисления площади треугольного участка по известному значению боковой стороны A в метрах и углов при основании α, β в градусах, минутах и секундах.

$$S = \frac{A^2 \sin\beta(\cos\beta + \sin\beta ctg\alpha)}{2}.$$

Площадь вычислять в гектарах с точностью до 0,1 га.

В программе предусмотреть проверку α+ β <180 (для этого углы необходимо предварительно перевести в градусы). В случае нарушения указанного соотношения, вывести сообщение «УГЛЫ НЕРЕАЛЬНЫ».

В программе воспользоваться функцией пользователя для перевода углов из градусной меры в радианную.

Форма запросов при вводе исходных данных:

СТОРОНА =

ПЕРВЫЙ УГОЛ: ГР., МИН., СЕК =

ВТОРОЙ УГОЛ: ГР., МИН., СЕК =

Форма вывода результата:

ПЛОЩАДЬ = значение S ГА.

№ 8.13. Написать программу для вычисления площади треугольного участка по известному значению основания C в метрах и углов при основании α, β в градусах, минутах и секундах. Площадь определить в гектарах с точностью до 0,01 га.

В программе предусмотреть проверку соотношения α+β<180° (для этого α и β необходимо предварительно перевести в градусы с десятичной дробной частью). В случае невыполнения этого соотношения вывести сообщение «СУММА УГЛОВ ПРЕВОСХОДИТ 180 ГР.».

Формула для вычисления площади:

$$S = \frac{C^2 \sin\alpha \sin\beta}{2\sin(\alpha + \beta)}.$$

В программе разработать функцию для перевода угла из градусной меры в радианную.

Форма запросов при вводе исходных данных:

СТОРОНА=

ПЕРВЫЙ УГОЛ: ГР., МИН., СЕК.=

ВТОРОЙ УГОЛ: ГР., МИН., СЕК.=

Форма вывода результата:

ПЛОЩАДЬ= значение S ГА.

№ 8.14. Написать программу для вычисления площади четырехугольного участка по заданным значениям двух противолежащих углов α, β и образующих эти углы сторон A, B и C, D

$$S = (A \cdot B \sin\alpha + C \cdot D \sin\beta)/2.$$

Углы вводить в градусах, минутах и секундах, стороны – в метрах, значение площади выводить в гектарах, округляя до 0,01 га. В программе предусмотреть проверку существования четырехугольника с указанными значениями сторон и противолежащих углов.

$$| A^2 + B^2 - 2AB\cos\alpha - C^2 - D^2 + 2CD\cos\beta | < 0{,}1.$$

Если указанное неравенство не выполняется, вывести сообщение «УЧАСТОК С УКАЗАННЫМИ ПАРАМЕТРАМИ НЕ СУЩЕСТВУЕТ» и передать управление операторам ввода исходных данных.

В программе предусмотреть следующие запросы при вводе исходных данных:

СТОРОНЫ A, B=

УГОЛ МЕЖДУ A, B: ГР., МИН., СЕК.=

СТОРОНЫ C, D=

УГОЛ МЕЖДУ C, D: ГР., МИН., СЕК.=

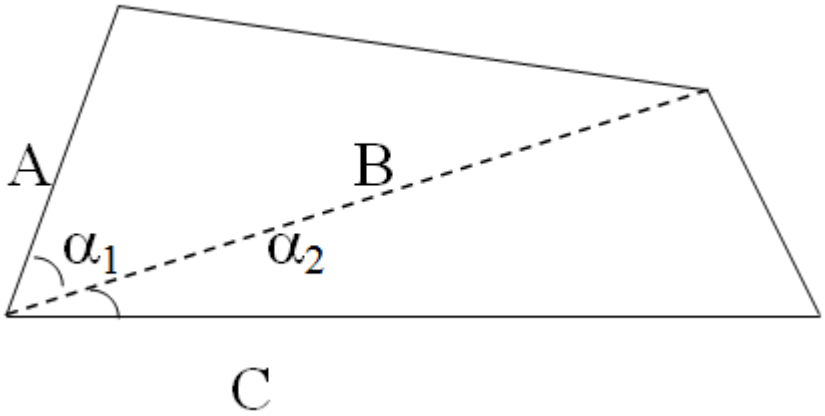
Форма вывода результата:

ПЛОЩАДЬ= значение S ГА.

В программе разработать функцию для перевода угла из градусной меры в радианную.

№ 8.15. Из вершины O участка, изображенного на рис.ниже, измерены расстояния A, B, C до остальных его вершин и углы α_1 α_2. Написать программу для вычисления площади этого участка по формуле

$$S = \frac{B(A\sin\alpha_1 + C\sin\alpha_2)}{2}.$$



Расстояния A, B, C измерены в метрах, углы α, β в градусах, минутах, секундах. Площадь вычислить в гектарах с точностью до 0,01 га.

В программе предусмотреть следующие запросы при вводе:

A, B, C=

УГОЛ МЕЖДУ A и B: ГР., МИН., СЕК.=

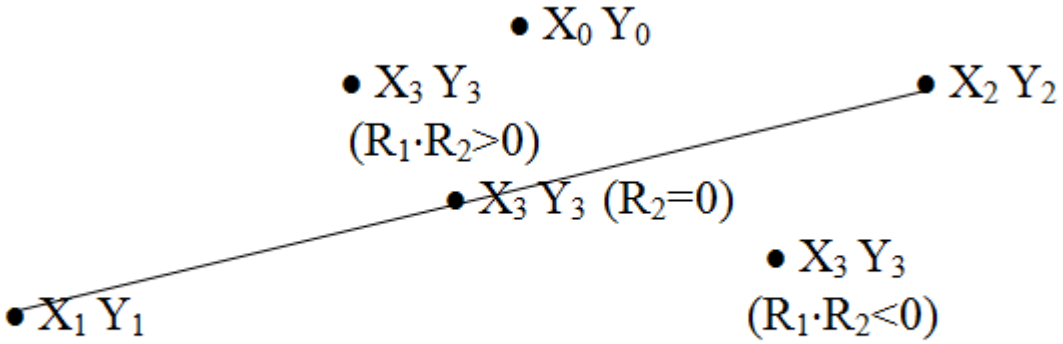
УГОЛ МЕЖДУ C и D: ГР., МИН., СЕК.=

Для вычисления синусов углов, заданных в градусах, минутах и секундах разработать функцию.

Форма вывода результата:

ПЛОЩАДЬ= значение S ГА.

№ 8.16. Написать программу, которая определяет, как расположена точка с координатами X_3, Y_3 по отношению к прямой, проведенной через точки X_1,Y_1 и X_2, Y_2 и точке с координатами X_0, Y_0 (см рис. ниже).



Для этого программа должна вначале вычислить величину

$$R_1 = \frac{Y_0 - Y_1}{Y_2 - Y_1} - \frac{X_0 - X_1}{X_2 - X_1},$$

Предусмотреть проверку знаменателя на равенство 0. Если знаменатель оказался нулевым ввести другие координаты точек.

Затем проверить условие R_1=0. Если это условие выполняется, необходимо вывести сообщение «КОНТРОЛЬНАЯ ТОЧКА ЛЕЖИТ НА ПРЯМОЙ» и передать управление операторам ввода X_0, Y_0 для исправления координат контрольной точки.

Если условие R_1=0 не выполняется, необходимо вычислить величину

$$R_2 = \frac{Y_3 - Y_1}{Y_2 - Y_1} - \frac{X_3 - X_1}{X_2 - X_1}.$$

Если R_1 и R_2 совпадут, значит, точки лежат по одну сторону от прямой, если не совпадут, значит, точки лежат по разные стороны от прямой, если R_2 =0, значит, точка X_3 Y_3 лежит на прямой.

Для вычисления R_1 и R_2 разработать функцию.

Форма запросов при вводе:

X1, Y1=

X2, Y2=

X0, Y0=

X3, Y3=

В качестве результата выводить одно из следующих сообщений:

«ТОЧКА номер точки ЛЕЖИТ ПО ОДНУ СТОРОНУ С КОНТРОЛЬНОЙ ТОЧКОЙ» или «ТОЧКА номер точки ЛЕЖИТ ПО РАЗНЫЕ СТОРОНЫ С КОНТРОЛЬНОЙ ТОЧКОЙ» или «ТОЧКА номер точки ЛЕЖИТ НА ПРЯМОЙ».

№ 8.17 Написать программу для вычисления по формуле :

$$\frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n+1} \cdot \frac{x^{2 \cdot n+1}}{2 \cdot n+1} = \frac{\pi}{2} - \left(x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots \right)$$

Для $|x|<1$. Для вычисления члена в разложении разработать функцию. Число членов в разложении должен задать пользователь программы.

№ 8.18 Написать программу для вычисления по формуле:

$$\frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n+1} \cdot \frac{1}{(2 \cdot n+1) \cdot x^{2 \cdot n+1}} = \frac{\pi}{2} - \left(\frac{1}{x} - \frac{1}{3x^3} + \frac{1}{5x^5} - \frac{1}{7x^7} + \ldots \right)$$

Для $|x|>1$. Число членов в разложении должен задать пользователь программы. Для вычисления члена в разложении разработать функцию.

№ 8.19 Написать программу для вычисления по формуле:

$$1-\sum_{n=1}^{\infty}(-1)^{n+1}\cdot\frac{2^{2\cdot n-1}\cdot x^{2\cdot n}}{(2\cdot n)!}=1-\left(\frac{2^1\cdot x^2}{2!}-\frac{2^3\cdot x^4}{4!}+\frac{2^5\cdot x^6}{6!}-\frac{2^7\cdot x^8}{8!}+\ldots\right)$$

Для $x<1$. Число членов в разложении должен задать пользователь программы. Для вычисления члена в разложении разработать функцию.

№ 8.20 Написать программу вычисления количества сочетаний из n элементов по m по формуле :

$$C_n^m=\frac{n!}{m!(n-m)!}$$

Для вычисления факториала в программе предусмотреть функцию.

Глава 9. Работа с внешними файлами

В данной главе рассматриваются следующие вопросы: процедуры работы с файлами, создание главного меню, использование стандартных диалогов.

Теория

Процедуры работы с файлами

Во всех предыдущих параграфах ввод исходных данных осуществлялся во время исполнения программы с клавиатуры с использованием компонент Edit, Memo, StringGrid и других. Однако, часто удобнее заранее готовить исходные данные в файлах, проверять правильность их подготовки, а при работе программы автоматически вводить уже подготовленные файлы. При больших объемах исходных данных только так и можно эффективно работать.

Файл – это именованная структура данных, представляющая собой последовательность элементов данного типа, причем количество элементов последовательности практически не ограничено.

Файл должен быть объявлен в разделе описания переменных. В общем виде объявление файла:

Имя: file of Тип Элементов;

Примеры описания файлов:

Var Job: file of char; // файл символьного типа

Fz: file of integer; // файл целого типа

Test: file of real; // файл вещественного типа

Файл, элементами которого могут быть данные разных типов, называется текстовым.

Описание текстового файла в общем виде:

Имя: TextFile;

где Имя – имя файловой переменной, TextFile – обозначение типа, показывающее, что Имя – это файловая переменная, представляющая текстовый файл.

My_File: TextFile; // файл текстового типа.

Подготовить текстовый файл можно, например, в программе Блокнот (Все программы – Стандартные – Блокнот) или непосредственно в Delphi (Файл – Создать – Text). В последнем случае файл можно создавать непосредственно в папке проекта.

Внешний файл, содержащий одномерный массив действительных чисел из пяти элементов имеет вид:

5
3.16
7.04
-2.17
4.57
0.12

Здесь первый элемент-длина файла, каждый следующий элемент вводится в новой строке (через Enter).

Внешний файл, содержащий двумерный массив 2x4 целых чисел имеет вид:

2 4
6 9 1 3
8 5 0 7

Здесь в первой строке размерность двухмерного массива. Остальные элементы вводятся построчно. Элементы в строке вводятся через пробел, переход к следующей строке через Enter.

Для того чтобы программа могла считывать исходные данные из файла (ввод данных) или записывать результаты своей ра-

боты в файл (вывод данных), необходимо связать файловую переменную с конкретным файлом (задать имя файла).

Процедура **AssignFile** устанавливает физическую связь между файловой переменной f и строковой переменной ИмяФайла:

AssignFile (f, ИмяФайла: string);

Примеры вызова процедуры AssignFile:

AssignFile(f, ‘d:\Мои документы\result.txt’);

AssignFile(Fin, ‘test011.txt’);

FileName:=’otchet1.txt’; AssignFile(Fot, FileName);

Часто, при создании проекта удобнее не указывать конкретное имя файла исходных данных, а предусмотреть, что при выполнении программы пользователь введет это имя, например, в компонент Edit1.Text.

Тогда вызов процедуры AssignFile запишется так:

FileName:= Edit1.Text ; AssignFile(Fot, FileName);

Или еще проще:

AssignFile(Fot, Edit1.Text);

Прежде чем читать информацию из файла или записывать информацию в файл необходимо его открыть.

Открыть файл для чтения можно с помощью процедуры **Reset(f)**, где **f** – имя файловой переменной. Процедура **Reset** ищет файл, открывает его и устанавливает указатель на начало файла. Если файл не найден, выдается сообщение об ошибке ввода/вывода и программа завершает свою работу.

Открыть файл для записи позволяет процедура **Rewrite(f),** где **f** – имя файловой переменной. Процедура **Rewrite(f)** ищет файл, открывает его, стирает в нем старую информацию и устанавливает указатель на начало файла. Если файл не найден, он создается заново.

Для открытия файла с целью добавления в него информации служит процедура **AppendFile(f)**, где **f** – имя файловой переменной. Процедура **AppendFile** открывает файл и устанавливает указатель на конец файла. Если файл не найден, выдается сообщение об ошибке ввода/вывода и программа завершает свою работу. В Delphi XE данная процедура имеет название **Append(f)**.

Для чтения (ввода) информации из файла предназначены процедуры:

Read(f,[список ввода]);

Readln(f,[список ввода]);

где f – имя файловой переменной,

список ввода – переменные, элементы массивов.

Процедура Readln имеет следующие особенности:

– после ввода данных указатель переносится на начало следующей строки;

– процедура используется только для текстовых файлов.

Для записи (вывода) данных в файл существуют процедуры:

Write(f,[список вывода]);

Writeln(f,[список вывода]);

где f – имя файловой переменной,

список вывода – переменные, элементы массивов, константы.

Отличительными особенностями процедуры Writeln также является перенос указателя на новую строку после вывода данных и применимость ее только в случае текстовых файлов.

Seek(f, N) перемещает указатель файла в нужную позицию N.

Часто в программах при работе с файлами оказываются полезными функции EOF(f) и EOLN(f).

EOF(f) – логическая функция, принимает значение TRUE (истина), если обнаружен конец файла.

EOLN(f) – логическая функция, принимает значение TRUE (истина), если обнаружен конец строки.

FileSize(f) – функция, которая определяет, сколько элементов записано в файле.

В Delphi XE при работе с текстовыми файлами хорошую помощь могут оказать функции:

SeekEoln(f) − если строка начинается с пробелов, они пропускаются, указатель файла перемещается на значащие символы.

SeekEof(f) − если встречается пустая строка, то она пропускается. Указатель файла перемещается на строку, в которой есть значащие символы.

После окончания всех действий с файлом он должен быть закрыт с помощью процедуры **CloseFile(f)**, где **f** – имя файловой переменной.

Пример

В параграфе 6 описана задача подсчета количества положительных, отрицательных и нулевых элементов в массиве, введенном в компонент Memo1(Задача 6b). Данные вводились с клавиатуры при исполнении программы. Ниже приведена процедура, решающая ту же задачу в случае, когда исходные данные подготовлены во внешнем файле, имя которого при исполнении программы вводится в Edit1.text.

```
procedure TForm1.Button1Click(Sender: TObject);
var f:textfile;
imfile:string;
a:array[1..100] of real;
n,i:integer;
sp,so,s:integer;
begin
sp:=0; so:=0; s:=0;
imfile:= edit1.text;
assignfile(f,imfile);
reset(f);
readln(f,n);
for i:=1 to n do
begin
readln(f,a[i]);
memo1.lines.add(floatTostr(a[i]));
if a[i]<0 then so:=so+1 else if a[i]>0 then
sp:=sp+1 else s:= s+1;
end;
CloseFile(f);
label1.caption:= 'колич.отриц. чисел =' + inttostr(so)+#13+
'колич.положит. чисел =' + inttostr(sp) +#13+ ‘колич.нулей =’+
inttostr(s)
end;
end.
```

Главное меню

Компонент находится на вкладке Standard.

MainMenu позволяет поместить главное меню в программу.

При помещении MainMenu на форму это выглядит, как просто иконка. Иконки данного типа называют «невидимыми компонентами», поскольку они невидимы во время выполнения программы. Создание меню включает три шага: (1) помещение MainMenu на форму, (2) вызов Дизайнера Меню через свойство Items в Инспекторе Объектов, (3) определение пунктов меню в Дизайнере Меню**.**

Чтобы создать процедуру, обрабатывающую действия, связанные с выбранным пунктом меню, необходимо произвести два щелчка по пункту на форме, и в редакторе кода будет подготовлена заготовка процедуры, подобная этой:

```
procedure TForm1.N1Click(Sender: TObject);
begin

end;
```

Использование стандартных диалогов

На вкладке Dialogs (диалоги) представлены компоненты для вызова стандартных диалогов Windows. Внешний вид диалогов зависит от используемой версии Windows. Объекты, представленные на данной вкладке невидимы во время выполнения и вызов диалогов происходит программно.

 OpenDialog находится на вкладке Dialogs .

Позволяет выбрать файл из списка (подобно тому, как открывается файл в Word с помощью команды **Открыть**.). Одним из главных свойств компонента OpenDialog (кроме самого имени компонента) является свойство NameFile, которому присваивается имя выбранного из панели диалога файла. В программе для проверки, выбран файл или еще нет, используется функция Execute. Функция логическая. Принимает значение Истина, если файл выбран и Ложь в противном случае. Фрагмент программы для выбора файла из панели диалога может выглядеть так:

```
if not Opendialog1.Execute then Exit;
NameFile := Opendialog1.FileName;
```

 Компонент SaveDialog (находится также на вкладке Dialogs) используется для выбора имени файла, в котором будет сохране-

на информация из диалоговой панели. Обычно такая диалоговая панель вызывается командами **File→Save (Сохранить**) или **File→Save As (Сохранить как)**. Если пользователь указал имя файла и нажал кнопку Ok, то оно сохраняется как значение свойства FileName. Использование данного компонента аналогично использованию компонента OpenDialog. Например,

```
if not SaveDialog1.Execute then Exit;
NameFile:=SaveDialog1.FileName;
```

Организация вывода на принтер

Для вывода на принтер нужно воспользоваться рядом процедур, определенных в модуле Printers. То есть в разделе Uses модуля программы нужно обязательно подключить модуль Printers:

```
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms, Grids, DateUtils, Printers;
```

Далее в программном модуле нужно ассоциировать значение текстовой переменной с принтером с помощью процедуры AssignPrn и открыть файл для записи. Например,

```
Var
  TextToPrint: System.Text;
  ……….
  AssignPrn(TextToPrint); // ассоциировать с принтером
  Rewrite(TextToPrint); // открыть файл
```

После этого вызов стандартных процедур Write и Writeln приведет к выводу текста на устройство печати. Например, вывод содержимого компонента Memo на принтер можно осуществить с помощью следующих инструкций:

```
For Line:=1 to Memo1.Lines.Count do
      Writeln(TextToPrint, Memo1.Lines[Line]);
```

После того как вывод информации завершен, необходимо вызвать стандартную процедуру Close:

```
System.Close(TextToPrint);
```

Структура данных – запись

На практике мы часто сталкиваемся с задачами, где используются данные, состоящие из других данных. Например, инфор-

мация о студентах может состоять не только из фамилии и инициалов, паспортных данных, номера группы, но и оценок, полученных на экзаменах. Для работы с такой информацией в Delphi используется структура данных – *запись* (record).

Запись – это структура данных, состоящая из полей. Поля – это именованные компоненты разного типа.

Переменную-запись можно описать в разделе описания переменных в общем виде как:

```
Имя_переменной: record
                    Поле_1: Тип_1;
                    Поле_2: Тип_2;
                    Поле_К: Тип_К;
                    End;
```

Например,

```
Student: record
   f_name: string[20];
   l_name: string[20];
   day:    integer;
   month:  integer;
   year:   integer;
   address: string[50];
end;
```

Если в программе нужно использовать много переменных-записей одной структуры, то целесообразно в разделе Type объявить тип «запись»:

```
Имя = record
        Поле_1: Тип_1;
        Поле_2 : Тип_2;
        Поле_К : Тип_К;
End;
```

Например,

```
Type
   TDate = record
        Day: integer;
        Month: integer;
        Year: integer;
   End;
```

В разделе описания переменных переменные-записи могут быть объявлены так:

```
Var
    Birthday: TDate;
    Dat1, Dat2: TDate;
```

Для работы с записями в языке Delphi предусмотрен оператор With. Он позволяет не писать имя переменной-записи много раз , а работать просто с полями. В общем виде оператор With выглядит так:

```
With Имя do
  Begin
  {операторы программы}
  End;
```

где:

Имя – имя переменной-записи;

With – зарезервированное слово, означающее, что далее, до слова end, при обращении к полям записи «Имя», имя записи можно не указывать.

Например, если в программе объявлена запись Student (см. выше) и данные о студенте вводятся в поля Edit1, Edit2 и Edit3 диалогового окна, то вместо операторов

```
Student.f_name:=Edit1.Text;
Student.l_name:=Edit2.Text;
Student.address:=Edit3.Text;
```

можно записать

```
with student do
    begin
      f_name:=Edit1.Text;
      l_name:=Edit2.Text;
      address:=Edit3.Text;
    end;
```

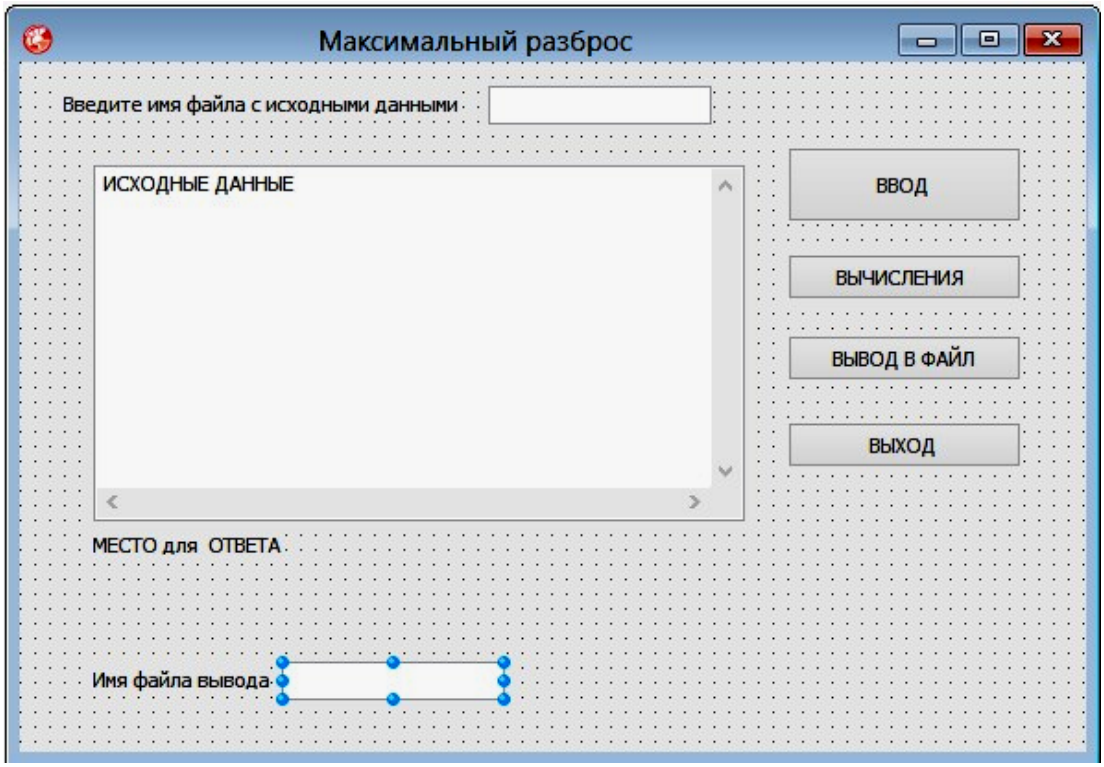
Практика

Задача 9а

Получен ряд измерений некоторой случайной величины. Найти количество измерений (длину ряда) и максимальный разброс значений. Максимальный разброс – модуль разности макси-

мального и минимального значений. Составить программу. Исходные данные ввести (считать) из файла. Вывод осуществить в окно программы и в файл.

Проект формы



Компоненты

Имя Компонента	Свойства компонента	Значение	Назначение
Form1	Caption	Максимальный разброс	Заголовок формы
Label1	Caption	Введите имя файла с исходными данными	Справочная информация для пользователя программы
Label2	Caption	МЕСТО для ОТВЕТА	Подсказка пользователю «МЕСТО для ОТВЕТА»
Edit1	Text	Очищено от значений по умолчанию	Поле для ввода полного имени файла с исходными данными
Label3	Caption Visible	Имя файла вывода FALSE	Подсказка пользователю По умолчанию компонент невидим
Edit2	Text Visible	Очищено от значений по умолчанию FALSE	Поле для ввода полного имени файла с результатом. Компонент по умолчанию невидим

Memo1	Lines ScrollBars	ИСХОДНЫЕ ДАННЫЕ ssBoth	Подсказка пользователю Вывод полос прокрутки
Button1	Caption	ВВОД	Кнопка запускает ввод исходных данных из файла
Button2	Caption	ВЫЧИСЛЕНИЯ	Кнопка для вычислений
Button3	Caption	ВЫВОД В ФАЙЛ	Кнопка запускает вывод результата в файл
Button4	Caption	ВЫХОД	Кнопка закрывает форму

Краткое описание программы

Процедура, связанная с событием клик по кнопке ВВОД

Если не возникает исключительных ситуаций (ошибки ввода/вывода или выполнения) файловой переменной FI ставится в соответствие имя файла, введенное в поле ввода. Файл открывается для чтения. Производится чтение элементов массива MAS[I] (массив вещественных чисел) с одновременным подсчетом их количества N и выводом на форму в поле текстового редактора Memo до тех пор, пока не достигнут конец файла. Файл закрывается. В случае возникновения исключительных ситуаций выдается сообщение «Файл не существует!».

Процедура, связанная с событием клик по кнопке ВЫЧИСЛЕНИЯ

Переменным MAX и MIN присваивается значение первого элемента массива. В цикле от 2 (начиная со второго элемента массива) до N производится сравнение значений MIN и MAX с текущим элементом массива MAS[I]. Если текущий элемент массива MAS[I] меньше MIN, то в MIN помещается текущее значение элемента массива. Если текущий элемент массива MAS[I] больше MAX, то в MAX помещается текущее значение элемента массива.

После выхода из цикла вычисляется RB − максимальный разброс (модуль разности максимального и минимального значений по модулю). Вывод RB на форму. Поле для ввода имени файла вывода становится видимым.

Процедура, связанная с событием клик по кнопке ВЫВОД В ФАЙЛ

Если не возникает исключительных ситуаций, имя файла вывода из поля ввода Edit2 ставится в соответствие файловой переменной

FO. Файл открывается для записи. В файл выводятся значения исходного массива и максимального разброса. Файл закрывается.

В случае возникновения исключительной ситуации выдается сообщение «Задайте имя Файла вывода!» (см. замечание к тексту модуля).

Текст модуля

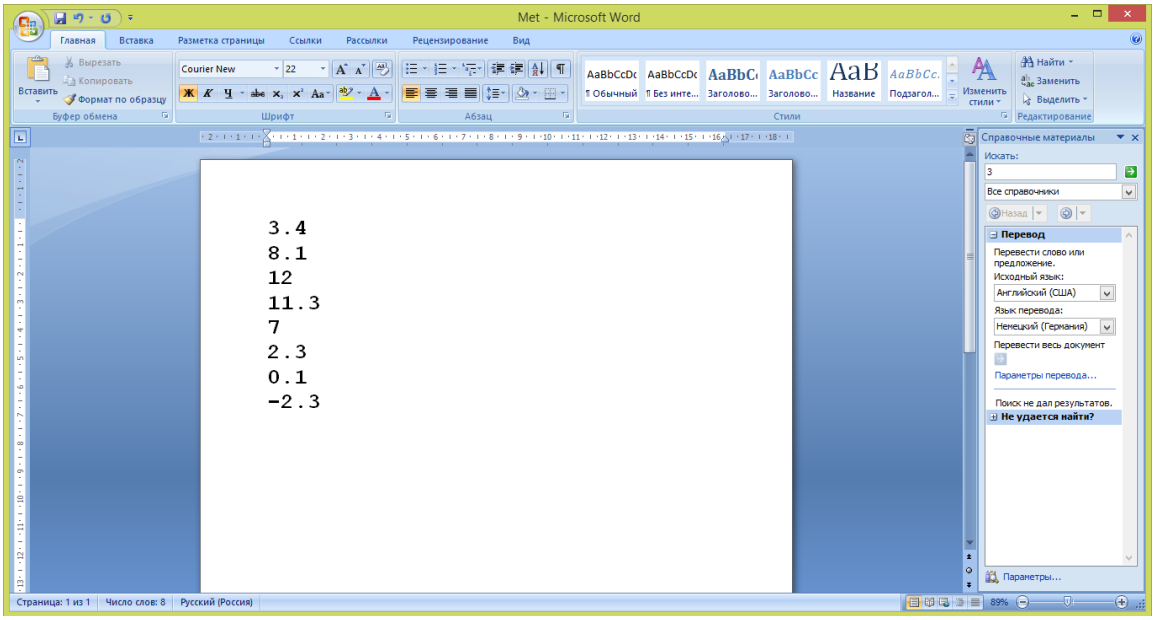
```
unit Unit1;

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;
  Memo1: TMemo;
  Button1: TButton;
  Button2: TButton;
  Label2: TLabel;
  Button3: TButton;
  Edit1: TEdit;
  Label3: TLabel;
  Edit2: TEdit;
  Button4: TButton;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
```

```
var
        Form1: TForm1;
 Mas : Array [1..100] of Real;
  N: Integer;
   I: Integer;
  Min, Max, RB: Real;
  FI, FO: TextFile;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin

  try
  AssignFile (FI,Edit1.Text);
  Reset (FI);
  I:=1;
  Repeat
   Readln(FI,Mas[I]);
   Memo1.Lines.Add (FloatToStrF(Mas[I],Fffixed,8,2));
// создание счетчика длинны файла N
   N:=I;
   I:=I+1;
  Until EOF(FI);
  CloseFile(FI);
Memo1.Lines.Add ('длина исходного массива равна '+IntToStr(N));
    except
   MessageDlg('Файл не существует!', mtError,[mbCancel],0);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);

begin
 Min:=Mas[1];  Max:=Mas[1];
```

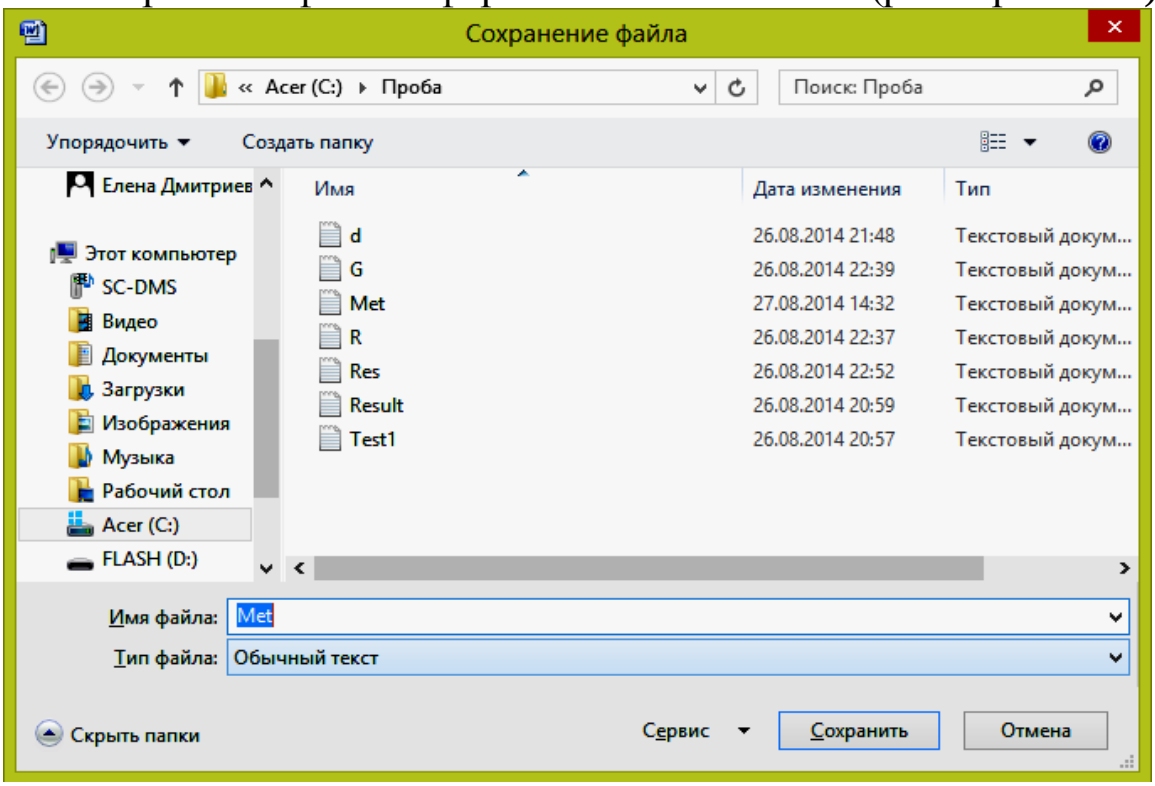
```
    for I := 2 to N do
   begin
   if Min>MAS[I] then Min:=Mas[I];
   if Max<MAS[I] then Max:=Mas[I];
   end;
  Memo1.Lines.Add ('минимум='+FloatToStrF(Min,Fffixed,8,2));
  Memo1.Lines.Add ('максимум='+FloatToStrF(Max,Fffixed,8,2));
   RB:=ABS(Max-Min);
   Label2.Caption:='Максимальный разброс значений массива равен ' +
   FloatToStrF(RB,Fffixed,8,2);
{При проектировании формы поля связанные с заданием имени файла вывода сделаны невидимыми (Visible=False). Видимыми они становятся, если пользователь нажал кнопку ВЫЧИСЛЕНИЯ. Чтобы сделать компонент видимым в процедуре свойству Visible присваивается значение True − истина.}
 Edit2.Visible:=True;
     Label3.Visible:=True;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 try
   AssignFile (FO,Edit2.text);
   ReWrite(FO);
   Writeln (FO,'Исходный массив:');
for I := 1 to N do
Writeln (FO,Mas[i]:8:2);
Writeln (FO,'Длина массива:  ', N);
Writeln (FO,'Минимум = ', MIN:8:2,' Максимум = ',MAX:8:2);
Writeln (FO,'Максимальный разброс значений массива равен ', RB:8:2);
   CloseFile(FO);
  Except
// Если не введено имя файла вывода, выдается следующее
//сообщение
   MessageDlg('Задайте имя Файла вывода!', mtError,[mbCancel],0);
  end;

end;
```

```
procedure TForm1.Button4Click(Sender: TObject);
begin
 Form1.Close;
end;

end.
```

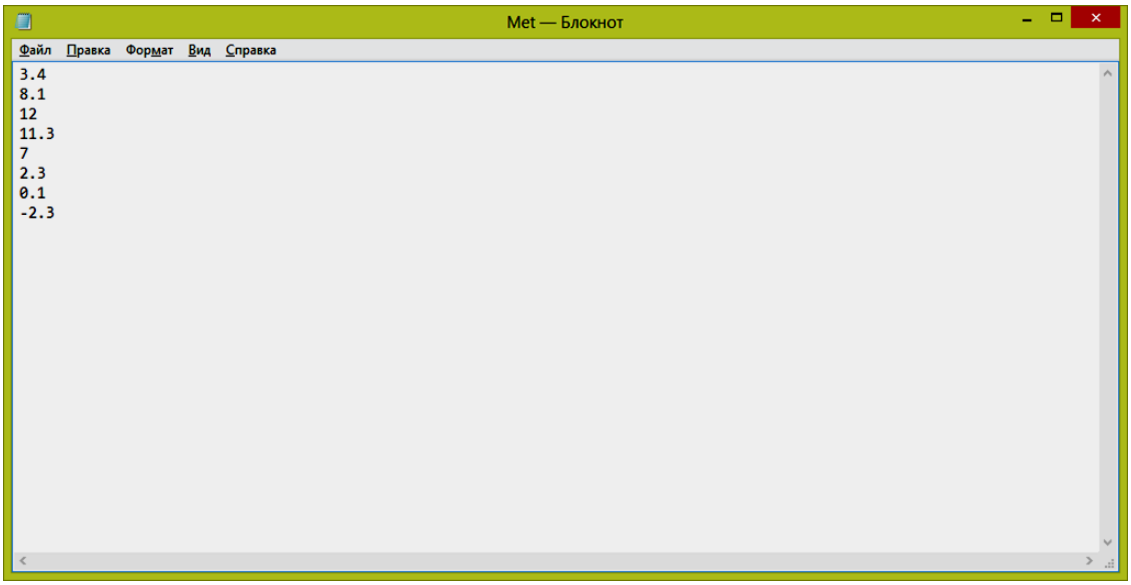
Подготовка файла с исходными данными в редакторе Word.



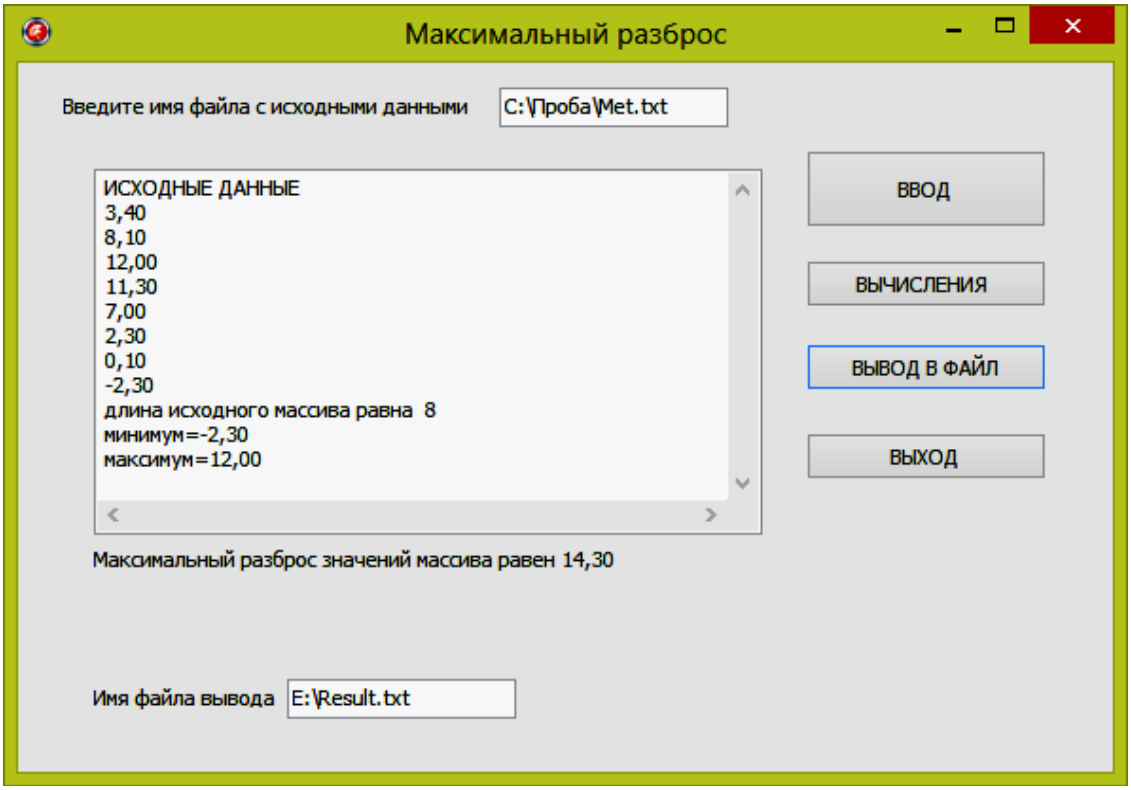
Сохранение файла в формате Обычный текст (расширение txt).



Можно создать файл с исходными данными в редакторе Блокнот.



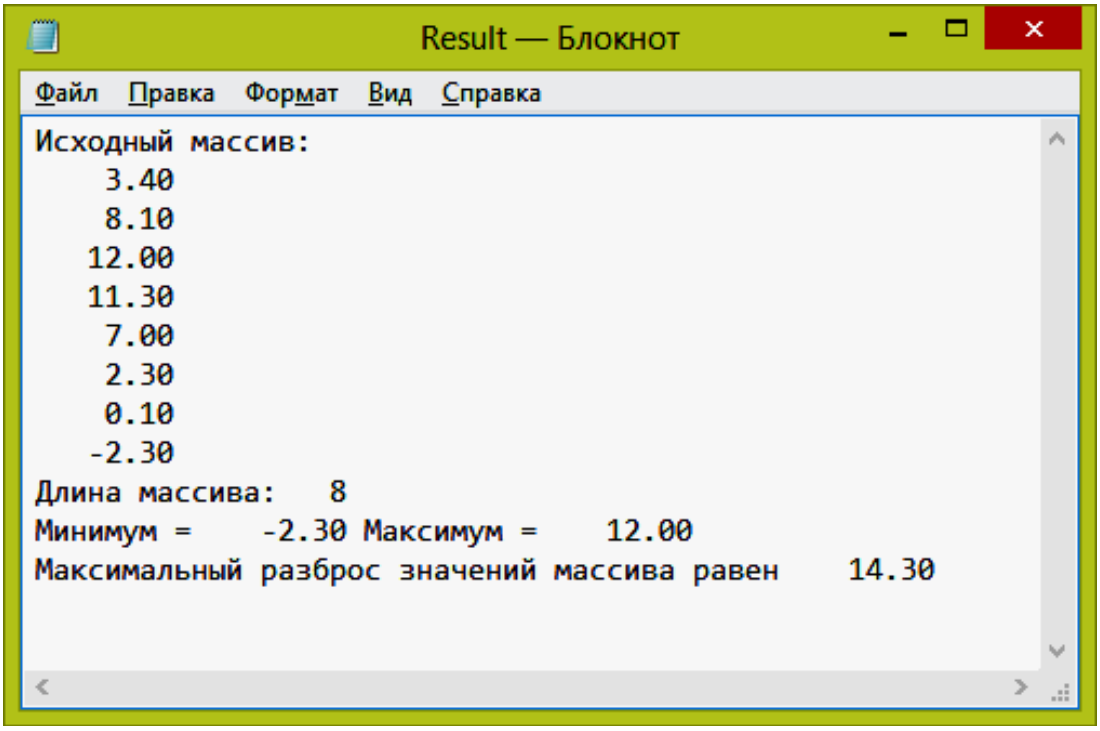
Результаты работы:



Замечание. Прежде чем нажимать кнопки ВВОД и ВЫВОД В ФАЙЛ, нужно корректно задать имя файла. После нажатия кнопки ВВОД на экране в поле редактора Memo появятся данные из файла. С помощью полосы прокрутки их можно все просмотреть и проверить.

Программа может выдавать сообщение «Файл не существует», не только в случаях, когда вы неправильно задали имя файла или вообще забыли его создать, но если вы набрали числа с десятичным разделителем запятая. Вещественные числа в программах имеют разделитель десятичная точка. В файлах с вещественными числами также используется разделитель десятичная точка. После ввода и проверки данных можно нажимать кнопку ВЫЧИСЛЕНИЯ. В окне появится результат счета и поле для ввода имени файла вывода.

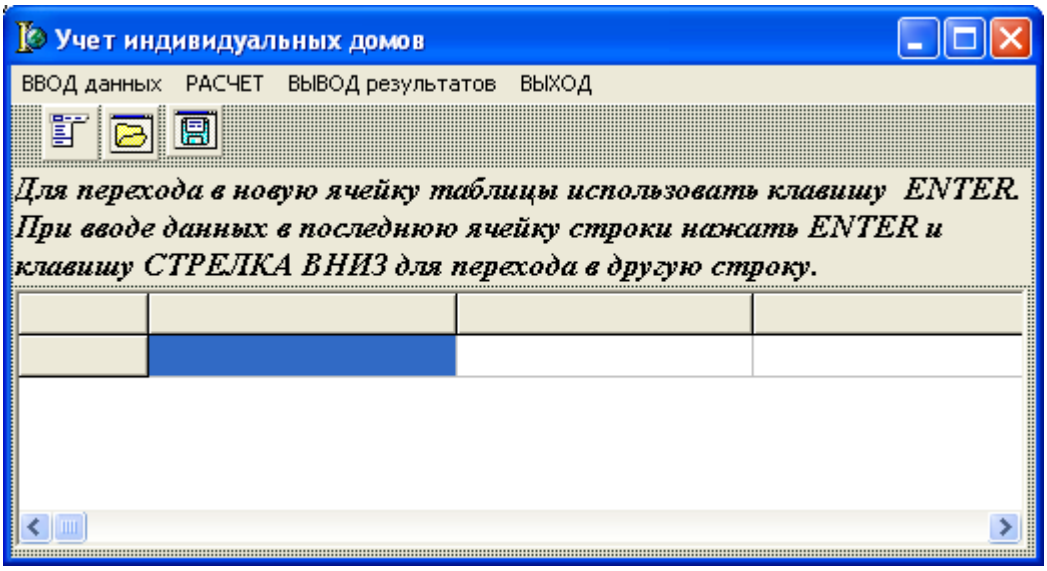
Файл с результатами работы программы

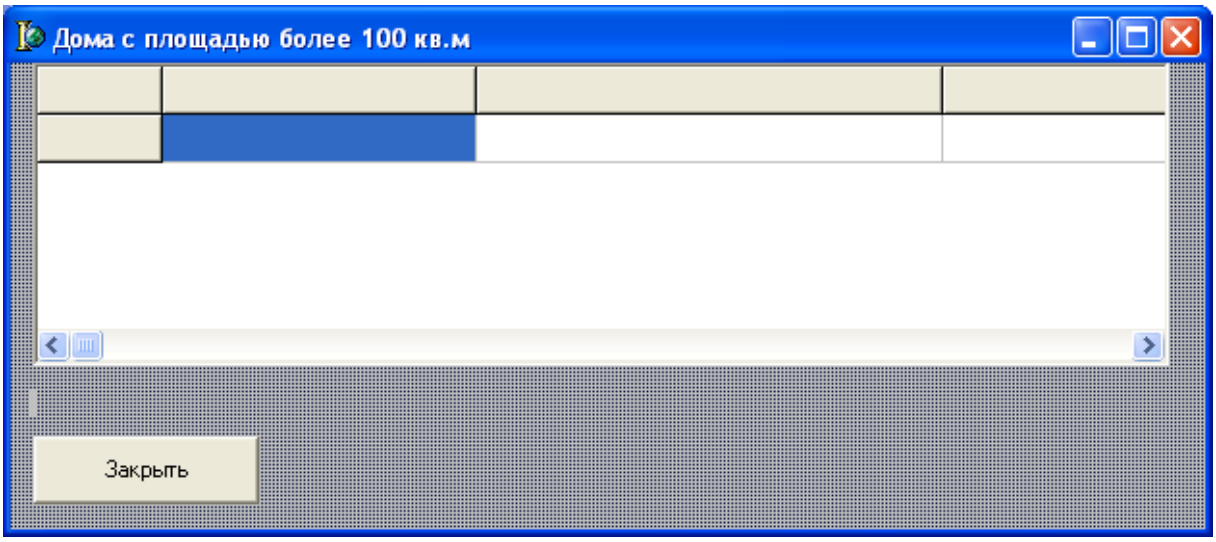


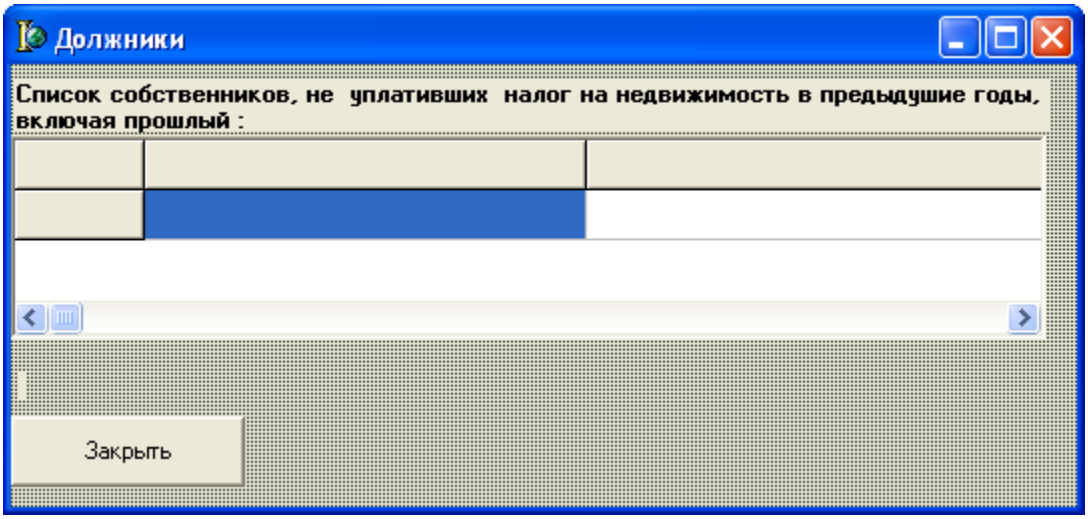
Задача 9b

Дана следующая информация об индивидуальных домах в некотором регионе: номер кадастрового дела, Фамилия И.О. собственника, адрес дома, общая площадь (кв.м), дата последней уплаты налога на недвижимость. Составить программу, которая: 1) находит собственников, не уплативших налог за предыдущий год; 2) выводит информацию о домах площадью более 100 кв.м. Предусмотреть возможности ввода исходных данных с клавиатуры и из файла на диске. Вывод предусмотреть в двух вариантах: в файл либо на принтер.

Для решения задачи предлагается создать три окна:





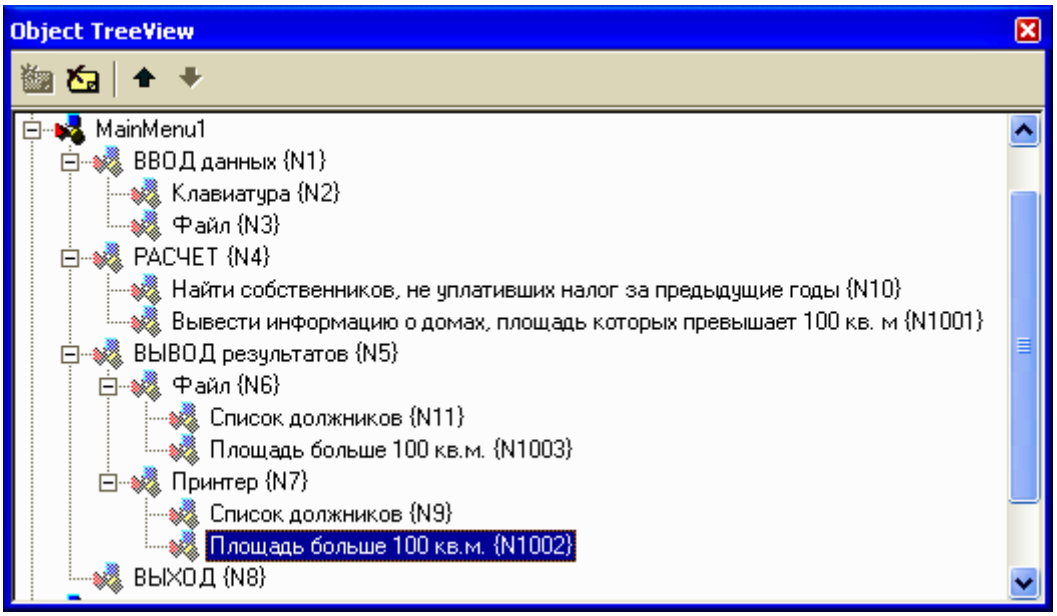


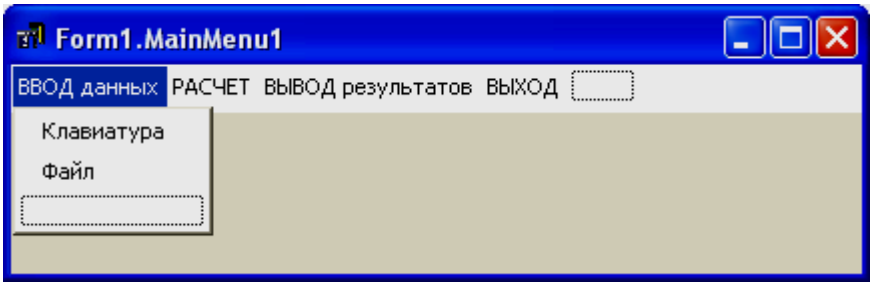
Компоненты

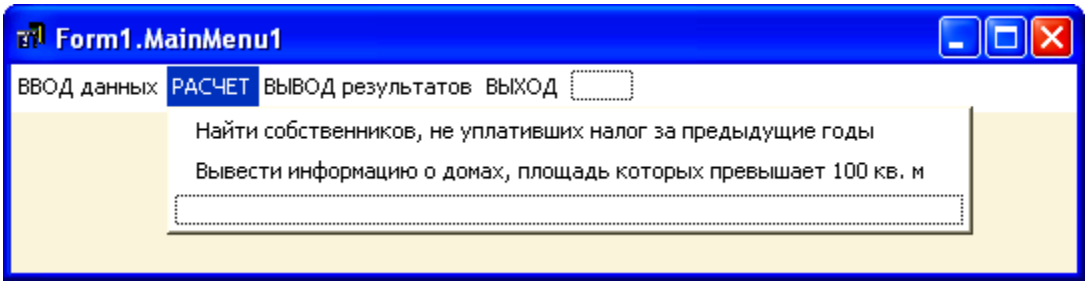
Имя Компонента	Свойства компонента	Значение	Назначение
Form1	Caption	Учет индивидуальных домов	Заголовок главной формы
Label1	Caption	Для перехода в новую ячейку таблицы использовать клавишу «ENTER». При вводе данных в последнюю ячейку строки нажать «ENTER» и клавишу «СТРЕЛКА ВНИЗ» для перехода в другую строку	Справочная информация для пользователя программы
	Visible	False	Строка подсказки невидима
Form1.StringGrid1	ColCount	6	Количество столбцов по умолчанию
	FixedCols	1	Количество фиксированных слева столбцов
	FixedRows	1	Количество фиксированных сверху строк
	Options: goFixedVertLine	 True	Опции: Разметка зафиксированных вертикальных линий
	goFixedHorzLine	True	Разметка зафиксированных горизонтальных линий
	goVertLine	True	Разметка вертикальных линий внутри таблицы
	goHorzLine	True	Разметка горизонтальных линий внутри таблицы
	goEditing	True	Признак допустимости редактирования содержимого ячеек таблицы.True — редактирование разрешено. False — запрещено
	goTabs	True	Разрешает использование клавиши Tab для перемещения в другую ячейку

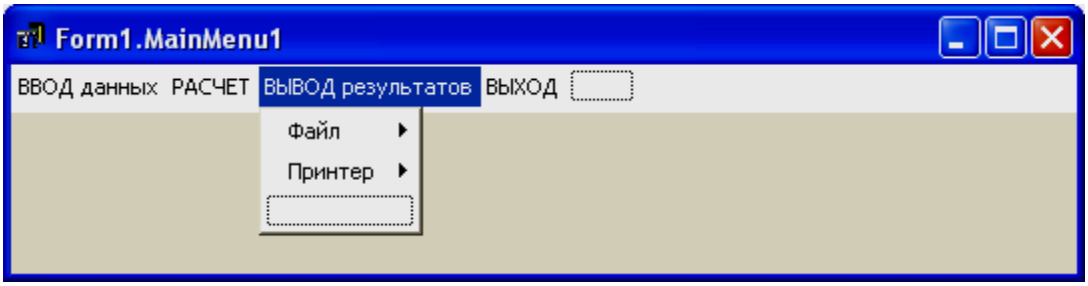
	goAlwaysShowEditor ScrollBars	True ssBoth	Признак нахождения компонента в режиме редактирования Выводятся обе полосы прокрутки
Form1.MainMenu	N1 N2 N3 N4 N5 N6 N7 N8 N10 N1001 N11 N1003 N9 N1002	Ввод данных Клавиатура Файл Расчет Вывод результатов Файл Принтер Вывод: Найти собственников, не уплативших налог за предыдущие годы Вывести информацию о домах, площадь которых превышает 100 кв. м Список должников Площадь больше 100 кв.м Список должников Площадь больше 100 кв.м	Названия пунктов меню
OpenDialog1	Name	Od1	Имя компонента
SaveDialog1	Name	SaveDialog1	Имя компонента
Form2	Caption	Дома с площадью более 100 кв.м	Заголовок формы
Form2.StringGrid1	Свойства аналогичны свойствам Form1.StringGrid1 (см. выше)		
Label1	Caption		Поле вывода результата
Form2.Button1	Caption	Закрыть	Кнопка для закрытия окна “ Дома с площадью более 100 кв.м ”
Form3	Caption	Должники	Заголовок формы
Form3.StringGrid1	Свойства аналогичны свойствам Form1.StringGrid1 (см. выше)		
Label1	Caption	Список собственников, не уплативших налог на недвижимость в предыдущие годы, включая прошлый	Справочная информация для пользователя программы
Label2	Caption		Поле для вывода результата
Form3.Button1	Caption	Закрыть	Кнопка для закрытия окна “Должники”

Для наглядности приведем структуру создаваемого главного меню:









Переменные и массивы

Обозначение в программе	Содержание	Тип
L	количество домов с площадью более 100 кв.м	целый
M	количество должников	целый
Delo_N	массив номеров кадастровых дел	строковый длиной 4
Famil	массив собственников индивидуальных домов	строковый длиной 20
Adres	массив адресов индивидуальных домов	строковый длиной 30
S	массив площадей индивидуальных домов	вещественный
D	массив дат последней уплаты налога на недвижимость	Объект типа дата
n	Счетчик объектов в файле или таблице	целый
F	Файловая переменная	текстовый
NameFile	Имя файла	строковый длиной 20
DA	День уплаты налога	Целый беззнаковый
M	Месяц уплаты налога	Целый беззнаковый
Y	Год уплаты налога	Целый беззнаковый
i	Счетчик строк, управляющая переменная в цикле	Целый
TextToPrint	Файловая переменная для вывода на принтер	Системный текс

Текст модуля

```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Menus, Grids, DateUtils, StdCtrls,Unit2,Unit3,
Printers;
// подключаются модули, связанные с 2-ой и 3-ей формами,
// библиотека функций и процедур работы с принтером
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
```

```
        N8: TMenuItem;
        N10: TMenuItem;
        N1001: TMenuItem;
        StringGrid1: TStringGrid;
        od1: TOpenDialog;
        SaveDialog1: TSaveDialog;
        N9: TMenuItem;
        N1002: TMenuItem;
        N11: TMenuItem;
        N1003: TMenuItem;
        Label1: TLabel;
          procedure N8Click(Sender: TObject);
          procedure N2Click(Sender: TObject);
          procedure N3Click(Sender: TObject);
          procedure FormCreate(Sender: TObject);
          procedure  StringGrid1KeyDown(Sender: TObject; var Key:
Word;
          Shift: TShiftState);
          procedure N10Click(Sender: TObject);
          procedure N1001Click(Sender: TObject);
          procedure N11Click(Sender: TObject);
          procedure N1003Click(Sender: TObject);
          procedure N9Click(Sender: TObject);
          procedure N1002Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
    L:Integer;// количество домов с площадью более 100 кв.м
    M:Integer;// количество должников
    Delo_N:Array[1..1000] of String[4];{ массив номеров кадастро-
вых дел}
    Famil :Array[1..1000] of String[20]; { массив собственников
    индивидуальных домов}
    Adres :Array[1..1000] of String[30];{ массив адресов индивиду-
альных домов}
    S:Array[1..1000] of Real; { массив площадей индивидуальных
домов}
```

```
    D:Array[1..1000] of TDate;{ массив дат последней уплаты налога на недвижимость}
    end;
    var
     Form1: TForm1;
    implementation
    {$R *.dfm}
    // процедура закрывает главное окно – ВЫХОД
    procedure TForm1.N8Click(Sender: TObject);
    begin
    Form1.Close;
    end;
    //активизация ввода с клавиатуры
    procedure TForm1.N2Click(Sender: TObject);
    begin
    N2.Checked := True;// ставится галочка напротив пункта Клавиатура
    Label1.Visible:=True;// поле с подсказкой становится видимым
    StringGrid1.SetFocus;// фокус переносится в таблицу StringGrid
    StringGrid1.Options := StringGrid1.Options + [goEditing];
//в StringGrid добавляется свойство, разрешающее редактирование ячеек
    end;
    // процедура ввода исходных данных из файла на диске
procedure TForm1.N3Click(Sender: TObject);
  Var
    n:integer;// счетчик объектов в файле
    F:TextFile;// файловая переменная текстового типа
    NameFile:String[20];// имя файла
    DA,M,Y:Word;{ день, месяц, год последней уплаты
    налога на недвижимость, в файле эти числа вводятся через пробел }
    begin
    N3.Checked:=True;// ставится галочка напротив пункта «Файл»
    if not od1.Execute then Exit;// если в открывшемся списке
// не выбрано имя файла, то прекращается выполнение процедуры
    NameFile := od1.FileName;{ переменной имя_файла присваивается имя, выбранное из списка}
    AssignFile(F,NameFile);// процедура связывает файловую
    // переменную и конкретный файл
```

```
Reset(F);// процедура открывает файл для чтения
// счетчик объектов инициализируется единицей
n:=1;
{пока не достигнут конец файла, из файла читаются построчно:
номер кадастрового дела, фамилия собственника, адрес дома, площадь, дата в виде трех чисел на одной строке. Данные одновременно выводятся в таблицу в главном диалоговом окне}
While Not(EoF(F)) Do // пока не достигнут конец файла, делать
  Begin
  Readln(F,Delo_N[n]); // из файла читается номер кадастрового дела
  StringGrid1.Cells[1,n]:=Delo_N[n]; { номер кадастрового дела выводится в таблицу на 1 форме}
  Readln(F,Famil[n]); // из файла читается фамилия собственника
  StringGrid1.Cells[2,n]:=Famil[n];// фамилия выводится в форму на экран
  Readln(F,Adres[n]); // читается адрес
  StringGrid1.Cells[3,n]:=Adres[n];// адрес выводится в форму на экран
  Readln(F,S[n]);// читается площадь
  StringGrid1.Cells[4,n]:=FloatToStr(S[n]); {площадь выводится в таблицу на экране}
  Readln(F,DA,M,Y);D[n]:=EncodeDate(Y,M,DA);{преобразование трех чисел в объект типа дата – массив дат}
  StringGrid1.Cells[5,n]:=DateToStr(D[n]); {дата выводится в таблицу}
  n:=n+1;// счетчик объектов увеличивается на 1
  {если количество строк в таблице на форме меньше n, то ему присваивается значение счетчика объектов }
  if StringGrid1.RowCount < n then
  StringGrid1.RowCount := n;
  End;
  {после того как достигнут конец файла, файл закрывается}
CloseFile(F);
end;
{процедура выполняется при появлении диалогового окна (формы) на экране, связана с событием – создание формы}
procedure TForm1.FormCreate(Sender: TObject);
```

```
Var
 n:Integer; // счетчик объектов в таблице
begin
// заполнение шапки таблицы нулевой строки
 StringGrid1.Cells[0,0]:='Номер п/п';
 StringGrid1.Cells[1,0]:='Номер кадастрового дела';
 StringGrid1.Cells[2,0]:='Фамилия И.О. собственника';
 StringGrid1.Cells[3,0]:='Адрес';
 StringGrid1.Cells[4,0]:='Общая площадь (кв.м)';
 StringGrid1.Cells[5,0]:=
 'Дата последней уплаты налога на недвижимость';
 // цикл нумерации строк таблицы
 For n:=1 to 1000 Do
   StringGrid1.Cells[0,n]:=IntToStr(n);
end;
```

{Процедура, связанная с событием, нажатие клавиши в ячейке таблицы.

Проверяется правильность нажатия клавиш при вводе – «ENTER» (для перехода в ячейку справа) и последовательно клавиш «ENTER» и «СТРЕЛКА ВНИЗ» (для перехода на следующую строку). Здесь же происходит считывание данных из ячеек таблицы в массивы. Такая технология ввода позволила организовать ввод для заранее неизвестного количества данных.}

```
procedure TForm1.StringGrid1KeyDown(Sender: TObject; var
Key: Word;
   Shift: TShiftState);
Begin
// если нажата клавиша «СТРЕЛКА ВНИЗ»
 if Key = VK_DOWN then begin
 // то если текущий номер строки таблицы меньше предыдущего
 // значения количества строк в таблице, тогда ВЫХОД
     if StringGrid1.Row < pred(StringGrid1.RowCount) then Exit;
 // количество строк в таблице увеличивается на единицу
     StringGrid1.RowCount := StringGrid1.RowCount + 1;
     StringGrid1.Col := 1; // переход в начало строки
  // если в первую ячейку текущей строки ничего не введено,
  // то добавленная строка убирается
if (Length(StringGrid1.Cells[1,StringGrid1.Row]) = 0) then begin
```

```
StringGrid1.RowCount:=pred(StringGrid1.RowCount);Exit;
end;
end;
// если нажата клавиша «ENTER», то
if (Key = VK_RETURN) then begin
case StringGrid1.Col of
// если номер столбца таблицы равен 1 и если ячейка не пуста,
1 : if (Length(StringGrid1.Cells[1,StringGrid1.Row]) > 0)
// тогда содержимое ячейки с индексом равным номеру
// текущей строки помещается в массив «Номер кадастрового дела»
            then
            Delo_N[StringGrid1.Row] :=
StringGrid1.Cells[1,StringGrid1.Row]
// иначе количеству строк в таблице
// присваивается предыдущее значение и выход, т.е. программа будет
// ждать заполнения первой колонки в этой строке
            else begin
               StringGrid1.RowCount:=pred(StringGrid1.RowCount);
               Exit;
                 end;
// если текущий номер колонки таблице равен 2, то в массив
// собственников под номером равным номеру текущей строки
// таблицы, помещается содержимое ячейки, стоящей на пересечении
// второго столбца и текущей строки
          2 : Famil[StringGrid1.Row] :=
StringGrid1.Cells[2,StringGrid1.Row];
// аналогично предыдущему вводится адрес
          3 : Adres[StringGrid1.Row] :=
StringGrid1.Cells[3,StringGrid1.Row];
// если четвертая колонка не пуста, то строка символов преобразуется
// в вещественное число и помещается в массив площадей домов
          4 : if Length(StringGrid1.Cells[4,StringGrid1.Row]) > 0 then
    S[StringGrid1.Row]:=StrToFloat(StringGrid1.Cells[4,StringGrid1.
Row])
// иначе в массив вводится число 0.00
             else
             S[StringGrid1.Row] := 0.00;
   // если пятая колонка не пуста, то строка символов преобразуется
```

```
    // в дату и помещается в массив дат
          5 : if Length(StringGrid1.Cells[5,StringGrid1.Row]) > 0 then
     D[StringGrid1.Row] :=
StrToDate(StringGrid1.Cells[5,StringGrid1.Row])
    // иначе в массив дат помещается 0
              else
              D[StringGrid1.Row] := 0;
    end;
    // если текущий номер колонки меньше 5, то он увеличивается на 1
      if StringGrid1.Col < 5 then
        StringGrid1.Col := StringGrid1.Col +1
      end;

    end;
    { процедура, связанная с пунктом меню «Найти собственни-
ков, не уплативших налог за предыдущие годы»}
    procedure TForm1.N10Click(Sender: TObject);
    Var
    i,n:integer;
    begin
    // ставится галочка напротив пункта меню
    // «Найти собственников, не уплативших налог за предыдущие годы»
    N10.Checked:=True;
    // подписывается шапка таблицы
    Form3.StringGrid1.Cells[0,0]:='Номер по п/п';
    Form3.StringGrid1.Cells[1,0]:='Фамилия И.О. собственника';
Form3.StringGrid1.Cells[2,0]:=
'Дата последней уплаты налога на недвижимость';
    i:=0;// – номер строки в таблице
{ В цикле последняя дата уплаты налога собственником дома
сравнивается с текущей датой, установленной на компьютере.
Если разница превышает 1 год, собственник является должником.
Фамилии должников выводятся в таблицу в третьей форме. Если
дата неизвестна, то выводится сообщение: «нет сведений»}
    For n := 1 to pred(StringGrid1.RowCount) Do
    // функция YearsBetween(D1,D2) находит разницу между двумя датами
      IF YearsBetween(Date,d[n]) >= 1 then Begin
      Inc(i); // i:=i+1;
```

```
      Form3.StringGrid1.Cells[0,i]:=IntToStr(i);
      Form3.StringGrid1.Cells[1,i]:=Famil[n];
      if D[n]=0 then Form3.StringGrid1.Cells[2,i]:='нет сведений'
      else
          Form3.StringGrid1.Cells[2,i]:=DateToStr(D[n]);
    Form3.StringGrid1.RowCount:=Form3.StringGrid1.RowCount+1
                                                  end;
      // вывод количества должников
    M:=i;
    Form3.Label2.Caption:='Всего: '+ IntToStr(M)+' должников';
    //показать форму 3
    Form3.Show;
    end;
    {процедура, связанная с пунктом меню «Вывести информа-
цию о домах, площадь которых превышает 100 кв.м»}
    procedure TForm1.N1001Click(Sender: TObject);
    Var n,i:Integer;
    Begin
    // поставить галочку напротив выбранного пункта меню
      N1001.Checked:=True;
      Form2.StringGrid1.Cells[0,0]:='Номер по п/п';
      Form2.StringGrid1.Cells[1,0]:='Номер кадастрового дела';
      Form2.StringGrid1.Cells[2,0]:='Фамилия И.О. собственника';
      Form2.StringGrid1.Cells[3,0]:='Адрес';
      Form2.StringGrid1.Cells[4,0]:='Общая площадь (кв.м)';
Form2.StringGrid1.Cells[5,0]:=
'Дата последней уплаты налога на недвижимость';
    i:=0;
    { В цикле производится сравнение площадей всех домов со
      100 кв.м. Если площадь дома превышает 100 кв.м, то вся
      информация о нем выводится в таблицу на форме 2}
    For n:=1 to pred(Form1.StringGrid1.RowCount) Do
    IF S[n]>100 Then Begin
      Inc(i); // i:=i+1;
      Form2.StringGrid1.Cells[0,i]:=IntToStr(i);
      Form2.StringGrid1.Cells[1,i]:=Delo_N[n];
      Form2.StringGrid1.Cells[2,i]:=Famil[n];
      Form2.StringGrid1.Cells[3,i]:=Adres[n];
```

```
        Form2.StringGrid1.Cells[4,i]:=FloatToStrF(S[n],ffFixed,10,2)+
        ' кв.м';
        if D[n]=0 then Form2.StringGrid1.Cells[5,i]:='нет сведений'
        else
        Form2.StringGrid1.Cells[5,i]:=DateToStr(D[n]);
      Form2.StringGrid1.RowCount:=Form2.StringGrid1.RowCount+1
                          end;
        L:=i;
Form2.Label1.Caption:='Всего домов площадью более 100 кв.м:'
+IntToStr(L);
    Form2.Show;
    end;
    {процедура, связанная с пунктом меню «Вывод в файл» –
список должников}
    procedure TForm1.N11Click(Sender: TObject);
    Var f:TextFile;
        NameFile:String;
        i:Integer;
    begin
    N11.Checked:=True;
{Если из панели диалога не выбрано или не задано имя файла со-
хранения, то выход. Т.е. панель диалога будет находится на экра-
не, пока пользователь не задаст имя файла.}
    if not SaveDialog1.Execute then Exit;
// переменной ИмяФайла присваивается имя файла,
// заданное в панели диалога
    NameFile:=SaveDialog1.FileName;
// устанавливается физическая связь между файловой переменной
// и конкретным файлом на внешнем носителе
    AssignFile(f,NameFile);
// открыть файл для записи
    Rewrite(f);
// в следующем фрагменте программы информация о должниках
// из формы 3 записывается в файл
    Writeln (f,'список должников:');
    Writeln (f,'Номер п/п ','Фамилия И.О. собственника',
    ' Дата последней уплаты налога на недвижимость');
    For i:=1 To M do
```

```
  Writeln
(f,Form3.StringGrid1.Cells[0,i]:4,Form3.StringGrid1.Cells[1,i]:30,' ',
     Form3.StringGrid1.Cells[2,i]:30);
// закрытие файла
     CloseFile(f);
     end;

     {процедура, связанная с пунктом меню «Вывод в файл» –
список домов с площадью более 100 кв.м}
     procedure TForm1.N1003Click(Sender: TObject);
     Var f:TextFile;
          NameFile:String;
          i:Integer;
     begin
     N1003.Checked:=True;
// если не задано имя файла в панели диалога,
// панель диалога остается на экране
     if not SaveDialog1.Execute then Exit;
// переменной ИмяФайла присваивается имя файла,
// заданное в панели диалога
     NameFile:=SaveDialog1.FileName;
// устанавливается физическая связь между файловой переменной
// и конкретным файлом на внешнем носителе
     AssignFile(f,NameFile);
// открыть файл для записи
     Rewrite(f);
// в следующем фрагменте программы информация о домах с площадью
// более 100 кв.м из формы 2 записывается в файл
     Writeln (f,'Дома с площадью более 100 кв. м:');
Writeln (f,'Номер п/п ','Номер кадастрового дела ',
'Фамилия И.О. собственника  ', '     Адрес     ',
'Общая площадь (кв.м) ', 'Дата последней уплаты налога на недвижимость');
  For i:=1 To L do
  Writeln
(f,Form2.StringGrid1.Cells[0,i]:4,Form2.StringGrid1.Cells[1,i]:20,' ',
     Form2.StringGrid1.Cells[2,i]:30,
     ' ',Form2.StringGrid1.Cells[3,i]:30,
     Form2.StringGrid1.Cells[4,i]:16,
```

```
    ' ',Form2.StringGrid1.Cells[5,i]:20 );
    CloseFile(f);
    end;

    {процедура, связанная с пунктом меню «Вывод на принтер» –
список должников}
    procedure TForm1.N9Click(Sender: TObject);
    Var TextToPrint:System.Text;
          i:integer;
    begin
    N9.Checked:=True;
// устанавливается физическая связь с печатающим устройством
    AssignPrn(TextToPrint);
    // открытие для вывода
    Rewrite(TextToPrint);
// в следующем фрагменте осуществляется вывод списка должников
// из формы 3 на принтер
    Writeln (TextToPrint,'список должников:');
    Writeln (TextToPrint,'Номер п/п ','Фамилия И.О.',
    ' Дата последней уплаты ');
    Writeln (TextToPrint,'      ','собсвенника',
    ' налога на недвижимость ');
    For i:=1 To M do
    Writeln (TextToPrint,Form3.StringGrid1.Cells[0,i]:4,
    Form3.StringGrid1.Cells[1,i]:30,
    ' ',Form3.StringGrid1.Cells[2,i]:30);
    System.Close(TextToPrint);
    end;

    {процедура, связанная с пунктом меню «Вывод на принтер» –
список домов с площадью более 100 кв.м}
    procedure TForm1.N1002Click(Sender: TObject);
    Var TextToPrint:System.Text;
        i:integer;
    begin
    N1002.Checked:=True;
    // ассоциация с принтером
    AssignPrn(TextToPrint);
```

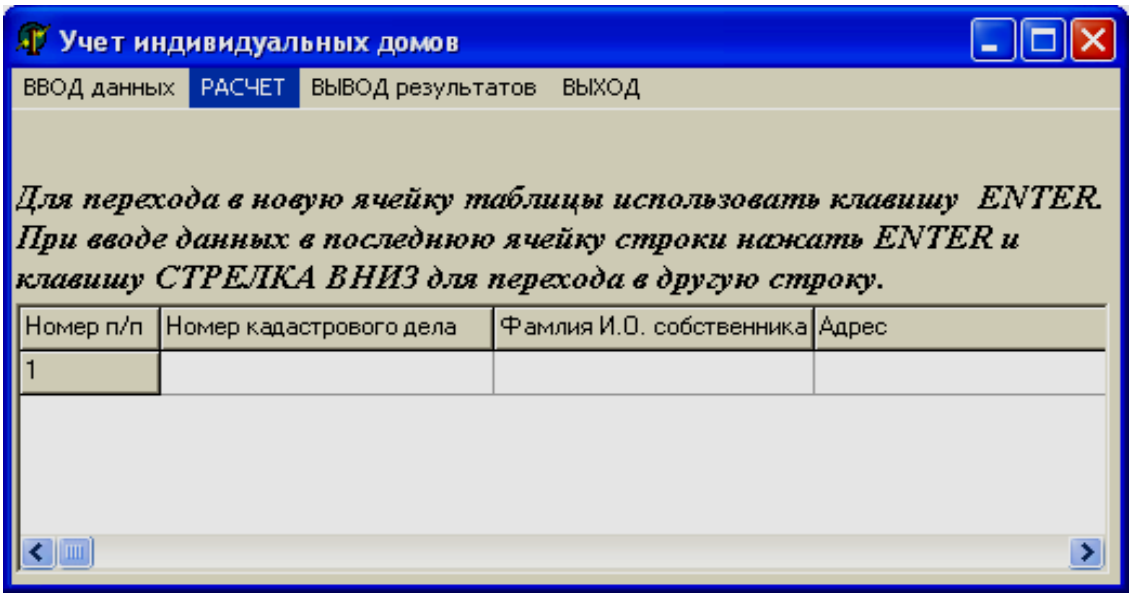
```
    // открытие для вывода
    Rewrite(TextToPrint);
// в следующем фрагменте осуществляется вывод списка
// домов с площадью более 100 кв.м из формы 2 на принтер
    Writeln (TextToPrint,'Дома с площадью более 100 кв. м:');
    Writeln (TextToPrint,'N п/п','Номер кад. дела ',
    'Фамилия И.О.собств.','     Адрес     ',
'Общ. площ.(кв.м)','Дата посл. Уплаты налога на недвижимость');
    For i:=1 To L do
    begin
      Writeln (TextToPrint,Form2.StringGrid1.Cells[0,i]:4,
      Form2.StringGrid1.Cells[1,i]:20,' ',
      Form2.StringGrid1.Cells[2,i]:20,' ',
      Form2.StringGrid1.Cells[3,i]:40,
      Form2.StringGrid1.Cells[4,i]:16,' ',
      Form2.StringGrid1.Cells[5,i]:20 );
    end;
    System.Close(TextToPrint);
    end;

    end.
    {модуль Unit2 связан с формой Form2, куда выводится список
домов с площадью превышающей 100 кв.м}
    unit Unit2;
    interface
    uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Con-
trols, Forms, Dialogs, Grids, StdCtrls;
    type
      TForm2 = class(TForm)
        StringGrid1: TStringGrid;
        Button1: TButton;
        Label1: TLabel;
        procedure Button1Click(Sender: TObject);
    private
      { Private declarations }
    public
        { Public declarations }
```

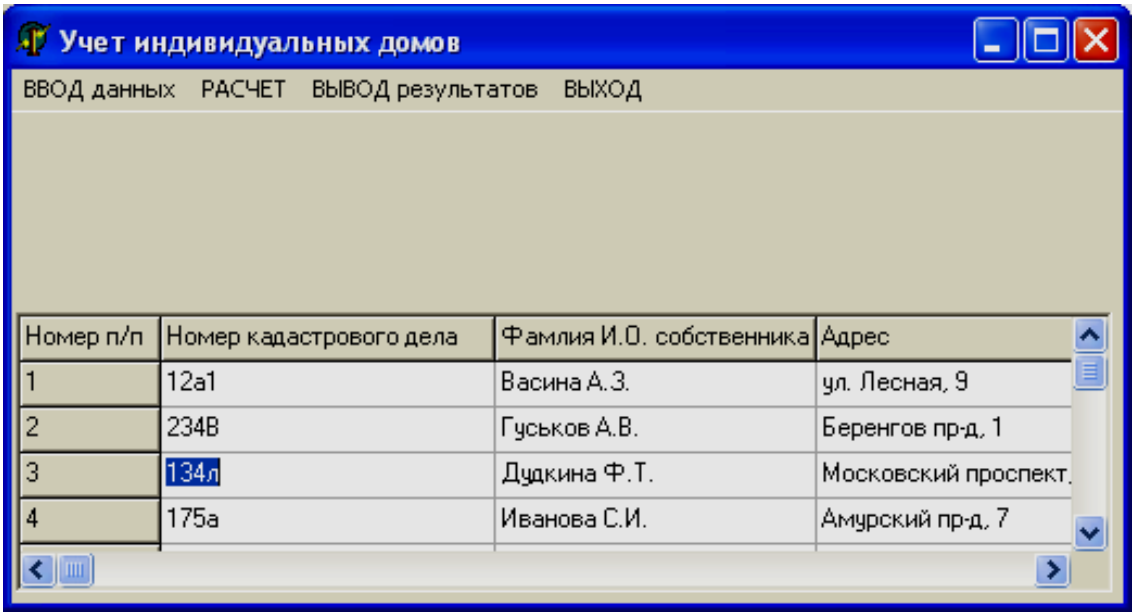
```
end;
var
Form2: TForm2;
implementation
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close
end;
end.
```

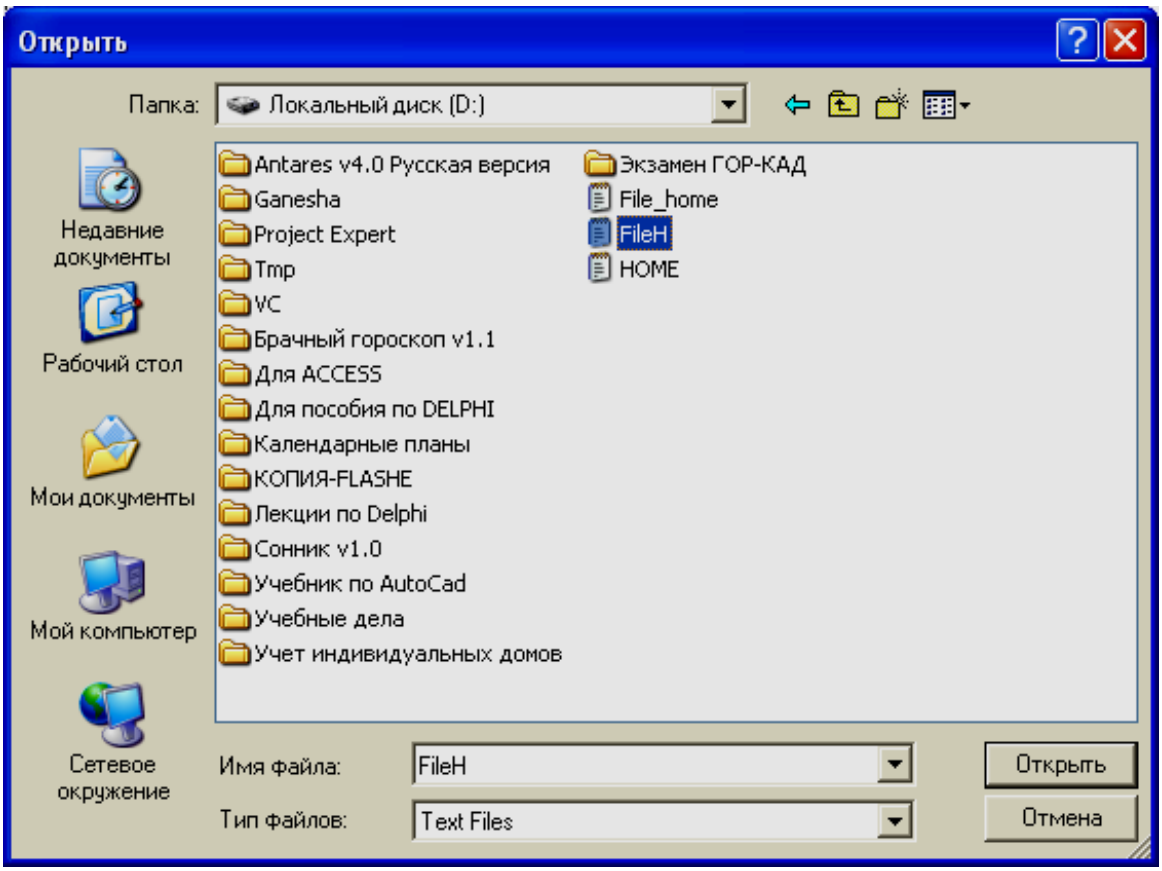
{модуль Unit3 связан с формой Form3, куда выводится список должников по уплате налога на недвижимость}

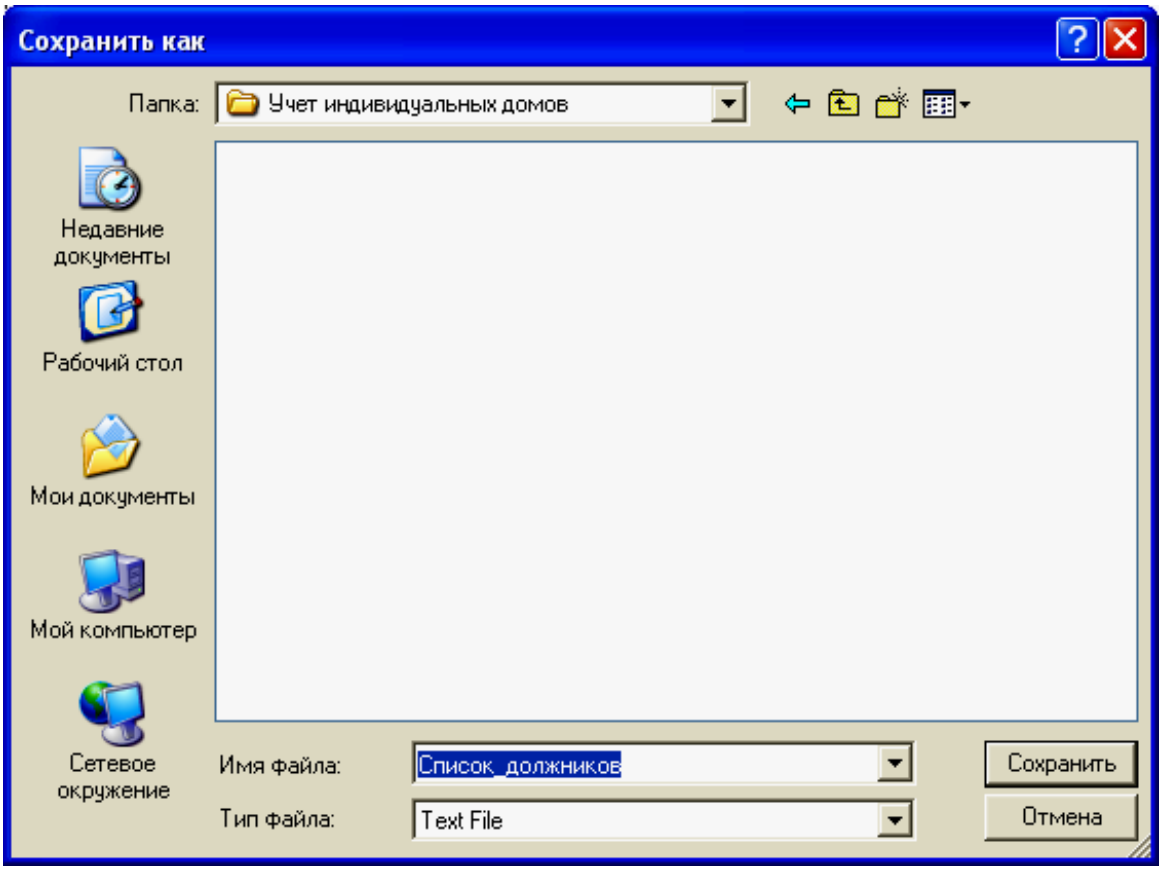
```
unit Unit3;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Grids;
type
  TForm3 = class(TForm)
    Label1: TLabel;
    StringGrid1: TStringGrid;
    Button1: TButton;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form3: TForm3;
implementation
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Close
end;
end.
```

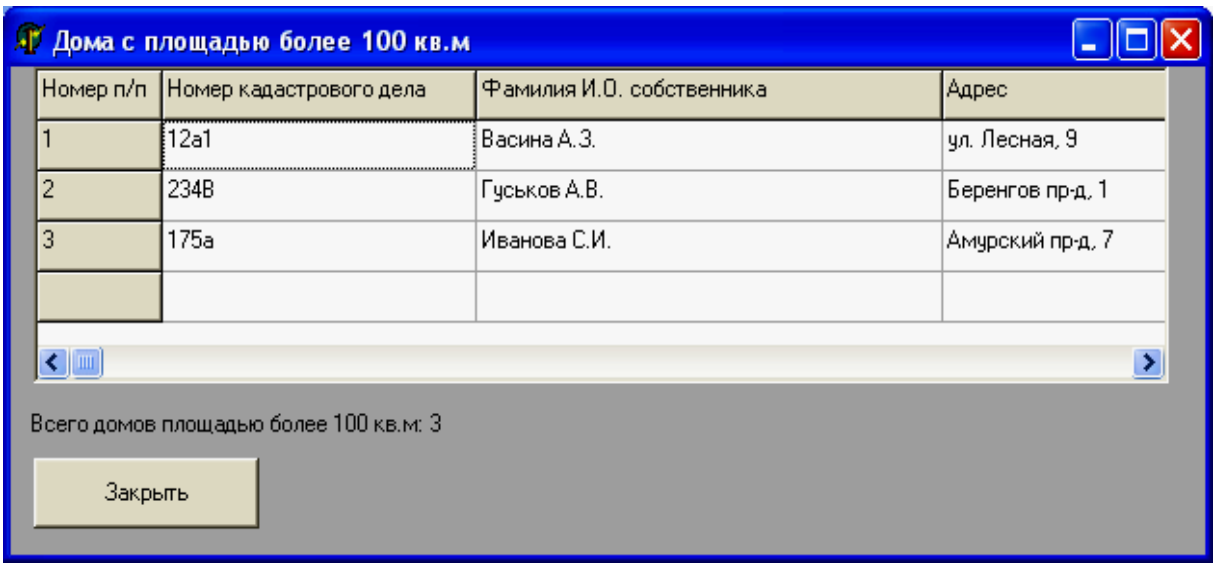
Вид некоторых диалоговых окон во время работы программы



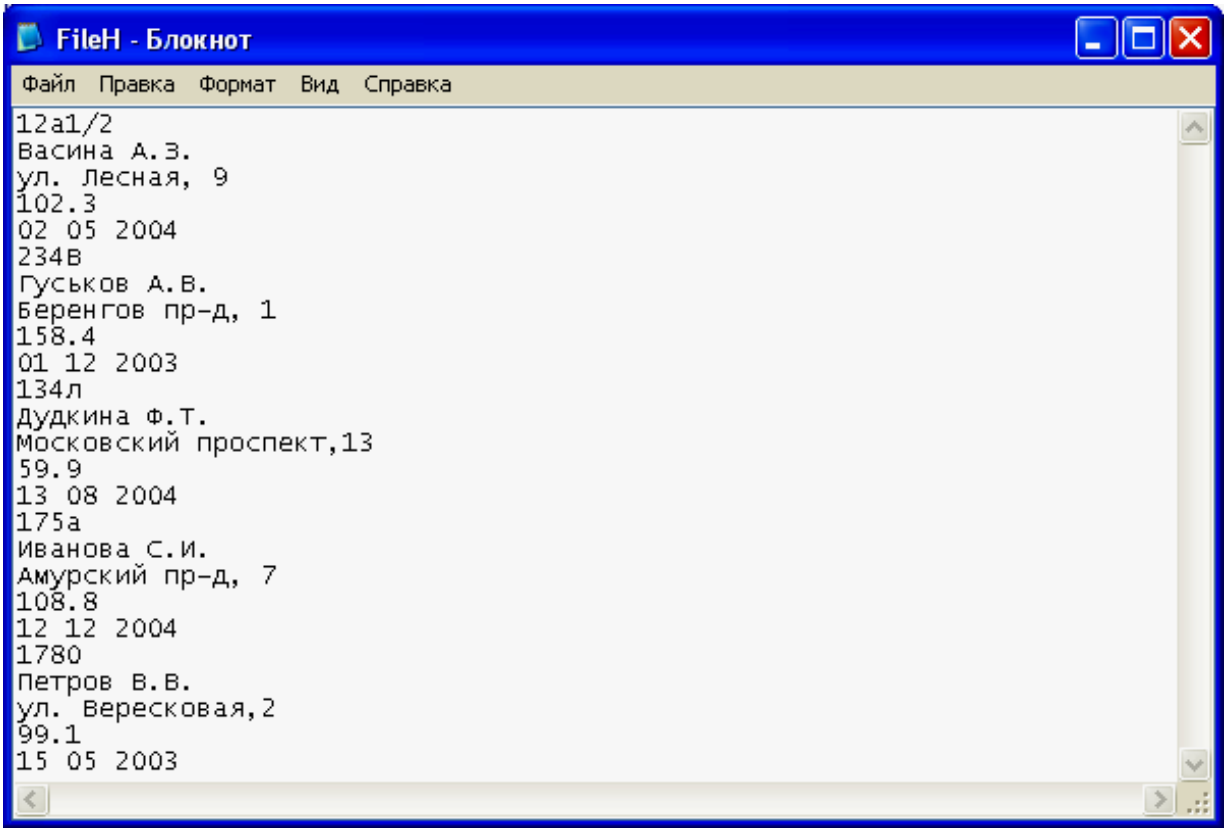



Открыть
Папка:
Локальный диск (D:)
Antares v4.0 Русская версия
Ganesha
Project Expert
Tmp
VC
Брачный гороскоп v1.1
Для ACCESS
Для пособия по DELPHI
Календарные планы
КОПИЯ-FLASHE
Лекции по Delphi
Сонник v1.0
Учебник по AutoCad
Учебные дела
Учет индивидуальных домов
Экзамен ГОР-КАД
File_home
FileH
HOME
Недавние документы
Рабочий стол
Мои документы
Мой компьютер
Сетевое окружение
Имя файла:
FileH
Тип файлов:
Text Files
Открыть
Отмена


Сохранить как
Папка:
Учет индивидуальных домов
Недавние документы
Рабочий стол
Мои документы
Мой компьютер
Сетевое окружение
Имя файла:
Список_должников
Тип файла:
Text File
Сохранить
Отмена



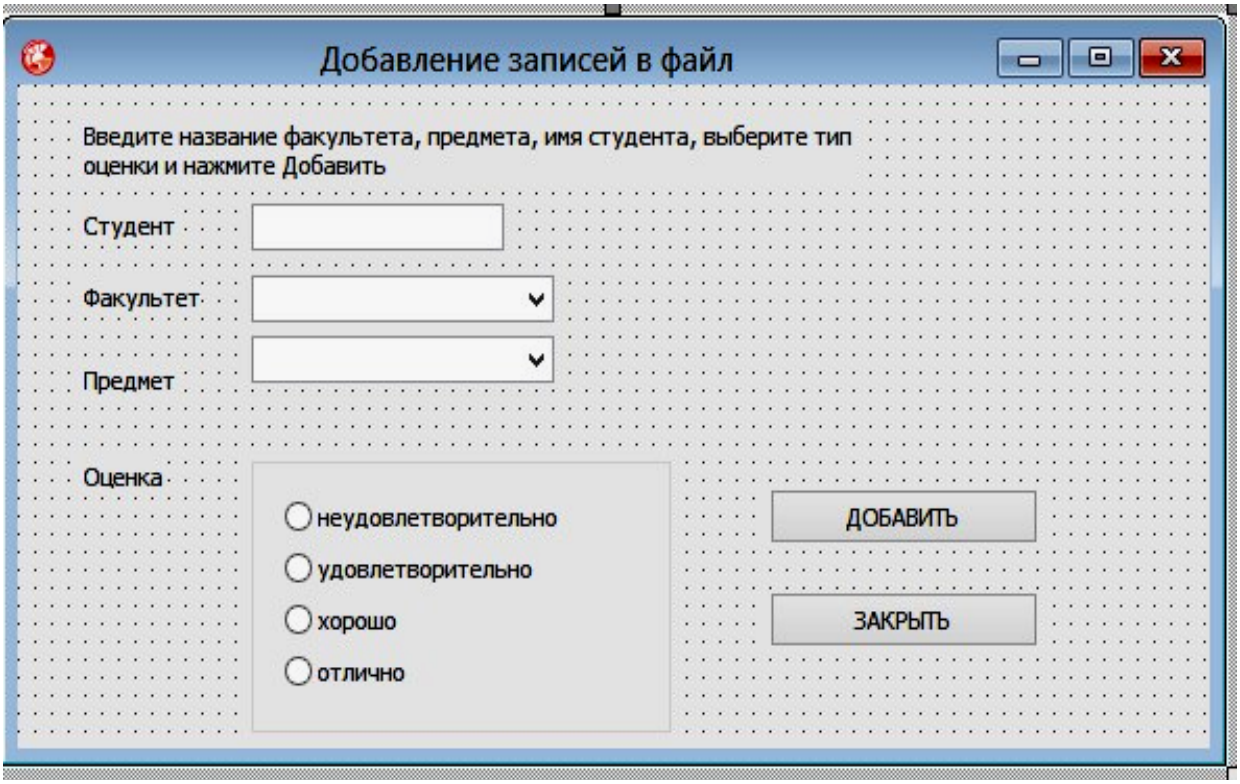
Файл с исходной информацией удобнее подготовить с помощью стандартной программы «Блокнот». Каждое поле записи набирается на отдельной строке. Десятичный разделитель – точка. Дата уплаты налога в файле набирается в виде трех чисел на одной строке через пробел. Для приведенного выше примера файл с исходными данными выглядит так:



Задача 9с

Составить программу, которая записывает в файл введенные пользователем данные о результатах экзаменов, формируя таким образом простую базу данных. Исходные данные вводятся в поля диалогового окна и сохраняются в файле, компонентами которого являются записи типа TOzenka.

Проект формы



Компоненты

Имя Компонента	Свойства компонента	Значение	Назначение
Form1	Caption	Добавление записей в файл	Заголовок формы
Label1	Caption WordWrap	Введите название факультета, предмета, имя студента, выберите тип оценки и нажмите Добавить True	Справочная информация для пользователя программы Перенос не уместившихся слов на новую строку
Label2	Caption	Студент	Подсказка пользователю
Edit1	Text	Очищено от значений по умолчанию	Поле для ввода ФИО студента
Label3	Caption	Факультет	Подсказка пользователю
Label4	Caption	Предмет	Подсказка пользователю
Label5	Caption	Оценка	Подсказка пользователю
ComboBox1	Text Items	Очищено от значений по умолчанию Архитектура Городской кадастр Землеустройство Кадастр недвижимости Юриспруденция	Список для выбора факультета
ComboBox2	Text Items	Очищено от значений по умолчанию Высшая математика Геодезия Информатика История	Список для выбора предмета
RadioGroup1	Caption Items	Должно быть очищено неудовлетворительно удовлетворительно хорошо отлично	Группа переключателей для выбора оценки Названия оценок
Button1	Caption Enabled	ДОБАВИТЬ False	Запуск добавления записи в файл Первоначально кнопка недоступна, чтобы не было попытки добавить запись в несуществующий файл
Button2	Caption	ЗАВЕРШИТЬ	Закрытие программы

Замечание. Чтобы создать события FormActivate и FormClose (соответственно и процедуры для них), нужно для объекта Form1 перейти на вкладку События (Events) в инспекторе объектов и в поле OnActivate два раза щелкнуть мышью. Создастся событие FormActivate и заготовка процедуры для этого события в редакторе кода. Аналогично можно создать событие FormClose. Два раза щелкнуть по событию OnClose.

Текст модуля

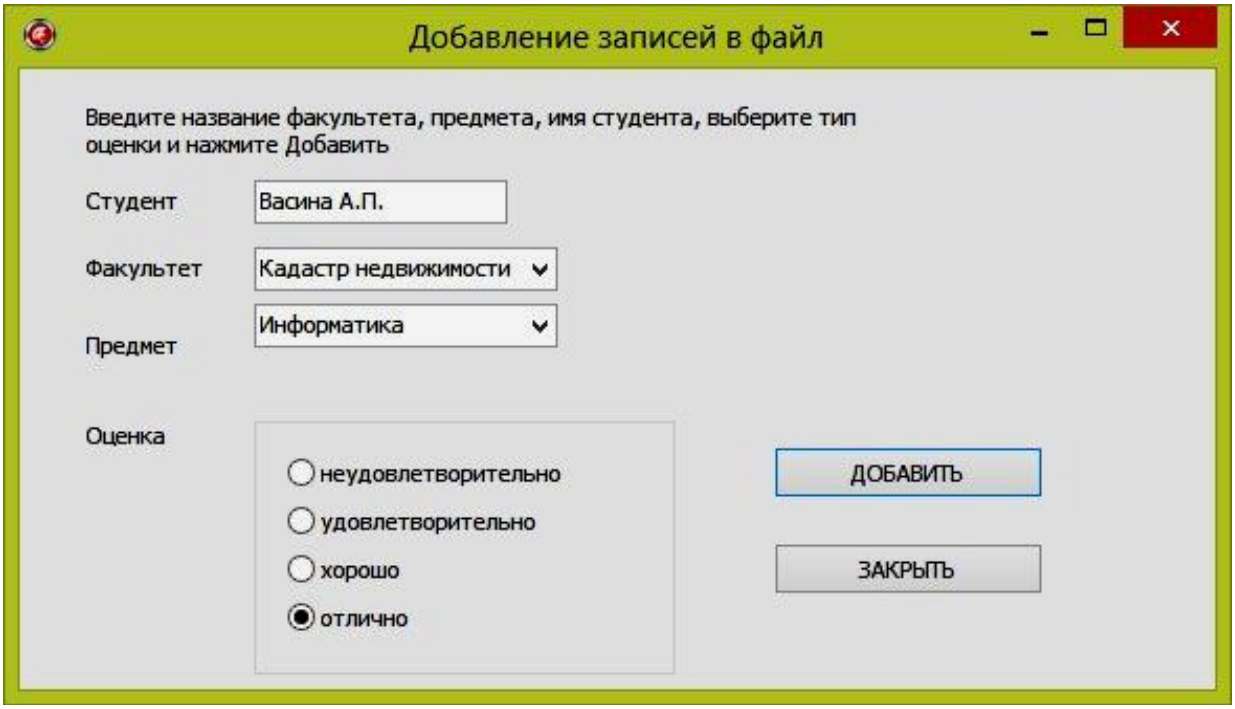
```
unit Unit1;
interface

uses
 Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
 Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls;
type
 TForm1 = class(TForm)
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
  Edit1: TEdit;
  ComboBox1: TComboBox;
  ComboBox2: TComboBox;
  Button1: TButton;
  Button2: TButton;
  RadioGroup1: TRadioGroup;
   procedure FormActivate(Sender: TObject);
   procedure FormClose(Sender: TObject; var Action: TCloseAction);
   procedure Button2Click(Sender: TObject);
   procedure Button1Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
```

```
 end;
Type
 // тип оценки
  TKind= (NeUd, Ud, Hor,Otl);

 // запись файла
   TOzenka=record
    Facultet: string[20]; // факультет
    Predmet:  string[20]; // предмет
    person: string[40]; // студент
    Oz:  TKind;    // оценка
   end;
var
 Form1: TForm1;
 f: file of TOzenka; // файл записей – база данных
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
 Ozenka: TOzenka;
 oz :TKind;
begin
 with Ozenka do
 begin
   Facultet := ComboBox1.Text;
   Predmet:= ComboBox2.Text;
   person := Edit1.Text;
   case RadioGroup1.ItemIndex of
    0: oz := NeUd;
    1: oz:= Ud;
    2: oz := Hor;
    3: oz := Otl;
   end;
 end;
 write(f,Ozenka ); // записать содержимое полей записи в файл
end;
```

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

procedure TForm1.FormActivate(Sender: TObject);
var
 resp : word; // ответ пользователя
begin
{ Если на вашем компьютере нет диска D, вместо него укажите другой имеющийся,
 кроме диска C:, который является системным}
 AssignFile(f,'D:\Ozenki.db');
 {$I-}// отключение сообщений компилятора об ошибках I/O
 Reset(f); // открыть файл
 Seek( f, FileSize(f)); // указатель записи в конец файла
 {$I+}// включение сообщений компилятора об ошибках I/O
 if IOResult = 0
  then button1.enabled:=TRUE // теперь кнопка «Добавить» //доступна
  else
   begin
    resp:=MessageDlg('Файл базы данных не найден.'+
' создать новую БД?',mtInformation,[mbYes,mbNo],0);
    if resp = mrYes then
     begin
      {$I-}
      rewrite(f);
      {$I+}
      if IOResult = 0
       then button1.enabled:=TRUE
       else ShowMessage('Ошибка создания БД.');
     end;
   end;
end;
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
```

```
  CloseFile( f ); // закрыть файл
end;
end.
```

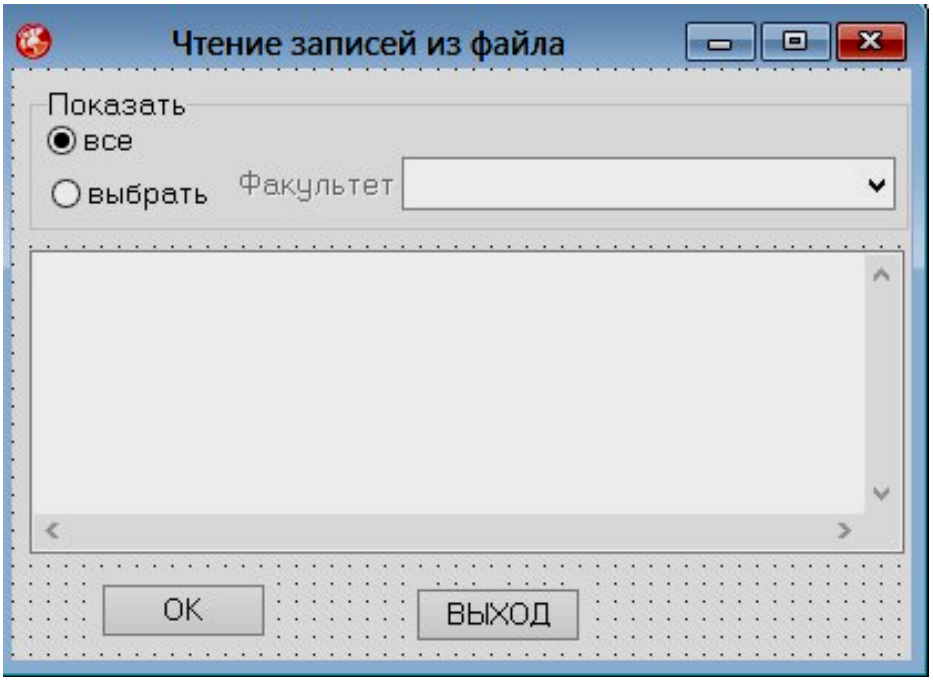
Так выглядит форма в процессе выполнения программы:



Задача 9d

Составить программу, которая читает и обрабатывает данные, записанные в файл программой *Добавление записи в файл*, и, в зависимости от того, какой из переключателей – *все* или – *выбрать* выбран, выводит список оценок, полученных студентами всех факультетов или факультета, название которого введено в поле «Факультет».

Проект формы



Компоненты

Имя Компонента	Свойства компонента	Значение	Назначение
Form1	Caption	Чтение записей из файла	Заголовок формы
GroupBox1	Caption	Показать	Название группы переключателей
RadioButton1	Caption Checked	все True	Подсказка пользователю Включено по умолчанию
RadioButton2	Caption Checked	выбрать True	Подсказка пользователю Выключено по умолчанию
Label1	Caption Enabled	Факультет False	Подсказка пользователю По умолчанию недоступно (пока не нажат переключатель Факультет при работе программы)
Combobox1	Text Items	Очищено от значения по умолчанию Архитектура Городской кадастр Кадастр недвижимости Землеустройство Юриспруденция	Список факультетов Названия факультетов
Memo1	Lines	Очищено от значений по умолчанию	Поле для вывода результатов
Button1	Caption	ОК	Запуск чтения
Button2	Caption	ВЫХОД	Закрытие программы

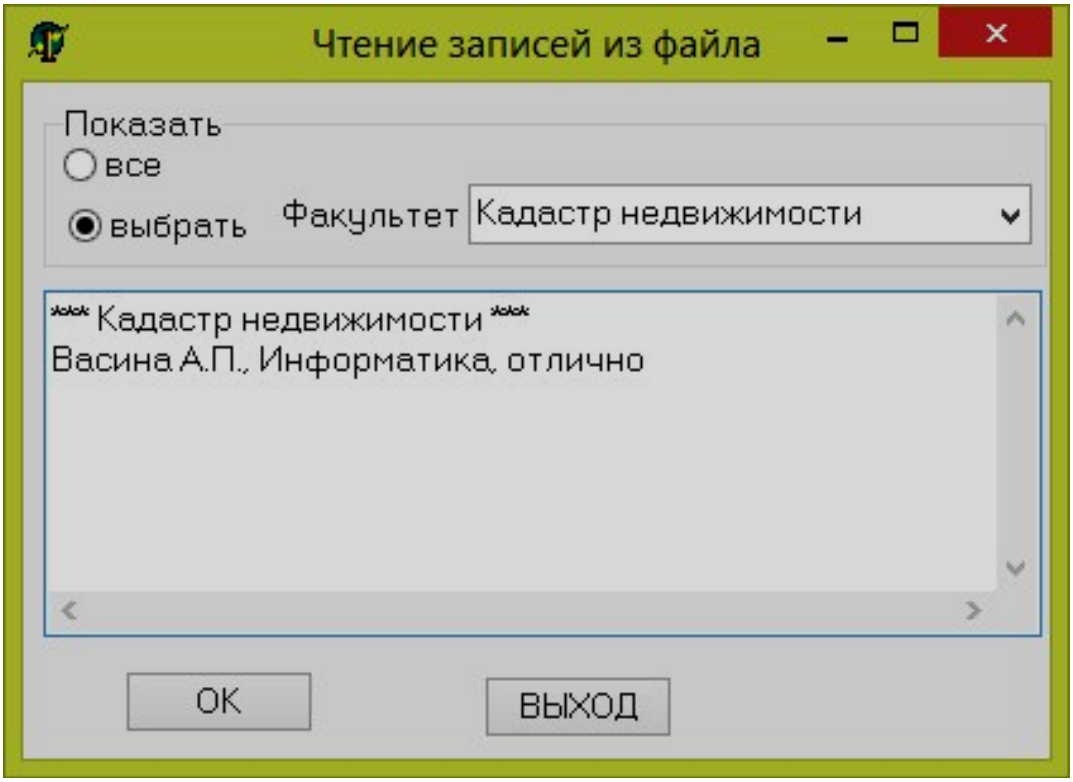
Текст модуля

```
unit rdrec_;

interface

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

type
 TForm1 = class(TForm)
  RadioButton1: TRadioButton; // переключатель Все
  RadioButton2: TRadioButton; // переключатель Выбрать
  ComboBox1: TComboBox;    // комбинированный список
             // для ввода названия страны
  Memo1: TMemo;        // поле вывода записей, удовлетворяющих
             // условию запроса
  Button1: TButton;      // кнопка OK
  GroupBox1: TGroupBox;
  Button2: TButton;
  Label1: TLabel;       // текст Факультет
  procedure Button1Click(Sender: TObject);
  procedure RadioButton2Click(Sender: TObject);
  procedure RadioButton1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}
```

```
procedure TForm1.Button1Click(Sender: TObject);
type
  // тип оценки
  TKind = (NeUd,Ud,Hor,Otl);

  // запись файла
  TOzenka = record
    Facultet:string[20];
    Predmet:string[20];
    person:string[40];
    kind:TKind;
  end;

var
  f: file of TOzenka; // файл записей
  rec: TOzenka;    // запись, прочитанная из файла
  n: integer;     // кол-во записей удовлетворяющих запросу
  st: string[80];

begin
{Должен быть указан диск (кроме диска C, он системный),
где находится ранее созданный файл}
 AssignFile(f,'D:\ozenki.db');
 {$I-}
 Reset(f); // открыть файл для чтения
 {$I-}
 if IOResult <> 0 then
  begin
   ShowMessage('Ошибка открытия файла БД.');
   Exit;
  end;
   Memo1.Clear; // очистить список поля Memo
 // обработка БД
 if RadioButton2.Checked then
    Memo1.Lines.Add('*** ' + ComboBox1.Text + ' ***');

 n := 0;
   while not EOF(f) do
```

```
  begin
    read(f, rec); // прочитать запись
    if RadioButton1.Checked or
      (rec.facultet = ComboBox1.Text) then
    begin
      n := n + 1;
      st := rec.person+ ', ' + rec.predmet;
      if RadioButton1.Checked then
        st := st + ', '+ rec.facultet;
      case rec.kind of
        NeUd: st := st+ ', неудовлетворительно ';
        Ud:  st := st+ ', удовлетворительно';
        Hor:  st := st+ ', хорошо';
        Otl:  st := st+ ', отлично';
      end;
      Memo1.Lines.Add(st);
    end;
  end;
  CloseFile(f);
  if n = 0 then
    ShowMessage('В БД нет запрашиваемой информации.');
end;

// переключатель Выбрать
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
   Label1.Enabled := True;
   ComboBox1.Enabled := True;  { теперь поле Факультет доступно}
   ComboBox1.SetFocus; // курсор в поле Факультет
end;

// переключатель Все
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
   Label1.Enabled := False;
   ComboBox1.Enabled := False; { теперь поле Факультет не дос-
тупно}
end;
```

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

end.
```



Вопросы для самоконтроля

1. Дайте определение файлу.
2. Каких типов бывают файлы?
3. Для чего служит файловая переменная?
4. Что представляет собой текстовый файл?
5. Назначение и формат процедуры AssignFile.
6. Назначение и формат процедуры Reset.
7. Назначение и формат процедуры Rewrite.
8. Назначение и формат процедуры Read.
9. Назначение и формат процедуры Readln.
10. Назначение и формат процедуры Write.
11. Назначение и формат процедуры Writeln.

12. Назначение и формат процедуры CloseFile.

13. Назначение и формат процедуры Seek.

14. Какие вы знаете функции для работы с файлами?

15. Какой компонент Delphi предназначен для создания главного меню и как его использовать?

16. Какой компонент Delphi предназначен для создания окна выбора открываемого файла и как его использовать?

17. Какой компонент Delphi предназначен для создания окна выбора файла для сохранения информации и как его использовать?

18. С какой целью программисты создают в программах структуру данных − запись?

19. Опишите методику использования структуры данных − запись.

20. Назначение оператора With.

21. Покажите фрагмент программы «Должники» (Задача 9b), где происходит чтение (ввод) исходных данных из файла.

22. Покажите фрагмент программы «Должники» (Задача 9b), где производится запись (вывод) результатов в файл.

23. Что произойдет в программе «Чтение записей из файла», если файл с указанным именем не будет найден?

24. Для чего в программе «Чтение записей из файла» применяется функция EOF?

25. Что произойдет при работе программы «Добавление записей в файл», если файл с указанным именем не будет найден?

Задачи для самостоятельного решения

№ 9.1 Подсчитать значение Q, вычисляемое по формуле

$$Q = \frac{\sum_{i=1}^{N} Y_i}{N!}$$

где Y_i – элементы массива Y с размерностью N. Массив Y ввести из файла на диске. Вывод обеспечить: 1) на экран; 2) в файл на диске.

№ 9.2 Дан массив X, содержащий вещественные числа. Найти и записать на место X(1) – наибольший элемент массива X, а на место X(N) – наименьший элемент массива (X1,X2,...,XN). Ввод исходного массива X осуществить из файла на диске. Вывод обеспечить:1) на экран; 2) в файл на диске.

№ 9.3 Дан массив Q(N), состоящий из отрицательных и положительных чисел. Составить из Q два массива: A, содержащий только положительные числа; B, содержащий только отрицательные. Массив Q вводится из файла на диске. Вывести исходный массив и массивы, полученные в процессе преобразования. Вывод обеспечить:1) на экран; 2) в файл на диске.

№ 9.4 Даны натуральное число N, действительные числа

X1,X2,..., Xn. Получить (1+r)/(1+s),где r – сумма всех тех членов последовательности X1, X2,..., Xn, которые не превосходят 1, a s – сумма членов, больших 1. Вывести сначала массив X. Затем r, s. Ввод исходных данных осуществить из файла на диске. Вывод – по желанию пользователя программы либо на экран или в файл на диске.

№ 9.5 Дана последовательность из N различных чисел. Найти сумму чисел этой последовательности, расположенных между максимальным и минимальным числами (в сумму включить и оба этих числа). Последовательность ввести из файла на диске. Вывести исходную последовательность в виде таблицы с указанием порядкового номера числа в последовательности и самого числа и сумму чисел. Вывод предусмотреть:1) в файл на диске; 2) на экран.

№ 9.6 Даны: n – целое число (n>=2), последовательность действительных чисел : a1, a2,..., an. Получить:

min(a1+a2,a2+a3,...,an-1+an);

max(a1,a1a2,a1a2a3,...,a1...an-1an).

Ввод исходных данных осуществить из файла на диске. Вывод – по желанию пользователя программы либо на экран, либо в файл на диске.

№ 9.7 Дан файл f1, который содержит номера телефонов сотрудников учреждения: указывается фамилия сотрудника, его инициалы и номер телефона. Найти телефон сотрудника по его фамилии и инициалам. Вывод по желанию пользователя либо на экран, либо в файл на диске.

№ 9.8 Дан файл f, содержащий сведения о кубиках: размер каждого кубика (длина ребра в см), его цвет (красный, желтый, зеленый или синий) и материал (деревянный, металлический, картонный). Найти: а) количество кубиков каждого из перечисленных цветов и их суммарный объем;

б) количество деревянных кубиков с ребром 3 см и количество металлических кубиков с ребром, большим 5 см. Вывод по желанию пользователя либо на экран, либо в файл на диске.

№ 9.9 Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Дан файл f, содержащий сведения о нескольких автомобилях. Найти: а) фамилии владельцев и номера автомобилей данной марки; б) количество автомобилей каждой марки. Вывод по желанию пользователя либо на экран, либо в файл на диске.

№ 9.10 Дан файл f, содержащий сведения о книгах. Сведения о каждой из книг – это фамилия автора, название и год издания.

а) Найти названия книг данного автора, изданных до 2002 года.

б) Определить, имеется ли книга с названием «Информатика». Если да, то сообщить фамилию автора и год издания. Если таких книг несколько, то сообщить имеющиеся сведения обо всех этих книгах. Вывод по желанию пользователя либо на экран, либо в файл на диске.

№ 9.11 В память компьютера вводятся по очереди координаты N точек. Определить , сколько из них попадает в кольцо с внутренним радиусом R1 и внешним R2. Обе окружности имеют общий центр с координатами X0 и Y0. Результаты обработки выдать либо в файл на диске, либо на экран.

№ 9.12 Имеется таблица со среднесуточными температурами за период с 1.06.XX по 31.08.XX с точностью до 0,1 градуса. Необходимо определить и вывести на экран, а по просьбе пользователя и в файл на диске, максимальную и минимальную среднесуточные температуры за указанный период, а также дни, в которые среднесуточная температура отличалась от максимальной или минимальной не более чем на 0,5 градуса. Исходные данные вводить из ранее подготовленного файла. Форма представления исходных данных в файле:

Июнь

1…...температура

2……–

–…...–

–…...–

30…...–

Июль, август – аналогично.

Форма вывода исходных данных и результатов на экран и в файл:
Среднесуточные температуры:

Июнь.

Дата температура Дата температура и т.д.

1…….–……...6……– всего 6 пар столбцов

2…….–……..7……–

3…….–……..8……–

4…….–……..9……–

5…….–…….10…...–

Июль, август – аналогично.

Максимальная температура XX градусов была:

в июне: число, число, и т.д.

в июле: число, число, и т.д.

в августе: число, число, и т.д.

Минимальная температура XX градусов была:

в июне: число, число, и т.д.

в июле: число, число, и т.д.

в августе: число, число, и т.д.

№ 9.13 Дан файл f, содержащий сведения о веществах: указывается название вещества, его удельный вес и проводимость (проводник, полупроводник, изолятор).

а) Найти удельные веса и названия всех полупроводников.

б) Выбрать данные о проводниках и упорядочить их по убыванию удельных весов.

Составить программу. Вывод по желанию пользователя либо на экран, либо в файл на диске.

№ 9.14 Даны два файла f1 и f2. Файл f1 – это инвентарный файл, содержащий сведения о том, сколько изделий разных видов продукции хранится на складе (вид продукции задается его порядковым номером). Файл f2 – это вспомогательный файл, содержащий сведения о том, на сколько уменьшилось или увеличилось количество изделий по некоторым видам продукции. Вспомогательный файл может содержать несколько сообщений по продукции одного вида или не содержать ни одного такого сообщения. Обновить инвентарный файл на основе вспомогательного, образовав новый файл g.

№ 9.15 Составить программу назначения студентов на стипендию по результатам экзаменационной сессии:

1) если сессия сдана на все «5»,то стипендия повышенная;

2) если на «4» и «5», то стипендия обыкновенная;

3) если есть хотя бы одна «3»,то стипендия не назначается.

Исходные данные ввести из файла на диске. Вывод обеспечить: 1) на экран дисплея; 2) в файл.

№ 9.16 Даны координаты n точек на плоскости:X1,Y1,..., Xn, Yn (n=20). Найти номера двух точек, расстояние между которыми наибольшее (считать, что такая пара точек единственная). Координаты точек вводятся из файла. Вывести координаты точек в виде таблицы, номера двух точек, расстояние между которыми наибольшее и само расстояние между ними. Вывод предусмотреть: 1) в файл; 2) на экран.

№ 9.17 Дан файл f, содержащий сведения об учениках школы . Сведения об ученике состоят из его имени, фамилии, названия класса (года обучения, буквы), в котором он учиться и среднего балла по результатам в последней четверти.

а) Выяснить, сколько учеников школы не имеют отметок ниже четырех.

б) Собрать в файле g сведения о лучших учениках школы, т.е. об учениках, имеющих средний балл не ниже 4,5.

Составить программу. Вывод по желанию пользователя либо на экран, либо в файл.

№ 9.18 Информация о температуре воздуха за месяц задана в виде массива. Определить:

а) сколько раз температура опускалась ниже 0 градусов по Цельсию;

б) температура скольких дней была ниже среднемесячной.

Результаты выдать на экран или в файл. Исходные данные должны быть подготовлены в файле.

№ 9.19 По результатам экзаменационной сессии подготовить протокол, который должен содержать данные о неуспевающих студентах (пофамильно), среднюю оценку каждого студента, средний балл по группе. В заголовке протокола указать название факультета и номер группы. Файл с исходными данными подготовить на диске.

№ 9.20 Подготовить на диске файл с фамилиями и оценками, полученными студентами группы во время сессии. Определить число неуспевающих студентов, фамилии студентов сдавших экзамены на отлично и средний балл по группе. Протокол обработки данных выдать в файл, либо вывести на экран.

Приложение1. 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, biMinimaze, biMaximaze и biHelp. Свойство biSystemMenu определяет доступность кнопки системного меню (значок в заголовке окна), biMinimaze – кнопки Свернуть, biMaximaze – 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
Title	Текст в заголовке окна. Если значение свойства не указано, то в заголовке отображается текст Открыть
Filter	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 Тип файлов. Каждый фильтр задается строкой вида описание|маска, например Текст|*.txt
FilterIndex	Если в списке Filter несколько элементов (например, Текст|*.txt|Все файлы|*.*), то значение свойства задает фильтр, который используется в момент появления диалога на экране
InitialDir	Каталог, содержимое которого отображается при появлении диалога на экране. Если значение свойства не указано, то в окне диалога отображается содержимое папки Мои документы
FileName	Имя файла, выбранного пользователем

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

Свойство	Описание
Title	Текст в заголовке окна. Если значение свойства не указано, то в заголовке отображается текст Сохранить как Filter 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 Тип файлов. Каждый фильтр задается строкой вида описание|маска, например Текст|*.txt
FilterIndex	Если в списке Filter несколько элементов (например, Текст|*.txt|Все файлы|*.*), то значение свойства задает фильтр, который используется в момент появления диалога на экране
InitialDir	Каталог, содержимое которого отображается при появлении диалога на экране. Если значение свойства не указано, то в окне диалога отображается содержимое папки Мои документы
FileName	Имя файла, введенное пользователем в поле Имя файла
DefaultExt	Расширение, которое будет добавлено к имени файла, если в поле Имя файла пользователь не задаст расширение файла

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

Свойство	Описание
Name	Имя компонента.
Items.Name	Имя пункта меню. Используется для доступа к пункту меню
Items.Caption	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 позволителен. 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

Учебное издание

Елена Евгеньевна Дмитриева
Михаил Ильич Коробочкин

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

Часть II. Программирование в среде Delphi

Учебное пособие

Редакционно-издательский отдел ГУЗ
Сдано в производство 08.05.2015. Подписано в печать 09.02.2015.
Формат 60 х 84 1/16. Объем 16,75 п.л., 14,8 уч.-изд.л.
Бумага офсетная. Тираж 300 экз. Заказ № _____.
_____
Участок оперативной полиграфии ГУЗ
105064, г. Москва, ул. Казакова, 15