МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Методическое пособие по курсу «Информатика и вычислительная техника» (раздел «Программирование в среде «Delphi») для студентов физико-металлургического факультета

Донецк 2008

УДК 681.3.06 (071)

Методическое пособие по курсу «Информатика и компьютерная техника» (раздел «Программирование в среде «Delphi») для студентов физико-металлургического факультета /В.Н.Павлыш, И.Ю.Анохина, И.Н.Кононенко/ Донецк, ДонНТУ – 2008 - 113с.

Методическое пособие предназначено для получения навыков работы в среде программирования Delphi. Состоит из семи лабораторных работ, каждая из которых включает теоретический материал, необходимый для выполнения индивидуального задания, самих индивидуальных заданий и примеров их выполнения.

Первая лабораторная работа посвящена описанию основных понятий и принципов работы в среде Delphi, вторая – разработке программы линейного вычислительного процесса. В третьей лабораторной работе рассматриваются принципы создания алгоритма разветвленного вычислительного процесса и его реализации в среде программирования. В четвертой излагаются основы работы с массивами с использованием структурного компонента StringGrid. В пятой лабораторной работе описываются принципы отображения графической информации. В шестой– уделяется внимание основным принципам и методам построения графиков функций. В седьмой работе описываются принципы создания и работы с файлами в Delphi.

Рассчитано на студентов физико-металлургического факультета и студентов других специальностей, которые изучают в пределах курса «Информатика и компьютерная техника» раздел «Программирование в среде «Delphi».

Рецензент:

Отв. за выпуск:

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО КУРСУ «ИНФОРМАТИКА И КОМПЬЮТЕРНАЯ ТЕХНИКА»

(раздел «Программирование в среде «Delphi»)

Утверждено
на заседание кафедры
«Вычислительная математика и
программирования»
Протокол № 6 от 11. 02. 2008 г.

Донецк 2008

ВСТУПЛЕНИЕ

Delphi- это современная система программирования, с помощью которой можно быстро и качественно создавать любые программы: от самого простого калькулятора до многоуровневой системы управления технологическим процессом. Программы в Delphi пишутся на языке Object Pascal, который является объектно-ориентированным развитием широко распространенного языка Turbo Pascal. Среда Delphi включает в себя полный ассортимент инструментов для разработки приложений.

В пособии, ориентированном на тех пользователей, которые начинают свой путь в программировании, невозможно рассмотреть все вопросы, связанные с программированием в Delphi, все компоненты и характеристики среды. Много интересных тем остались за его рамками. Вместе с тем, в нем рассмотрены фундаментальные понятия программирования, базовые структуры данных и методы работы с ними, основные возможности среды и методы работы в ней — все то, что должен знать и уметь пользователь, начинающий процесс изучения алгоритмизации и освоения прикладного программирования.

Для успешного овладения учебным материалом следует придерживаться общих указаний к работе с методическим пособием.

1. Внимательно изучите теоретический материал.
2. Попробуйте повторить описанные действия на компьютере.
3. После овладения теоретическим материалом приступайте к выполнению индивидуального задания.
4. Проверьте правильность его выполнения сами, а затем покажите преподавателю.

СОДЕРЖАНИЕ

Основные понятия

Каждая наука имеет свои основополагающие понятия. В физике это масса тела и скорость, плотность и температура. В математике это цифры и числа, математические операции. Одно из основополагающих понятий информатики – это ***алгоритм.***

Алгоритм - это описание последовательности действий, которые необходимо выполнить для решения поставленной задачи.

Слово алгоритм произошло от латинского написания слова аль – Хорезми (algorithm), под которым в средневековой Европе знали величайшего математика из Хорезма Мухамеда бен Мусу, жившего в 783-850 гг. Вы знакомы с одним из его творений – действия над числами «столбиком».

С понятием алгоритма мы сталкиваемся ежедневно. Когда у Вас спрашивают, как пройти куда- либо, Вы формулируете алгоритм действий: сначала направо, три поворота налево и две ступеньки вниз. Рецепты приготовления блюд – это кулинарные алгоритмы: все порезать, смешать и варить всю ночь. Составляя утром, планы на день, Вы и здесь не обходитесь без алгоритма: сначала поехать в институт, если получится, удрать со второй пары, если нет, то с третьей, заехать к другу, вернуться домой и сделать чертеж.

Под математическим алгоритмом понимают описание процесса преобразования исходных данных в конечные результаты.

К алгоритмам, создаваемым для их последующего преобразования в компьютерные программы, предъявляется ряд требований.

Алгоритм должен быть представлен как последовательное выполнение шагов. Это свойство алгоритма называется дискретность. Под шагом понимается каждое действие алгоритма. Допустим, требуется найти сумму натуральных чисел от 1 до 1000. Если Вы напишите просто знак суммы, то программа выполнена не будет. Это не понятно. А вот если расписать

процесс получения суммы как выполнение последовательности операций сложения, то такой набор элементарных действий будет выполнен.

Алгоритм выполняется по шагам. Каждый из шагов должен быть однозначно выполним. Это свойство называют определенностью.

К примеру, в одном из кулинарных рецептов сказано: слегка размешать, подогреть и влить молоко. Понятно ли такое описание? Что значит слегка, до какой температуры подогреть, сколько молока.

В алгоритме не должны присутствовать неопределённые слова: немного, чуть-чуть, слегка и т. д.

Результативность - алгоритм должен приводить к решению задачи за определенное число шагов.

Это свойство подразумевает, что каждый шаг (и алгоритм в целом) после своего завершения даёт среду, в которой все имеющиеся объекты однозначно определены. Если это по каким-либо причинам невозможно, то алгоритм должен сообщать, что решение задачи не существует.

Примером может служить алгоритм решения квадратного уравнения. В том случае, когда дискриминант принимает отрицательное значение, важно получить сообщение об отсутствии действительных корней, что и будет результатом работы алгоритма.

Каждый шаг алгоритма обязательно представляет собой какое-либо допустимое действие.

Массовость - алгоритм составляется в общем виде, т.е. он должен быть применим к ряду задач, различающихся исходными данными.

При разработке алгоритма стараются описать его таким образом, чтобы можно было запускать программу на расчет при разных исходных данных.

Например, рассчитывая траекторию движения снаряда как исходные данные задают его массу, начальную скорость, скорость ветра… Эти величины не вводятся в программу как константы, а задаются в качестве переменных, тогда для расчета траектории полета снаряда с другой массой достаточно ввести новые исходные данные, а не изменять текст программы.

В зависимости от поставленной задачи и последовательности выполняемых шагов различают следующие виды алгоритмов.

Линейный - шаги алгоритма следуют один за другим не повторяясь, действия происходят только в одной заранее намеченной последовательности. Это аналогично поочередному вычислению формул. Например, необходимо вычислить

$z=3x^2+y$

$y=x+x^2+x^3$

$x=a+b$, где a, b – исходные данные.

Если бы Вы производили расчеты вручную, то сначала определили бы значение x, затем y и только потом z. Расчеты велись бы без каких-либо проверок условий, последовательно, один расчет за другим. Это и есть линейный алгоритм.

Алгоритм с разветвленной структурой - в зависимости от выполнения или невыполнения какого-либо условия производятся различные последовательности действий. Каждая такая последовательность действий называется ветвью алгоритма, выполняется либо одна, либо другая его ветвь.

Если дискриминант квадратного уравнения больше нуля, то следует определить по формулам два корня; если он равен нулю, то определяется один корень и, если меньше нуля, то выдается сообщение об отсутствии

действительных корней. В зависимости от коэффициентов квадратного уравнения будет реализована только одна ветвь алгоритма и никогда все три вместе. Дискриминант не может быть одновременно и положительным, и отрицательным и нулевым.

Если Вы смотрите на светофор, то в зависимости от его цвета выполняется одно из трех действий. Красный – Вы стоите, переходить улицу нельзя, желтый – Вы ожидаете смены цвета и на зеленый – переходите. Цвет не может быть одновременно и желтым, и красным, и зеленым.

Алгоритм циклической структуры. В зависимости от выполнения или невыполнения какого-либо условия повторяется определенная последовательность действий, называемая телом цикла. Процесс такого типа называется циклическим. Каждое утро мы просыпаемся, чистим зубы, завтракаем… Эти действия повторяются изо дня в день, пока мы живы, такой циклический процесс называется циклом с неизвестным числом повторений.

Каждый понедельник мы идем на работу и так пять дней в неделю. Это циклический процесс с известным числом повторений.

При создании программы выполняется несколько обязательных этапов.

Постановка задачи - составление точного и понятного словесного описания того, как должна работать будущая программа, что должен делать пользователь в процессе ее работы.

Разработка интерфейса (интерфейс - способ общения) - создание экранной формы (окна программы).

Составление алгоритма. Определение, каким образом должен быть получен результат, выбор математических, статистических и других формул, создание модели.

Программирование - разработка программного кода на языке программирования.

Отладка программы – процесс локализации и исправления ошибок, выявленных во время выполнения программы.

Тестирование программы - проверка правильности ее работы.

Создание документации, описывающей работу программы, помощи.

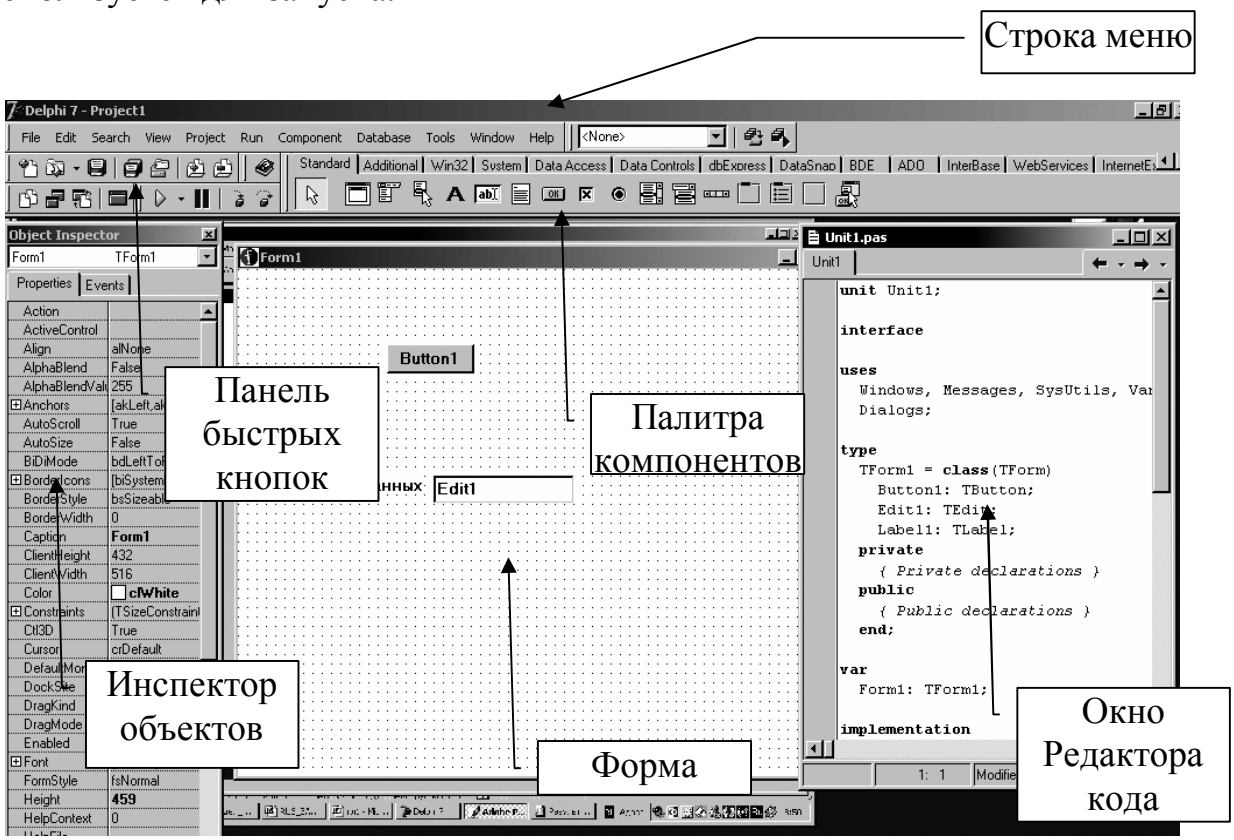
Лабораторная работа № 1

Основы работы в среде программирования Delphi

Цель работы: Изучить возможности среды программирования Delphi, правила создания, сохранения и запуска программ, работу с различными компонентами среды программирования.

Навыки: Приобрести навыки работы с различными компонентами среды Delphi, научиться создавать, сохранять и запускать программы. Освоить работу с компонентами Shape и Button.

Для запуска интегрированной среды разработки Delphi откройте пункт меню «Программы»|Borland Delphi|Delphi или найдите файл delphi32.exe, который используется для запуска.



Вид окна показан на рисунке.

В верхней части экрана расположена строка главного меню. Ниже строки меню размещены две инструментальные панели: панель быстрых кнопок, дублирующих наиболее часто используемые команды меню, и панель палитры компонентов. Палитра компонентов состоит из отдельных панелей, в которых сгруппированы компоненты по определенному принципу (Standard—наиболее часто используемые компоненты, System —системные компоненты: таймер, плейер… и ряд других панелей). Щелкая левой кнопкой мыши по закладке панели, Вы получаете в свое распоряжение находящиеся там компоненты.

Основой всех приложений, создаваемых в Windows, является форма. На ней размещаются все компоненты приложения. Редактор кода является программным редактором, в его окне пишется текст программы.

Инспектор объектов обеспечивает задание свойств компонент, помещенных на форму, и свойства самой формы.

В таблице 1 приводятся основные свойства компонентов Delphi.

Таблица 1 Свойства компонентов Delphi

Свойства	Комментарии
Autosize	True-высота элемента автоматически адаптируется к размеру символов текста; false- не адаптируется.
Caption	Строка текста, надпись на кнопке, метке, заголовок формы.
Color	Цвет фона компонента
Cursor	Определяет вид курсора мыши при попадании на объект.
Enabled	Определяет, реагирует ли компонент на события, связанные с мышью, клавиатурой, таймером.
Font	Задает свойства шрифта.

Height	Определяет высоту компонента
Hint	Определяет текст подсказки
Left	Координата левого края компоненты в пикселях
ShowHint	Определяет, показывать окно подсказки ил нет (true-показывать подсказку; false – не показывать)
Top	Координата верхнего края компоненты в пикселях
Width	Определяет ширину компонента
WindowState	Определяет размер окна формы

Одной из наиболее важных частей среды Delphi является Редактор кода. В этом окне создается текст программы, которая выполняет заданную последовательность действий.

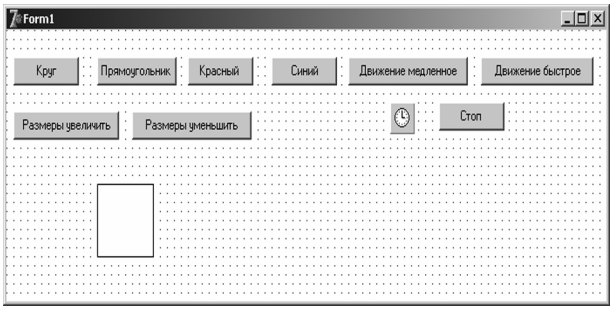
Ознакомимся с принципами работы Delphi на примере простейшего приложения.

Откройте страницу Additional. Найдите компонент **Shape**  и перенесите его на пустую форму. Для этого щелкните по нему левой кнопкой мыши, включая его, как кнопку, а затем поместите курсор в нужное место формы и щелкните опять левой кнопкой мыши.

Определите свойства объекта, используя Инспектор объектов. Свойств много, зададим только самые необходимые. Свойство «Shape» задает внешний вид компонента, это может быть круг (stCircle), прямоугольник (stRectangle) , эллипс(stEllipse) или фигуры с закругленными концами, тогда в название свойств добавляется слово Round. Задайте по своему усмотрению высоту, ширину и цвет объекта, их английские названия приведены в таблице 1.

1. Откройте страницу Standard и выберите кнопку **Button**  . Для этого объекта задаем свойство Caption, определяющее надпись, допустим «Круг», в строке Hint укажем подсказку «Нажми на кнопку для получения круга» и зададим ShowHint=true, чтобы подсказка высвечивалась при попадании курсора на кнопку.

2. Задайте аналогичные кнопки, руководствуясь приводимым ниже рисунком.



На нашем рисунке размещены следующие кнопки: кнопка «Круг», позволяющая изменить форму фигуры, превратив ее в круг; кнопка «Прямоугольник», превращающая фигуру в прямоугольник; кнопки «Красный» и «Синий», меняющая цвета; кнопки «Движение медленное» и «Движение быстрое», они будут задавать скорость движения объекта; кнопка «Стоп» останавливающая объект и две кнопки, задающие изменение объекта.

3. Так как мы предполагаем движение фигуры, то на форму помещаем **Timer** (страница System). Его свойства отличаются от стандартных свойств объектов, поэтому рассмотрим их отдельно. Во-первых, этот объект, хотя и выносится на форму, не виден на экране, поэтому помещать его можно в любое место. Свойство Interval задает время в миллисекундах, чем больше интервал, тем медленнее движется объект. Свойство Enabled задает в случае установки True начало отсчета времени, при значении равном False –остановку таймера.

4. Теперь, когда все объекты размещены на форме, сохраним наше приложение. Выполните команду File|Save all. Любой проект имеет, по-крайней мере, шесть файлов, связанных с ним.

5. Главный файл проекта, изначально называется PROJECT1.DPR. Введите свое название, при этом разрешается использовать прописные и строчные латинские буквы и цифры, при этом на первом месте в имени файла должна стоять буква.

6. Модуль программы |unit|, сохраняется как UNIT1.PAS, но его можно назвать любым другим именем, например, если файл с расширением DPR мы назвали gr.dpr, то второй файл назовем gr1.pas , тогда при

сортировке по имени все файлы, относящиеся к одному проекту будут расположены рядом, что облегчит их поиск, например, для копирования на дискету.

7. Файл главной формы с расширением DFM, используется для сохранения информации о внешнем виде главной формы.

8. Файл с расширением RES содержит иконку для проекта и создается автоматически.

9. Файл с расширением OPT является текстовым файлом для сохранения установок, связанных с данным проектом.

10. Файл с расширением DSK содержит информацию о состоянии рабочего пространства.

После компиляции программы появляется файл с расширением exe - исполняемый файл.

11. Теперь после задания внешнего вида формы, следует написать текст программы, чтобы определить выполняемые при нажатии на каждую кнопку действия. Щелкнем дважды по кнопке «Круг», автоматически появится скомпилированный текст процедуры.

unit f12; ---------- **модуль начинается с ключевого слова Unit, после которого следует имя файла.**

interface

uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, **Dialogs, ExtCtrls, StdCtrls; ----------список подключенных модулей**

type TForm1 = class(TForm)

Button1: TButton;
Shape1: TShape;
Button2: TButton;
Button3: TButton;
Button4: TButton;
Timer1: TTimer;

Список компонент, помещенных на форму

Button5: TButton;

Button6: TButton;

Button7: TButton;

Button8: TButton;

Button9: TButton;

procedure Button1Click(Sender: TObject); ----------**список реализованных процедур**

private { Private declarations }----------**описание переменных, функций и процедур, доступных только в этом модуле**

public { Public declarations } ----------**описание переменных, функций и процедур, доступных в других модулях**

end;

var Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);

begin

shape1. shape:=stCircle;----------этот оператор задает изменение внешнего вида компонента Shape

end;

Обратите внимание, что при создании программы, кроме оператора **shape1. shape:=stCircle;** Вы не вводите ничего, все компилируется средой Delphi автоматически. Для изменения вида фигуры необходимо указать имя объекта, который видоизменяется, в нашем случае это Shape1. Внешний вид определяет свойство Shape, если Вы откроете его, то увидите набор возможных модификаций фигуры. Укажите нужный .

12. Если щелкнуть по кнопке «Прямоугольник», фигура должна в него преобразоваться. Дважды щелкните по кнопке с аналогичным названием , появится скомпилированный текст программы и курсор будет мигать между

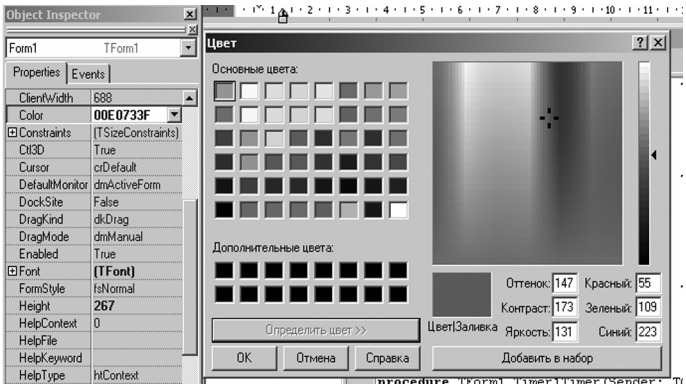
словами begin и end. Именно в этом месте и следует ввести оператор shape1. shape:=stRectangle ;

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

Здесь мигает курсор и сюда вводится текст программы!

```
end;
```

13. Для задания смены цветов рассмотрим технологию их образования. Компьютерные мониторы отображают цветовые гаммы, следуя технологии RGB (red –красный; green—зеленый; blue—синий).



Чтобы посмотреть, как действует технология RGB, откройте свойство Color формы. Щелкните дважды по появившемуся цвету, откроется диалоговое окно с набором цветных прямоугольников. Щелкните по кнопке «Определить цвет». Выберите любой цвет и посмотрите на набор цветов «Красный», «Зеленый» и «Синий» в правом нижнем углу экрана. Чтобы получить любые цвета, отображаемые на компьютере, эти цвета смешивают. В Вашем распоряжении от 0 до 255 значений, считайте литров краски. Например, в строке «красный» укажем максимально возможное количество —255, в остальных — по нулю. Цвет образца (правый нижний угол) изменится на «настоящий» красный, точно также, если указывать максимальную интенсивность для зеленого или синего, а остальные цвета не использовать, Вы получите ярко синий, ярко зеленый. Это логично, если Вы взяли карандаш или фломастер синего цвета и начали рисовать им, то это будет только синий цвет. А что будет, если по синему сверху порисовать зеленым карандашом?

Задайте красного и зеленого по 255, а синего не берите вообще (ноль), Вы получите чистый желтый цвет. Все по 128 даст серый, все по нулю—черный, все по 255 дает белый цвет.

Теперь посмотрим, как задать изменение цвета в программе. У объекта Shape есть свойство Brush, которое в свою очередь распадается на два подсвойства: Color и Style. Первое из них позволяет задавать цвет, а второе – тип заливки, т.е. штриховку. Оператор изменения цвета выглядит следующим образом: shape1.brush.Color:=rgb(255,0,0); Это задание красного цвета объекта. Введите такой же оператор, но с другим набором составляющих RGB при щелчке по кнопке «Синий».

14. Зададим движение. На экран был помещен компонент Timer . Установите вначале свойство Enabled равным false, это означает, что таймер находится в неактивном, нерабочем состоянии. Тогда при щелчке по кнопке «Движение медленное» включим таймер и зададим большую длительность интервала:

timer1.enabled:=true; timer1. interval:=500.

Щелкаем дважды по таймеру. Пусть объект должен двигаться слева направо, это значит, что значение координаты Left будет увеличиваться, как увеличивается расстояние от левого края формы. Это задается оператором:

shape1.Left:= shape1.Left +5;

цифра пять выбрана нами произвольно, она устанавливает, на сколько пикселей должен смещаться объект каждые 500 миллисекунд.

Быстрое движение объекта задается в виде :

timer1.enabled:=true; timer1. interval:=10;

т.е. опять включается таймер, но интервал значительно меньше, значит те же пять пикселей фигура должна пройти не за 500, а за 10 миллисекунд.

Чтобы остановить движение, следует выключить таймер timer1.enabled:=false.

15. Изменение размеров происходит по командам:

shape1.Width:= shape1.Width*2;

shape1.height:= shape1.height*2;--------Это увеличение в два раза и высоты и ширины фигуры. Уменьшение объекта задается с помощью тех же операторов, но следует использовать знаки минус или делить.

Чтобы приложение лучше выглядело, мы можем задать цвет самой формы (Свойство Color), цветные буквы на кнопках (Свойство Font).

16. Теперь, когда готова программная часть приложения, сохраним его еще раз и запустим. Для запуска используется пункт меню Run (напротив него указана быстрая клавиша F9. Можно использовать и треугольник, он синего (Delphi3) или зеленого (Delphi7) цвета.

Так как в программе не был реализован выход, то воспользуйтесь стандартной кнопкой закрытия окна в Windows, если хотите вернуться в режим программирования.

Иногда возникает ощущение, что программа «зависла». Это может быть, например, по причине неправильного ввода исходных данных. Выполните команду Run|Program Reset (выполнение команды можно заменить использованием клавиш Ctrl+F2) и произойдет возврат в среду программирования.

В конце каждой темы мы будем выносить основные положения, которые изложены в ее рамках.

Основные положения темы:

- ✓ Для создания нового приложения выполняется команда File|New.
- ✓ Для сохранения File|Save All.
- ✓ Для запуска на выполнение Run|Run.
- ✓ Для перезагрузки программы команда Run|Program Reset.

- ✓ Компонент Button (кнопка) позволяет задать выполнение определенных операций при нажатии на нее. Основным свойством компонента является свойство Caption (текст на кнопке).
- ✓ Компонент Shape используется для помещения на форму прямоугольника, или квадрата, или круга или эллипса. Основным свойством является свойство Shape (вид фигуры), свойство Brush (задает цвет и стиль штриховки).
- ✓ Компонент Timer используется для задания процессов, изменяющихся во времени, например. Движение фигуры, изменение ее размеров …Основными свойствами является свойство Enabled (включить или выключить таймер) и свойство Interval, задающее скорость процессов.

Индивидуальные задания.

1. Создать программу, позволяющую демонстрировать геометрические фигуры в пределах свойств компонента Shape. Предусмотреть изменение размеров и цвета фигур, их движения по вертикали и горизонтали одновременно.

2. Создать программу, позволяющую демонстрировать планеты Солнечной системы. В программе реализовать набор кнопок, позволяющих при выборе определенной планеты перемещать ее на передний план изображения, увеличивая при этом размеры.

3. Создать программу, изображающую попадание мяча в ворота.

4. С использованием компонента Shape создайте программу, позволяющую изобразить дом. Затем из набора предложенных образцов выберите размеры, тип и материал для окон. При выборе определенного типа окна, оно должно перемещаться таким образом, чтобы разместиться на доме. Окна могут быть круглые, с закруглениями, квадратные, прямоугольные. Предусмотреть изменение размеров и цвета окон.

5. Создайте программу, показывающую движение металла в валках в прокатном стане с изменением размера полосы.

6. С использованием компонента Shape создайте программу, позволяющую изобразить дом с дверью. Создать программу, позволяющую выбрать вариант двери: двойная, одинарная, металлическая, деревянная, ее толщину, цвет и т.д. Предусмотреть изменение размеров и цвета двери.

7. Создать программу, позволяющую выбрать вариант дома: одноэтажный, двухэтажный, с верандой или без, из дерева или кирпича. Предусмотреть изменение размеров и цвета дома.

8. Создать программу, позволяющую выбрать вариант косметики: тени произвольного цвета, глаза - круглые, квадратные, овальные.

9. Создать программу, позволяющую выбрать вариант оформления комнаты: обои (цвет, рисунок), краска(произвольный цвет). Потолок (обои, дерево, побелка) и.т.д.

10. Создать программу, позволяющую выбрать вариант холодильника- цвет, дизайн, мощность.

11. Создать программу, изображающую передвижение паровоза.

12. Создать программу, демонстрирующую изменение дерева в зависимости от сезона. Летом – зеленые яблоки, осенью – красные и падают и т.д.

13. Создать программу, изображающую рост растения.

14. Изобразите кота, причем отдельно должны быть кнопки, позволяющие задать цвет его туловища, глаз, хвоста. Организуйте выбор размеров кота и его движения по диаметру экрана.

15. Изобразите изменение человека во времени: младенец, взрослый, старик.

16. Создайте программу, позволяющую показать, как колба наполняется водой.

17. Создайте программу, позволяющую продемонстрировать, как в зависимости от выбора карандаша меняется цвет объекта. Карандаш должен двигаться.

18. Изобразите собаку, забегающую в будку.

19. Изобразите движение лодки по реке.

20. Изобразите падение снега и появление сугроба.

21. Изобразите включение|выключение компьютера.

22. Создайте программу, позволяющую показать, как плывут облака и идет дождь.

23. Создайте программу, изображающую смену дня и ночи с цветовым оформлением и движением солнца или луны по небу.

24. Создайте программу, демонстрирующую, как разлетается на части объект при попадании в него мяча.

25. Создайте программу, демонстрирующую попадание в цель при стрельбе.

Лабораторная работа № 2 Разработка программы линейного вычислительного процесса

Цель работы: Программирование линейного вычислительного процесса.

Навыки: Ввод, вывод информации. Программирование формул.

Постановка задачи:

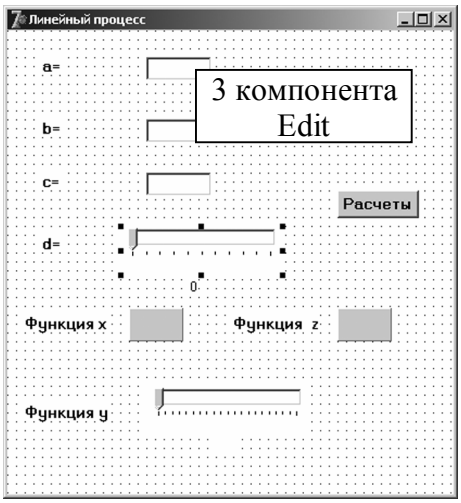
Исходные данные : a,b-дробные числа; c,d –целые.

Определить x:=sin(a+2·b);

$y:=e^3+ (c+3\cdot d)^{0.5}$;

$z:= (x+y)^2$ и вывести значения на экран.

Расположим на форме компоненты, необходимые для работы.



Для ввода a,b,c используем компонент Edit. Значение переменной a будет вводиться в Edit1, значение b - в Edit2 и c в Edit3.

Edit - компонент, который используется для ввода/вывода символьной сроки. Основным свойством является свойство Text, первоначально в этом окне отображается имя компонента, если окно при вводе информации должно быть пустым, его просто стирают; если есть стартовые исходные данные, то их можно ввести в этом свойстве.

Для удобства работы с программой , введем поясняющие надписи. Для этого используют компонент **Label.**

Label – компонент, предназначенный для отображения текстовой информации. Информация записывается в свойстве Caption (в нашем примере это надписи вида a=; b=; c= и другие).

Для ввода информации можно использовать компонент **TrackBar**. Он принадлежит к панеди кнопок Win32. Это элемент управления с ползунком, который пользователь может перемещать с помощью мыши. К его основным свойствам относятся:

Min, Max – минимально и максимально допустимые значения, в которых изменяется отображаемая на ползунке информация.

Orientation- определяют горизонтальную (выбрать из списка trHorizontal) или вертикальную (trVertical) ориентацию ползунка.

Frequency – частота отображения рисок на шкале.

TickMarks- положение шкалы (tmBoth –снизу и сверху; tmBottomRight внизу ползунка; tmTopLeft –сверху).

Position – основное программируемое свойство, определяющее позицию расположения ползунка.

Для вывода информации в этой программе будем использовать компонент Panel.

PANEL - компонент, который используется для вывода результатов. Основным для него является свойство Caption , которое и используется для вывода данных.

Кроме этих объектов на экран помещена уже известная вам кнопка Button с надписью «Расчеты».

Так как поставленная задача предполагает операции с числами, первым делом следует организовать ввод информации. У среды Delphi есть важное свойство: вся вводимая информация воспринимается как текстовая, даже при вводе чисел. Для проведения расчетов сначала вводимую информацию преобразовывают в числа. Производят расчеты, а затем полученные результаты преобразуют в текстовую информацию для вывода на экран.

Дважды щелнем по кнопке «Расчеты» для компиляции процедуры. Это приведет к генерации текста

procedure TForm1.Button1Click(Sender: TObject);

begin

После begin введем текст программы:

a:=strtofloat(edit1.text);

b:=strtofloat(edit2.text);

c:=strtoint(edit3.text).

Функция strtofloat используется для преобразования строки в число с плавающей запятой (дробное), strtoint – аналогично, но в целое.

Оператор a:=strtofloat(edit1.text) показывает, что значение переменной a будет считано из компонента Edit1 (как вы помните, ввод данных происходит за счет свойства text). Преобразовано в дробное число.

Для считывания информации из компонента TrackBar щелкнем по нему дважды. Автоматически скомпилируется процедура TrackBar1Change. При изменении позиции ползунка в переменную d считывается значение этой позиции. Нами организован вывод на экран показателя ползунка. Для этого использована метка Label и оператор label7.Caption:=inttostr(d).

Функция inttostr выполняет преобразование целого числа в его текстовый эквивалент.

procedure TForm1. TrackBar1Change(Sender: TObject);

begin d:= TrackBar1.position;

label7.Caption:=inttostr(d);

end;

Для проведения расчетов дописываем текст procedure TForm1.Button1Click. Для расчетов используют оператор присваивания , который имеет формат

Переменная:=выражение;

где в левой части указывается имя переменной, которая вычисляется. А справа – как произвести вычисления.

Delphi позволяет использовать шесть арифметических операций:

+ сложение; **-** вычитание; ***** умножение; / деление; **div** целочисленное деление; **mod** – остаток от целочисленного деления. Кроме этого

используется парное количество круглых скобок () и стандартные математические функции.

В таблице приведены наиболее часто используемые математические функции.

Функция	**Значение**	**Функция**	**Значение**
Abs (n)	Модуль числа n	Arctan (n)	Арктангенс n
Sqrt (n)	Квадратный корень из n	Exp(n)	Экспонента n
Sqr (n)	Квадрат n	Ln(n)	Натуральный логарифм n
Sin (n)	Синус n	Rardom(n)	Случайное целое число в диапазоне от 0 к n-1
Cos (n)	Косинус n	Trunc(n)	Целая часть дробного числа
PI	Значение π=3.1415926		

Величина угла тригонометрических функций должна быть выражена в радианах. Для превращения величины угла из градусов в радианы используется формула $\frac{a \cdot Pi}{180}$, где: a - величина угла в градусах; 3.1415926 - число π.

Запрограммированные формулы выглядят следующим образом:

```
x:=sin(a+2*b);
y:=exp(3)+sqrt(c+3*d);
z:=sqr(x+y).
```

При программировании формулы следует помнить, что сначала выполняются все расчеты функций, затем операции умножения и деления в порядке их следования слева направо, и только потом сложения и вычитания в том же порядке слева направо. Это компьютерный принцип **приоритета операций**. Чтобы лучше уяснить понятие приоритета операций, давайте рассмотрим простое математическое выражение $2*2+3^2-5+9$. Вспомним

школьные годы и попробуем произвести вычисления. Что мы делаем в начале? Умножаем два на два и запоминаем, что результат равен четырем. Затем возведем в квадрат три, получим девять. Формула примет вид 4+9-5+9. А дальше последовательно 4+9=13. затем вычтем из тринадцати пять 13-5=8 и в конце прибавим 9. Результат равен 17. В принципе мы сейчас работали как компьютер, а, если точнее, то компьютер сделали под нас. Сначала были найдены две старшие операции: возведение в степень и умножения, мы посчитали и запомнили результат или записали его на листе бумаги, затем слева направо производили все операции сложения и вычитания, подряд, в том порядке, в каком они следуют в формуле.

Круглые скобки, как и в математике, меняют приоритет операций.

Теперь требуется организовать вывод на экран в компоненты Panel и компонент TrackBar. Вывод на экран можно организовать несколькими способами. Первый способ заключается в использовании функции floattostr (преобразование дробного числа в строку и имеет вид

panel2.caption:=floattostr(z); - вывод в компонент Panel. Количество знаков, выводимых после запятой равно 11.

Второй способ состоит в использовании функции str. Функция имеет формат str(число: количество выводимых символов: из них после запятой, имя строки, в которое число преобразуется). Этот вариант позволяет выдавать на экран число с заданной степенью точности.

```
str(x:5:2,ss);
  panel1.caption:=ss;
```

Третий способ – вывод в компонент trackbar1.

```
  trackbar1.position:=trunc(y);
   label11.Caption:=floattostr(trunc(y));
```

Для вывода необходимо преобразование числа с отбрасыванием его дробной части. В компонент label11 выводится полученное значение для отображения его величины на экране.

Все переменные, которые использовались в программе, должны быть описаны. В разделе Var конкретной процедуры описываются переменные, которые будут использованы только в ней.

В нашем случае описание имеет вид

var ss:string; a,b,x,y,z:real; c:integer;

В таблице приведены варианты описания данных.

Тип	Размер в байтах	Диапазон значений	Тип	Размер в байтах	Диапазон значений
Byte	1	0÷255	Real		8
Word	2	0÷65535	Single		4
LongWord	4	0÷4294967295	Double		8
ShortInt	1	-128÷127	Char	1 символ	1
SmallInt	2	-32768÷32767	String	Максимальная длина символов	255
Integer	4	-2147483648÷2147483647			

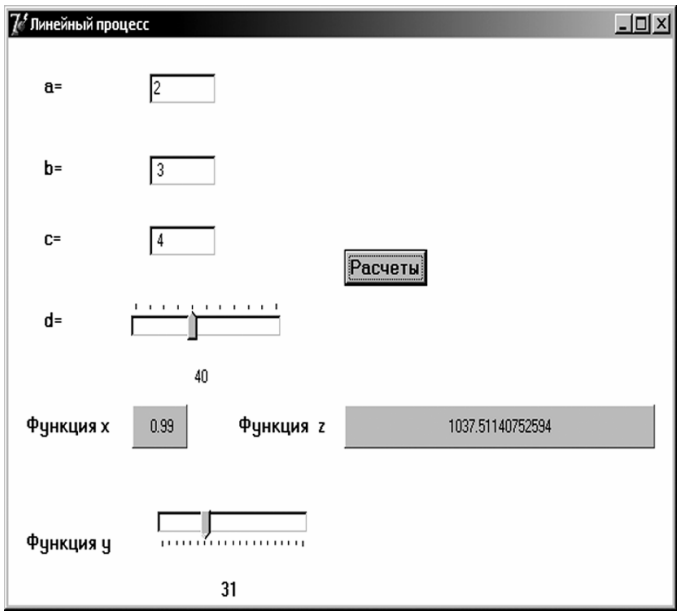
Переменная d используется в двух процедурах. Ее описание помещают в раздел

public d:integer;

Листинг процедуры, которая происходит при щелчке по кнопке «**Расчеты»** имеет следующий вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var ss:string; a,b,x,y,z:real; c:integer;
begin
a:=strtofloat(edit1.text);
b:=strtofloat(edit2.text);
c:=strtoint(edit3.text);
x:=sin(a+2*b);
```

```
y:=exp(3)+sqrt(c+3*d);
z:=sqr(x+y);
  str(x:5:2,ss);
  panel1.caption:=ss;
  trackbar1.position:=trunc(y);
   label11.Caption:=floattostr(trunc(y));
```




```
panel2.caption:=floattostr(z);
        end;
```

Листинг процедуры, которая происходит при перемещении ползунка, имеет следующий вид:

```
procedure TForm1.sChange(Sender: TObject);
begin          d:= s.position;
  label7.Caption:=inttostr(d);
end;
```

Нами приводится копия экрана, демонстрирующая работу программы.

Обратите внимание на значение функции z. Если нет необходимости в таком большом количестве знаков после запятой, то функцию floattostr не используют для вывода информации.

Основные положения темы:

- ✓ Для ввода|вывода информации используют компоненты Edit, Panel и TrackBar.
- ✓ Вся вводимая информация должна быть преобразована из текста в числовые данные соответствующего типа для последующих расчетов.

- ✓ Вся выводимая информация должна быть преобразована из числовых типов в текст для последующего вывода на экран.
- ✓ При расчетах формул используется оператор присваивания.
- ✓ При программировании формул можно использовать числа, переменные, функции, знаки операций и парные круглые скобки.
- ✓ При программировании следует учитывать приоритет операций.

Индивидуальные задания. *Осуществить ввод входных данных, выполнить расчет за формулами, вывести на экран результаты расчетов, сделать проверочный расчет в Microsoft Excel и сравнить полученные результаты.*

№ варианта	Исходные данные	Формулы, по которым выполняют расчеты	Результаты расчетов
1	X, Y	$Z=\frac{tg\,(x^2-y)}{\cos(3+a^3)}$ $A=\frac{2+z^4-\pi/12}{3-y^3}$ $D=\sqrt[3]{a+8^x}-5+tg\,(2\pi/3)$	A, Z
2	A,B,C	$X=\frac{2\ln\lvert D\rvert}{b}+\frac{3}{c^2}-\frac{4\sin b}{a^3}$	X, D
3	X, M, N	$P=2\cdot\sin(x^3-1{,}3m)+\sqrt{6{,}7s}$ $S=2\cos(n^4-\pi/3)+8*\sin(m^2-0{,}3)$	P, S
4	X,Y,C	$Z=tg\,(x+\frac{\pi}{8})+3\frac{x}{W^2-5}-12$ $W=\ln(x^3-2c)+\frac{\sqrt{c}}{y+4}-12$	Z,W

5	X,C	$A = \sin(\ \pi\ /\ 6\ -\ x\ )\ +\ \cos(\ \ 2\,\pi\ /\ 3\ -\ 3\,b\ )\ -\ 0\,,5$ $B\ =\ 6\,tg\ (\,x\ +\ \pi\ /12\ )\ -\ 2\ /\ c\ +\ 3\ /\ x^{\,2}\ +\ 3$	A, B
6	X,Y	$A\ =\ \ln(\ \ x^{\,3}\ -\ 2\,b\ )\ +\ \dfrac{b^{\,2}}{y\ +\ 4}\ -\ 12$ $B\ =\ \sin(\ \ \pi\ /\ 6\ -\ x\ )\ +\ \cos(\ \ 2\,\pi\ /\ 3\ -\ 3\ y\ )$	A, B
7	X,Y	$A\ =\ \dfrac{2\ +\ x^{\,4}\ -\ \pi\ /12}{3\ -\ c^{\,3}}\ -\ \ 2\ ,8\ \ y^{\,3}$ $C\ =\ \dfrac{tg\ \ (\,z^{\,2}\ -\ \ x\ )}{\cos(\ \ \ 3\ +\ \ x^{\,3}\ )}\ -\ \ 5$	A, C
8	X,Y, Z	$A=\dfrac{1+b^2-3z}{\sqrt{1+x^3}}$ $D=2a+\dfrac{\sin^5 y}{3+7z}$ $A=\sqrt{1\ +\ \ B^{\,2}\ -\ \cos\ ^{\,2}\ x}$	A, D
9	X,Y	$B\ =\ \sqrt[5]{Y}\ \dfrac{2\ +\ x^{\,4}\ -\ \pi\ /12}{3\ -\ x^{\,3}}$ $Y\ =\ \dfrac{1\ +\ \lvert C\rvert}{\sqrt[3]{1\ +\ x\ +\ x^{\,2}}}$	A, B
10	X,C	$A=\dfrac{1+x^2-3y}{\sqrt{1+x^3}}$	A, Y
11	X,F	$G=2\ln(1+x^2)+R\dfrac{2-F}{3+F}$ $R=(1+x)^{\frac{3}{5}}\sqrt[3]{F}$	G, F
12	X	$A=\dfrac{1+B}{1+x^2}$ $B=\sqrt{1+\dfrac{x}{1+C}},$ $C=2\lvert\sin(3x)\rvert$	A, B, C

13	X	$A = \frac{1+B}{\sqrt{1+x^2}} + Z$ $B = -A + 2e^{-2Z}$ $Z = \lvert 2 - X \rvert^{1/3}$	A, B, Z
14	M, N, K	$X = 3\sin M^3 - \cos^2 N$ $Y = 3\sqrt{1+K^2} + \lvert \sin^3 X \rvert$	X,Y
15	B	$A = 3\sin^2 B - \cos C + D/3$ $D = \sqrt{2+B^2}$ $C = \frac{\sqrt[7]{B}}{2B}$	C, D, A
16	X,Y	$A = \frac{3x^2}{1+Y^2}$ $C = \frac{\cos(A^2 - x)}{\ln\lvert x \rvert (3+y^3)}$	A, C
17	X, R	$A = 3x + \sqrt{1+x^2} + B^4$ $B = \sqrt{1+\frac{2x}{1+R^4}}]$ $Z = 2\sin 3R + \sqrt[3]{B^5}$,	A, B, Z
18	M, N, K	$S = \sqrt[3]{M+N^3} - 5K^2 + \cos(2\pi M/3)$ $D = \frac{2}{S} + \frac{3}{N^2} - \frac{7}{M^3} + 3K^3\sqrt[6]{M^3}$	S, D
19	X,Y	$Q = \frac{1+3\cdot tg(\frac{2\pi x}{3})}{2+\sqrt[3]{1+Z}}$ $Z = \ln(x^3 - 2y) + \frac{x^2}{y^3 - 4x}$	Q, Z
20	X,Y	$A = \frac{3+\sin B}{1+x^2}$ $B = 2x^2\cos^2 Y$ $C = \frac{2 - AB^3}{2AB} + \ln\lvert B \rvert$	A, B, C

21	S, D, F	$A=\frac{2+s^4-\frac{\pi d^3}{12}}{3-s^3}f-3{,}3$ $B=\frac{tg(s^2-d)}{\sin(3d+f^3)}+\sqrt[5]{f}$	A, B
22	X, Z	$A=\ln(x^3-2B)+\frac{x^2}{z^3+4}-12$ $B=\cos(\frac{\pi x}{6}-\frac{x}{z})+\cos(2\pi-3x)$	A, B
23	X	$H=\cos(x^3-\frac{\pi x}{6})+8\sin^5(x^2-0{,}3)$ $D=tg(H+\frac{\pi Z}{8})+3\frac{x}{Z^2-5H}-12Z$ $Z=\sqrt[7]{\sin^4(x)}$	H, D, Z
24	X	$A=2\sin^3(x^3-6{,}3\frac{B-5}{2BC})+\sqrt{C}$ $B=\cos^3(x^4-\pi/3)$ $C=\frac{45}{3B\sqrt[3]{x^4}}$	A, B, C
25	N, K	$L=\frac{2+M^4-\pi n/12}{3-k^2}$ $M=\frac{tg^2(n^2-k)}{\cos(3+n^3)}$	L, M
26	X,Y, Z	$S=\cos(y^4-\pi/3)+\sin^3(x^2-0{,}7)+\frac{x}{C-z}$ $C=tg^3(\sqrt{x}+\frac{8}{4-y})+3\frac{x}{x^2-5}-z$	S, C
27	X,Y,Z	$A=\frac{2+B^4-\pi/12}{3-x^3}+\sqrt[3]{\lvert\sin(x)\rvert}$ $B=\frac{\sin(x^2-x)}{\ln\lvert y\rvert(3+z^3)}$	A, B

Лабораторная работа № 3 Разработка программы разветвленного вычислительного процесса

Цель работы: Изучение правил алгоритмизации и программирования при использовании разветвленного вычислительного процесса.

Навыки: Использование операторов **IF** и **CASE** при программировании разветвлений.

На практике редко встречаются задания, алгоритм решения которых является линейным.

Точки алгоритма, в которых выполняется выбор последующего хода программы, называются точками выбора. Выбор шага решения задачи осуществляется в зависимости от выполнения некоторого условия.

В повседневной жизни условие обычно формулируется в виде вопроса, на который можно ответить **да или нет.** Например:

- Величина дискриминанта положительная ?
- Ответ правилен?
- Сумма покупки больше 500 гривен?

В программе условие — это выражение логического типа, которое может принимать одно из двух значений: **True** (истина) или **False** (ложь).

В языке Delphi есть шесть операторов сравнения (>больше, < меньше, = равно, <> не равно, >= больше или равно, <= меньше или равно).

Из простых условий с помощью логических операторов: **and** — «логическое И», **or** – «логическое ИЛИ» и **not** – «нет» можно строить сложные условия.

Оператор **if** позволяет выбрать один из двух возможных вариантов развития программы. Выбор осуществляется в зависимости от выполнения условия. В общем виде Оператор **if** записывается так:

if условие then оператор1 else оператор2.

Если условие выполняется, то действует оператор 1 иначе оператор 2.

Например, необходимо запрограммировать следующее условие:

$$z = \begin{cases} \dfrac{a\ sin(x)}{x} & x \le 0 \\ \dfrac{x}{(x-6)} & x > 0 \end{cases}$$

If x<=0 then z:=a*sin(x) else z:=x/(x-6);

В зависимости от введенного значения x расчет производится по первой формуле, если x меньше нуля и по второй, если введенное значение больше или равно нулю.

По правилам языка после слов **then** или **else** можно ставить один, но любой оператор, это может быть оператор присваивания, вывода на экран, ввода данных, еще один условный оператор и т.д. Если по условию задачи после **then** или **else** требуется поставить более одного оператора используют составной оператор. Составной оператор состоит из группы операторов, помещенных в операторные скобки **begin** и **end.**

Допустимы вложенные операторы if . Например, выполняемая последовательность действий в зависимости от цвета светофора может быть запрограммирована следующим образом:

If svet=’зеленый’ **then** showmessage(‘переходим улицу’) **else if** svet=’желтый’ **then** showmessage(‘ожидаем’) **else** showmessage(‘стоим’);

Использованный в примере оператор showmessage используется для вывода строки текста, причем строка помещается в одинарные кавычки.

Рассмотрим процедуру нахождения корней квадратного уравнения. В общем виде оно записывается как *ax2 + bx+ c = 0*.

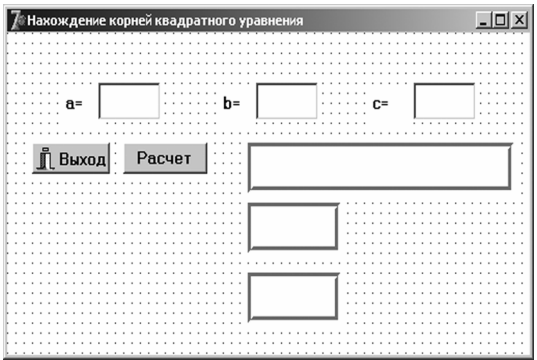
Для ввода данных использованы три компонента Edit. Для вывода результатов три компонента Panel.

Текст программы выглядит следующим образом:

procedure TForm1.Button1Click(Sender: TObject); {расчет производится при щелчке по кнопке «Расчет»}

var ss:string; a,b,c,d,x1,x2:real; {*описание используемых в программе переменных: a, b, c – коэффициенты уравнения; d- дискриминант; x1,x2 – корни уравнения }*

begin



a:=strtofloat(edit1.text); *{Ввод исходных данных}*

b:=strtofloat(edit2.text);

```
c:=strtoint(edit3.text);
 d:=b*b-4*a*c; {Расчет дискриминанта}
 str(d:1:2,ss);
  panel1.caption:='дискриминант ='+ss; {Вывод дискриминанта на экран, причем кроме этого выводится текст; текстовые переменные соединяются знаком плюс для вывода в одну строку}
  if d>0 then begin x1:=(-b+sqrt(d))/2/a;  x2:=(-b-sqrt(d))/2/a;
{Если дискриминант больше нуля, определяются оба корня}
  panel2.Visible:=true;  panel3.Visible:=true; {Первоначально оба компонента Panel на экране не отображаются, т.к. неизвестно, есть ли у уравнения корни, если они есть, то их выводят на экран, для этого свойство компонента Visible полагается равным true }
   str(x1:1:2,ss);
  panel2.caption:='x1 ='+ss; {Вывод первого корня в Panel2}
   str(x2:1:2,ss);
  panel3.caption:='x2 ='+ss; {Вывод второго  корня в Panel3}
   end else if d=0  then begin x1:=-b/2/a;  str(x1:1:2,ss); {Если корень один, то его вычисляют и выводят на экран, второй компонент Panel становится невидимым}
    panel2.Visible:=true;  panel3.Visible:=false;
  panel2.caption:=' Один корень x ='+ss; end else  begin panel2.Visible:=true; panel2.Width:=120;  {При выводе данных изменяется ширина компонента Panel в зависимости от длины выводимого текста}
  panel2.Width:=180;  panel3.Visible:=false;  panel2.caption:=' Действительных корней нет';  {Если корней нет, то это сообщение выводится на экран, при этом меняется ширина компонента Panel }end;
  end;
```

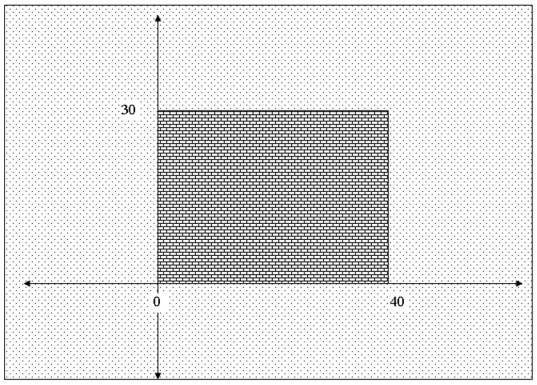
Ниже приводятся экраны расчета корней уравнения для

отрицательного, нулевого и положительного дискриминантов.

При записи сложных условий важно учитывать то, что логические операторы имеют высший приоритет, чем операторы сравнения, и потому простые условия следует брать в скобки.

Например, пусть условие предоставления скидки сформулировано таким образом: «Скидка предоставляется, если сумма покупки превышает 300 руб. и день покупки — воскресенье», Если день недели обозначен как переменная Day целого типа, и равенство ее значения единице отвечает воскресенью, то условие предоставления скидки можно записать:

If (Summa > 300) **and** (Day = 1) then Showmessage (‘Скидка’) else Showmessage (‘ Нет скидки’);



Пусть в зависимости от вводимого значения координаты точки (x, y) следует выдать на экран сообщение: точка попала в прямоугольник или нет. Оператор вывода имеет вид:

If (x>0) and (x<40) and (y<30) and (y>0)

then showmessage (‘Точка находится внутри прямоугольника’) else showmessage (‘Точка вне прямоугольника’);

Выбор в точке разветвления алгоритма дежурного шага программы может быть реализованным с помощью оператора **case**. Если оператор **if**

позволяет выбрать один из двух возможных вариантов, инструкция **case** — один из нескольких.

Множественный выбор может быть реализован с помощью вложенных одна в другую инструкций **if**. Такой подход не всегда удобен, особенно в том случае, если количество вариантов хода программы большое.

В языке Delphi есть оператор **case**, который позволяет эффективно реализовать множественный выбор. В общем виде она записывается таким образом:

case [Селектор] of

список1: оператор 1 ;

список 2: оператор 2;

список M: оператор N

else

оператор

end;

где:

✓ Селектор — выражение, значение которого определяет последующий ход выполнения программы (то есть последовательность инструкций, которая будет выполнена); Селектор должен иметь порядковый тип. Порядковыми типами называются такие типы данных, у которых значения упорядочены и для каждого текущего значения можно указать предыдущее и следующее. Примером являются данные целого типа или данные типа char . Отметим, что такая особенность уменьшает область действия оператора case , т.к. в качестве селектора нельзя использовать дробные числа, текстовые строки, в отличие от оператора if , у которого нет ограничения указанного вида.

✓ Список N — список констант. Если константы являются диапазоном чисел, то вместо списка можно указать первую и последнюю константу диапазона, разделив их двумя точками. Например, список 1, 2, 3, 4, 5, 6 может быть заменен диапазоном 1..6.

Выполняется инструкция **case** таким образом:

1. Сначала вычисляется значение выражения-селектора.

2. Значение выражения-селектора последовательно сравнивается с константами из списков констант.

3. Если значение выражения совпадает с константой из списка, то выполняется соответствующая этому списку группа инструкций. На этом выполнение инструкции **case** завершается.

4. Если значение выражения-селектора не совпадает ни с одной константой из всех списков, то выполняется последовательность инструкций, следующая за else.

Синтаксис инструкции **case** позволяет не писать **else** и соответствующую последовательность инструкций. В этом случае, если значение выражения не совпадает ни с одной константой из всех списков, то выполняется следующая за case инструкция программы.

```
case n   of
1,2,3,4,5: day:='Рабочий день. ' ;
6: day:='Суббота!';
7: day:='Воскресенье!';
end;
```

Допустим, переменная m хранит номер месяца, тогда с помощью **case** легко организовать расчет количества дней в каждом месяце:

```
case m_ of
1,3,5,7,8,10,12: day:=31;
4,6,9.11: day:=30;
2: if  year mod 4 = 0 then day:=29 else day;=28;
 end;
```

Если номер месяца равен 1 или 3, или 5, или 7…, то количество дней в этом месяце равно 31. Аналогично действует проверка на четвертый месяц, шестой и остальные. Исключение составляет февраль. В нем может быть 28 или 29 дней в зависимости от того, является ли год високосным или нет.

В предыдущей теме мы рассматривали арифметические операции, к которым кроме обычных сложения, вычитания, деления и умножения относятся операции div и mod. Div – целочисленное деление, при котором дробная часть отбрасывается без округления. Например, 11 div 3 =3; 27 div 5 =5. Операция mod вычисляет ту часть, которая не делилась при целочисленном делении. 11 mod 3=2, 27 mod 5=2.

Признаком високосного года является деление его порядкового номера на четыре без остатка. 2000 год был високосным и число 2000 делится без остатка на четыре. Это же относится и к 2008 году, а вот 2007 год високосным не был, число 2007/4=501.75. Таким образом, если остаток от целочисленного деления равен нулю, то год был високосным.

Эта проверка реализована с помощью оператора If

if year mod 4 = 0 **then** day:=29 **else** day;=28;

если остаток от деления на четыре равен нулю, то в феврале високосного года 29 дней, в противном случае 28.

Основные положения темы:

- ✓ Для организации различного вида действий в зависимости от проверки условий используется оператор **If**. Он имеет формат **if условие then оператор1 else оператор2.**
- ✓ После слов **then** и **else** разрешается использовать только один оператор. Если требуется выполнить множественную последовательность действий, операторы помещаются в операторные скобки **begin** оператор1, оператор2, оператор3… **end .**
- ✓ Допустимы вложенные операторы **If .**
- ✓ При программировании условий разрешается использовать шесть операций сравнения и три логических оператора.
- ✓ Для организации множественного выбора используют оператор **case** . Оператор **case** позволяет использовать в качестве селектора данные только порядкового типа.

Индивидуальные задания. *Осуществить ввод входных данных, выполнить расчет по формулам для любого значения из отмеченных диапазонов, вывести на экран результаты расчетов за заданными исходными данными, сделать проверочный расчет в Microsoft Excel и сравнить полученные результаты, выполнить предложенные дополнительные условия..*

№ №	Выражения для расчетов	Исходные данные	Дополнительные условия
1	2	3	4
1	$z=\begin{cases}\dfrac{b\ln x}{x}+m\cos x & 2\le x\le 6.5\\ \dfrac{2cx+\sin(mx)}{bx+1} & 6.5<x\le 8.5\end{cases}$ где $b=\sqrt{x^3}$	c=2,7; m=2,4; x=2;	Напечатать, каким является z: (z>0, z<0 z=0).
2	$z=\begin{cases}\dfrac{a\sin\left(\frac{\pi}{2}x\right)+b\cos\left(\frac{\pi}{2}x\right)}{d\sin(\pi x)+\cos(\pi x)} & 0\le x<2.4\\ \dfrac{\lg x}{d(x-6)} & 2.4\le x\le 3.8\end{cases}$ $d=\sqrt{ab}$	a=2,1; b=3,2; x=1	Напечатать, что больше, d или z.
3	$g=\begin{cases}\dfrac{\ln(x+c)}{k(x+7)} & \lvert x\rvert\le 5\\ arctg+x\dfrac{\sin(pp\,x}{2\cos(pp\,x} & \lvert x\rvert>5\end{cases}$ где $c=\sin^5 x$	k=0.2 X=-10	Напечатать что больше c или g .
4	$L=\begin{cases}\dfrac{0.3e^x\sin^3 x}{4.6(x+2)} & 0\le x\le p/2\\ a\cdot\sqrt{b\sin^2(x/2)} & p/2<x\le p\end{cases}$ где $b=\ln a$	a=2.4 X= 0	Напечатать при каких значениях знаменатель

			равняется 0.
5	$C=\begin{cases}\dfrac{x}{5.4-x^2} & x^2+y^2\geq 5.4\\ \dfrac{\sqrt{\lvert x^2+e^{2x}\rvert}}{\sqrt[3]{x^4}} & x^2+y^2<5.4\end{cases}$ где $y=3.2tgx$	X=-5	Напечатать что больше y или c.
6	$F=\begin{cases}\dfrac{ax^3}{96}+\dfrac{5x-3}{\sqrt{\lvert 2x-7\rvert}} & 0\leq x\leq 4\\ (x^2+3)e^x-\dfrac{\ln x}{kx} & 4<x<7\end{cases}$ где $k=0.9-ctgx$	a=2.2 X=0	Напечатать при каких значениях x знаменатель равняется 0
7	$M=\begin{cases}\dfrac{\lg^2(x+c)}{4(x^2+b^2)} & \lvert x\rvert\leq 6\\ \dfrac{\sin 3x+x^4}{tgx} & \lvert x\rvert>6\end{cases}$ где $c=\sqrt{\dfrac{\lvert x\rvert}{ab}}$	b=1.3 X=-10	Напечатать при каких значениях x расчеты невозможны.
8	$\mathcal{Z}=\begin{cases}z-\dfrac{5z^3}{1,7\ln z} & x>3\\ \cos^2(3z) & 2\leq x\leq 3\\ e^{-z} & x<2\end{cases}$ где $z=\sqrt{2x}$	X=1	Напечатать, каким является γ(>γ0, <γ0 =0).

9	$F=\begin{cases} ln\left\|tg(\frac{xy}{c})\right\| & siny>=cosx \\ \frac{ysinx-xcos(cy)}{5c} & siny<cosx \end{cases}$ где y=$\sqrt{\|x\|}$	c=2.375 X=-5	Напечатать, при каких значениях x расчеты невозможны.
10	$R=\begin{cases} \cos^2 x(1+\frac{a-8,3}{bx}) & x\le 5 \\ b(x^3-1) & -5<x\le 1 \\ \lg^2(2x^3)+b & x>1 \end{cases}$ где $b=\cos^3(a+x)$	a=8.73 X=-10	Напечатать пределы интервала, в который попало введенное значение x
11	$Д=\begin{cases} \frac{tg(\cos^2(y+7.2))}{pd} & y\ge pd \\ \ln\left\|arctg(\frac{y}{d})\right\| & y<pd \end{cases}$ где y= *k=1.7-ctg(x+a)*	d=5.2 X=-10	Напечатать, каким является δ (>0, <0 δ=0).
12	$U=\begin{cases} \frac{e^x}{\sqrt{x^2+pa^2}} & (x+a)^2<10 \\ \cos^2(ky) & 10\le(x+a)^2\le 50 \\ e^{-y} & (x+a)^2>50 \end{cases}$ где *k=1.7-ctg(x+a)*	a=2.5 X=0	Напечатать, каким является u: u>0, u<0 u=0.
13	$\varphi = y + \frac{0.48\,x}{1+\cos^2 x}$ где	X= -1.5	Если y>φ, то по окончании расчетов вывести на экран: «y

	$y=\begin{cases} 1+e^{-z} & z>3 \\ \cos(z)+z^2 & -1\le z\le 3 \\ z+\sqrt{\lvert z\rvert} & <-1 \end{cases}$		больше » или «у меньше ».
14	$B=\begin{cases} 4x^2+k\sin^2(x/2) & x<5 \\ \dfrac{5x-3.95e^{2x}}{k\lg x^2} & x\ge 10 \\ 2.5x^2+10\sqrt{\lvert tgx\rvert} & 5\le x<1 \end{cases}$ где k=3 x^3	X=10	Напечатать при каких x числитель равняется 0
15	$e=\dfrac{3\sin(щs+x)}{2-\cos(x-4щ)}$ где $щ=\begin{cases} p/2-sinx & x<-\pi/4 \\ p-lnx & x>p/4 \\ 0 & -p/4\le x\le p/4 \end{cases}$	X=-2.21 k=0.38	Напечатать, что больше, ω или ε.
16	$H=\begin{cases} \sqrt[3]{\lvert t^2+4.8\rvert} & x<6 \\ arctg(t-1.6) & x=6 \\ \dfrac{1}{t\,e} & x>6 \end{cases}$ где $t=x+tg(x\lvert 4)$	X=2.2	Если H>t, то по окончании расчетов вывести на экран: «H больше t» или «H меньше t».
17	$H=\begin{cases} \dfrac{z^3}{2(e^{1.4\omega}+1)} & z<0 \\ 10z+\cos(p\omega) & 0\le z\le 5 \\ \lvert\sin(\pi z/3\rvert-\pi/3\sqrt{\omega} & z>5 \end{cases}$	Z=0.72	Сделать вывод о том, что больше ω или H при заданных начальных

	где w=0.3+z		данных
18	$y=\begin{cases} e^{\alpha}+2.3x^2 & 1<=x<5 \\ \dfrac{tgx-3.7x^2}{\ln\alpha} & x>5 \end{cases}$ где α=0.5x2-6	X=0.5	Напечатать, каким является y: y>0, y<0 y=0.
19	$y=\dfrac{(1,5x-w)^2}{tgw}$ где $W=\sin^4(x/3)$	X=-2	Напечатать, что больше,y или W.
20	$R=\begin{cases} \dfrac{1.5x^2-1}{2S} & 0<=x<=3 \\ \dfrac{6.5\sin(x^2)+2}{\sqrt{\lvert s\rvert}} & x>3 \\ 14\lg(x+4) & x<0 \end{cases}$, $S=x^3+\ln^2\lvert x\rvert+1$	X=-3	Напечатать, что больше, R или S.
21	$v=\begin{cases} \dfrac{\lg^2(x+0.42)-\alpha^x}{5,7-x} & x>3 \\ \dfrac{\sqrt{x^2+2\sin^2(\pi\alpha/3)}}{x^3} & 0.2<=x<=3 \end{cases}$ где α=0.9x3+16	X=1.6	При каком значении X расчеты невозможны
22	$L=\begin{cases} 5,3x^2+0,8b+29 & x<0 \\ \ln(\lvert b\rvert)\div 0,5^{-x} & 0.5<=x<=2.7 \\ \sqrt{\lvert 5,3x/18-3,6b\rvert} & x>2.7 \end{cases}$ где *b=0.5x2-sin2x*	X=-3	Напечатать, при каких X расчеты невозможны.

23	$N=\begin{cases}\ln\dfrac{\sqrt{\lvert x\rvert+a}}{\sin(a+x)} & x<0\\ \sqrt{\cos^3(a+x)} & x>0\end{cases}$ где $\alpha=1.3x^2-4$	X=-3	Напечатать, что больше: N или X.
24	$M=\begin{cases}0.95k^2+4.8\sin x & x<0\\ (0.17k^3+3.6)tgx & 0<=x<=1.5\\ 0.38^{(k+3.9)} & x>1.5\end{cases}$ где *k=5.7x2-6*	X=-2	Напечатать, при каких значениях X расчеты невозможны .
25	$W=\begin{cases}\dfrac{\sqrt{\lvert x^5\rvert-5.7n}}{2x} & x <= -2\\ \dfrac{\sin x+3n}{1+\mathrm{tg}^2 x} & -2 < x <= 1\end{cases}$ где *n=0.2x³-1*	X=-4	Напечатать, каким является w: w>0, w<0 w=0.
26	$\alpha=\begin{cases}\dfrac{1.8+x}{3.72-x}+r^x & x>0\\ \dfrac{\cos(x+\ln(\lvert x\rvert)}{\sqrt{x^2-4.75r}} & x<=0\end{cases}$ где *r=tg2x+0.5x2*	X=-2	Напечатать при каких отрицательных x значениях X знаменатель равняется 0
27	$F=\begin{cases}\ln^3(x-5)+4.3x^2 & 1<=x<5\\ \dfrac{tgx-5.8\sqrt[3]{x}+b}{\ln x} & x>5\end{cases}$ где $b=\cos^5(x+\pi/4)-3/2$	X=0.5	Напечатать, каким является f: f>0, f<0 f=0.

Лабораторная работа № 4 Разработка программ с использованием циклических процессов. Программы обработки одномерных массивов

Цель работы: Изучение основных компонентов, определения их свойств при создании циклического вычислительного процесса в среде разработки Delphi.

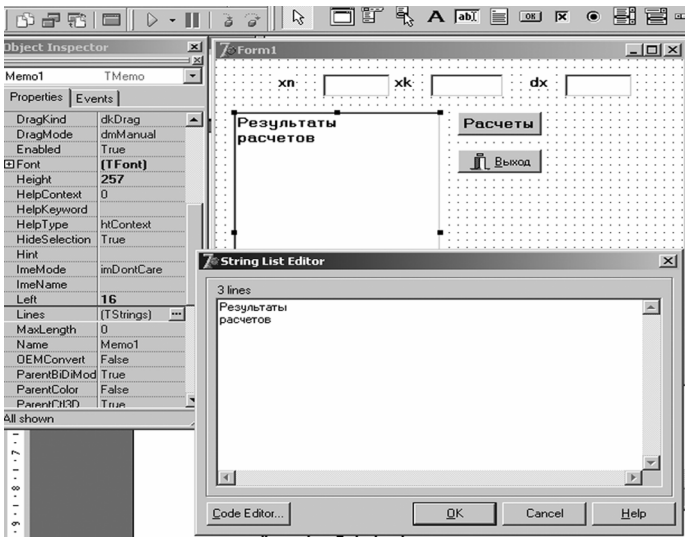
Навыки: Применение операторов цикла **while, for и repeat .** Компонент Stringgrid как базовый элемент при программировании таблиц.

Алгоритмы решения многих заданий являются циклическими, это означает, что для достижения результата определенная последовательность действий должна быть выполненная несколько раз. Например, программа контроля знаний выводит вопрос, принимает ответ, добавляет оценку за ответ к сумме баллов, потом повторяет это действие еще и еще раз, и так до тех пор, пока испытуемый не ответит на все вопросы.

Алгоритм, в котором есть последовательность операций (группа инструкций), которая должна быть выполненная несколько раз, называется циклическим, а сама последовательность операций именуется циклом. В программе цикл может быть реализован с помощью инструкций **for while** и **repeat**.

Задание1. Заданы начальное, конечное значение x и шаг. Определить для каждого x значение y, определяемое по формуле $y= 3x^2 -9$ и вывести значения x и соответствующие y на экран.

На рисунке показана форма с используемыми компонентами (часть1 рисунка). Для ввода начального, конечного значения и шага использованы, как обычно, компоненты Edit, а вот для вывода результатов нами взят новый компонент Memo из стандартной панели компонентов. Для ввода заголовка использовано свойство Lines(часть2 рисунка), при



щелчке по этому свойству открывается окно, в которое можно помещать любой текст (часть3 рисунка).

Допустим, требуется произвести расчеты от -10 до 10 с шагом, равным 2. Если бы мы производили расчеты вручную, то сначала взяли бы значение, равное -10 и определили величину y, затем прибавили бы шаг, для полученного значения x=-8 определили y и так до тех пор, пока x не превысит значение, равное 10.

Для организации такого вычислительного процесса используется оператор цикла с предусловием **while** . Ниже приводится процедура, организующая описанный выше процесс.

```
procedure TForm1.Button1Click(Sender: TObject);
var x,dx,y,xn,xk: real; {Описание локальных переменных}
begin
xn := strtofloat(edit1.Text);
dx:= strtofloat(edit3.Text); {Ввод исходных данных}
xk:= strtofloat(edit2.Text);
x:=xn;                    {Полагаем текущее значение x равным
начальному значению }
while x<=xk do begin   {Условие работы цикла}
y := 3*sqr(x)-9;   {Расчет по формуле}
 memo1.lines.add('x='+floattostr(x)+'   y='+floattostr(y)); {Вывод в
компонент Мемо результатов расчета y}
x :=x + dx; {Прибавление шага к предыдущему значению X}
end;
 end;
```

Оператор while

Оператор **while** используется в том случае, когда некоторую последовательность действий нужно выполнить несколько раз, причем число повторений цикла и может быть определено только во время работы программы.

Типичными примерами использования цикла **while** является вычисление с заданной точностью, поиск в массиве или в файле.

В общем виде оператор while записывается следующим образом:

while [условие] **do begin** { операторы, которые нужно выполнить несколько раз} **end**

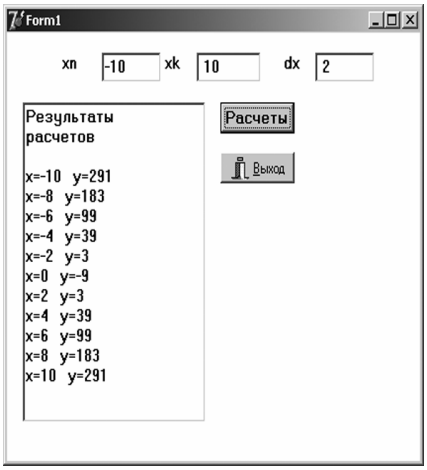
где [условие] — выражение логического типа, который определяет условие выполнения инструкций цикла, программируется по тем же принципам, что и условия в условных операторах. После оператора Do разрешается ставить только один оператор, если их в цикле требуется поставить несколько, используют операторные скобки Begin и End , образуя тем самым составной оператор.

Выполнение оператора while:

1. Сначала проверяется условие.

2. Если условие не выполняется, то на этом работа цикла и оператора while завершается.

3. Если условие выполняется, то последовательно выполняются расположенные между **begin** и **end** операторы тела цикла. После этого опять проверяется выполнение условия. Если условие выполняется, то операторы цикла выполняются еще раз. И так до тех пор, пока условие не станет невыполненным.



Нами приведена копия экрана с результатами выполнения программы. Рассмотрим оператор вывода данных на экран

memo1.lines.add('x='+floattostr(x)+' y='+floattostr(y));

при выводе использован компонент memo1, событие lines.add использовано для добавления новых строк при выводе информации.

Та часть информации, которая помещена в кавычки, выводится на экран без изменений, floattostr(x) – выводит само значение x, знак + использован для объединения текстовых фрагментов.

Отличительной особенностью оператора While является, во-первых, то, что сначала идет проверка условия входа в цикл, а только затем его выполнения и, во – вторых, в качестве параметров цикла (начального, конечного значения и шага цикла), можно использовать любые значения.

Вторым оператором цикла является оператор **Repeat**, который называют оператором цикла с постусловием. Сначала хотя бы один раз выполняется цикл и только затем осуществляется проверка на возможность его повторения.

В общем виде оператор **repeat** записывается таким образом:

Repeat {операторы} **unti1** [условие]

где [условие]— выражение логического типа, который определяет условие завершения цикла. Все операторы, которые необходимо выполнить в цикле помещаются между словами **Repeat и Unti1.** В отличие от оператора While не требуется использование операторных скобок Begin и End.

Изменим постановку задачи, которую мы рассматривали при изучении оператора While. Произвести расчеты y по формуле $y=3\sqrt{x}-9$, причем расчеты прекратить, как только значение x станет отрицательным.

Мы не приводим внешний вид формы, т.к. она выполнена аналогично. Текст программы приводится ниже. Как видно, внутри цикла выполняются три оператора : расчет y, вывод его значений на экран и изменение значения x с заданным шагом.

```
procedure TForm1.Button1Click(Sender: TObject);
var x,dx,y,xn: real;
begin
xn := strtofloat(edit1.Text);
dx:= strtofloat(edit3.Text);
x:=xn;
```

```
repeat
y := 3*sqrt(x)-9;
 memo1.lines.add('x='+floattostr(x)+'   y='+floattostr(y));
x :=x - dx;
 until x<0;
 end;
```

Оператор **for** используется в том случае, если некоторую последовательность действий нужно выполнить несколько раз, причем число повторений предварительно известно.

В общем виде оператор **for** записывается в виде:

for [счетчик] := [нач_знач] **to** [кон_знач] **do begin**

{операторы} **end**

где:

- счетчик — переменная-счетчик числа повторений инструкций цикла;
- нач_знач-- выражение, которое определяет начальное значение счетчика циклов;
- кон_знач — выражение, которое определяет конечное значение счетчика циклов.

Переменная счетчик, начальное и конечное значения должны быть целого типа. Количество повторений инструкций цикла можно вычислить по формуле:

[кон_знач] —[нач_знач]+ 1.

Если между** begin **и** end **находится только один оператор, то слова** begin **и** end **можно не писать.

Постановка задачи: Ввести массив X предварительно задав количество его элементов. Определить элементы массива У по формулам:

$$y_i = 2\sqrt{x_i}, при\_x_i > 0$$

$$y_i = 30 \sin x_i, при\_x_i \le 0$$

Найти элементы массива Z как большего значения их элементов массивов X и У. Найти сумму элементов массива У и произведение его ненулевых элементов.

Массив- это структура данных, позволяющая хранить под одним именем совокупность данных любого, но обязательно одинакового типа. Массив характеризуется именем, типом хранимой информации и количеством его элементов. Массив, как и любая переменная программы, перед использованием должен быть объявлен в разделе описания переменных. В общем виде описание массива выглядит таким образом:

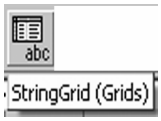
Имя: **array** [нижний_индекс. .верхний_индекс] **of** тип

где:

- имя — имя массива;
- array — зарезервировано слово языка Delphi, который помечает, что объявляемое имя является именем массива;
- нижний_индекс и верхний_и«декс — целые константы, которые определяют диапазон изменения индекса элементов массива и, неявно, количество элементов (размер) массива;
- тип — тип элементов массива.

Примеры описания массивов:

temp:array[1..50] **of** real; коеf:array[0. .4] **of** integer; nik:array[1..20] **of** **string**;

Для работы с массивами часто используют компонент **StringGrid**, который находится 

на странице Additional.

Компонент **StringGrid** является таблицей, которая содержит строки. Данные таблицы могут быть только для чтения или редактируемыми. Таблица может иметь полосы прокрутки, причем заданное число первых строк и столбцов может быть фиксированным и не прокручиваться, они называются фиксированными. Таким образом, можно задать заглавия

столбцов и строк, постоянно присутствующие в окне компонента. Каждой ячейке таблицы может быть поставлен в соответствие некоторый объект.

Основные свойства компонента

1. Cells - строка, содержащая в ячейке с индексами строки и столбцов.
2. Cols - список строк, содержащихся в столбце с индексом Index.
3. Rows - список строк, содержащихся в столбце с индексом Index.
4. Objects - объект, связанный со строкой.
5. Colcount , Rowcount- определяют число столбцов и строк.
6. Fixedcols, Fixedrows – определяют число не прокручиваемых столбцов и строк.

7.Fixedcolor - цвет фона фиксированных ячеек.

8.Leftcol, Toprow - определяют индексы первого видимого на экране в данный момент столбца и строки.

9.Scrollbars - наличие полос прокрутки.

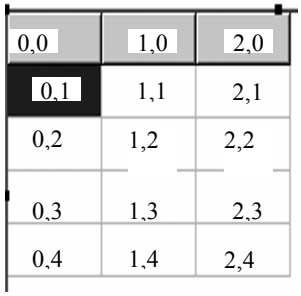
10. Options: - множество свойств:

- Gofixedvertline, Gofixedhorsline - наличие разделителей в фиксированных ячейках;
- Govertline, Gohorsline - наличие разделителей в обычных ячейках;
- Gocolsizing, Gorowsizing- возможность для пользователя с помощью мыши изменять размеры;
- Gocolmoving, Gorowmoving- возможность перемещать столбцы и строки;
- Goediting- возможность редактировать содержимое таблицы.

В основном компонент **StringGrid** используется для выбора пользователем каких-то значений, отображенных в ячейках. Свойства Col и Row показывают индексы столбца и колонки выделенной ячейки. Возможно также выделение пользователем множества ячеек, строк и столбцов.

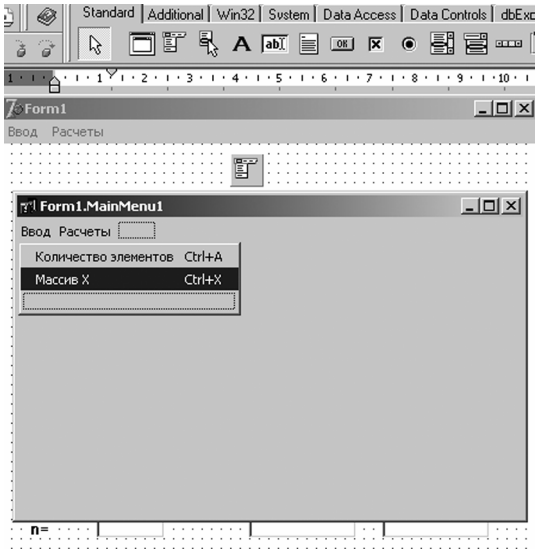
Для решения поставленной выше задачи на форму помещен объект **StringGrid,** причем для него задано количество столбцов Colcount равным

трем, количество строк Rowcount первоначально задается равным пяти. Так как предполагалось, что в первой строке будут находиться заголовки, т.е. названия вводимых и вычисляемых массивов, то свойство Fixedcols=0 (нет столбца как заголовка) и Fixedrows=1 (одна строка используется как заголовок). В наборе свойств Options свойство Goediting=true, это делается для того, чтобы в таблицу при запуске программы можно было вводить исходные данные. Остальные свойства (цвета, шрифт…) задаются по своему усмотрению.



Важным при работе с таблицами является нумерация ее ячеек. Первая ячейка, расположенная в верхнем левом углу таблицы имеет номер 0,0 ; следующая слева направо ячейка нумеруется как 1,0; затем 2,0… Первым в нумерации следует номер столбца от нуля и дальше, вторым номером идет номер строки, также нумеруемой от нуля.

Для оптимальной работы с программой организуем стандартное меню. Для ввода меню используется компонент MainMenu панели Standard.

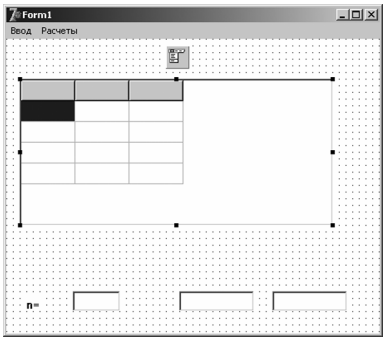


Поместите объект MainMenu на экран в любое место, сам объект при запуске программы не отображается. Вместо него в верхней строке экрана появляется строка меню, как и в стандартном окне Windows. При двойном щелчке по кнопке MainMenu появляется окно для ввода пунктов меню. Пунктирные прямоугольники предназначены для ввода (свойство Caption) названий пунктов меню. Перемещая курсор по пунктирным прямоугольникам то вниз, то налево. Вы создаете дополнительные пункты меню.

Обратите внимание на свойство ShortCut , которое используется для задания клавиш быстрого вызова, открыв список этого свойства, Вы получаете возможность выбрать одну из предлагаемых комбинаций клавиш.

На рисунке показана форма с вынесенными на нее объектами Stringgrid , MainMenu и тремя компонентами Edit, которые будут использоваться для ввода количества элементов в массиве X и вывода результатов определения суммы и произведения элементов.

Если мы хотим, чтобы сразу при загрузке формы появилась таблица с заголовками, то дважды щелкаем по самой форме и в скомпилированной процедуре вводим текст:



procedure TForm1.FormCreate(Sender: TObject);

begin

stringgrid1.Cells[0,0]:='X'; *{Задание заголовка нулевого столбца}*

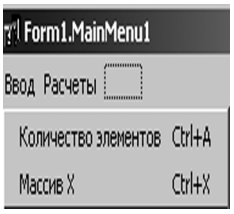
stringgrid1.Cells[1,0]:='Y'; *{Задание заголовка первого столбца}*

stringgrid1.ColWidths[0]:=50; *{Задание ширины нулевого столбца}*

stringgrid1.ColWidths[1]:=150; *{Задание ширины первого столбца}*

end;

Пункт меню **Ввод|Количество элементов** используется для ввода в компонент Edit1 предполагаемого количества элементов и изменения количества строк таблицы в соответствие с введенным значением.



procedure TForm1.N2Click(Sender: TObject);

begin

N:=strtoint(edit1.text); *{Ввод количества элементов}*

stringgrid1.RowCount:=n+1; *{Количество строк в таблице больше на единицу введенного числа, т.к. одна строка отводится под заголовок}*

end;

procedure TForm1.N3Click(Sender: TObject);

var i:integer;begin

```
for i:=1 to n do
x[i]:=strtoint(stringgrid1.Cells[0,i] );
end;
```

Пункт меню **Ввод|Массив X** используется для ввода элементов массива X. Обратите внимание на то, что каждый элемент массива вводится отдельно. Происходит считывание поэлементно содержимого каждой ячейки нулевого столбца и сохранение считанного содержимого в массиве.

```
procedure TForm1.N3Click(Sender: TObject);
var i:integer; {Описание локальных переменных}
begin
for i:=1 to n do{Организация цикла для выполнения повторяющейся}
x[i]:=strtoint(stringgrid1.Cells[0,i] ); {последовательности действий записи
считанных значений в массив}
end;
```

Пункт меню **Расчеты|Массив У** используется для вычисления элементов массива У. В цикле происходит вычисление элементов массива У, преобразование их в строковый тип и запись в первый столбец таблицы.

```
procedure TForm1.N4Click(Sender: TObject);
var i:integer;xx:string; {Описание локальных переменных}
begin
for i:=1 to n do begin{Организация цикла для выполнения
повторяющейся}
if x[i]>0 then y[i]:=2*sqrt(x[i])
else y[i]:=30*sin(x[i]); {последовательности вычислений}
  str(y[i]:1:2,xx); {Преобразование в строковый тип}
stringgrid1.Cells[1,i]:='y['+inttostr(i)+']='+xx; {Заполнение значениями
массива У первого столбца таблицы}
end;  end;
```

Пункт меню **Расчеты|Массив Z** используется для вычисления элементов массива Z. В цикле происходит вычисление элементов массива Z, преобразование их в строковый тип и запись во второй столбец таблицы.

```
procedure TForm1.Z1Click(Sender: TObject);
var i:integer;xx:string; {Описание локальных переменных}
begin
stringgrid1.Cells[2,0]:='Z'; {Создание заголовка второго столбца}
stringgrid1.ColWidths[2]:=150;
for i:=1 to n do     begin
if x[i]>y[i] then z[i]:=x[i] else z[i]:=y[i]; {Вычисление элементов массива Z}
str(z[i]:1:2,xx); {Преобразование в строковый тип}
stringgrid1.Cells[2,i]:='z['+inttostr(i)+']='+ xx; {Заполнение значениями массива Z второго столбца таблицы}
end;     end;
```

Пункт меню **Расчеты|Сумма** используется для вычисления суммы элементов массива У. Для вычисления суммы сначала устанавливают ее значение равным нулю. Это позволяет выполнять вычисления многократно, не прибавляя к предыдущему значению следующие. В цикле происходит накапливание значений суммы. Сначала s=0, затем к этому значению прибавляется первое значение элемента массива У, т.е s:=s+y[1], затем к уже существующему значению добавляется еще одно s:=s+y[1]+y[2] и так n раз, которые работает цикл.

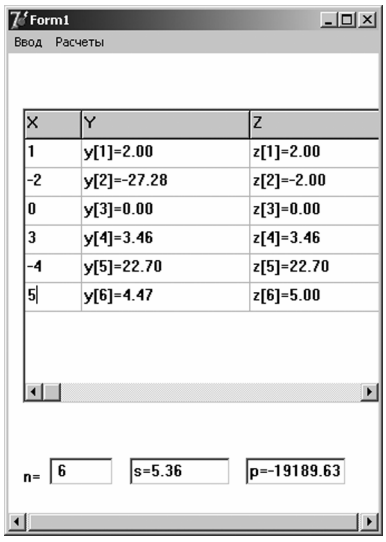
```
procedure TForm1.N6Click(Sender: TObject);
var i:integer;xx:string;  s:real; {Описание локальных переменных}
begin
s:=0;  {Установка начального значения суммы равным нулю}
 for i:=1 to n do
 s:=s+y[i]; {Накапливание суммы}
      str(s:1:2,xx); {Преобразование в строковый тип}
```

```
edit2.Text:='s='+xx; {Выдача результата в компонент Edi.}
end;
```

Пункт меню **Расчеты|Произведение** используется для вычисления произведения ненулевых элементов массива У. Для вычисления произведения сначала устанавливают его значение равным единице. Если бы, как и в предыдущем случае значение установили равным нулю, это привело бы к нулевому результату вычислений, т.к. умножение на ноль дает в результате ноль. В цикле происходит накапливание значений произведения. Сначала p=1, затем к этому значению прибавляется первое значение элемента массива У, т.е p:=p*y[1], затем к уже существующему значению добавляется еще одно p:=p*y[1]*y[2] и так n раз, пока работает цикл. Отличие от предыдущей процедуры заключается в том, что предварительно происходит проверка на равенство нулю элементов.

```
procedure TForm1.N7Click(Sender: TObject);
var i:integer;xx:string;   p:real; {Описание локальных переменных}
begin
p:=1;   for i:=1 to n do
  if y[i]<>0 {Произведение только ненулевых элементов}
then  p:=p*y[i]; {Накапливание произведения}
      str(p:1:2,xx); {Преобразование в строковый тип}
edit3.Text:='p='+xx; {Выдача результата в компонент Edit}
end;
```

Часть переменных была описана как локальные в каждой процедуре. Это означало, что они будут использованы только в этой процедуре. Параметр переменной цикла, в наших процедурах это i должен обязательно описан в той процедуре, в какой он используется. Массивы X,Y,Z описываются как глобальные, т.к. они используются в нескольких процедурах.



public n:integer;x:array[1..20] of integer; *{Описание целочисленного массива X}*

y,z:array[1..20] of real; *{Описание вещественных массивов Y и Z}*

На рисунке представлены результаты работы программы. Значение n было введено равным шести. Что привело к увеличению количества строк в таблице. После ввода массива X сформированы два других массива и рассчитаны сумма и произведение.

Иногда, по условию задачи цикл необходимо прервать, не выполнив заданного количества повторений. Для этого можно использовать операторы Break и Exit. Оператор Break используется для прерывания любого цикла, организованного операторами For , Repeat или While. Оператор Exit используется для прерывания не только цикла, но и всей программы.

Например, найти первый положительный элемент массива X.

procedure TForm1.N3Click(Sender: TObject);

var i,n; *{Описание цпеременных i,n}*

x:array[1..20] of integer; *{Описание целочисленного массива X}*

begin

for i:=1 to n do

x[i]:=strtoint(stringgrid1.Cells[0,i]); *{Ввод целочисленного массива X}*

if x[i]> 0 then begin edt1.text:=inttostr(x[i]); break;end; *{Если элемент массива X больше нуля, он выдается в компонент Edit1 и выполнение цикла прекращается}*

end;

Основные положения темы:

✓ Для организации циклических процессов используются операторы цикла **repeat** (цикл с постусловием) и **while** (цикл с предусловием).

- ✓ После слова **do в** операторе **While** разрешается использовать только один оператор. Если требуется выполнить множественную последовательность действий, операторы помещаются в операторные скобки **begin** оператор1, оператор2, оператор3… **end .**
- ✓ В качестве параметров цикла могут выступать как целые, так и вещественные числа.
- ✓ Для вывода информации в несколько строк можно использовать компонент **Memo**.
- ✓ Оператор **For** используется для организации цикла с известным числом повторений, при этом параметры цикла (начальное и конечное значения, переменная цикла) должны быть описаны как целые.
- ✓ Для работы с совокупностью элементов одного типа можно использовать такую структуру как массив. Массив должен быть описан в виде x:array[1..20] of integer; для целочисленных типов и как У:array[1..20] of real ; для вещественных.
- ✓ Для работы с массивами удобно использовать компонент **StringGrid (таблицу).**
- ✓ Для удобства выполнения действий в программе можно организовать меню, для этого используют компонент **MainMenu**.
- ✓ Для экстренного выхода из цикла применяют оператор **Break** (только прекращение цикла и продолжение выполнения основной программы) и оператор **Exit** (прекращение выполнения не только цикла, но и программы).

Индивидуальные задания. *Составить программу для обработки одномерных массивов. Для ввода исходных данных и записи полученных результатов расчета использовать компонент StringGrid. Ответ на дополнительные вопросы вывести с помощью компонента Panel. Использовать компонент MainMenu.*

N	Выражение	Исходные данные массива	Данные для вывода	Дополнительные вопросы
1	2	3	4	5
1.	$B_i = \begin{cases} \sqrt{2 \sin A_i + 0.5} & , A_i \le 2 \\ 4.5 A_i + \sqrt[5]{A_i^3} & , A_i > 2 \end{cases}$	i = 1 ÷ 20	Массивы A, B	Суммы элементов для обоих массивов, минимальное значение массива B
2.	$Z_i = \begin{cases} \dfrac{\sqrt{\lvert A_i \rvert \sin(\pi A_i)}}{4 A_i} & , A_i < 1.5 \\ A_i + e^{A_i + 5} & , A_i \ge 1.5 \end{cases}$	i = 1 ÷ 10	Массивы A, Z	Разницу между наибольшим и наименьшим значением массива Z
3.	$D_i = \begin{cases} 4.7^{\sin^3(X_i)} + \ln(X_i^2) & 5 \le X_i \le 15 \\ e^{X_i} + \sqrt[3]{X_i} & 0 < X_i < 5 \\ X_i + 3.2 & X_i \le 0 \text{ или } X_i > 15 \end{cases}$	i = 1 ÷ 25	Массивы Д, X	Количество отрицательных (<0) элементов массива *Д*, наибольшее значение элементов

				массива X
4.	$Y_i = \begin{cases} \sqrt{\lvert X_i \rvert} - \dfrac{X_i^2}{X_i + 1} - 0.2 & , X_i < 1.5 \\ \sin^2(\pi X_i) - X_i & , X_i \geq 1.5 \end{cases}$	i = 1 ÷ 20	Массивы X, B	Номера максимального и минимального элементов массива B
5.	$Z_i = \begin{cases} Y_i - 0.3\dfrac{Y_i^2}{Y_i + 1} & , Y_i > 1 \\ \dfrac{1}{2}\cos(Y_i) & , -1 \leq Y_i > 1 \\ 2\sin(Y_i) & , Y_i < -1 \end{cases}$	i = 1 ÷ 20	Массивы B, Z	Суммы элементов для отмеченных массивов, наибольшее значение массива B
6.	$Z_i = \begin{cases} \lvert 1 - B_i \cos(B_i) \rvert & , B_i > 0.2 \\ \sqrt{1 + B_i^2} & , B_i \leq 0.2 \end{cases}$	i = 1 ÷ 10	Массивы B, Z	Процент отрицательных (<0) элементов, суммы всех элементов массива Z
7.	$Y_i = \begin{cases} 20X_i - 0.27 & , X_i < -1.5 \\ X_i^3 + 5.5 & , -1.5 \leq X_i < 1.5 \\ 4\sin^2(X_i) & , X_i \geq 1.5 \end{cases}$	i = 1 ÷ 15	Массивы X, B	Среднее арифметическое значение

				элементов массива B, сумму элементов массива X
8.	$B_i = \begin{cases} Y_i + \dfrac{0.49Y_i}{2+\cos(Y_i)} & , \ X_i < -2.5 \\ \dfrac{1.5}{X_i} + \dfrac{X_i}{X_i^2 - 1} & , \ X_i \geq 2.5 \end{cases}$	i = 1 ÷ 10	Массивы B, Z	Произве-дение ненулевых элементов и наимень-шее значение элементов массива B
9.	$Y_i = \begin{cases} 1 + e^{-T_i} & T_i > 3 \\ \cos(T_i) + T_i & -1 \leq T_i \leq 3 \\ T_i + \sqrt{\lvert T_i \rvert} & T_i < -1 \end{cases}$	i = 1 ÷ 12	Массивы T, B	Суммы элементов для массивов, наимень-шее значение массива B
10.	$Z_i = \begin{cases} \ln\lvert Y_i \rvert + 1 & Y_i > 0.15 \\ Y_i / 2 + \sqrt{2\sin \pi Y_i} & Y_i \leq 0.15 \end{cases}$	i = 1 ÷ 10	Массивы B, Z	Количество элементов массива Y, которые больше среднего арифмети-ческого

				массива Z.
11.	$F_i = \begin{cases} 10\sqrt{A_i^2 + 1} - e^{\frac{2A_i}{1.7}} \\ A_i - \sqrt{\ln(A_i)} \end{cases}$ $A_i \le 6$ $A_i > 6$	i = 1 ÷ 20	Массивы A, F	Сумму элементов массива F и минимальный элемент массива A.
12.	$R_i = \begin{cases} Y_i - Y_i\sqrt{1 + 0.5\cos(Y_i)} & Y_i < 1.5 \\ 2\sqrt{1 + e^{0.5Y_i}} & Y_i \ge 1.5 \end{cases}$	i= 1 ÷ 15	Массивы Y, R	Номера максимального и минимального элементов массива R
13.	$R_i = \begin{cases} \frac{1.5X_i^2}{2 + ctg(X_i)} + \lvert\sin(\pi X_i)\rvert & X_i \le 0 \\ X_i + e^{\frac{1}{X_i}} & X_i > 0 \end{cases}$	i = 1 ÷ 15	Массивы X, R	Номера максимального и минимального элементов массива R
14.	$Z_i = \begin{cases} \frac{1.5X_i^2}{2.3 - X_i} - 4.6\lvert\sin(\pi X_i)\rvert & X_i \ge 0 \\ 6.5X_i - 1/2\left(\sqrt{X_i^2 + 1}\right) & X_i < 0 \end{cases}$	i = 1 ÷ 10	Массивы X, Z	Сумму положительных (>0) элементов массива X

				и количество отрицательных (<0) элементов массива Z.
15.	$T_i = \begin{cases} \dfrac{3\sin(\omega_i + 2.2)}{7.3 + \cos(\pi - \omega_i^3)} & , \ \omega_i > 0.2 \\ 2\omega_i + \sqrt{e^{\omega_i}} & , \ \omega_i \le 0.2 \end{cases}$	i = 1 ÷ 15	Массивы, T	Среднее арифметическое значение элементов массива T, сумму элементов массива ω
16.	$\sigma_i = \begin{cases} \dfrac{\pi}{3\sin(\pi X_i/3)} + 5\cos^2\{X_i\} & X_i \ge 0 \\ \sqrt{1+X_i^2} + (1-X_i)^{\frac{2}{3}} & X_i < 0 \end{cases}$	i = 1 ÷ 20	Массивы X –	Количество элементов массива Y, которые больше среднего арифметического массива Z.
17.	$B_i = \begin{cases} 0.5A_i + Sin(A_i) & A_i < 0 \\ 4\lvert A_i \rvert^{0.6} + 15 & 0 \le A_i \le 10 \\ 10 / A_i^2 - 5e & A_i > 10 \end{cases}$	i = 1 ÷ 12	Массивы A, B	Разницу между наибольшим и наимень-

				шим значением массива Z
18.	$F_i = \begin{cases} 2 + 3.1\left(X_i\right)\sin\left(X_i\right) \\ 1 + \cos^2\left(\pi / 2X_i\right) \\ X_i - \sqrt{\ln\left(X_i\right)} \end{cases}$ $X_i < 0$ $0 \le X_i \le 10$ $X_i > 10$	i = 1 ÷ 15	Массивы X, F	Номера мак-симального и минималь-ного элементов массива F
19.	$Z_i = \begin{cases} \dfrac{1}{\sqrt{\left\|X_i^{\,5}\right\| tgX_i + 1.5}} & X_i > 0.5 \\ 10\sqrt{X_i^{\,2} + 1} + e^{X_i} & X_i < 0.5 \end{cases}$	i = 1 ÷ 10	Массивы X, Z	Произведе ние
20.	$Z_i = \begin{cases} 3Y_i + Y_i\sqrt{1 - \sin\left(\dfrac{\pi}{2}Y_i\right)} & Y_i > 0.5 \\ 3\ln\left(1 + e^{Y_i}\right) & Y_i < 0.5 \end{cases}$	i = 1 ÷ 15	Массивы B, Z	ненулевых элементов и наиболь-шее значение массива Z
21.	$Q_i = \begin{cases} e^{2X_i} \Big/ \left(1 + \ln\left\|2X_i\right\|\right) & X_i \ge 0 \\ \sqrt{\left\|\sin\left(X_i\right) - \cos\left(X_i\right)\right\|}; & X_i < 0 \end{cases}$	i = 1 ÷ 20	Массивы X, Q	Сумму всех элементов массива Q и разницу между наиболь-шим и наимень-

				шим значением массива X
22.	$L_i = \begin{cases} \dfrac{N_i + 0.1N_i}{0.1\left(N_i - 0.1N_i\right)} & 0 \le N_i \le 5 \\ arctg\left(2N_i\right) & N_i > 5 \\ 0.2 - \dfrac{X_i^{\,2}}{0.6 + 0.01X_i} & N_i < 0 \end{cases}$	i = 1 ÷ 10	Массивы L, N	Сумму элементов массива L и наибольший элемент массива N.
23.	$Y_i = \begin{cases} \sin^2\left(\pi X_i\right) + 5\cos\left(\dfrac{\pi}{3}X_i\right) & X_i \ge 0 \\ 2tg\sqrt{X_i^{\,2} + 7} & X_i < 0 \end{cases}$	i = 1 ÷ 20	Массивы X, B	Количество отрицательных (<0) элементов массива *B*, наибольшее значение элементов массива X
24.	$A_i = \begin{cases} \dfrac{2}{7}Y_i - Y_i\sqrt{1 + 0.5\cos\left(Y_i\right)} & Y_i < 0 \\ 2\sqrt{1 + e^{0.5Y_i}} & Y_i \ge 0 \end{cases}$	i = 1 ÷ 14	Массивы B, A	Сумму положительных (>0) элементов и количество отрицательных (<0) элементов массива A.

25.	$Z_i = \begin{cases} 2 + 2(X_i)\sin(X_i) & X_i < 0 \\ 1 + \cos^2(\pi X_i / 2) & 0 \le X_i \le 10 \\ X_i - \sqrt{\ln(X_i)} & X_i > 10 \end{cases}$	i = 1 ÷ 20	Массивы X, Z	Количество элементов массива X, которые больше среднего арифметического массива Z.
26.	$B_i = \begin{cases} 3.5A_i + Sin(\frac{\pi}{4} A_i) \\ 2.2\lvert A_i \rvert + 0.5 \\ 10 / A_i^2 - 5e \end{cases}$ $A_i < 0$ $0 \le A_i \le 10$ $A_i > 10$	i = 1 ÷ 10	Массивы A, B	Разницу между наибольшим и наименьшим значением массива B
27.	$D_i = \begin{cases} 2X_i + 1.4(X_i) + \sin(X_i) & X_i < 0 \\ 1 + \cos^2(\pi / 2X_i) & 0 \le X_i \le 10 \\ X_i - \sqrt{\ln(X_i)} & X_i > 10 \end{cases}$	и = 1 ÷ 10	Массивы X, D	Сумму элементов массива L и наибольший элемент массива N.

Лабораторная работа № 5 Графическое представление информации в среде разработки Delphi

Цель работы: Изучение основных компонентов и методов графического представления информации в среде разработки Delphi.

Навыки: Навыки работы со свойством Canvas.

Delphi позволяет разрабатывать программы, которые могут выводить графику: схемы, чертежи, иллюстрации. Программа выводит графику на поверхность объекта (формы или компонента Image). Поверхности объекта отвечает свойство canvas. Для того, чтобы вывести на экран графический элемент (прямую линию, круг, прямоугольник и т. д.), необходимо применить к свойству canvas этого объекта соответствующий метод. Свойства позволяют задать характеристики графических примитивов, которые выводятся: цвет, толщину и стиль линий; цвет и вид заполнения областей; характеристики шрифта при выведении текстовой информации.

Методы выведения графических примитивов рассматривают свойство Canvas как некоторое абстрактное полотно, на котором они могут рисовать (canvas переводится как «холст», «полотно для рисования»). Полотно состоит из отдельных точек — пикселей. Положение пикселя характеризуется его горизонтальной (X) и вертикальной (Y) координатами. Левый верхний пиксель имеет координаты (0, 0). Координаты растут сверху вниз и слева направо. Значения координат правой нижней точки полотна зависят от размера холста.

Размер полотна можно получить, обратившись к свойствам Height и Width области иллюстрации (Image) или к свойствам формы**:** ClientHeight и Clientwidth.

Карандаш (Pen) применяется для вычеркивания линий и контуров на полотне, а кисть (Brush) — для закрашивания областей, ограниченных контурами.Значения свойств этих объектов определяют вид графических элементов, которые выводятся.

Свойство Pen (карандаш или перо)

PEN имеет четыре свойства, которые можно изменить: Цвет, Ширина, Стиль, и Метод.

Color- : Изменяет цвет пера ;

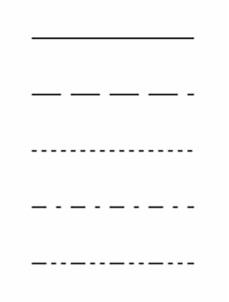
Width-: Изменяет ширину пера ;

Style -: Изменяет стиль пера ;

Mode - Изменяет метод пера ;

По умолчанию, т.е., если не задавать свойства, рисование происходит черным карандашом толщиной равной 1.

Pen.Color - Определяет цвет, используемый для изображения. Чтобы изменить цвет применяют оператор Canvas.Pen.Color := RGB(255,0,0), где выбор цветов осуществляется по технологии RGB, описанной в первой теме. В нашем примере выбран карандаш красного цвета.



Pen.Style -Определяет стиль, которым рисуются линии.

psSolid Сплошная линия.

psDash Линия из серии разрывов

psDot Линия из точек

psDashDot Линия из разрывов и точек

psDashDotDot Линия вида точка, точка, разрыв.

psClear Линия не изображается (использованный, чтобы пренебречь линию вокруг форм, которые вытягивают контур, используя текущее перо).

Например, Canvas.Pen.Style := psDashDotDot; задает изображение фигуры карандашом в виде двух точек и разрыва.

Pen.Width – Задает толщину карандаша.

Canvas.Pen.Width:=5;

Если значение свойства Canvas.Pen.width больше единицы, то пунктирные линии будут выведены как сплошные.

Свойство brush

Свойство позволяет заполнить области, включая внутреннюю часть форм. Brush имеет три свойства:

Color: Изменяет цвет;

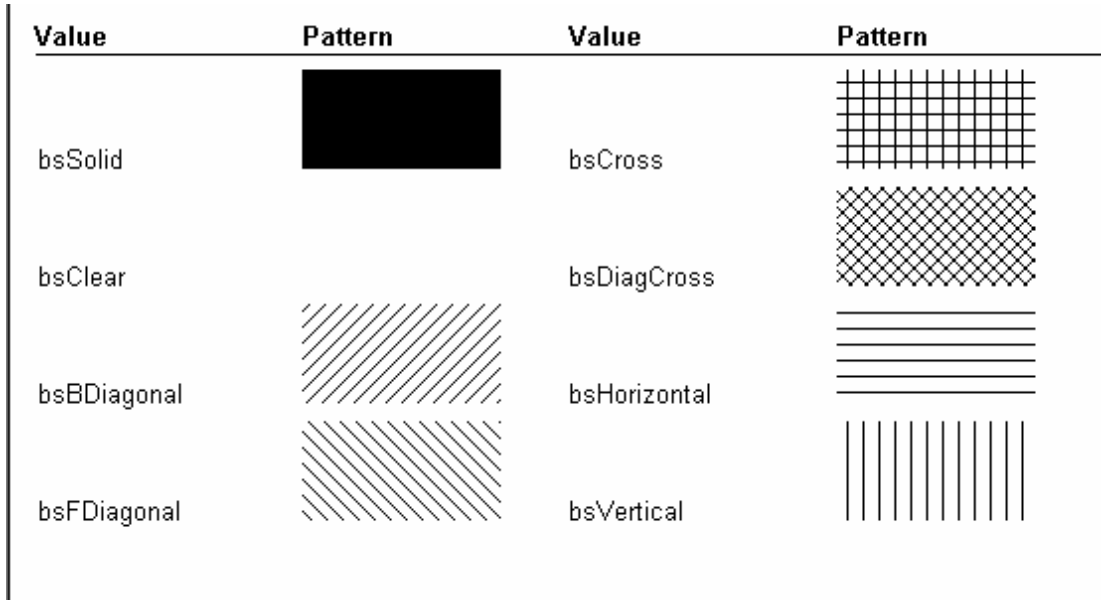
Style: Изменяет стиль заполнения области;

Bitmap: Использует точечные рисунки в качестве образца Brush.

Canvas.brush.color:=rgb(0,255,0); задает зеленый цвет заполняющий фигуру.

Brush.Style

. Возможные варианты заполнения областей показаны ниже.



Оператор Canvas.brush.style := bsDiagCross ; задает последующее заполнение области в виде решетки.

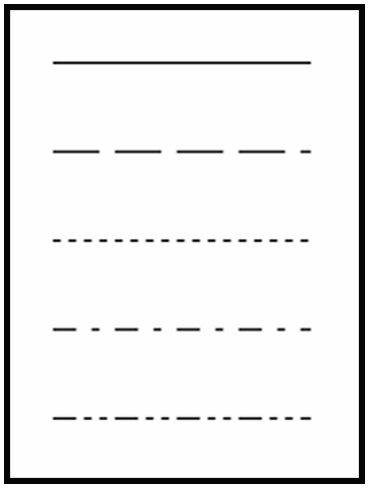
Canvas.MoveTo(x, y);

Используется для установки карандаша в позиции, определенные координатами x,y, где x, y целые числа.

LineTo(x, y); передвигает текущую позицию к конечной точке линии.

Процедура изображает набор линии разного стиля

```
procedure TForm1.N12Click(Sender: TObject);
var i:integer;
begin  for i:=0 to 4 do    begin
case i of
   0: Canvas.Pen.Style := psSolid;
   1: Canvas.Pen.Style := psDash;
   2: Canvas.Pen.Style := psDot;
   3: Canvas.Pen.Style := psDashDot;
```



```
    4: Canvas.Pen.Style := psDashDotDot;  end;
    Canvas.moveto(50,50+i*40 );
    Canvas. lineto( 150  , 50+i*40);
  end;
 end;
```

В программе организован цикл, позволяющий с помощью оператора Case перебрать всевозможные варианты стиля карандаша. Рисуется пять линий единичной толщины, смещенных по вертикали на 40 пикселей относительно друг друга.

Canvas.Ellipse

procedure Ellipse(X1, Y1, X2, Y2: Integer);

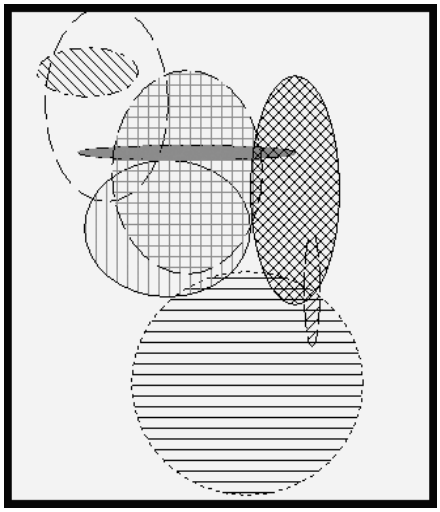
Изображает круг или эллипс. Верхняя левая точка прямоугольника в координатах (X1, Y1) элемента изображения и правая точка есть (X2, Y2). Если точки прямоугольника формируют квадрат, изображается круг.

Программа выполняет следующие функции:

- ✓ При включении таймера каждые пятьдесят миллисекунд рисуется эллипс с произвольными координатами.
- ✓ Для вычисления произвольных координат использован оператор RND (N) – генерация случайного числа целого типа в интервале от 0 до N. Оператор Randomize включает генератор случайных чисел, в противном случае при каждом новом запуске программы генерировались бы одни и те же случайные числа.
- ✓ Чтобы не загромождать экран большим количеством эллипсов, установлен ограничитель, реализованный в виде if k>8 then exit; Если количество эллипсов превысит восемь, выполнение программы будет остановлено.
- ✓ Выбор стиля заливки и карандаша организуется случайным образом.

```
procedure TForm1.Timer1Timer(Sender: TObject);
Var   X, Y: Integer;
```

```
begin
 Timer1.Interval := 50;   Randomize; k:=k+1;     if k>8 then exit;
 X := Random(form1.Width - 200);   Y := Random(form1.Height - 200);
 Canvas.Pen.Color := Rgb(0,0,0);Canvas.Pen.width := 1;
if k mod 3=0 then
Canvas.brush.Color:=rgb(255,0,0)  else
 if k mod 3=1 then
Canvas.brush.Color:=rgb(0,255,0)  else
  Canvas.brush.Color:=rgb(0,0,255) ;
 Canvas.brush.style := bsDiagCross;
case Random(5) of
  0: Canvas.Pen.Style := psSolid;
  1: Canvas.Pen.Style := psDash;
  2: Canvas.Pen.Style := psDot;
  3: Canvas.Pen.Style := psDashDot;
  4: Canvas.Pen.Style := psDashDotDot;  end;
  case k of
1: Canvas.brush.style := bsSolid   ;
2: Canvas.brush.style := bsClear  ;
3: Canvas.brush.style := bsHorizontal  ;
4:  Canvas.brush.style := bsVertical   ;
5:  Canvas.brush.style := bsFDiagonal   ;
6:  Canvas.brush.style := bsBDiagonal   ;
7:  Canvas.brush.style := bsCross   ;
8: Canvas.brush.style := bsDiagCross   ;    end;
Canvas.ellipse(X, Y, X + 10+Random(200), Y + 10+
Random(200));
end;
```



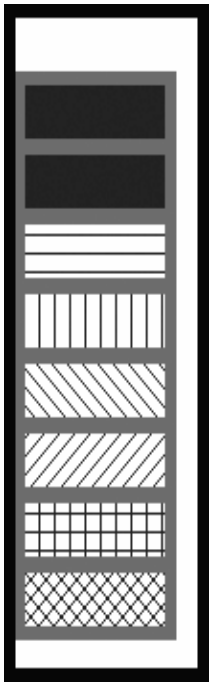
Canvas. Rectangle

Rectangle(X1, Y1, X2, Y2);

Изображает квадрат или прямоугольник. Верхняя левая точка прямоугольника в координатах (X1, Y1) элемента изображения и правая точка есть (X2, Y2).

Приведенная ниже программа демонстрирует изображение нескольких прямоугольников разного типа заливки. Каждый прямоугольник обведен черной линией с толщиной, равной шести.

```
procedure TForm1.N10Click(Sender: TObject);
var i:integer;
begin     timer1.Enabled:=false;
for i:=1 to 8 do    begin
case i of
  1: Canvas.brush.style := bsSolid   ;
  2: Canvas.brush.style := bsClear  ;
  3: Canvas.brush.style := bsHorizontal  ;
  4:  Canvas.brush.style := bsVertical   ;
   5:  Canvas.brush.style := bsFDiagonal   ;
   6:  Canvas.brush.style := bsBDiagonal   ;
   7:  Canvas.brush.style := bsCross   ;
   8: Canvas.brush.style := bsDiagCross   ;    end;
     Canvas.brush.Color:=rgb(0,0,255) ;
      Canvas.pen.width:=6;
Canvas.pen.color:=rgb(255,0,0);
   Canvas.rectangle(10,50+i*30,90,i*30+ 80);
end; end;
```



Изображение закругленных прямоугольников.

Чтобы изобразить закругленный прямоугольник , используется метод RoundRect. Первые четыре параметра, передаваемые в RoundRect, являются

координатами прямоугольника, следующие два параметра указывают на то, как изобразить закругленные углы.

На примере изображен закругленный прямоугольник в верхнем левом квадрате формы, закругляя углы, как секции круга с диаметром 10 pixels: Canvas.RoundRect(0, 0, 200 , 250, 10, 10);

Изображение дуг

Procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);

Arc рисует дугу, состоящую из эллипса, ограниченного точками (X1,Y1) and (X2,Y2). Рисуется дуга против часовой стрелки от начальной точки, которая определяется пересечением эллипса и точки (X3,Y3) и конечной точки (X4, Y4).

Ломаная замкнутая линия

Procedure Polygon([Point(10, 10), Point(30, 10), Point(130, 30), Point(240, 120)]);

Рисует замкнутую ломаную линию с координатами точек, определенных в Point.

Вывод текста в графическом режиме

Для выведения текста на поверхность графического объекта используется метод TextOut. Инструкция вызова метода TextOut в общем виде выглядит таким образом:

Canvas.TextOut(X, Y, Текст)

где:

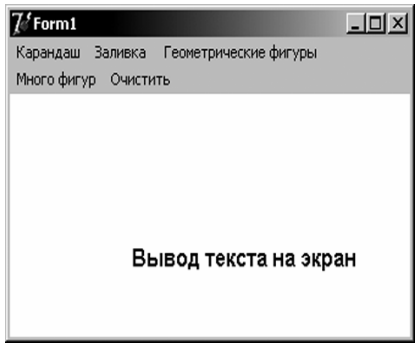
- **X, Y** — координаты точки графической поверхности, от которой выполняется выведение текста.
- Текст — переменная или константа символьного типа, значение которой определяет текст, который выводится методом.

Шрифт, который используется для выведения текста, определяется значением свойства **Font** соответствующего объекта **canvas**.

Свойства объекта TFont

Name	Как значение следует использовать название шрифта, например? Arial
Size	Размер шрифта в пунктах (points). Пункт— это единица измерения размера шрифта, используемая в полиграфии. Один пункт равен 1|72 дюймы
Style	Стиль изображения символов: нормальным, полужирным (fsBold), курсивным (fsltalic), подчеркнутым (fsUnderline), перечеркнутым (fsStrikeOut). Свойство Style позволяет комбинировать необходимые стили. Например, инструкция
Color	программы, которая устанавливает стиль «полужирный курсив», выглядит так:
	Canvas . Font : = [fsBold, fs Italic];

Ниже приводится текст программы, выводящий на экран надпись и результаты ее работы.




```
procedure TForm1.N7Click(Sender: TObject);
        begin   canvas. font.name:='Arial';
         canvas. font.size:=12;
            canvas. font.style:=[fsBold];
             canvas. font.color:=rgb(0,0,255)  ;
        canvas.TextOut(100,100,'Вывод текста на экран');
        end;
```

Текст выводится красным цветом, размером 12, жирным и стилем Arial.

Для очистки экрана от изображений используется команда **Refresh**.

Основные положения темы:

✓Для создания графических изображений в среде Delphi можно использовать свойство **Canvas**.

✓Методы **Canvas** позволяют варьировать при изображении цветом, толщиной и стилем карандаша; выбирать заливку объектов разного цвета и вида штриховки.

✓При создании изображений необходимо учитывать особенности системы координат. Центр координат располагается в левом верхнем углу экрана, таким образом ось X направлена, как обычно, слева направо, а вот ось У сверху вниз.

✓При задании координат объектов используются только целые положительные числа.

✓Оператор **Refresh** используется для очистки экрана.

✓Для задания случайных значений чисел используются операторы **RND и Randomize.**

Индивидуальные задания. *Создать программу для построения изображений с использованием меню, предусмотреть изменение цвета, стиля, подписей.*

1. Создать изображение круга и вписанного в него треугольника.
2. Создать изображение куба параллелепипеда.
3. Создать изображение нескольких звездочек разного размера.
4. Создать изображение эллипса и указать его диаметры.
5. Создать изображение однополостного гиперболоида.
6. Создать изображение двуполостного гиперболоида.
7. Создать изображение эллиптического параболоида.
8. Создать изображение гиперболического параболоида.
9. Создать изображение детской пирамиды.
10. Создать изображение треугольника, указав его высоту, медиану и биссектрису.
11. Создать изображение трапеции и вписать в нее эллипс.

12. Создать изображение трех концентрических окружностей с указанием их радиусов.

13. Создать изображение песочных часов.

14. Создать изображение пирамиды.

15. Создать изображение координатных осей и эллипса с центром в точке (0,0).

16. Создать изображение прямоугольного, равнобедренного и равностороннего треугольников.

17. Создать изображение окружности, вписанной в треугольник.

18. Создать изображение пары пересекающихся плоскостей.

19. Создать изображение пары параллельных плоскостей.

20. Создать изображение цилиндра, указав его оси и радиус.

21 .Создать изображение ромба, приведя координатные оси с соответствующими подписями.

22. Создать изображение параболической трапеции.

23. Создать изображение циклоиды.

24. Создать изображение эллипса и провести к нему в трех точках касательные.

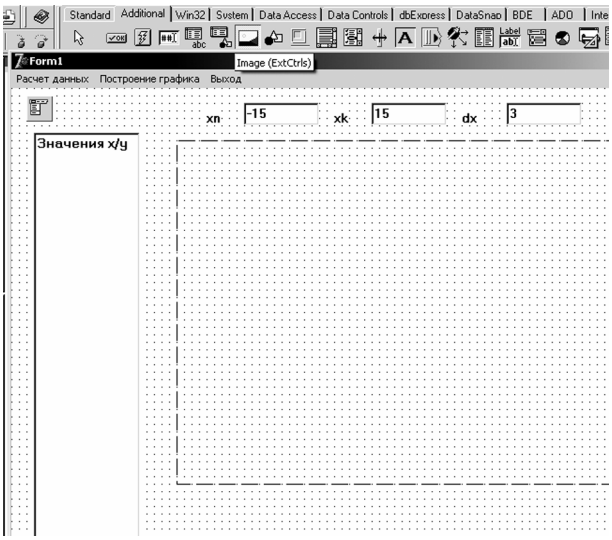
25. Создать изображение вписанных квадратов.

Лабораторная работа № 6 Построение графиков функций

Цель работы: Изучение основных принципов построения графиков функций в среде программирования Delphi.

Навыки: Расчет координат точек при использовании компонента Image.

Достаточно часто результаты расчетов удобно представить в виде графика. Для большей информативности и наглядности графики изображают на фоне координатных осей и оцифрованной сетки.



На рисунке показана форма с объектами, необходимыми для построения графика. Нами использован компонент MainMenu для организации меню, состоящего из трех пунктов: «**Расчет данных»** для определения координат точек графика; **«Построение графика»** как основная часть программы и пункта **«Выход»** для завершения работы с программой.

Для графика использован ранее неизвестный Вам компонент **Image** панели Additional. Этот компонент используется как полотно для рисования с помощью метода Canvas, для загрузки графических файлов, если они вставляются в программу из внешнего источника. Основным свойством является свойство Picture, определяющее какой файл вставляется в приложение.

В нашем случае не будет использовано это свойство, т.к. мы создаем изображение, а не берем готовый файл.

Для ввода начальных, конечных значений X взяты три компонента Edit.

«Расчет данных». При щелчке по этому пункту меню выполняются следующие действия:

- ✓ Вводятся начальное, конечное значение X и шаг, определяющие в каком интервале по оси абсцисс строится график. Данные преобразуются в числовые значения. Это организовано с помощью операторов
- ✓ xn:=strtoint(edit1.Text); xk:=strtoint(edit2.Text); dx:=strtoint(edit3.Text);
- ✓ Значения У, которые будут откладываться по оси ординат записываются в массив. Для этого вводится переменная i, определяющая

порядковый номер элемента в массиве. При x:=xn номер элемента массива равен единице.

✓ С помощью оператора while x<=xk do begin организован цикл, организующий перебор всех значений X с заданным шагом. Т.к. в цикле выполняются четыре оператора, после оператора do указана операторная скобка begin.

✓ В теле цикла происходит расчет значений y, вывод значений на экран, а также изменение порядкового номера элемента массива (i:=i+1;) и соответствующее изменение значения x на введенный шаг (x:=x+dx;).

✓ После завершения цикла определяется количество точек графика по формуле n:=i-1;

Текст программы процедуры, компилируемый при выборе пункта меню **«Расчет данных»** приводится ниже**.**

```
procedure TForm1.N1Click(Sender: TObject);
  var i:integer;   xx:string;
begin   xn:=strtoint(edit1.Text);  xk:=strtoint(edit2.Text);
dx:=strtoint(edit3.Text);
x:=xn;    i:=1;
  while x<=xk do begin
  y[i]:=2*sqr(x/2 )-8*(x/8)  +20*sin(x)-50; str( y[i] :1:1,xx);
 memo1.lines.add('x ='+ inttostr(x) +  '  y  ='+xx);
  i:=i+1;
  x:=x+dx;
end;  n:=i-1;  end;
```

«Построение графика». При щелчке по этому пункту меню выполняется ряд действий. Рассмотрим их поэтапно.

Изображение графика будет создаваться на компоненте Image с помощью метода Canvas. Это приведет к тому, что при заказывании любых опций, например, цвета корандаша придется набирать, например, image1.Canvas. pen.Color:=rgb(0,0,0). Чтобы избежать этого введем новый

оператор **With**. Это оператор присоединения, используемый для сокращения при обращении к полям записей.

Оператор **With** имеет формат

With объект do begin операторы end;

В операторах, следующих за словом do можно не указывать название объекта, это название будет автоматически присоединяться, причем только к тем операторам, к которым это присоединение необходимо.

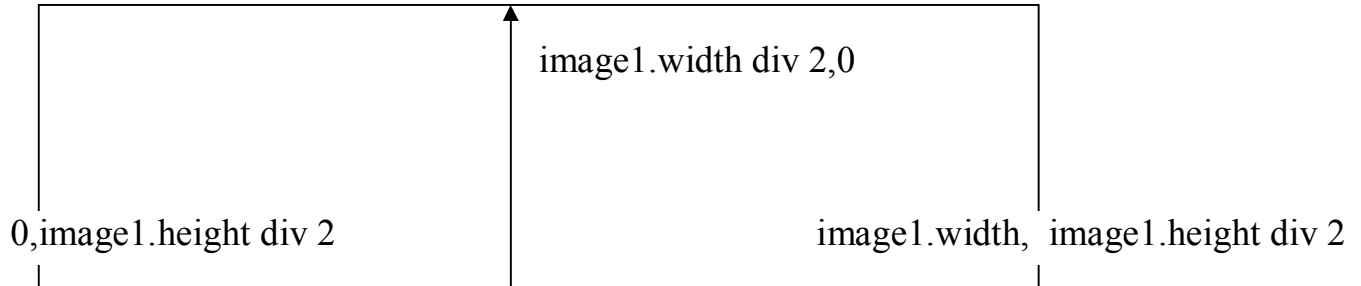
Для построения осей координат следует определить центр системы координат. Анализируем имеющиеся данные, т.к. у нас могут быть и отрицательные, и положительные значения как по оси X, так и по оси У, то логично расположить центр координат в центре компонета Image. Определим координаты его центра. Для этого производим несложные расчеты. xc:=image1.width div 2;yc:=image1.height div 2. Напомним, что операция div является целочисленным делением, следовательно. Если ширина компонента Image равнялась, допустим, 301 пикселю, а высота 400, то центр будет располагаться в точке (150, 200) от левого края Image.

Операторы pen.Width:=5; и pen.Color:=rgb(0,0,0); задают изображение координатных осей черного цвета толщиной в пять пикселей.

Для изображения осей использованы операторы

```
moveto(0,image1.height div 2);
lineto(image1.width,  image1.height div 2 );
moveto(image1.width div 2,0);
lineto(image1.width div 2,image1.height );
```

Сначала карандаш с помощью оператора moveto( ) ставится в самую левую точку экрана, а затем с помощью оператора lineto() протягивается по всей ширине Image. На приводимом ниже рисунке показаны четыре координаты, определяющие начала и концы осей.

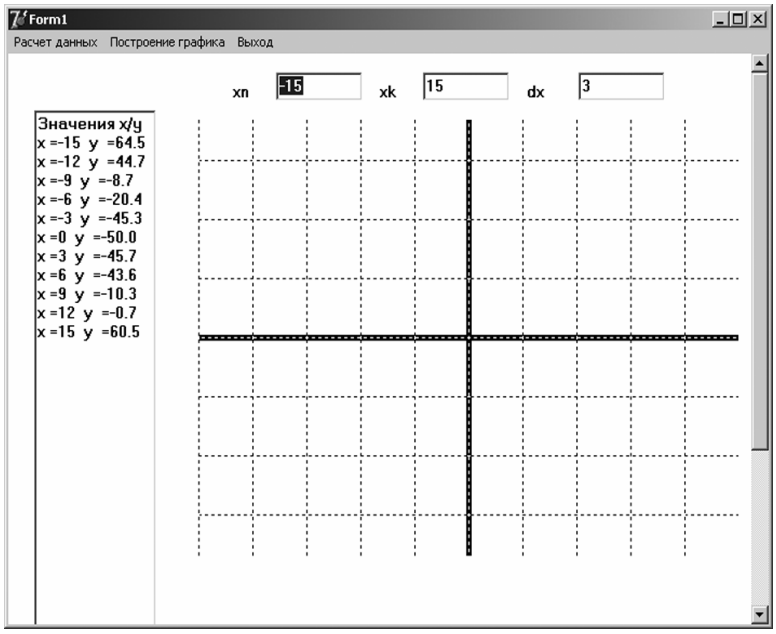


Для изображения масштабной сетки меняется толщина карандаша и стиль.

pen.Width:=1;pen.Color:=rgb(0,0,0); pen.Style:=psDot;

Определим, с какой частотой должна выводиться на экран координатная сетка. Допустим, слева и справа от оси ординат должно строиться одинаковое количество вертикальных линий. Тогда их количество определяют оператором m:=xc div 5;

Цикл




```
for i:=0 to 8 do begin
                moveto(xc+i*m,
0);lineto(xc+m*i  , height );
moveto(xc-i*m,   0);lineto(xc-
m*i  ,height );
moveto(0,yc-m*i);
lineto(2*xc,  yc-m*i );
 moveto(0,yc+m*i);
lineto(2*xc,  yc+m*i );
        end;
```

создает изображение масштабной сетки. Результат показан на рисунке.

В левой части экрана в компоненте Memo показаны значения X и У, на основании которых будет строится график, в правой – построенная ось координат и масштабная сетка.

Идея построения непосредственно графика состоит в следующем. График строят с помощью набора линий. Сначала карандаш устанавливается в первую точку с помощью оператора moveto, а затем в цикле происходит добавление по одной линии, соединяющей соседние точки графика. Если был задан небольшой шаг. То изображение графика будет выглядеть как построенное по точкам, если шаг задан большим, то получится кусочная кривая. Предварительно задается цвет графика и толщина его линии.

Необходимым при построении графика является определение масштаба. Представьте себе, что строится график синуса. Как известно, значения у варьируется в пределах от -1 до 1. Если оставить построение графика без масштаба, то мы не сможем заметить график, т.к не сможем заметить один пиксель то выше оси, то ниже ее.

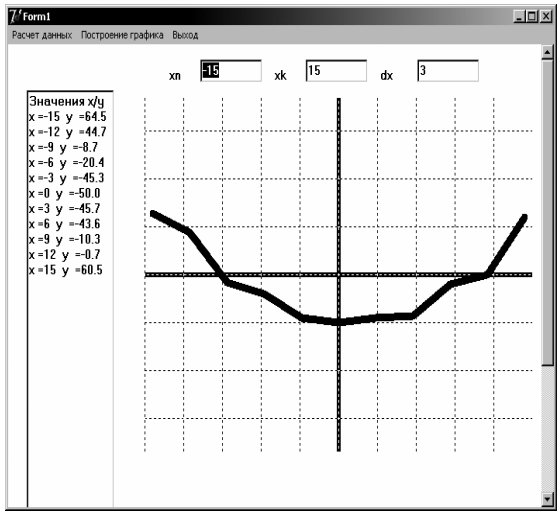
Следовательно, при необходимости, следует подбирать масштаб для построения графика. Нами определялся масштаб с помощью оператора: mx:=trunc(2*xc/(xk-xn));

Обратите внимание на использованную функцию trunc. При построении графических изображений все координаты должны быть целыми, функция trunc и производит преобразование вещественных данных к целому типу.

Таким образом, координата точки графика определяется следующим образом

(x*mx+xc,yc- trunc(y[i])) – к центру координат прибавляют значение x, увеличенное на величину масштаба. Аналогично с координатой по оси У, за исключением того, что значение вычитается. Это связано с тем, что номера пикселей увеличиваются сверху вниз, а не наоборот. нулевое значение находится в верхнем углу экрана, а самое большое в его нижнем углу.

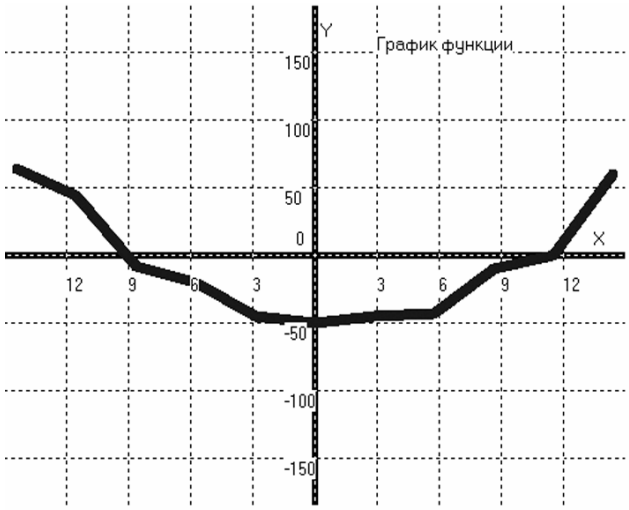
Ниже приводится текст программы и результаты ее выполнения.




```
x:=xn;                                  i:=1;
pen.color:=rgb(0,0,255);  pen.width:=8;
mx:=trunc(2*xc/(xk-xn));
moveto( (x)*mx+xc,yc- trunc(y[i]));
while x<=xk do begin
px:= x*mx +xc;
 py:=yc-trunc(y[i]);
image1.canvas.lineto (px,py);
    x:=x+dx;  i:=i+1;
end;
```

И последнее. Следует ввести подписи осей и название графика. Используем метод **Textout.** Нами задана высота букв и их цвет. С учетом масштаба откладывались надписи на масштабной сетке.

```
font.Size:=10; font.color:=rgb(0,0,0);
textout(xc+5,10, 'Y');
textout(2*xc-25,yc-20,
'X');
textout( xc-15,yc-20, '0');
textout( xc+50,20, 'График
функции');
for i:=1 to 3 do
textout(xc-24,yc-m*i,
inttostr(50*i));
for i:=1 to 3 do
textout(xc-24,yc+m*i, inttostr(-50*i));
for i:=1 to 4 do
textout(xc+m*i,yc+14, inttostr(3*i));
```



```
for i:=1 to 4 do
textout(xc-m*i,yc+14, inttostr(3*i));
```

Так как массив У и значения, определяющие интервал х, использовались в двух процедурах (расчет данных и построение графика), то они описывались в разделе

```
public   y:array[1..20] of real;   x,xn,dx,xk,n:integer;
```

Ниже приводится полный листинг программы.

```
public   y:array[1..20] of real;   x,xn,dx,xk,n:integer;
   { Public declarations }
 end;
var
 Form1: TForm1;
implementation
{$R *.dfm}
```

procedure TForm1.N1Click(Sender: TObject); - Расчет данных

```
 var i:integer;   xx:string;
begin   xn:=strtoint(edit1.Text);  xk:=strtoint(edit2.Text);
dx:=strtoint(edit3.Text);
x:=xn;    i:=1;
 while x<=xk do begin
 y[i]:=2*sqr(x/2 )-8*(x/8)  +20*sin(x)-50; str( y[i] :1:1,xx);
memo1.lines.add('x ='+ inttostr(x) +  '  y  ='+xx);
 i:=i+1;
 x:=x+dx;
end;  n:=i-1;  end;
```

procedure TForm1.N2Click(Sender: TObject); - Построение графика

```
var xc,yc,i,m,px,py,mx:integer;
begin
```

```
with image1.Canvas do begin
xc:=image1.width div 2;yc:=image1.height div 2;
pen.Width:=5;pen.Color:=rgb(0,0,0);
moveto(0,image1.height div 2);
lineto(image1.width,  image1.height div 2 );
moveto(image1.width div 2,0);
lineto(image1.width div 2,image1.height );
pen.Width:=1;pen.Color:=rgb(0,0,0);   pen.Style:=psDot;
m:=xc div 5;     { showmessage(inttostr(m));  }
for i:=0 to 8 do begin
moveto(xc+i*m,  0);lineto(xc+m*i  , height );
 moveto(xc-i*m,  0);lineto(xc-m*i  ,height );
moveto(0,yc-m*i);
lineto(2*xc,  yc-m*i );
 moveto(0,yc+m*i);
lineto(2*xc,  yc+m*i );
end;
x:=xn;  i:=1;   pen.color:=rgb(0,0,255);  pen.width:=8;
mx:=trunc(2*xc/(xk-xn));
moveto( (x)*mx+xc,yc- trunc(y[i]));
while x<=xk do begin
px:= x*mx +xc;
 py:=yc-trunc(y[i]);
image1.canvas.lineto (px,py);
    x:=x+dx;  i:=i+1;
end;            font.Size:=10;
textout(xc+5,10, 'Y');
textout(2*xc-25,yc-20, 'X');
textout( xc-15,yc-20, '0');
textout( xc+50,20, 'График функции');
```

```
for i:=1 to 3 do
textout(xc-24,yc-m*i, inttostr(50*i));
for i:=1 to 3 do
textout(xc-24,yc+m*i, inttostr(-50*i));
for i:=1 to 4 do
textout(xc+m*i,yc+14, inttostr(3*i));
for i:=1 to 4 do
textout(xc-m*i,yc+14, inttostr(3*i));
  end;
end;
```

procedure TForm1.N3Click(Sender: TObject); - Прекращение работы программы.

```
begin
close;  end;end.
```

Основные положения темы:

- ✓ Для построения графиков в среде Delphi можно использовать свойство **Canvas**.
- ✓ Для сокращения записи при обращении к свойствам Canvas можно использовать оператор **With**.
- ✓ При построения графиков необходимо учитывать особенности системы координат и при необходимости вводить масштабирование.
- ✓ При задании координат точек графика используются только целые положительные числа.
- ✓ Подрисуночные надписи, поясняющий текст удобно вводить с помощью метода **Textout**.

Индивидуальные задания. *Создать программу для построения графика отмеченной функции, в которой предусмотреть изменение цвета и типа графической зависимости, подпись построения и вехе графика избрать согласно заданию.*

N	Выражение	Интервал построения	Формат графической зависимости	Формат осей	Формат подписи графика
1	2	3	4	5	6
1.	$F = 8Sin(x^2 - 0.3)$	[-15;15]	Сплошная синяя линия толщиной 3 пикселя	Сплошная черная тонкая линия	Полужирный красный Arial 12 размера
2.	$F = \sqrt[3]{x^2 - 4.3}$	[-18;18]	Сплошная зеленая линия толщиной 2 пикселя	Сплошная синяя тонкая линия	Курсив (Arial) красного цвета 14 размера
3.	$Y = 3a^2 + 8a - 3$	[-10;10]	Пунктирная синяя линия с короткими штрихами	Сплошная черная тонкая линия	Подчеркнутый (Arial) зеленого цвета 12 размера
4.	$Y = 2a^2 - 7a + 6$	[-16;16]	Пунктирная зеленая линия с длинными штрихами	Сплошная синяя тонкая линия	Полужирный курсив (Arial) зеленого цвета 10 размера
5.	$Y = -\sin(4x) + 8x$	[-20;20]	Сплошная	Сплош-	Подчеркну-

			зеленая линия толщиной 2 пикселя	ная черная тонкая линия	тый (Arial) зеленого цвета 12 размера
6.	$F = 3\cos x + 5$	[-15;15]	Пунктирная синяя линия с короткими штрихами	Сплош-ная зеленая тонкая линия	Полужир-ный зеленый Arial 12 размера
7.	$Y = -tg(4x + 8)$	[-18;18]	Сплошная черная линия толщиной 2 пикселя	Сплош-ная синяя тонкая линия	Курсив (Arial) красного цвета 14 размера
8.	$F = \frac{1}{3}\cos(x/2 + 5)$	[-10;10]	Пунктирная зеленая линия с короткими штрихами	Сплош-ная черная тонкая линия	Полужир-ный красный Arial 12 размера
9.	$Y = 3\sin 5x - 2\cos 5x - 3$	[-16;16]	Пунктирная синяя линия с длинными штрихами	Сплошна я черная тонкая линия	Курсив (Arial) синего цвета 14 размера
10.	$F = 6tg(5\pi x/2 - 7)$	[-18;18]	Сплошная зеленая линия толщиной 2 пикселя	Сплош-ная синяя тонкая линия	Подчеркну-тый (Arial) зеленого цвета 12 размера
11.	$F = \sqrt[3]{5x^2 - 2.6}$	[-10;10]	Пунктирная синяя линия	Сплош-ная синяя	Курсив (Arial)

			с короткими штрихами	тонкая линия	красного цвета 14 размера
12.	$F = 7Cos(x^2 - 2.8)$	[-16;16]	Пунктирная зеленая линия с длинными штрихами	Сплошная черная тонкая линия	Подчеркнутый (Arial) зеленого цвета 12 размера
13.	$Y = a^2 - 4a - 5$	[-20;20]	Сплошная зеленая линия толщиной 2 пикселя	Сплошная синяя тонкая линия	Полужирный курсив (Arial) зеленого цвета 10 размера
14.	$F = \frac{1}{3}\cos(x/2 + 5)$	[-15;15]	Пунктирная синяя линия с короткими штрихами	Сплошная черная тонкая линия	Подчеркнутый (Arial) зеленого цвета 12 размера
15.	$Y = 4\cos^2 x - 4\sin x - 1$	[-18;18]	Сплошная черная линия толщиной 2 пикселя	Сплошная зеленая тонкая линия	Полужирный зеленый Arial 10 размера
16.	$Y = 3a^2 + 8a - 3$	[-10;10]	Пунктирная зеленая линия с короткими штрихами	Сплошная черная тонкая линия	Полужирный красный Arial 12 размера
17.	$T = 2\sin^2 x - \cos x - 1$	[-18;18]	Сплошная	Сплош-	Курсив

			синяя линия толщиной 3 пикселя	ная синяя тонкая линия	(Arial) красного цвета 12 размера
18.	$F = \frac{2}{3}\cos(4x/5+7)$	[-10;10]	Сплошная зеленая линия толщиной 2 пикселя	Сплош-ная черная тонкая линия	Пидкресле-ний (Arial) зеленого цвета 12 размера
19.	$F = 4ctgx - 7$	[-15;15]	Пунктирная синяя линия с короткими штрихами	Сплош-ная синяя тонкая линия	Курсив (Arial) красного цвета 14 размера
20.	$Y = -\sin(3x+2.6)$	[-20;20]	Пунктирная зеленая линия с длинными штрихами	Сплош-ная синяя тонкая линия	Полужирны й курсив (Arial) зеленого цвета 10 размера
21.	$F = \frac{4}{5}\sin(3x+\sqrt{\lvert x\rvert})$	[-17;17]	Пунктирная синяя линия с короткими штрихами	Сплош-ная черная тонкая линия	Полужир-ный красный Arial 12 размера
22.	$Y = 3\sin(5x+11)-7$	[-16;16]	Пунктирная зеленая линия с длинными штрихами	Сплош-ная синяя тонкая линия	Курсив (Arial) красного цвета 10 размера

23.	$F = 2ctg(\pi x/6 - 1)$	[-20;20]	Сплошная зеленая линия толщиной 2 пикселя	Сплошная черная тонкая линия	Подчеркнутый (Arial) зеленого цвета 12 размера
24.	$F = \frac{5z^3 - 1}{2z} - 6z$	[0;35]	Пунктирная синяя линия с короткими штрихами	Сплошная синяя тонкая линия	Курсив (Arial) красного цвета 14 размера
25.	$Z = 4\ln x - 2.8x^2$	0;18[	Сплошная черная линия толщиной 2 пикселя	Сплошная синяя тонкая линия	Полужирный курсив (Arial) зеленого цвета 11 размера
26.	$F = 25\ln(\lvert 5\pi x \rvert / 2 - 7)$	]-10;10[	Пунктирная зеленая линия с короткими штрихами	Сплошная черная тонкая линия	Полужирный красный Arial 12 размера
27.	$Y = \sqrt[5]{7x^4 - 3.2x^2 + 8}$	[-18;18]	Сплошная синяя линия толщиной 3 пикселя	Сплошная синяя тонкая линия	Курсив (Arial) красного цвета 10 размера

Лабораторная работа № 7 Процедуры для работы с файлами

Цель работы: Изучение алгоритмов записи информации в файл и считывания из него. Работа с несколькими формами.

Навыки: Навыки проведения расчетов в одномерных массивах с использованием файловой системы.

Файл представляет собой объект, состоящий из последовательности компонент одного типа. Длина файла определяется как число этих компонент. Различают файлы трех типов: текстовые, типизированные и нетипизированные.

Описывают файлы в зависимости от их типа как:

f1:textfile; - текстовый файл;

f1:file of real; - файл, в который записаны действительные числа;

f1: file ; - описание нетипизированного файла.

Файловая система состоит из двух уровней: логических и физических файлов. Логический файл описывается как переменная одного из файловых типов, физический файл – это реально существующий файл. находящийся в одной из папок Windows. Использование логического файла позволяет пользователю не задумываться о проблемах ввода-вывода информации, т.к. все особенности физических файлов учитываются при создании логического файла.

Для связывания логического и физического файлов используется процедура **Assignfile**, которая имеет один из двух форматов:

Assignfile (f1,'A: text.txt'); -связывает файловую переменную f1 с файлом text.txt, который находится на диске A:

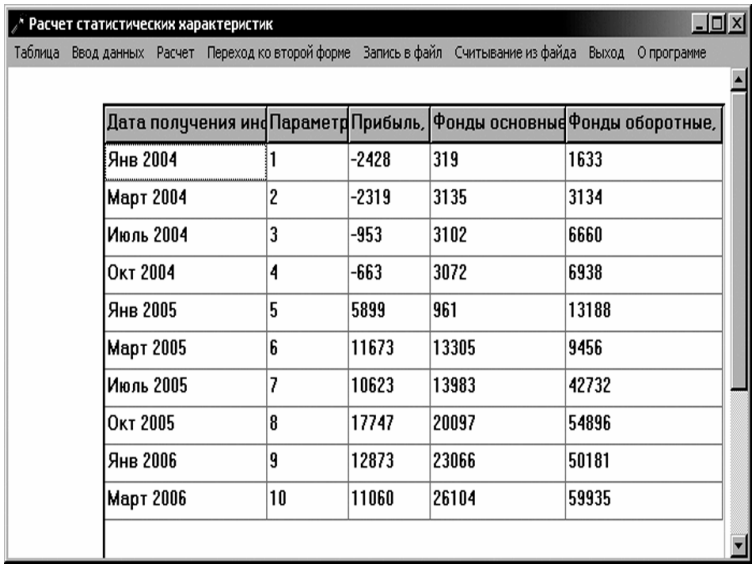
или

Assignfile (f1,Opendialog1.filename); -связывает файловую переменную f1 с файлом, который будет указан при появлении диалогового окна Opendialog1.

Для записи данных в файл используется процедура **Rewrite**(F1). При использовании этой процедуры происходит создание нового файла.

Процедура **Reset**(F1) позволяет открыть существующий файл для считывания из него информации. После выполнения операций чтения/записи файлы должны быть закрыты. Закрытие файла позволяет предохранить хранящуюся в них информацию от повреждений при сбое в работе программы. Кроме этого только закрытые файлы можно удалять или переименовывать. Для закрытия файла используется процедура **Closefile**(F1);

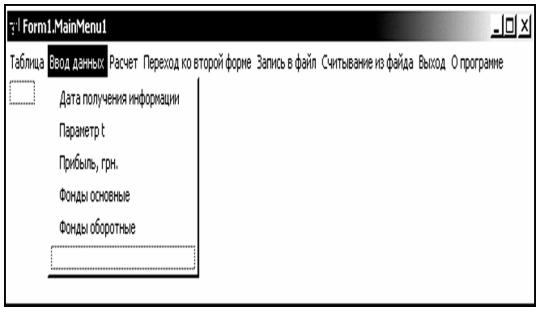
Использование файлов рассмотрим на примере программы, которая позволяет вводить исходные данные в таблицу, записывать их в файл, считывать из него, производить расчеты и строить графики.



Дата получения инф	Параметр	Прибыль,	Фонды основные	Фонды оборотные,
Янв 2004	1	-2428	319	1633
Март 2004	2	-2319	3135	3134
Июль 2004	3	-953	3102	6660
Окт 2004	4	-663	3072	6938
Янв 2005	5	5899	961	13188
Март 2005	6	11673	13305	9456
Июль 2005	7	10623	13983	42732
Окт 2005	8	17747	20097	54896
Янв 2006	9	12873	23066	50181
Март 2006	10	11060	26104	59935

На рисунке показана таблица с введенными исходными данными. В разделе 4 мы рассмотрели один из вариантов ввода данных в таблицу, когда данные вводились при ее появлении экране. Вторым возможным вариантом является ввод исходных данных программно. Этот вариант используют в том случае, когда данные относительно постоянны. При загрузке программы появляется таблица с уже введенными данными, которые при необходимости можно поменять. Например, при создании таблицы, содержащей расписание движения поездов, данные меняются не так и часто. Отдельные исправления вносятся уже при работе с программой.

На рисунке показано созданное нами меню для работы с программой.



Меню «Таблица» используется для загрузки шапки таблицы, «Ввод данных» позволяет выбрать, данные из какого

столбца должны вводиться программно, и вывести эти данные на экран. «Переход во вторую форму» открывает новую форму, на которой строится график, для записи/считывания информации введены следующие два пункта меню, пункт «О программе» выводит на экран описание функций, выполняемых программой. Работа прекращается при выборе пункта «Выход».

Текст приводимой ниже процедуры формирует шапку таблицы и определяет ее размеры.

```
procedure TForm1.N1Click(Sender: TObject);
begin Memo1.Visible:= false; {Компонент Memo, используемый для ввода текста описания функций программы, делаем невидимым}
StringGrid1.show; {Таблица выводится на экран}
with stringgrid1 do{Оператор присоединения для использования таблицы}
begin colwidths[0]:=150; colwidths[1]:=75; {Определяем ширину каждого}
colwidths[2]:=75; colwidths[3]:=125; colwidths[4]:=145; {столбца отдельно}
width:= colwidths[0] + colwidths[1] + colwidths[2] +
colwidths[3] + colwidths[4]+10 ; {Из суммы размеров столбцов находим общую ширину таблицы}
cells [0,0]:= 'Дата получения информации'; {Вводим «шапку» таблицы}
cells [1,0]:= 'Параметр t'; cells [2,0]:= 'Прибыль, грн.';
cells [3,0]:= 'Фонды основные, грн.'; cells [4,0]:= 'Фонды оборотные, грн.';
end;end;
```

Для ввода данных в первый столбец таблицы использована процедура:

```
procedure TForm1.N3Click(Sender: TObject);
begin with stringgrid1 do begin
cells [0,1]:= 'Янв 2004'; cells [0,2]:= 'Март 2004';
cells [0,3]:= 'Июль 2004';cells [0,4]:= 'Окт 2004';
cells [0,5]:= 'Янв 2005';cells [0,6]:= 'Март 2005';
cells [0,7]:= 'Июль 2005';cells [0,8]:= 'Окт 2005';
cells [0,9]:= 'Янв 2006';cells [0,10]:= 'Март 2006'; end; end;
```

В каждую ячейку заносится информация, которая появится на экране при выборе этого пункта меню. Мы не приводим тексты программ, используемых для заполнения остальных столбцов таблицы, т.к. они отличаются только номером столбца и заносимой в ячейки информацией.

По условию задачи требуется рассчитать три статистических характеристики: математическое ожидание по формуле $M_x = \frac{\sum_{i=1}^{N} x_i}{N}$, где M_x – математическое ожидание, x_i – возможные значения исследуемой величины, N – количество данных. Дисперсию находим как $D_x = \frac{\sum_{i=1}^{N} (x_i - M_x)^2}{N}$, а среднее квадратическое отклонение как корень из дисперсии $\sigma_x = \sqrt{D_x}$.

Сначала определяется математическое ожидание. Чтобы с данными таблицы можно было производить различные действия, их следует считать в массивы, при этом массив данных типа дата остается текстовым, т.к. он используется только для построения графика, а остальные массивы преобразуются в данные числовых типов.

```
procedure TForm1.Mx1Click(Sender: TObject);
var i:integer; xx:string; {Описание локальных переменных}
begin   with stringgrid1 do begin
 for i:=1 to 10 do  begin N[i]:=cells[0,i]; Fob[i]:=strtoint(cells[2,i]); {Данные }
Pr[i]:=strtoint(cells[2,i]); {каждого столбца}
FOSN[i]:=strtoint(cells[2,i]); end; {заносятся в массивы}
M1:=0;   cells[0,11]:='Mx'; {Устанавливаем начальное значение Mx равным
нулю и вводим в нулевом столбце обозначение вычисляемой величины}
for i:=1 to 10 do M1:=M1+pr[i]; {Накапливаем сумму массива значений
прибыли}
M1:=1/10*M1;    str(M1:1:2,xx);cells[2,11]:= xx;  {Находим среднее
арифметическое и выводим его значение на экран}
```

```
M2:=0; for i:=1 to 10 do M2:=M2+Fosn[i]; M2:=1/10*M2;
str(M2:1:2,xx);cells[3,11]:= xx;  {занесятся в массивы}
M3:=0; for i:=1 to 10 do M3:=M3+Fosn[i]; {Производим аналогичные
вычисления для двух других массивов}
M3:=1/10*M3;    str(M3:1:2,xx);cells[4,11]:= xx;
stringgrid1.RowCount:=stringgrid1.RowCount+1;  {Увеличение числа строк на
одну для вывода значений математических ожиданий}
end;  end;
```

Процедура, используемая для расчета значений дисперсий, организована аналогично.

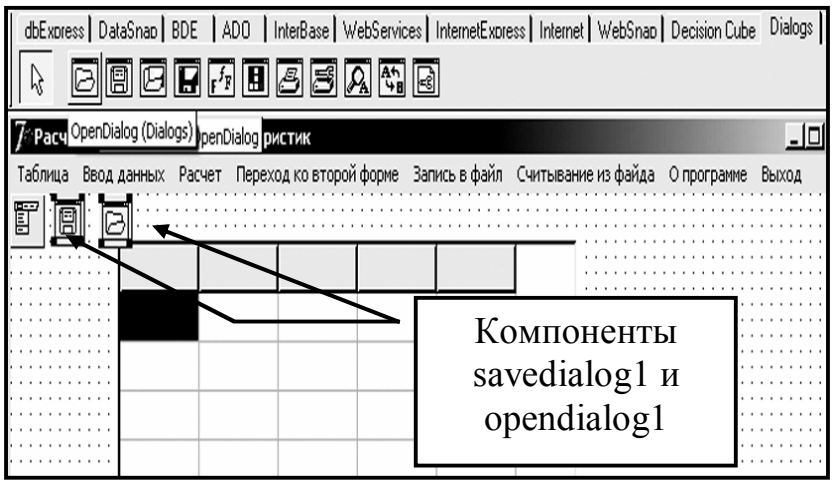
```
procedure TForm1.Dx1Click(Sender: TObject);
var i: integer;   xx:string;
begin    stringgrid1.RowCount:=stringgrid1.RowCount+1;
d1:=0;  stringgrid1.cells[0,12]:='Dx';
for i:=1 to 10 do d1:=d1+sqr(pr[i]-M1);
d1:=d1/10;    str(d1:1:2,xx);stringgrid1.cells[2,12]:= xx;
d2:=0;for i:=1 to 10 do d2:=d2+sqr(fosn[i]-M2);
d2:=d2/10;    str(d2:1:2,xx);stringgrid1.cells[3,12]:= xx;
d3:=0; for i:=1 to 10 do d3:=d3+sqr(fob[i]-M2);
d3:=d3/10;    str(d3:1:2,xx);stringgrid1.cells[4,12]:= xx;
end;
```

Расчет значений среднего квадратического отклонения реализован по приводимому алгоритму.

```
procedure TForm1.x1Click(Sender: TObject);
var xx: string;
begin   stringgrid1.RowCount:=stringgrid1.RowCount+1;
 stringgrid1.cells[0,13]:='Ср.кв.откл';
 q1:= sqrt(d1); str(q1:1:2,xx);stringgrid1.cells[2,13]:= xx;
 q2:= sqrt(d2); str(q2:1:2,xx);stringgrid1.cells[3,13]:= xx;
  q3:= sqrt(d3); str(q3:1:2,xx);stringgrid1.cells[4,13]:= xx; end;
```

Запись введенных данных файл происходит в несколько этапов. Сначала помещаем на форму компоненты **Savedialog1 и Opendialog1** панели Dialogs.

При создании приложений часто приходится выполнять стандартные операции в виде операций открытия файлов, записи в них , выбора шрифтов, цветовой палитры, печати. Разработчики Delphi предусмотрели автоматизированное выполнение этих операций, включив их в библиотеку страницы Dialogs.



Свойства компонентов Savedialog и Opendialog практически одинаковы:

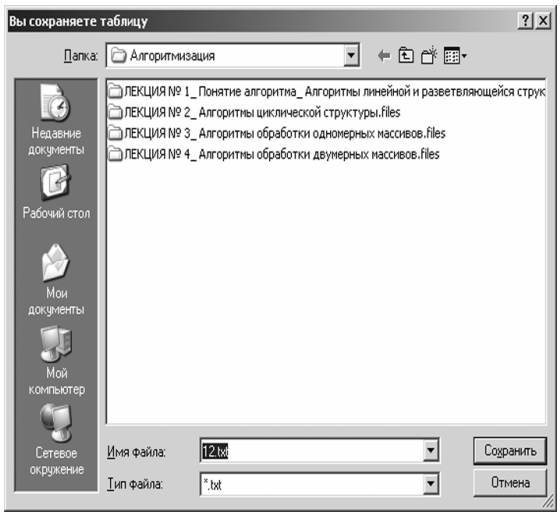
✓ FileName – строковая переменная, хранящая имя выбранного пользователем файла;

✓ Filter – ограничивает типы файлов, которые появляются в диалоговом окне работы с ними. Например, шаблоны *.txt| *.doc позволят просматривать только файлы указанных двух типов. Обратите внимание, вместо привычной запятой разграничителем шаблонов выступает черта | .

✓ Свойство InitialDir определяет диск и папку, которые будут открыты при использовании диалоговых окон, в нашем примере это диск H:, а в нем папка ABC.

✓ Свойство Title задает заголовок окна, если свойство оставить незаполненным, появится стандартное сообщение Windows. Как видно из

рисунков, нами задан текст «Вы сохраняете таблицу», который и появляется вместо стандартного.



Основным методом описанных компонент является Execute. Если при открытии диалогового окна пользователь выбрал файл или ввел его имя, то функция Execute возвращает true, если выбор не сделан, нажата кнопка «Отмена», клавиша «Esc» , то возвращается значение false.

Стандартное обращение к диалоговому компоненту имеет вид:

```
if savedialog1.execute then
 begin  assignfile(F1,savedialog1.filename);
 rewrite(F1); {Далее следуют операторы записи в файл} end;
```

Если введено имя файла, в котором будет сохраняться информация, то связывается файловая переменная с именем этого файла и файл открывается для записи в него информации.

```
procedure TForm1.N10Click(Sender: TObject);
var xx:string; i:integer;
begin  showmessage('Введите название файла, в который будет записываться информация');
if savedialog1.execute then
 begin  assignfile(F1,savedialog1.filename);
 rewrite(F1);
xx:='Дата получения информации';  writeln(F1,xx);
```

```
for i:=1 to 10 do begin
xx:= stringgrid1.cells[0,i]; writeln(F1,xx);
end;
 xx:='Параметр t';  writeln(F1,xx);
  for i:=1 to 10 do begin
xx:= stringgrid1.cells[1,i]; writeln(F1,xx);
end;
xx:='Прибыль, грн.';writeln(F1,xx);
  for i:=1 to 10 do begin
xx:= stringgrid1.cells[2,i]; writeln(F1,xx);
end;
xx:='Фонды основные, грн.'; writeln(F1,xx);
  for i:=1 to 10 do begin
xx:= stringgrid1.cells[3,i]; writeln(F1,xx);
end;
xx:='Фонды оборотные, грн.'; writeln(F1,xx);
  for i:=1 to 10 do begin
xx:= stringgrid1.cells[4,i]; writeln(F1,xx);
end;
 end;
  closefile(F1);
end;
```

Индивидуальные задания

Создать программу, состоящую из меню, позволяющего вводить исходные данные, записывать их в файл, считывать из файла, редактировать, производить расчеты и отображать информацию графически. На графиках должны быть изображены оси и сетка, каждый график отобразить индивидуальным цветом, толщина линий графика и осей должны отличаться. Дать названия графику и осям.

Вариант № 1. Создать программу, состоящую из меню, позволяющего вводить исходные данные, записывать их в файл, считывать, редактировать, производить расчеты и отображать информацию графически.

Объем внешней торговли России, (млрд.дол. в текущих ценах)

Годы	1999	2000	2001	2002	2003	2004
Оборот	183,3	121,0	97,2	101,4	116.7	135,7
Экспорт	88,5	66,8	54,2	59,2	66,2	77,8
Импорт	94,8	54,2	43,0	42,2	50,5	57,9

Определить, какой процент составляют импорт и экспорт относительно оборота. Построить графики зависимостей, отложив по оси X – годы, а по оси У – объемы экспорта и импорта.

Вариант № 2. Определить, на какие континенты экспортируется больше всего продукции и на какие меньше всего. Вывести названия этих континентов с помощью компонента Panel.

Страны	2001	2002	2003
I. Дальнее зарубежье	78,2	78,5	83,2
в том числе:			
1) Европа	58,2	53,5	56,0
в том числе Западная Европа	44,0	44,4	46,0

Бывшие страны СЭВ	14,2	9,1	10,0
2) Азия	16,2	16,6	17,2
в том числе Япония	3,1	4,2	4,5
Китай	5,3	4,4	4,0
НИС и АСЕАН	2,0	3,1	
3) Америка	2,7	7,1	9,1
в том числе США	1,4	5,1	5,7
4) Африка	1,1	0,9	0,9

Построить графики зависимостей, отложив по оси Х – страны, а по оси У – объемы экспорта по каждому году отдельно.

Вариант № 3. Найти, на сколько процентов возрос импорт и экспорт в 2004г. по сравнению с 2003.

Страны	Экспорт		Импорт	
	2003	2004	141	107
Азербайджан	175	86	141	107
Армения	154	127	53	75
Белоруссия	2940	3103	2094	2088
Грузия	63	49	53	68
Казахстан	2198	2416	1996	2726
Киргизия	104	105	98	101
Молдавия	542	413	476	136
Таджикистан	143	190	90	167
Туркменистан	112	93	60	61
Узбекистан	786	882	852	889
Украина	6701	6698	4404	6617

Построить графики зависимостей, отложив по оси Х – страны, а по оси У – полученные проценты по экспорту и импорту отдельно.

Вариант№ 4. Определить, как изменился экспорт услуг по сравнению с 1998г. Построить графики зависимостей, отложив по оси Х названия видов услуг, а по оси У – объемы мирового экспорта за три года.

Мировой экспорт услуг (млрд. дол.)

Годы	1988	1994	2004
Все виды услуг	653,2	853,0	1100
Транспорт	167,4	209,2	250,4
в т.ч.пассажирский фрахт др. виды транспорта	36,2 83,6 47,7	49,6 103,3 56,3	56,9 125 68,1
Путешествия	190,1	246,9	321,1
Правительственные услуги	43,4	47,0	49,5
Другие виды услуг	252,4	349,9	479,1

Вариант № 5. В таблице приведены данные по импорту алюминия. Ввести данные в таблицу, определить суммарный объем импорта, наименование продукции, которая импортируется больше всего.

Продукция	Объем импорта
Алюминиевый лист	4321 тонны
Фольга	5031 тонна
Трубы	25 тыс. т
Листовые металлоконструкции	893 тонны,
Алюминиевый прокат	1380 тонн

Построить графики зависимостей, отложив по оси Х названия продукции, а по оси У – объемы импорта.

Вариант № 6. В таблице приведены данные о функционировании Селидовского хлебокомбината, полученные за период с 1 января 1994г. по 1 января 2004г. Одним из показателей экономической деятельности предприятия является уровень жизненного цикла (УЖЦ), определяемый как отношение прибыли к сумме основных и производственных фондов. Определите уровень жизненного цикла для каждого приведенного периода, постройте график, отложив по оси Х – дату получения информации, а по оси У – УЖЦ. Выделите разным цветом периоды спада и роста уровня ЖЦ.

Данные о балансе Селидовского хлебокомбината

Дата получения информации	Параметр t	Прибыль, грн.	Фонды основные, грн.	Фонды оборотные, грн.
01.01.1994	1	-242832	319327	1633266
01.04.1994	2	-2319	313503	3134348
01.07.1994	3	-953	310268	6660783
01.10.1994	4	-6635	307259	6938099
01.01.1995	5	5899	961	13188
01.04.1995	6	11673	13305	9456
01.07.1995	7	10623	13983	42732
01.10.1995	8	17747	20097	54896
01.01.1996	9	12873	23066	50181
01.04.1996	10	11060	26104	59935
01.07.1996	11	22314	104435	65842

Вариант № 7. В таблице представлена информация о выпуске основных видов продукции заводами черной металлургии. Найдите сумму произведенной продукции отдельно по каждому виду производства за все годы и определить отношение производства в каждом году относительно этой суммы. Постройте графики, отложив по оси Х – годы, а по оси У – объемы производства.

Продукция, годы	1995	1996	1997	1998	1999	2000	2001	2002	2003
Чугун	55,3	54,9	58,7	61	62,3	45,2	49	50,2	51
Сталь	76,2	75,1	77,2	78,2	79,3	80	67	81,2	83
Прокат	59,3	64	68	63	66	69	71	73	78

Вариант № 8. Анализ тенденций социально – экономических показателей Украины

	1995	1996	1997	1998	1999	2000	2001	2002
Валовой внутренний продукт	54516	81519	93365	102593	130442	170070	204190	220556
Сводный бюджет. Доходы.	20689.9	30218.7	28112.0	28915.8	32876.4	49117.9	54934.6	60812.1
Индекс потребитель-ских цен	281.7	139.7	110.1	120.0	119.2	125.8	106.1	99.4

Найти средний индекс потребительских цен и отношение в процентах индекса в каждом году к среднему. Построить графики зависимостей, отложив по оси Х годы, а по оси У – значения каждого из трех приведенных параметров.

Вариант № 9. Уровень жизненного цикла (УЖЦ) определяется на основании балансов предприятий как отношение прибыли к сумме основных и производственных фондов. Определить уровень жизненного цикла для каждого приведенного периода, построить график, отложив по оси Х – дату, а по оси У – УЖЦ.

Прибыль	Фонды основные	Фонды оборотные	Дата получения информации

6629	8039	38871	01.04.1992
2371	8670	67721	01.07.1992
20891	11589	161266	01.10.1992
12863	125789	273967	01.01.1993
86645	135935	536260	01.04.1993
162188	212508	1400451	01.07.1993
117036	1842273	3197510	01.10.1993
549153	2727723	5194615	01.01.1994
1114	2979	8466	01.04.1994

Вариант№ 10. Найти, какой процент составляет производство товаров в каждом году относительно общей суммы за указанный период по каждой отрасли отдельно и по всем отраслям вместе. Дать графическую интерпретацию информации, отложив по оси Х – годы, а по оси У – объемы выпуска. Определите, можно ли поместить данные на одном графике или требуются отдельно графики по каждой отрасли.

Выпуск товаров и услуг, млн.грн.	Годы				
	1996	1997	1998	2000	2002
Всего в отраслях экономики	175960	204421	220679	276246	373893
Промышленность	77848	85590	95561	128516	175532
Строительство	9773	9658	9925	11335	14426
Сельское хозяйство	26746	30032	32758	37683	54356
Транспорт и связь	17062	19798	20612	25418	33442

Вариант № 11. Найти разницу между соответствующими

показателями фондов для двух предприятий, построить графики, отложив по оси Х – даты, а по оси У- значения основных фондов для обоих предприятий.

ЦОФ «Россия»		ЦОФ «Украина»	
Фонды основные	Фонды оборотные	Фонды основные	Фонды оборотные
5451	1325	5732	867
1652574	3319994	1116492	104864
2422	27544	1327423	926656
82735	520958	3813	18379
85589	475642	44103	685321
5538296	4876509	3148	2296
5767000	3198000	3486	3292
5616	5555	3737	2902
5680	4422	3800	1987
5784	4679	3712	2034
5784	4679	3651	1998
5784	4679	3596	1950
5784	4679	3688	1892

Вариант № 12.В таблице приводятся данные об отказах, зарегистрированных при испытаниях автомобиля Найти средние показатели, построить график, отложив по оси Х – пройденное автомобилем расстояние, а по оси У- число отказов. Найти среднее количество отказов.

Пройденное расстояние, тыс. км	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
Число отказов	0	3	8	10	8	12	16	15	20	22	28	29	32	40	52	100

Вариант № 13 . В таблицах приводятся данные по двум фирмам, каждая таблица содержит информацию об изменении валовых издержек в зависимости от объема реализации. Найти отношение объемов валовых издержек к объему реализации и средние показатели по каждому из параметров для каждой фирмы отдельно.

Построить графики зависимостей для каждой фирмы отдельно, отложив по оси Х – валовые издержки, а по оси У- объем реализации.

Таблица1. Данные о фирме 1.

Валовые издержки	122	148	152	179	198	210
Объем реализации	234	256	267	288	300	321

Таблица2. Данные о фирме 2.

Валовые издержки	234	117	142	186	212	239	278
Объем реализации	333	245	256	290	288	321	340

Вариант № 14. В таблице представлены технологические показатели работы агрегата. Найти средний расход топлива, максимальную производительность, построить график, отложив по оси Х – расход, а по оси У – производительность агрегата.

Производитель-ность агрегата, Х	2220	2300	2345	2650	2730	2800	2840	2910
Расход топлива, Y	1200	1234	1280	1790	1800	1820	1870	1923

Вариант № 15. В таблице приводятся данные изменения объема прибыли в зависимости от сезона продажи и объема реализации. Найти, на какой сезон приходится максимальная прибыль, на какой минимальная. Построить графики зависимостей, отложив по оси х – объем реализации, а по

оси У – прибыль.

Сезон	весна	весна	весна	весна	весна	весна	весна	весна
Прибыль	14	16	18	21	25	30	34	54
Объем реализации	234	245	247	278	299	324	345	378
Сезон	лето	лето	лето	лето	осень	осень	осень	зима
Прибыль	25	28	45	57	23	32	45	34
Объем реализации	278	289	312	345	218	256	276	456
Сезон	зима	зима	зима	зима				
Прибыль	23	26	28	29				
Объем реализации	345	389	489	502				

Вариант № 16. В таблице приведены объемы заготовок лома и отходов цветных металлов, сделанные металлургическими предприятиями Украины в разные годы. Найти средние объемы по каждому виду лома и по всем вместе. Построить график, отложив по оси Х- вид лома, а по оси У- средний объем.

Виды лома и отходов	Объемы заготовок
медь	140668, 134567, 123900, 111876, 345678, 789012
свинец	102564, 90050, 890123
алюминий	530620, 450800, 600100, 390600, 290234, 502312
цинк	53433, 80700, 100000, 30000, 60700, 56000, 568000
магний	2851, 5000, 4000, 3000, 2900
никель	16535, 11456, 12900, 17800, 23000
олово	901, 800, 700, 750, 600, 340, 450
кобальт	820, 900, 700, 560, 450, 600
титан	29405 , 30000, 23000, 12900
ртуть	70, 80, 60, 45, 78, 90, 100
вольфрам	547, 700, 600, 560, 450, 340, 400

Вариант № 17. Приводимые в таблице экспериментальные данные предлагается описать теоретическими уравнениями :

$$\sigma_b = 54{,}4 + 1{,}9 * N;$$

$$\sigma_t = 41 - 0{,}031 * T + 5{,}3 * N - 0{,}3 * \tau$$

Произвести расчеты по уравнениям и сравнить с экспериментальными данными. Оценить погрешность расчета по формулам.

Найти и поместить в новую таблицу расчетные значения, построить график, отложив по оси X- величину τ, а по оси y -σ_b.

Технологические параметры	T	1150	1150	1150	1050	1050	1050	1150	1150	1150	1100	1100	1100
	N	8	6	8	4	6	8	8	6	4	4	6	8
	τ	1	5	5	3	1	3	3	1	1	5	5	3
Механические свойства	σ_b	64	70,4	63	66	65,2	61,9	62	64,5	63	66,8	67	69
	σ_t	45,6	44,3	48,9	42,1	44,2	46,9	49,1	42	44	45,6	47,8	49

Вариант № 18. Для расчета коэффициента трения при захвате используются следующие методики:

1) Методика Экелунда С.

$\mu_{зах}$=1.05–0.0005 T – для стальных валков;

$\mu_{зах}$=0.8·(1.05–0.0005 T) – для чугунных валков с закаленной поверхностью.

2.) Методика Гелеи Ш.

$\mu_{зах}$=1.05-0.0005 t – 0.056 v – для стальных валков;

$\mu_{зах}$=0.82-0.0005 t – 0.056 v – для чугунных закаленных валков.

где v – окружная скорость валков, м/с,

t – температура прокатки, С.

Для известных значений t определить $\mu_{зах}$ по обеим формулам при v=5 м/с,

$\mu_{зах}$	0.38	0.36	0.33	0.32	0.315	0.3	0.28
t	700	800	900	960	1000	1050	1100

Построить график, отложив по оси X- t, а по оси У - $\mu_{зах}$, полученное при расчетах по обеим формулам, и приводимое в таблице.

Вариант № 19. В таблице приведены данные о структуре иностранных инвестиций в 2004 г.

Топливно-энергетический комплекс	49,5%
Торговля и общественное питание	9,8%
Строительство	5,6%
Деревообрабатывающая промышленность	4,7%
Машиностроение	4,1%
Прочие отрасли	26,8%

По приводимым данным определить, в какую отрасль промышленности вкладываются наибольшие инвестиции. Вывести название с помощью компонента Panel. Построить график, отложив по оси X- названия отраслей, а по оси У -.проценты.

Вариант № 20. В таблице приводятся данные зависимости времени передачи файлов в зависимости от размеров. Определить средние показатели времени и размера файлов, построить график, отложив по оси X- время передачи, а по оси У – размер файла.

Время передачи	137	136	123	86	74	66	65	51	40
Размер файла	148939	143678	123567	110429	99274	87345	73222	51455	45890

Вариант № 21. В таблице приведена прибыль двух предприятий, найти разницу, и напечатать название предприятия с максимальным средним значением прибыли. Определить, на сколько процентов каждое значение отличается от среднего. Построить график, отложив по оси Х- порядковый номер значения, а по оси У – прибыли двух предприятий.

Предприятие № 1	0.8	2.5	4.6	0.5	2	3.6	-2.9	0.7	4.7
Предприятие № 2	-152.6	-78.6	-69.5	-85.1	-48	65	139.3	-12.8	126.7

Вариант № 22. В таблице приведены данные контроля химического состава стали 08Ю. Найти средние значения каждого показателя, а также на сколько отличается каждое значение от среднего, построить график, отложив по оси Х – порядковый номер эксперимента, а по оси У – величины показателей (три графика на одном листе).

08Ю	Номер эксперимента									
	1	2	3	4	5	6	7	8	9	10
C	0,030	0,045	0,034	0,050	0,057	0,071	0,043	0,051	0,087	0,09
MN	0,210	0,435	0,290	0,372	0,490	0,569	0,267	0,321	0,432	0,58
SI	0,017	0,023	0,034	0,045	0,038	0,073	0,100	0,087	0,092	0,13

Вариант № 23. В таблице представлена информация об интенсивности износа валков СПХН. Найти средние значения показателей, поместить в новую таблицу максимальное и минимальное значение твердости и соответствующие ей величины интенсивности. Построить график, отложив по оси Х- интенсивность, по У – твердость.

Твердость	270	275	278	283	290	294	300	305	308
Интенсивность износа	0,011	0,105	0,111	01	0095	0,088	0,088	0,085	0,089

Вариант № 24. В таблице указан химический состав сталей.

Сталь	C	Si	Mn	Cr	Ni
ШХ6	1.02	0.32	0.33	0.62	0.15
ШХ9	1.09	0.27	0.31	1.1	0.12
ШХ15	0.98	0.31	0.30	1.51	0.19
Р9	0.82	0.33	0.38	4.3	0.32
Р18	0.68	0.37	0.39	4.1	0.32
У8	0.78	0.3	0.28	0.15	0.2
У10	1.02	0.16	0.18	0.11	0.13
У12	1.21	0.22	0.30	0.15	0.17

Найти названия сталей, содержащих наибольший процент Mn, найти средние показатели процентного содержания Ni во всех сталях. Построить график, отложив по оси X – название металла, а по оси У – его средние показатели.

Вариант № 25. В таблицах приводятся результаты испытаний 2 автомобилей на полигоне. Испытания состояли в проведении пробега на 100000 км.

Пройденное расстояние, тыс. км. (авто 1)	4	8	12	16	20	24	28	32	36	40	44	48	52	56
Число отказов (авто 1)	0	3	8	10	8	12	16	15	20	22	28	29	32	40
Пройденное расстояние, тыс. км. (авто 1)	4	8	12	16	20	24	28	32	36	40	44	48	52	56
Число (авто 1) отказов	2	2	5	8	9	8	7	4	9	15	18	21	26	27

Найти, какой автомобиль имеет наибольшее число отказов, среднее количество пройденных км. Построить график, отложив по оси X- число отказов, а по оси У – км.

Заключение

Среда программирования Delphi еще в 1993 году заняла третье место по результатам исследований американской лаборатории тестирования программных продуктов и была признана лучшей средой для программирования непрофессионалов.

С тех пор Delphi сохраняет лидирующие позиции на рынке программных продуктов.

С одной стороны. Среда имеет огромные возможности, начиная от простейших расчетов и графической интерпретации информации, которые мы продемонстрировали в методической разработке, до создания приложений для работы в Интернет, в Access и Excel.

В отличие, допустим, от популярного ныне Visual Basic, среда не требует предварительных знаний ни в одном пакете Microsoft Office.

Вторая часть предлагаемого методического пособия будет посвящена теоретическому материалу, необходимому для выполнения курсовой работы в третьем семестре и будет включать описание работы с файлами, создание многооконных приложений и работу с большими массивами информации в сфере статистических исследований.