МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Г.Г. ШВАЧИЧ, О.В.ОВСЯННІКОВ,
Л.М. ЄФАНОВА, Ю.С. ІВАЩЕНКО

ОСНОВИ ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ DELPHI

Частина І

Затверджено на засіданні Вченої ради академії
як навчальний посібник. Протокол № 1 від 29.01.2013

Дніпропетровськ НМетАУ 2013

УДК 004(075.8
Швачич Г.Г., Овсянніков О.В., Єфанова Л.М., Іващенко Ю.С. Основи програмування в середовищі DELPHI. Частина І: Навч. посібник (російською мовою). – Дніпропетровськ: НМетАУ, 2013. – 60 с.

Викладені основи об’єктно-орієнтованого програмування в інтегрованому середовищі розробки додатків, візуального проектування та організації обчислювальних процесів.

Наведені правила написання програмних конструкцій, приклади реалізації типових задач в середовищі Delphi.

Призначений для студентів спеціальностей 7.05050311 – металургійне обладнання та 7.05050201 – технології машинобудування. Табл. 4. Бібліогр.: 5 найм.

Друкується за авторською редакцією.

Відповідальний за випуск Г.Г. Швачич, д-р техн. наук, проф.

Рецензенти: О.Г. Савенчук, канд. техн. наук, доц. (НГУ)
Г.І. Толстіков, канд. техн. наук, доц. (ДІПОпром)

ВВЕДЕНИЕ

Структура и содержание учебного пособия отвечают требованиям образовательной программы высшего образования Украины. В состав пособия вошли основные темы основ информатики и языка программирования Object Pascal, визуального проектирования и разработки приложений в среде Delphi.

Учебное пособие состоит из двух частей. Первая часть посвящена основам программирования, основное внимание в ней уделено методике программирования и правилам написания программных конструкций. Во второй части рассматриваются: работа с компонентами, решение инженерных задач, разработка полнофункциональных Windows приложений. Каждая тема учебного пособия содержит теоретический и практический материал. Практический материал представлен в виде лабораторных работ, в которых рассматривается реализация конкретных решений задач. Рассмотрение каждой темы завершается самостоятельной работой – индивидуальными заданиями.

1. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

1.1. Состав и структура проекта

Основой *Delphi* является графическая среда разработки приложений, называемая интегрированной средой разработки *(Integrated Development Environment, IDE)*. Основой создаваемого в среде *Delphi* приложения всегда является форма (*Form*). В форме можно размещать различные компоненты, такие как: поля ввода, кнопки, таблицы, меню, панели и другие. Программный код таких компонентов автоматически генерируется *Delphi* при их установке в форму. Для создания многих приложений порой бывает достаточно разместить в форме стандартные компоненты, так как их количество в *Delphi* очень велико.

1.2. Главные составные части среды программирования

- Дизайнер Форм (*Form Designer*).
- Окно Редактора Исходного Текста (*Editor Window*).
- Палитра Компонентов (*Component Palette*).
- Инспектор Объектов (*Object Inspector*).
- Меню (*Menu System*).
- Панель инструментов (*SpeedBar*).

Имеются, конечно, и другие составляющие *Delphi*, такие как: древовидный просмотр состава проекта, интуитивный помощник написания кода, менеджер проекта и многие другие, используемые для точной настройки программы и среды программирования.

Программирование в среде *Delphi* предполагает частое переключение между Дизайнером Форм и Окном Редактора Исходного Текста (которое для краткости называют Редактор).

Дизайнер Форм среды *Delphi* интуитивно понятен и прост в использовании. Он первоначально состоит из одного пустого окна, которое заполняется нужными объектами, выбранными в Палитре Компонентов.

Палитра Компонентов позволяет выбрать нужные объекты для размещения их в Дизайнере Форм. Для использования Палитры Компонентов необходимо с помощью указателя мыши выбрать один из объектов, нажать и отпустить левую клавишу мыши, затем переместить курсор в рабочее поле

дизайнера форм и оставить в нем выбранный объект, нажав и отпустив левую клавишу мыши.

При установке компонента в форму ему присваивается собственное имя, включающее порядковый номер. Например, при размещении в форме двух кнопок - компонентов *Button* - им будут присвоены имена *Button1* и *Button2* соответственно. Имя компонента является его идентификатором, поэтому двум компонентам, установленным в форму, нельзя присваивать одинаковые имена. Имена компонентов, как и другие их свойства, в дальнейшем можно изменять. Любой компонент, установленный в форму, является *Экземпляром* своего *Класса*. Например, поле ввода *Edit1* является экземпляром класса *TEdit*. Учитывая тот факт, что компоненты, представленные в Палитре компонентов среды *Delphi*, также являются *Экземплярами* своего К*ласса*, следует различать понятия названия компонента, как *класса* и употребление его *имени*. Когда говорится о компоненте, как о представителе *класса*, то употребляется слово *TComponent*, например: *TEdit*. Когда компонент называется по имени, представленному в Палитре Компонентов, то употребляется слово *Component*, например: *Edit*. Палитра Компонентов использует постраничную группировку объектов. В верхней части Палитры находится набор страниц - *Standard, Additional, Dialogs и др*. Слева от Дизайнера форм размещен Инспектор объектов. Информация в Инспекторе Объектов изменяется в зависимости от объекта, выбранного в форме. Необходимо отметить, что каждый компонент является настоящим объектом, которым можно управлять при помощи Инспектора Объектов.

Инспектор Объектов состоит из двух страниц, каждую из которых можно использовать для определения поведения данного компонента. Первая страница - это список свойств (Properties), вторая - список событий (Events). Если необходимо изменить какое-либо свойство объекта, то обычно это выполняется в Инспекторе Объектов. К примеру, можно изменить имя, название и размер компонента *Panel*, изменяя свойства *Name, Caption, Left, Top, Height, и Width* в окне редактора свойств Инспектора объектов.

Для переключения между страницами свойств и событий используются закладки в верхней части Инспектора Объектов. Страница событий связана с Редактором. Если дважды щелкнуть мышью по правой стороне какого-нибудь пункта, то соответствующая данному событию конструкция программного кода

автоматически запишется в Редактор. При этом Редактор получит фокус - окно Редактора станет активным и появится на переднем плане экрана, и сразу появится возможность добавить код обработчика данного события.

Меню в среде *Delphi* предоставляет быстрый и гибкий интерфейс. Это обусловлено тем, что помимо команд системного меню, управление может осуществляться при помощи “горячих клавиш“.

Наиболее часто используемыми в работе командами являются команды меню *File* (*New, New Application, New Form, Open, Save As, Save Project As, Save All*), команды меню *Edit* (*Cut, Copy, Paste, Delete*), команды меню *View* (*Project Manager, Project Source*), команды меню *Project* (*Add to Project, Options*), а также команды меню ***Run, Components и Help***.

1.3. Проект Delphi

Проект любой разрабатываемой программы является ее основой, а исполняемый файл (приложение) – результатом разработки. Как правило, для проектирования реальных приложений требуется значительное количество времени, что приводит к необходимости хранения файлов проекта. В проект программы можно вносить изменения в любое время, также к разрабатываемому проекту можно подключать составные части ранее разработанных проектов, что существенно сокращает время разработки. Поэтому знание структуры проекта в целом и его составных частей (файлов) является обязательным для разработчика программного продукта.

Папки, содержащие файлы проектов, можно перемещать с диска на диск, переносить на другие компьютеры и переименовывать, но файлы, относящиеся непосредственно к проекту, переименовывать нельзя. Следует отметить, что проекты приложений, разработанные в ранних версиях среды *Delphi*, совместимы с поздними версиями. Однако прямой обратной совместимостью версии *Delphi* не обладают. Также следует обратить внимание на тот факт, что если в проекте использованы компоненты третьих лиц, то при перемещении на другой компьютер соответствующие компоненты также должны быть перемещены и инсталлированы в среде *Delphi*.

Разработку любого нового проекта рекомендуется начинать с создания новой папки, которая может содержать вложенные папки для хранения дополнительной информации, отдельных проектов, входящих в комплексный

проект или учебное задание.

После создания структуры папок для хранения файлов проекта необходимо выбрать команду меню *File / Save Project As*. Сохранить нужно два файла. Первый - модуль (*unit*), содержащий программный код, второй - главный файл проекта, который идентифицирует программу. Отметим, что при первом сохранении нового проекта, файлам проекта рекомендуется присваивать уникальные имена.

Предположим, что мы начали работу над новым проектом, который назовем *Labwork_1***.** Первым сохраняется файл модуля, которому присвоим имя *Lab_1.PAS***.** Вторым сохраняется главный файл проекта с именем *Labwork_1.DPR*. Сохранять файл проекта и файл модуля с одинаковыми именами нельзя. Сохранять указанные файлы необходимо до компиляции программы. Это обусловлено тем, что среда *Delphi* будет знать, где в дальнейшем размещать другие файлы проекта, включая скомпилированную программу. В процессе сохранения к указанным файлам *Delphi* добавит следующие файлы: *Labwork_1.DOF, Labwork_1.RES и Lab_1.DFM*. Для создания исполняемого файла необходимо скомпилировать проект. Компиляция выполняется командой системного меню *RUN* / RUN. После выполнения этого действия в рабочей папке появятся еще два файла*: Labwork.exe и Lab_1.DCU*.

Итак, при создании нового проекта среда *Delphi* сформирует следующие файлы:

- *Labwork_1.DPR* – файл проекта. Он содержит код главной программы, написанной на языке *Object Pascal*. В файле проекта содержатся ссылки на все формы проекта и относящиеся к ним модули. В нем также содержится код инициализации приложения.
- *Lab_1.DFM* – файл формы, для которого декларируется тип, который определяет форму как Класс. Класс - это объектный тип. Объявление нового класса всегда содержится в отдельном модуле. В нашем случае это *Lab_1.PAS*. Каждая форма является компонентом, следовательно, графическим объектом. Объектно-ориентированное программирование предполагает, что свойства объекта должны быть декларированы и определены. Таким образом, все свойства соответствующей формы хранятся в двоичном файле *Lab_1.DFM.*

- *Lab_1.PAS* - Pascal файл. Стандартный идентификатор класса формы. Этот файл содержит весь программный код, относящийся к данному модулю.
- *Labwork_1.RES* – файл ресурсов приложения. Представляет собой двоичный файл, содержащий пиктограммы, графические изображения, курсоры и строки.
- *Labwork_1.DOF* – текстовый файл, который содержит опции проекта, такие как: настройки компилятора и компоновщика, имена служебных каталогов и условные директивы.
- *Lab_1.DCU* – двоичный, скомпилированный файл PAS файл.
- *Labwork_1.EXE* - исполняемый файл (Приложение). В данном случае *Labwork_1.EXE* - готовая программа, которая может функционировать под управлением операционной системы Windows.

Помимо указанных файлов *Delphi* может создавать и другие файлы. Это файлы временного хранения, имеющие расширение *~Pa, ~Df.*

2. ЛИНЕЙНЫЙ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС

2.1. Структура модуля

Модуль состоит из последовательности разделов. Каждый раздел начинается ключевым словом и продолжается до начала следующего раздела.

unit Имя Модуля;

interface // раздел интерфейса

{ Здесь находятся описания процедур и функций модуля, которые могут использоваться другими модулями.)

const // раздел объявления констант

{ Здесь находятся объявления глобальных констант модуля, которые могут использоваться процедурами и функциями модуля.}

type // раздел объявления типов

{ Здесь находятся объявления глобальных типов модуля,

которые могут использоваться процедурами и функциями модуля }

var // раздел объявления переменных

{ Здесь находятся объявления глобальных переменных модуля, которые могут использоваться процедурами и функциями модуля }

implementation // раздел реализации

{ Здесь находятся описания (текст) процедур и функций

Предположим, что нам необходимо разработать программу вычисления значения ***y*** по формуле (1) при различных значениях ***x*.**

$$y = a \cdot e^{x} + \ln\sqrt{b + s \cdot \sin\left(\frac{\pi}{4}\right)}\ , \qquad (2.1)$$

где значения **a**, **b, s** являются константами, например:
a = 3,112; b = 5,85; s = 9,48.

Ввод, вычисление и вывод результата можно реализовать в три строки программного текста, который имеет следующий вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var
x:double; InpX: string;
begin
  InpX := InputBox('Ввод значения X', 'Введите вместо (0) действительное
число', '0');   // 1
  x := StrToFloat(InpX);    // 2

  Form1.Caption := FloatToStr(3.112 * Exp(x) + ln(SQRT(5.85 + 9.48 *
sin(pi/4))));          // 3
End;
```

Для начинающего такая запись программного кода может оказаться сложной в понимании, а студент, освоивший основы программирования, прочтет ее без затруднения.

Итак, в первой строке переменной **InpX** присваивается значение поля ввода диалогового окна **InputBox.** Во второй строке выполняется преобразование числа, введенного, как текст, в действительный тип данных, значение которого получает переменная **x.** В третьей строке в заголовок формы **Form1** выводится результат (как текст), вычисленный по приведенной

формуле. Данный программный код реализован в обработчике события щелчка кнопки **Button1.**

В данной программе возможно возникновение **исключительной ситуации** (сбоя работы программы – выдаче сообщения Windows), в случае, если пользователь введет в поле ввода **InputBox** символы текста вместо символов чисел.

Обработку **исключительных ситуаций** мы будем рассматривать после получения достаточных навыков, а на данном этапе разберемся с некоторыми правилами и простыми приемами при написании программ.

На начальном шаге написания программы целесообразно выполнить вычисления выражения (1.1) по частям.

$$y = a \cdot e^{x} + \ln(c)\,. \qquad (2.2)$$

C вычисляется по формуле:

$$c = \sqrt{b + s \cdot \sin\left(\frac{\pi}{4}\right)}\,. \qquad (2.3)$$

Примечание: Следует заметить, что на практике сложные (громоздкие) выражения часто разбиваются на части для удобства чтения программного кода. При этом затраченный объем памяти возрастает незначительно.

2.2. Лабораторная работа №1

Прежде чем приступить к разработке приложения (программы), надо создать в **Delphi** новый проект, который сохранить в своей папке с именами файлов по умолчанию, выполнив команду **Save Project As.** Установить в форму кнопку **Button1** и повторно сохранить проект, выполнив команду **Save All**.
В нашей задаче результат ***y*** и параметр ***c*** будут действительными числами. Также константы ***b*** и ***s*** являются действительными числами по условию задачи. Поэтому, в зависимости от требуемой точности, можно применить один из известных типов представления действительного числа, например: **real**, **single**, **double** и т.д.

В выражениях могут присутствовать функции расширенной математики (см. приложение 1), которые объявлены в библиотеке **math.pas**. Данный модуль, по умолчанию **проекта Delphi**, не описан в интерфейсе **unit'**а.

Поэтому необходимо самостоятельно объявить **math.pas в** разделе **interface.** Добавить модуль **math** в раздел **uses,** как показано ниже:

```
Unit Unit1;
interface
uses
  Windows, Messages, SysUtils,  Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, math;       // Добавлен модуль расширенной математики
```

После добавления модуля выполнить команду **Save All**.

Представим алгоритм нашей программы по шагам:

Шаг 1. Определим, какие константы и переменные мы будем использовать, определим их имена, типы и значения.

Из анализа формул и условия задачи очевидно, что нам необходимы три константы ***a, b,*** и ***s,*** для которых выберем тип double.

a = 3.112; b = 5.85; s = 9.48;

Примечание: в дальнейшем константы определим не в теле реализации формулы, а отдельно, для получения навыков в программировании.

Шаг 2. Определим имена и типы переменных.

Из анализа формул и последовательности вычислений очевидно, что необходимо использовать три переменных*:* ***x, y, c*** – типа double.

Шаг 3. Выберем способ ввода значения ***x*.**

Первично мы не будем использовать диалоговое окно для ввода значений ***x*** и их присвоения переменным, а выполним это действие в теле программы.

Шаг 4. Определим, в какой последовательности мы будем выполнять вычисления, т.е. какие составные части выражения (формулы 1) будем выполнять в первую очередь, а какие в последующем.

Очевидно, что перед вычислением значения ***y*** необходимо предварительно вычислить значение ***c*.**

1). $c = \sqrt{b + s \cdot \sin\left(\frac{\pi}{4}\right)}$; 2). $y = a \cdot e^{x} + \ln(c)$.

Шаг 5. Определим место и форму представления результатов вычисления.

Первоначально будем выводить значения ***y*** в заголовок формы.

Программу реализуем в обработчике события щелчка кнопки **Button1.**

Для создания пустого обработчика события необходимо выполнить двойной щелчок по кнопке, установленной в форме, либо на странице **Events** Инспектора объектов, выполнить двойной щелчок в поле, справа от события **OnClick**. Ниже приводится вид пустой процедуры обработки события.

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

Между словами **procedure** и **begin** первыми определяются константы, для описания которых имеется зарезервированное слово **const**.

Выполним описание используемых констант.

```
procedure TForm1.Button1Click(Sender: TObject);
{определение констант}
const
a:double = 3.112;
b:double = 5.85;
s:double = 9.48;
begin

end;
```

Выполнить команду RUN с целью проверки синтаксических либо грамматических ошибок. Если приложение откомпилировалось правильно, то ошибок нет, в противном случае в редакторе кода будет выделена строка, содержащая ошибку, а в нижнем окне редактора кода сообщение о виде ошибки.

Для объявления переменных имеется зарезервированное слово **Var**. Переменные всегда объявляются после описания констант**,** в случае, если мы используем константы.

```
procedure TForm1.Button1Click(Sender: TObject);
{определение констант}
const
a:double = 3.112;   b:double = 5.85; s:double = 9.48; // можно писать в одну строку через (;)

{объявление переменных}
var
x, y, c:double;
begin

end;
```

Выполнить объявление переменных. После выполнения команды **RUN** сохранить проект с помощью команды **SaveAll**.

Приступим к написанию собственно исполняемого кода, который должен находиться между операторными скобками **begin и end,** т.е. в теле процедуры**:**

```
procedure TForm1.Button1Click(Sender: TObject);
{определение констант}
const
a:double = 3.112;  b:double = 5.85; s:double = 9.48;
{объявление переменных}
var
x, y, c:double;
begin
  x := 5.5;                              // Присвоение значения переменной- x
 -
  c := SQRT(b + s * sin(pi/4));          // Вычисление параметра  - c
  y := a * Exp(x) + ln(c);               // Вычисление значения  - y
  Form1.Caption := FloatToStr(y);        // Вывод результата
end;
```

Так как мы имеем дело с линейной программой, то после написания каждой строки программы, завершающейся символом (;), регулярно выполняем команду **RUN** и, при необходимости, исправляем синтаксические и грамматические ошибки.

Сохранить отлаженную программу, выполнив команду **Save All**.

Правильно написанная программа должна показать значение

y = 762,746287875005 при значении x = 5.5.

Продолжим разработку нашей программы. Для ввода значений *x* используем диалоговое окно **InputBox**. Так как все параметры функции диалогового окна имеют строковый тип, необходимо вводить действительные числа и объявить еще одну переменную типа **String**, например **InpX,** значение которой затем преобразовать в действительное число.

```
procedure TForm1.Button1Click(Sender: TObject);
{определение констант}
const
a:double = 3.112;  b:double = 5.85; s:double = 9.48;
{объявление переменных}
var
x, y, c:double;
InpX: string;                          // Объявлена доп. Переменная InpX
begin
  InpX := InputBox('Ввод значения X', 'Введите вместо (5,5) новое число', '5,5');
  x := StrToFloat(InpX);                    // Присв. Знач. переменной – x с преобразов. Типа.
  c := SQRT(b + s * sin(pi/4));             // Вычисление параметра  - c
  y := a * Exp(x) + ln(c);                  // Вычисление значения  - y
  Form1.Caption := FloatToStr(y);           // Вывод результата
end;
```

Внеся дополнения в программу, выполнить команду **RUN** и сохранить отлаженную программу, командой **Save All.**

Далее мы можем обеспечить вывод результата работы нашей программы в окне сообщения **ShowMessage**. Так как результат представляет собой действительное число, а параметр вывода ShowMessage имеет тип String, необходимо объявить дополнительную переменную **OutY**, которая получит значение FloatToStr(y) – преобразование числа в символы текста.

```
procedure TForm1.Button1Click(Sender: TObject);
{определение констант}
const
a:double = 3.112;  b:double = 5.85; s:double = 9.48;
{объявление переменных}
var
x, y, c:double;
InpX, OutY: string;                                  // Объявлена доп. Переменная
OutY
begin
  InpX := InputBox('Ввод значения X', 'Введите вместо (5,5) новое число',
'5,5');
  x := StrToFloat(InpX);                              // Присв. Знач. переменной – x с
преобразов. Типа.
  c := SQRT(b + s * sin(pi/4));                       // Вычисление параметра  - c
  y := a * Exp(x) + ln(c);                            // Вычисление значения  - y
 OutY := FloatToStr(y);                              // Присв. пер.OutY  преобр. Знач. - y
 ShowMessage('Результат = ' + OutY);   // Вывод окна сообщения
end;
```

Вывести результат в окно сообщения можно и не используя переменной OutY, например: ShowMessage('Результат = ' + FloatToStr(y)).

2.3. Индивидуальные задания к лабораторной работе №1

Составить линейные программы вычисления приведенных ниже выражений, используя функции InputBox, ShowMessage.

№	Выражение	Константы
1	$y=(a-0{,}4)\cdot tg^2\left(\frac{\pi}{8}\right)-2{,}6e^{-X}\sqrt{\lvert 1+b\cdot\sin(x)\rvert}$	A = 6, 34 B = 8,59
2	$y=\sqrt{\lvert a\cdot x+\cos^2\left(\frac{\pi}{4}\right)\cdot e^{-X}\cdot\sin^2\left(\frac{\pi}{8}\right)\rvert}$	A = 12, 32
3	$y=(a+b)\cdot tg^2\left(\frac{\pi}{8}\right)+2{,}6e^{-X_i}\cdot\sqrt{\lvert 1+c\cdot\sin(x)\rvert}$	A= 49,45 B= 82,34 C=15,14
4	$y=\cos(2\cdot\pi)^b-e^{\frac{X}{a}}\cdot a\cdot\sin(\frac{\pi}{4})^{\frac{X}{a}}$	A = - 45.25 B = 10,12
5	$y=\cos(a\cdot\pi)^b-e^{\sqrt{\lvert X\rvert}}\cdot b\cdot\cos(x^{a\cdot X})$	A = 0,03 B = - 2.25
6	$y=a\cdot\cos(x^3)-b\cdot\sin(\sqrt{\lvert x\rvert})$	A = 0,81 B = 1,62
7	$y=\cos(a\cdot\pi)^b-e^{\frac{X}{2}}\cdot a\cdot\cos(\frac{\pi}{4})^X$	A = 0,333 B=1,666
8	$y=a\cdot\cos(x^3)-b\cdot\sin(x^5)$	A=1,468 B = 3,24
9	$y=\cos(b\cdot\pi)^a\cdot e^{\frac{X}{2}}+a\cdot\cos(\frac{x}{4})$	A=12,36 B=1,76
10	$y=a\cdot\cos(x^5)-b\cdot\sin(\sqrt{\lvert x\rvert})$	A=0,06 B=3,21
11	$y=(a-0{,}4)\cdot tg^2\left(\frac{\pi}{8}\right)-2{,}6e^{-X}\sqrt{\lvert 1+b\cdot\sin(x)\rvert}$	A = 6, 34 B = 8,59
12	$y=(a+b)\cdot tg^2\left(\frac{\pi}{8}\right)+2{,}6e^{-X_i}\cdot\sqrt{\lvert 1+c\cdot\sin(x)\rvert}$	A= 49,45 B= 82,34 C=15,14
13	$y=\cos(a\cdot\pi)^b-e^{\sqrt{\lvert X\rvert}}\cdot b\cdot\cos(x^{a\cdot X})$	A = 0,03 B = - 2.25

14	$y = \cos(a \cdot \pi)^b - e^{\frac{X}{2}} \cdot a \cdot \cos(\frac{\pi}{4})^X$	A = 0,333 B=1,666
15	$y = \cos(b \cdot \pi)^a \cdot e^{\frac{X}{2}} + a \cdot \cos(\frac{x}{4})$	A=12,36 B=1,76
16	$y = a \cdot \cos(x^5) - b \cdot \sin(\sqrt{\lvert x \rvert})$	A=0,06 B=3,21
17	$y = \sqrt{\lvert a \cdot x + \cos^2\left(\frac{\pi}{4}\right) \cdot e^{-X} \cdot \sin^2\left(\frac{\pi}{8}\right) \rvert}$	A = 12, 32
18	$y = \cos(2 \cdot \pi)^b - e^{\frac{X}{a}} \cdot a \cdot \sin(\frac{\pi}{4})^{\frac{X}{a}}$	A = - 45.25 B = 10,12
19	$y = a \cdot \cos(x^3) - b \cdot \sin(\sqrt{\lvert x \rvert})$	A = 0,81 B = 1,62
20	$y = a \cdot \cos(x^3) - b \cdot \sin(x^5)$	A=1,468 B = 3,24
21	$y = a \cdot \cos(x^5) - b \cdot \sin(\sqrt{\lvert x \rvert})$	A=0,06 B=3,21
22	$y = (a - 0{,}4) \cdot tg^2\left(\frac{\pi}{8}\right) - 2{,}6e^{-X} \sqrt{\lvert 1 + b \cdot \sin(x) \rvert}$	A = 6, 34 B = 8,59
23	$y = (a + b) \cdot tg^2\left(\frac{\pi}{8}\right) + 2{,}6e^{-X_i} \cdot \sqrt{\lvert 1 + c \cdot \sin(x) \rvert}$	A= 49,45 B= 82,34 C=15,14
24	$y = \cos(a \cdot \pi)^b - e^{\sqrt{\lvert X \rvert}} \cdot b \cdot \cos(x^{a \cdot X})$	A = 0,03 B = - 2.25
25	$y = \cos(a \cdot \pi)^b - e^{\frac{X}{2}} \cdot a \cdot \cos(\frac{\pi}{4})^X$	A = 0,333 B=1,666

3. РАЗВЕТВЛЯЮЩИЙСЯ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС

Предположим, что нам необходимо вычислить значение R по разным формулам, в зависимости от некоторых условий. В приведенном ниже примере, если $x \geq y$, R необходимо вычислять по формуле $R = a \cdot x + b \cdot y$, в противном случае, по формуле $R = (y - x) * y$. В приведенных выражениях a и b являются константами, значения которых соответственно равны: a = 10, b = 15.

$$R = \begin{cases} a \cdot x + b \cdot y \ \ при: x \geq y \\ (y - x) \cdot y \end{cases} . \qquad (3.1)$$

Для проверки условий используется конструкция, состоящая из операторов **if** (если)**, then** (то)**, else**(также, в противном случае). Конструкция работает следующим образом. Между операторами **if** и **then** проверяются условия, между операторами **then** и **else** выполняются действия (группа действий), соответствующие условию. Действия (группа действий), не соответствующие условию, выполняются после оператора **else**.

В нашем случае фрагмент программного кода будет иметь следующий вид:

```
if x >= y then R:= a * x  + b * y else R:= (y – x) * y;
```

или

```
if x >= y then
R:= a * x  + b * y
else
R:= (y – x) * y;
```

Что в общем одно и то же, так как программная строка заканчивается символом (;)- точка с запятой.

Если нам необходимо выполнить группу действий, например, помимо вычислений вывести сообщение с указанием ветви, по которой выполнялось вычисление, то группа действий должна заключаться в операторные скобки **begin** и **end**. Рассмотрим такой пример:

```
if x >= y then
begin
  R:= a * x  + b * y;
```

```
  ShowMessage('Условия выполнены (ветвь 1)' );
end else
begin
  R:= (y – x) * y;
  ShowMessage('Условия не выполнены (ветвь 2)' );
end;
```

Следует отметить тот факт, что условия могут быть составными, например, в нашем примере изменим условие:

$$R = \begin{cases} a \cdot x + b \cdot y \ \ при: x \geq y \ \ и \ \ a < b \\ (y - x) \cdot y \end{cases} . \qquad (3.2)$$

Тогда фрагмент программного кода будет иметь следующий вид:

if (x >= y) **and** (a < b) **then** R:= a * x + b * y **else** R:= (y – x) * y;

Обратите внимание на запись проверки отдельных условий, которые заключается в скобки (x >= y) **and** (a < b).

В случае, если нет необходимости выполнять какие либо действия при невыполнении условий, оператор **else** исключается из конструкции:

```
f:= false;
if x >= y then f: = true;
```

В данной записи устанавливается начальное значение булевой переменной **f** в состояние **false**. Далее проверяется условие **x >= y**. Если условие выполняется, то **f** принимает значение **true**, в противном случае **f** остается в состоянии **false**.

В общем случае в конструкции **if .. then .. else** возможно реализовать вычисления непосредственно в блоке сравнения. Рассмотрим в несколько расширенном виде задачу (3.1), в которой определим, что значения переменных **x** и **y** вычисляются по формулам: $x = \sin(\frac{\pi}{4}) * t_i$, $y = \cos(\frac{\pi}{4}) * t_i$ Тогда программная строка, реализующая выражение (1), может иметь следующий вид:

if sin(pi / 4) * t >= cos(pi / 4) * t **then** R:= a * x + b * y **else** R:= (y – x)*y;

3.1. Программный код, реализуемый инструкцией case

В разветвляющихся вычислительных процессах множественный выбор реализуется при помощи вложенных одна в другую инструкций if. Такой подход не всегда удобен, особенно в том случае, если количество вариантов хода программы велико. В языке Delphi есть инструкция case, которая позволяет эффективно реализовать множественный выбор. В общем виде она записывается следующим образом:

```
case Селектор of список1:
begin
{ инструкции 1 } end; список2:
begin
{ инструкции 2 } end; списокM:
begin
{ инструкции N }
end;
else
begin
{ инструкции )
end;
end;
```

Инструкции между begin и end выполняются, если значение выражения, записанного после case, совпадает с константой из соответствующего списка. Если это не так, то выполняются инструкции, находящиеся после else, между begin и end.

Ниже приведен пример программного кода, реализуемый инструкцией case.

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

Выполняется инструкция **case** следующим 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.

3.2. Лабораторная работа №2

Произвести вычисления функции *R* в соответствии с выражением:

$$R = \begin{cases} a \cdot x + b \cdot \dfrac{y}{x} & при: x \geq y \\ (y - x) \cdot \dfrac{x}{y} & \end{cases}, \qquad (3.3)$$

где: ***x*** и ***y*** целые числа, ***a*** = 4, ***b*** = 6 - константы.

Как и в предыдущей работе, используем форму и кнопку. Значения ***x*** и ***y*** вводим с использованием **InputBox,** а результат с пояснениями представим с помощью функции **ShowMessage.**

Выполним анализ нашей задачи. Константы a и b являются целыми числами, по условиям задачи ***x*** и ***y*** - целые числа. Так как в выражении присутствуют дроби, результат ***R*** будет действительным числом. Также в нашем выражении нет специальных математических функций и нет необходимости применения модуля расширенной математики ***math***.

Выполним начальные действия по сохранению проекта – команда **Save Project As**. Установим в форму кнопку Button1. и повторно сохраним проект командой Save All. Далее создаём пустой обработчик события щелчка для кнопки.

```
procedure TForm1.Button1Click(Sender: TObject);
begin


end;
```

Опишите константы и объявите необходимые переменные.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a:integer = 4;  b:integer = 6;
var
x, y:integer;
R:real;
```

```
InpX, OutY: string;
begin
end;
```

Выполнить команду **RUN**. Закрыть приложение, а затем выполнить команду **Save All**.

Линейная часть программы, отвечающая за ввод значений ***x*** и ***y***.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a:integer = 4;  b:integer = 6;
var
x, y:integer;
R:real;
InpN, OutN: string;
begin
  InpN := InputBox('Ввод данных', 'Введите целое число X', '0');
  x := StrToInt(InpN);
  InpN := InputBox('Ввод данных', 'Введите целое число Y', '0');
  y := StrToInt(InpN);
end;
```

Выполнить команду **RUN**. Проверить последовательный вывод диалогового окна **InputBox**. Закрыть приложение, и затем выполнить команду **Save All**.

Далее создадим конструкцию **if then else,** которая в нашем случае будет иметь полный вид:

```
procedure TForm1.Button1Click(Sender: TObject);
const
a:integer = 4;  b:integer = 6;
var
x, y:integer;
R:real;
InpN, OutN: string;
begin
  InpN := InputBox('Ввод данных', 'Введите целое число X', '0');
```

```
  x := StrToInt(InpN);
  InpN := InputBox('Ввод данных', 'Введите целое число Y', '0');
  y := StrToInt(InpN);
  if x >= y then
  begin
  end else
  begin
  end;
end;
```

Выполните команду **RUN**. Закройте приложение, и затем выполните команду **Save All**.

Далее нам осталось описать ветви нашей программы:

```
procedure TForm1.Button1Click(Sender: TObject);
const
a:integer = 4;  b:integer = 6;
var
x, y:integer;

R:real;
InpN, OutR: string;
begin
  InpN := InputBox('Ввод данных', 'Введите целое число X', '0');
  x := StrToInt(InpN);
  InpN := InputBox('Ввод данных', 'Введите целое число Y', '0');
  y := StrToInt(InpN);
  if x >= y then                                  // если x >= y, то выполняется:

begin
  R := a * x + b * (y / x);                       // вычисление по ветви 1

OutR := 'Усновие x >= y, R = ' + FloatToStr(R);   присвоение результата

 end else                                         // в противном случае
  begin
```

```
  R := (y - x) * x / y;                                  // вычисление по ветви 1
  OutR := 'Усновие x < y, R = ' + FloatToStr(R);         // присвоение результата
  end;
 ShowMessage(OutR);                                      // вывод
результата
end;
```

Выполните команду **RUN**. Проверьте правильность функционирования программы. Закройте приложение, и затем выполните команду **Save All**. и проанализируйте работу программы при разных значениях ***x*** и ***y***.

3.3. Индивидуальные задания к лабораторной работе №2

№	Условие	Значение констант и переменных
1	$R=\begin{cases}a\cdot X+b\cdot Y \ при: X<Y\\(b-X)\cdot Y\end{cases}$	A = 3,24, b = 0,05 X и Y действительные числа
2	$R=\begin{cases}(a+b)\cdot X+(a-b)\cdot Y \ при: X\geq Y\\a\cdot b-X\end{cases}$	A = 8,11, b = 6,26 X и Y целые числа
3	$R=\begin{cases}(a+b)\cdot X+(a-b)\cdot Y \ при: X<Y\\(a\cdot b-X)\cdot Y\end{cases}$	A = 8, b = 6 X и Y целые числа
4	$R=\begin{cases}(a-b)\cdot X^2+(a+b)\cdot Y^2 \ при: X\geq Y\\(a+b)\cdot(b-\frac{X}{Y})\end{cases}$	A = 10, b = 60 X и Y целые числа
5	$R=\begin{cases}(a-b)\cdot X^2+(a+b)\cdot Y^2 \ при: X<Y\\(a+b)\cdot(b-\frac{X}{Y})\end{cases}$	A = 0,8, b = 6, 2 X и Y действительные числа
6	$R=\begin{cases}(a+b)\cdot(X+Y) \ при: \ \frac{X}{Y}\cdot\lvert X-Y\rvert\geq 0\\(a-b)\cdot(X-Y)\end{cases}$	A = 30, b = 50 X и Y целые числа
7	$R=\begin{cases}(a+b)\cdot(X+Y) \ при: \ \frac{X}{Y}\cdot\lvert X-Y\rvert<0\\(a-b)\cdot(X-Y)\end{cases}$	A = 60, b = 30 X и Y целые числа
8	$R=\begin{cases}a\cdot X-b\cdot Y \ при: X<Y\\X-Y\end{cases}$	A = 0,9, b = 3, 2 X и Y действительные числа

9	$R = \begin{cases} a \cdot X - b \cdot Y \ \ при: X \geq Y \\ X^2 - Y^2 \end{cases}$	A = 0,7, b = 9, 2 X и Y действительные числа
10	$R = \begin{cases} \lvert a \cdot X + b \cdot Y \rvert \ \ при: X < Y \\ \lvert b \cdot X - a \cdot Y \rvert \end{cases}$	A = 30, b = 20 X и Y целые числа
11	$R = \begin{cases} a \cdot X + b \cdot Y \ при: X < Y \\ (b - X) \cdot Y \end{cases}$	A = 3,24, b = 0,05 X и Y действительные числа
12	$R = \begin{cases} (a + b) \cdot X + (a - b) \cdot Y \ при: X < Y \\ (a \cdot b - X) \cdot Y \end{cases}$	A = 8, b = 6 X и Y целые числа
13	$R = \begin{cases} (a - b) \cdot X^2 + (a + b) \cdot Y^2 \ при: X < Y \\ (a + b) \cdot (b - \frac{X}{Y}) \end{cases}$	A = 0,8, b = 6, 2 X и Y действительные числа
14	$R = \begin{cases} (a + b) \cdot (X + Y) \ при: \ \frac{X}{Y} \cdot \lvert X - Y \rvert < 0 \\ (a - b) \cdot (X - Y) \end{cases}$	A = 60, b = 30 X и Y целые числа
15	$R = \begin{cases} a \cdot X - b \cdot Y \ \ при: X \geq Y \\ X^2 - Y^2 \end{cases}$	A = 0,7, b = 9, 2 X и Y действительные числа
16	$R = \begin{cases} \lvert a \cdot X + b \cdot Y \rvert \ \ при: X < Y \\ \lvert b \cdot X - a \cdot Y \rvert \end{cases}$	A = 30, b = 20 X и Y целые числа
17	$R = \begin{cases} (a + b) \cdot X + (a - b) \cdot Y \ при: X \geq Y \\ a \cdot b - X \end{cases}$	A = 8,11, b = 6,26 X и Y целые числа
18	$R = \begin{cases} (a - b) \cdot X^2 + (a + b) \cdot Y^2 \ при: X \geq Y \\ (a + b) \cdot (b - \frac{X}{Y}) \end{cases}$	A = 10, b = 60 X и Y целые числа
19	$R = \begin{cases} (a + b) \cdot (X + Y) \ при: \ \frac{X}{Y} \cdot \lvert X - Y \rvert \geq 0 \\ (a - b) \cdot (X - Y) \end{cases}$	A = 30, b = 50 X и Y целые числа
20	$R = \begin{cases} a \cdot X - b \cdot Y \ при: X < Y \\ X - Y \end{cases}$	A = 0,9, b = 3, 2 X и Y действительные числа
21	$R = \begin{cases} \lvert a \cdot X + b \cdot Y \rvert \ \ при: X < Y \\ \lvert b \cdot X - a \cdot Y \rvert \end{cases}$	A = 30, b = 20 X и Y целые числа

22	$R = \begin{cases} a \cdot X + b \cdot Y \ \ при: X < Y \\ (b - X) \cdot Y \end{cases}$	A = 3,24, b = 0,05 X и Y действительные числа
23	$R = \begin{cases} (a + b) \cdot X + (a - b) \cdot Y \ \ при: X < Y \\ (a \cdot b - X) \cdot Y \end{cases}$	A = 8, b = 6 X и Y целые числа
24	$R = \begin{cases} (a - b) \cdot X^2 + (a + b) \cdot Y^2 \ \ при: X < Y \\ (a + b) \cdot (b - \frac{X}{Y}) \end{cases}$	A = 0,8, b = 6, 2 X и Y действительные числа
25	$R = \begin{cases} (a + b) \cdot (X + Y) \ \ при: \ \frac{X}{Y} \cdot \lvert X - Y \rvert < 0 \\ (a - b) \cdot (X - Y) \end{cases}$	A = 60, b = 30 X и Y целые числа

4. ЦИКЛИЧЕСКИЙ ВЫЧИСЛИТЕЛЬНЫЙ ПРОЦЕСС

Существует три вида конструкции циклов, определенных операторами: **Repeat**, **While** и **For**. Выбор конструкции зависит от характера циклического процесса. Если известны начальные и конечные состояния, то рекомендуется применять оператор **For**. Если же существует уверенность в том, что программный блок должен быть выполнен, по меньшей мере, один раз, и может быть установлена необходимость повторного выполнения данного блока, то следует использовать оператор **Repeat**. Этот оператор называется оператором цикла с условием завершения. Если неизвестно, должен ли выполняться программный блок вообще, рекомендуется применять оператор **While**.

В случае выбора цикла-счетчика (оператор **For**) имеется возможность организации обратного счета с использованием оператора **downto**.

4.1. Конструкция цикла счетчика (for .. to .. do)

Предположим, нам необходимо вычислить сумму ***s*** и произведение ***p*** всех значений элементов одномерного массива ***a[..]***. Для решения подобных задач наиболее подходит конструкция цикла – счетчика, образованного операторами (**for**, **to**, **do**).

Рассмотрим следующий пример, в котором массив задан последовательностью действительных чисел:

const a: array[1..5] **of** real = (2.34, -7.82, 5.55, -0.23, 1.76);

Для наглядности, представим указанный массив строкой таблицы:

Таблица 4.1

Порядковый № (индекс *i*)	1	2	3	4	5
Значение	2.34	-7.82	5.55	-0.23	1.76

Становится очевидным тот факт, что каждому порядковому номеру (индексу) соответствует конкретное значение строки таблицы (массива), то есть, обратившись по индексу ***i,*** мы получим конкретное значение ***a***. Например: значение a[4] будет равно -0.23.

Следовательно, чтобы решить поставленную задачу, нам необходимо для вычисления суммы ***s*** просуммировать все значения элементов массива по индексу ***i***, а для вычисления произведения ***p*** перемножить все элементы, т.е:

$$s = \sum_{i=1}^{n} a_i \ , \qquad (4.1)$$

$$p = \prod_{i=1}^{n} a_i \ , \qquad (4.2)$$

где**: i = 1, 2, .. , n.**

Стоит вопрос, какие начальные значения должны иметь переменные ***s*** и ***p***. Очевидно, что для вычисления суммы ***s*** начальное значение переменной должно быть равно 0, а для вычисления произведения ***p*** начальное значение переменной ***p*** должно быть равно 1.

Тогда, фрагмент текста программы, реализующий нашу задачу, будет иметь вид:

```
Const
a: array[1..5] of  real = (2.34, -7.82,  5.55,  -0.23, 1.76);          // Описание массива
var
i: integer;                                   // объявление переменной – счетчика цикла
s, p: real;          // объявление переменных – аккумуляторов суммы и произведения
OutR: string;                                 // объявление переменной результата
begin
  s := 0;  p := 1;                    // Определение начальных значений
```

```
переменных
 for i := 1 to 5 do                    // В цикле от i = 1 до 5
 begin
 s := s + a[i];                        // вычисляем сумму s
 p := p * a[i];                        // вычисляем произведение p
 end;
                        {Преобразуем результаты в строку}
 OutR := 'Сумма s = ' + FloatToStr(s) + ' Произведение p = ' + FloatToStr(p);
 ShowMessage(OutR);                    // выводим результат
end;
```

В указанном выше примере массив a[..] описан константой. Можно описать данный массив как переменную. Тогда программа имеет следующий вид:

```
Var
a: array[1..5] of real;                                    // Объявление
массива a[..]
i: integer;
s, p: real;
OutR: string;
begin
 a[1] := 2.34;  a[2] := -7.82; a[3] := 5.55;               // Присвоение
значений
 a[4] := -0.23; a[5] := 1.76;                              // элементам
массива a[..]
 s := 0;  p := 1;
 for i := Low(a) to High(a) do
 begin
 s := s + a[i];
 p := p * a[i];
 end;
 OutR := 'Сумма s = ' + FloatToStr(s) + ' Произведение p = ' + FloatToStr(p);
 ShowMessage(OutR);
end;
```

Следует заметить, что на практике бывают случаи, когда пользователь не знает заранее ни размеров массива, ни границ индексов, например: $\mathbf{i_n = -x;\ i_k = x}$. Для таких случаев в **Delphi** имеются функции **Low** и **High** – возвращающие нижнюю и верхнюю границы массива соответственно. Следовательно, более корректно будет написать цикл с использованием указанных функций:

```
for i := Low(a) to High(a) do
begin
s := s + a[i];
p := p * a[i];
end;
```

Обратите внимание на тот факт, что в теле цикла используется внутренняя рекурсия, при которой переменные ***s*** и ***p*** ссылаются сами на себя, а точнее - на свои предыдущие значения: **s := s + a[i]** и **p := p * a[i]**, где: **a[i]** означает значение элемента массива по счетчику **i**.

4.2. Лабораторная работа №3

Рассмотрим подробнее правило написания программы и алгоритм ее реализации. Для разработки программы будем использовать форму и кнопку **Button**, установленную в форму.

Шаг 1. Определим типы переменных и констант. Создадим процедуру обработчика события щелчка кнопки, где опишем константы и объявим переменные.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array[1..5] of  real = (2.34, -7.82,  5.55,  -0.23, 1.76);          // Описание массива
var
i: integer;                                                  // объявление переменной – счетчика цикла
s, p: real;              // объявление переменных – аккумуляторов суммы и произведения
```

```
OutR: string;                                    // объявление переменной результата
begin

end;
```

Шаг 2. Определим начальные значения переменных и опишем вывод результата.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array[1..5] of  real = (2.34, -7.82,  5.55,  -0.23, 1.76);          // Описание массива
var
i: integer;                                  // объявление переменной – счетчика цикла
s, p: real;              // объявление переменных – аккумуляторов суммы и произведения
OutR: string;                                // объявление переменной результата
begin
  s := 0;  p := 1;                    // Определение начальных значений переменных

                                {Преобразуем результаты в строку}
 OutR := 'Сумма s = ' + FloatToStr(s) + ' Произведение p = ' + FloatToStr(p);
  ShowMessage(OutR);                 // выводим результат
end;
```

Запустим программу на выполнение (команда RUN). В результате выполнения программы должно быть выведено окно сообщения с результирующей строкой:

Сумма **s = 0** Произведение **p = 1.**

Закрыть приложение и сохранить проект.

Шаг 3. Опишем конструкцию пустого цикла.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array[1..5] of  real = (2.34, -7.82,  5.55,  -0.23, 1.76);          // Описание массива
var
i: integer;                                  // объявление переменной – счетчика цикла
s, p: real;              // объявление переменных – аккумуляторов суммы и произведения
OutR: string;                                // объявление переменной результата
begin
  s := 0;  p := 1;                    // Определение начальных значений переменных

for i := Low(a) to High(a) do  // В цикле от нижней границы до верхней границы масс. A[..]
  begin

  end;
                              {Преобразуем результаты в строку}
 OutR := 'Сумма s = ' + FloatToStr(s) + ' Произведение p = ' + FloatToStr(p);
  ShowMessage(OutR);                // выводим результат
end;
```

С целью проверки отсутствия ошибок запустим программу на выполнение.

Шаг 4. Опишем действия внутри конструкции цикла.

```
procedure TForm1.Button1Click(Sender: TObject);
const
```

```
a: array[1..5] of  real = (2.34, -7.82,  5.55,  -0.23, 1.76);          // Описание массива
var
i: integer;                                   // объявление переменной – счетчика цикла
s, p: real;          // объявление переменных – аккумуляторов суммы и произведения
OutR: string;                                 // объявление переменной результата
begin
  s := 0;  p := 1;                    // Определение начальных значений переменных

 for i := Low(a) to High(a) do  // В цикле от нижней границы до верхней границы масс. A[..]
  begin
  s := s + a[i];                      // вычисляем сумму элементов массива  s
  p := p * a[i];                      // вычисляем произведение элементов массива p
  end;
                              {Преобразуем результаты в строку}
 OutR := 'Сумма s = ' + FloatToStr(s) + ' Произведение p = ' + FloatToStr(p);
  ShowMessage(OutR);                  // выводим результат
end;
```

Запустить программу. В результате ее выполнения должно быть выведено сообщение:

Сумма s = 1,6 Произведение p = 41,110816032.

Сохранить проект программы.

Внесем некоторые изменения в постановку задачи и посмотрим, какие изменения в программе последуют за этим.

Предположим, необходимо вычислить сумму ***s*** только положительных значений элементов массива и произведение ***p*** только отрицательных

элементов. Для этого будет достаточно включить в тело цикла проверку условий вида:

if a[i] >= 0 **then** s := s + a[i]; *// вычисляем сумму элементов массива s при a[i] >= 0*

if a[i] < 0 **then** p := p * a[i]; *// вычисляем произведение элементов массива p при a[i] < 0*

Вводим значения элементов массива с использованием функции **InputBox**.

```
procedure TForm1.Button1Click(Sender: TObject);
var
a: array[1..5] of real;
i: integer;
s, p: real;
OutR, InpN, temp : string;
begin
  for i := Low(a) to High(a) do
  begin
  Temp := 'Введите значение ' + IntToStr(i) + ' элемента массива' ;
  InpN := InputBox('Ввод данных', Temp, '0');
  a[i] := StrToFloat(InpN);
  end;

  s := 0;  p := 1;
  for i := Low(a) to High(a) do
  begin
  if a[i] >= 0 then s := s + a[i] else p := p * a[i];
  end;
  OutR := 'Сумма s = ' + FloatToStr(s) + ' Произведение p = ' + FloatToStr(p);
  ShowMessage(OutR);
end;
```

4.3. Индивидуальные задания к лабораторной работе №3

№	Задан массив a[..]	Решить задачу (написать программу)
1	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить сумму четных значений элементов массива.
2	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить сумму нечетных значений элементов массива.
3	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить произведение значений элементов массива, кратных 3.
4	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить произведение значений элементов массива, кратных 4.
5	12, 34, -11, -1, 16, 21, -7, 5	Найти наименьшее значение элемента в массиве.
6	12, 34, -11, -1, 16, 21, -7, 5	Найти наибольшее значение элемента в массиве.
7	12, 34, -11, -1, 16, 21, -7, 5	Найти номер (индекс) элемента массива, имеющего наименьшее значение.
8	12, 34, -11, -1, 16, 21, -7, 5	Найти номер (индекс) элемента массива, имеющего наибольшее значение.
9	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить среднее значение элементов массива.
10	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить значение $x = p/s$, где p – произведение всех элементов массива, а s – сумма всех элементов.
11	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить сумму четных значений элементов массива.
12	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить произведение значений элементов массива, кратных 3.
13	12, 34, -11, -1, 16, 21, -7, 5	Найти наименьшее значение элемента в массиве.
14	12, 34, -11, -1, 16, 21, -7, 5	Найти номер (индекс) элемента массива, имеющего наименьшее значение.

15	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить среднее значение элементов массива.
16	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить значение ***x = p/s,*** где ***p*** – произведение всех элементов массива, а ***s*** – сумма всех элементов.
17	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить сумму нечетных значений элементов массива.
18	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить произведение значений элементов массива, кратных 4.
19	12, 34, -11, -1, 16, 21, -7, 5	Найти наибольшее значение элемента в массиве.
20	12, 34, -11, -1, 16, 21, -7, 5	Найти номер (индекс) элемента массива, имеющего наибольшее значение.
21	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить значение ***x = p/s,*** где ***p*** – произведение всех элементов массива, а ***s*** – сумма всех элементов.
22	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить сумму четных значений элементов массива.
23	1, 6, 3, 7, 12, 11, 5, 4, 17, 8	Вычислить произведение значений элементов массива, кратных 3.
24	12, 34, -11, -1, 16, 21, -7, 5	Найти наименьшее значение элемента в массиве.
25	12, 34, -11, -1, 16, 21, -7, 5	Найти номер (индекс) элемента массива, имеющего наименьшее значение.

4.4. Конструкция цикла Repeat … Until

Цикл **Repeat** (повторять) наиболее удобно применять в случаях, когда необходимо выполнять вычисления до достижения некоторых значений **Until,** например**: x >= n,** либо выполнять вычисления с заданным шагом ***i***, являющимся действительным числом.

Рассмотрим следующий пример. Предположим, что нам необходимо найти значение ***x,*** при котором функция ***y = f(x)*** достигнет некоторого значения.

Примечание: Заведомо известно, что функция **y = f(x)** будет иметь заданное значение в выбранном диапазоне значений **x.**

Рассмотрим функцию $y = e^x / 2 * x$ и определим, при каких значениях ***x*** значение ***y*** будет равно 7. Для решения задачи выберем диапазон значений ***x*** изменяющийся в пределах -1 ..3 с шагом 0,01 (точность вычисления).

4.5. Лабораторная работа №4

Для решения задачи создадим в **Delphi** новый проект приложения. Установим в форму кнопку. В обработчике события щелчка кнопки напишем следующий текст программы:

```
procedure TForm1.Button1Click(Sender: TObject);
var
x, y: double;                                   // Объявление переменных
x и y
OutR: string;                                   // Объявление переменной
сообщения
begin
  x := - 1;                                     // Определения начального значения
переменной x
  repeat                                        // Повторение до выполнения
условия until
 if x <> 0 then  y := exp(x)/2*x;               // Вычисление значения y при x <> 0
  x := x + 0.01;                                // Увеличение значения x
  until y >= 7;                                 // Завершение цикла по условию y >=
7
  OutR := 'Значение x = ' + FloatToStr(x);      // Формирование строки
сообщения
  ShowMessage(OutR);                            // Вывод сообщения
end;
```

В результате выполнения программы будет выведено сообщение: “Значение x = 1,97999999999998”. Из текста программы видно, что точность

вычисления зависит от величины приращения ***x***. Например, если установить шаг, равный 0,000001, то результат будет более точным и равен 1,9640599999404.

Следует обратить внимание на тот факт, что сравнение выхода из цикла типа **repeat until** должно быть не точным, а соответствовать условию (>=) - для возрастающей функции, либо (<=) - для убывающей функции, т.к. четкое равенство может не выполняться, и цикл не будет иметь завершения.

Стоит вопрос, а если не известно, достигнет ли функция заданного значения, и завершится ли цикл вообще? В таких случаях обычно используется проверка завершения цикла для двух и более значений, например:

```
procedure TForm1.Button1Click(Sender: TObject);
var
x, y: double;
OutR: string;
begin
  x := - 1;
  repeat
  if x <> 0 then y := exp(x)/2*x;
  x := x + 0.000001;
  until (y >= 7) or (x = 3);              // комплексная проверка условий y >= 7 или x >= 3
  if y >= 7  then                                    // если y >= 7 то:
  OutR := 'Значение x = ' + FloatToStr(x)          //формируем сообщение 'Значение x = ‘
  else
 OutR := 'Значение не достигнуто ';       //формируем сообщение 'Значение не достигнуто ‘
  ShowMessage(OutR);
end;
```

В приведенном выше примере цикл всегда будет иметь завершение либо по условию достижения функцией заданного значения, либо по достижению граничных условий (верхней границы значений x > = 3). Например, если мы будем исследовать заданную функцию в диапазоне значений x от -1 до 1, мы получим сообщение: “Значение не достигнуто”.

4.6. Индивидуальные задания к лабораторной работе №4

№	Задания (вычисление выполнить с точностью 0,001)
1	Найти площадь отрицательной области, ограниченной кривой, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений *x:* *-2,5 < x > 2,5.*
2	Найти протяженность кривой $y = \frac{9{,}95 * x}{e^x}$ в условных единицах точности вычисления, заданной функцией в диапазоне значений *x: 0 < x > 3,5.*
3	Найти площадь, ограниченную кривой, заданной функцией $y = \frac{e^x}{0{,}5 * x}$ и осью *x* в диапазоне значений *x: -3 < x > 1*
4	Найти минимальное значение функции $y = \frac{e^x}{0{,}5 * x}$ в диапазоне значений *x: -2 < x > 1.*
5	Найти максимальное значение функции $y = \frac{9{,}95 * x}{e^x}$ в диапазоне значений *x: 0 < x > 2*
6	Найти протяженность кривой в условных единицах точности вычисления, заданной функцией $y = \frac{e^x}{0{,}5 * x}$ в диапазоне значений *x: -3 < x > 0,5.*
7	Найти протяженность кривой $y = \frac{9{,}95 * x}{e^x}$ в условных единицах точности вычисления, заданной функцией в диапазоне значений *x: 0 < x > 3,5.*
8	Найти площадь, ограниченную кривой, заданной функцией $y = \frac{e^x}{0{,}5 * x}$ и осью *x* в диапазоне значений *x: -3 < x > 1.*
9	Найти площадь положительной области, ограниченной кривой, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений x: *-2,5 < x > 2,5.*

10	Найти протяженность кривой в условных единицах точности вычисления, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений *x: -2,5 < x > 2,5*.
11	Найти площадь отрицательной области, ограниченной кривой, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений *x:* *-2,5 < x > 2,5*.
12	Найти площадь, ограниченную кривой, заданной функцией $y = \frac{9{,}95 * x}{e^x}$ и осью *x* в диапазоне значений *x: 0 < x > 3,5*.
13	Найти максимальное значение функции $y = \frac{9{,}95 * x}{e^x}$ в диапазоне значений *x: 0 < x > 2*
14	Найти протяженность кривой $y = \frac{9{,}95 * x}{e^x}$ в условных единицах точности вычисления, заданной функцией в диапазоне значений *x: 0 < x > 3,5*.
15	Найти площадь положительной области, ограниченной кривой, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений x: *-2,5 < x > 2,5*.
16	Найти протяженность кривой в условных единицах точности вычисления, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений *x: -2,5 < x > 2,5*.
17	Найти протяженность кривой в условных единицах точности вычисления, заданной функцией $y = \frac{e^x}{0{,}5 * x}$ в диапазоне значений *x: -3 < x > 0,5*.
18	Найти минимальное значение функции $y = \frac{e^x}{0{,}5 * x}$ в диапазоне значений *x: -2 < x > 1*.
19	Найти протяженность кривой $y = \frac{9{,}95 * x}{e^x}$ в условных единицах точности вычисления, заданной функцией в диапазоне значений *x: 0 < x > 3,5*.

20	Найти площадь, ограниченную кривой, заданной функцией $y = \frac{9{,}95 * x}{e^x}$ и осью x в диапазоне значений x: $0 < x > 3{,}5$. Площадь вычислять в условных единицах точности вычисления значения функции.
21	Найти протяженность кривой в условных единицах точности вычисления, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений x: $-2{,}5 < x > 2{,}5$.
22	Найти площадь отрицательной области, ограниченной кривой, заданной функцией $y = \frac{\sin(x)}{e^x}$ в диапазоне значений x: $-2{,}5 < x > 2{,}5$.
23	Найти площадь, ограниченную кривой, заданной функцией $y = \frac{e^x}{0{,}5 * x}$ и осью x в диапазоне значений x: $-3 < x > 1$. Площадь вычислять в условных единицах точности вычисления значения функции.
24	Найти максимальное значение функции $y = \frac{9{,}95 * x}{e^x}$ в диапазоне значений x: $0 < x > 2$
25	Найти протяженность кривой $y = \frac{9{,}95 * x}{e^x}$ в условных единицах точности вычисления, заданной функцией в диапазоне значений x: $0 < x > 3{,}5$.

4.7. Конструкция цикла While … Do

Различие между оператором **while** и оператором **repeat** состоит в следующем. В конструкции **while** сначала анализируется условие выхода из цикла, после чего вызывается сам оператор. В конструкции **repeat .. until** все происходит наоборот. Число циклов в конструкции **while** может быть 0 или больше, а в конструкции **repeat .. until** - 1 или больше. В конструкции **while** итерация производится до тех пор, пока выполняется условие, в конструкции **repeat .. until** итерация выполняется только до тех пор, пока условие не будет выполнено. При написании программного кода необходимо учитывать, что в

случае применения конструкции **while .. do** вначале должно быть инициализировано условие выхода из цикла, например**:**

```
var Number: integer;
begin
Number := 9;
while Number > 0 do
Number := Number - 2;
end;
```

В результате выполнения указанного примера переменная Number примет значение, равное 1. В случае, если переменная Number будет иметь начальное значение меньше 0, цикл не будет выполняться вообще.

4.8. Лабораторная работа №5

Пусть нам задана нелинейная функция, в которой необходимо вычислить значение ***r*** при значениях ***x,*** изменяемых в заданном диапазоне с шагом ***i***.

$$r = \sum_{i=0}^{1}(2x_i + 3) - \sum_{i=1}^{2}({x_i}^2 + 3) \quad при \;\; i = 0{,}01 \qquad (4.1)$$

Данную задачу можно было бы решить, используя конструкции **if .. then .. else** в цикле **repeat**, с использованием счетчика ***i*** в диапазоне значений (0..2). Однако, для решения данной задачи можно не использовать конструкцию **if .. then .. else**, а проверку условия вычисления сумм выполнить непосредственно в циклах **while** .. **do**, выполнив последовательно два цикла в диапазонах значений ***i*** (0..1) и (1..2).

Для выполнения работы создадим в Delphi новый проект, содержащий форму и кнопку **Button.**

Рассмотрим по шагам алгоритм решения задачи.

Шаг 1. Определим имена констант и переменные, их типы. Опишем константы и переменные в обработчике события щелчка кнопки **Button1.**

```
procedure TForm1.Button1Click(Sender: TObject);
const
i:real = 0.01;                                    // приращение
счетчика
var
x, s1, s2: Real;                                  // используемые
переменные
OutR: string;                                     // переменная
результата
begin

end;
```

Выполнить команду **Save All**.

Шаг 2. Определим начальные значения переменных и вывод результата.

```
procedure TForm1.Button1Click(Sender: TObject);
const
i:real = 0.01;
var
x, s1, s2: Real;
OutR: string;
begin
  x := 0; s1 := 0; s2 := 0;

  OutR := FloatToStr(s1 - s2);   // Вычисление разности сумм и приведение R к
типу строка
  ShowMessage('R =  ' +  OutR);                   //Вывод
результата
end;
```

Выполнить команду **RUN** и проверить результат вывода. Выполнить команду **Save All**.

Шаг 3. Напишем пустые конструкции циклов вычисления сумм.

```
procedure TForm1.Button1Click(Sender: TObject);
const
i:real = 0.01;
var
x, s1, s2: Real;
OutR: string;
begin
  x := 0; s1 := 0; s2 := 0;
  while x <= 1 do       // тело цикла вычисления первой суммы
  begin
   end;

  while x <= 2 do      // тело цикла вычисления второй суммы
  begin

  end;

  OutR := FloatToStr(s1 - s2);
  ShowMessage('R = ' +  OutR);
end;
```

Выполнить команду **Save All**.

Примечание: Выполнять команду RUN на данном этапе разработки нельзя, т.к. еще не описаны изменения счетчиков x и программа “зависнет” – т.е. не будет выхода их циклов.

Шаг 4. В теле конструкций опишем вычисления сумм в соответствии с формулой.

```
procedure TForm1.Button1Click(Sender: TObject);
const
i:real = 0.01;
var
x, s1, s2: Real;
OutR: string;
```

```
begin
  x := 0; s1 := 0; s2 := 0;
  while x <= 1 do
  begin
   s1 := s1 + 2*x + 3;                    // вычисления 1-й суммы
   x := x + i;                            // вычисление  значения x
  end;
  while x <= 2 do
  begin
   s2 := s2 + Sqr(x) + 3;                 // вычисления 2-й суммы
   x := x + i;                            // вычисление  значения x
  end;

  OutR := FloatToStr(s1 - s2);
  ShowMessage('R = ' +  OutR);
end;
```

Сохранить проект (**Save ALL**) и выполнить команду **Run**. В результате работы программы должно быть выведено сообщение: “R = -132,875”.

Следует отметить, что перед началом второго цикла нет необходимости переопределять значение переменной x := 1, так как после выполнения первого цикла значение x будет равно 1.

5. РАБОТА С МНОГОМЕРНЫМИ МАССИВАМИ

Описать таблицу в **Delphi** можно в виде двумерного массива как константу, если заведомо известны и не изменяются значения таблицы:
Пусть нам задана таблица целых чисел вида:

Таблица 5.1

X/Y	***X1***	***X2***	***X3***
Y1	-4	-3	-2
Y2	-1	0	1
Y3	2	3	4

Необходимо выполнить некоторые операции над таблицей, например, найти сумму всех значений таблицы.

```
Const
a: array [1..3,1..3] of integer = ((-4, -3, -2),
                                   (-1,  0,  1),
                                   ( 2,  3,  4));
```

В виде переменной типа двумерный массив, последовательно присвоить значения элементам массива:

```
Var
b: array [1..3,1..3] of integer;
begin
  b[1,1] := -4; b[1,2] := -3; b[1,3] := -2;
  b[2,1] := -1; b[2,2] := 0;  b[2,3] := 1;
  b[3,1] :=  2; b[3,2] := 3;  b[3,3] := 4;
 end;
```

Таким образом, доступ к элементам массива обеспечивается посредством указателей, заключенных в квадратные скобки и разделенных запятой.

Для того, чтобы решить поставленную задачу, необходимо описать массив (таблицу) и просканировать по столбцам и строкам все его элементы с накоплением на переменной ***s*** суммы всех элементов. Вследствие того, что размер массива известен, и шагом сканирования будет целое число, воспользуемся конструкцией циклов счетчиков **for..to..do**.

5.1. Лабораторная работа №6

Рассмотрим последовательность создания программы:

Шаг 1. Создадим в **Delphi** проект, содержащий форму **Form1** и кнопку **Button1.**

Шаг 2. Опишем и объявим необходимые константы и переменные в обработчике события щелчка кнопки.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array [1..3,1..3] of integer = ((-4, -3, -2),       // Константа – двумерный
массив данных
                                  (-1,  0,  1),
                                  ( 2,  3,  4));
var
i, j, s: integer;                                  // Переменные циклов i, j и суммы
- s
OutR: string;                                      // Переменная результата
begin

end;
```

Шаг 3. Определим начальное значение переменной аккумулятора **s** и опишем конструкцию вложенных циклов, а также вывод результата **OutR**;

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array [1..3,1..3] of integer = ((-4, -3, -2),
                                  (-1,  0,  1),
                                  ( 2,  3,  4));
var
i, j, s: integer;
OutR: string;
begin
  s := 0;
  for j := 1 to 3 do    // в цикле от 1 до 3 определяем индексы строк (сканируем
ось Y )
  begin
    for i := 1 to 3 do  // в цикле от 1 до 3 определяем индексы столбцов
(сканируем ось X )
    begin


    end;
```

```
  end;
  OutR := IntToStr(s);                    // приводим результат к
строковому типу
  ShowMessage(OutR);                      // выводим результат
end;
```

Шаг 4. В теле вложенного цикла описываем действие (вычислим сумму элементов массива).

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array [1..3,1..3] of integer = ((-4, -3, -2),
                                   (-1,  0,  1),
                                   ( 2,  3,  4));
var
i, j, s: integer;
OutR: string;
begin
  s := 0;
  for j := 1 to 3 do    // в цикле от 1 до 3 определяем индексы строк (сканируем
ось Y )
  begin
    for i := 1 to 3 do  // в цикле от 1 до 3 определяем индексы столбцов
(сканируем ось X )
    begin
    s := s + a[i,j];                  // накопление суммы значений элементов
массива
    end;
  end;
  OutR := IntToStr(s);                    // приводим результат к
строковому типу
  ShowMessage(OutR);                      // выводим результат
end;
```

Шаг 5. Сохранить проект и выполнить команду Run.

Для данного массива результат суммы всех его элементов будет равен нулю.

В нашей задаче в теле вложенного цикла нет блоков, также отсутствуют какие либо действия до выполнения вложенного цикла. В связи с этим можно оптимизировать текст программы, исключив лишние операторные скобки **begin .. end**.

Следует отметить тот факт, что для обработки отдельных строк и столбцов двумерного массива (таблицы) нет необходимости применять вложенные циклы, например, для вычисления суммы элементов первой строки исходный текст программы будет иметь следующий вид:

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array [1..3,1..3] of integer = ((-4, -3, -2), (-1,  0,  1), ( 2,  3,  4));
var
i, j, s: integer;
OutR: string;
begin
  i := 1;                                                    // Указываем первую
строку
  s := 0;
    for j := 1 to 3 do  s := s + a[i,j];    // Сканируем столбцы и подсчитываем
сумму
  OutR := IntToStr(s);
  ShowMessage(OutR);
end;
```

Также для вычисления суммы элементов главной и дополнительной диагоналей квадратного массива нет необходимости применять вложенные циклы. Ниже приводится текст программы, вычисляющей сумму элементов главной и дополнительной диагоналей массива.

```
procedure TForm1.Button1Click(Sender: TObject);
const
a: array [1..3,1..3] of integer = ((-1, -3, -2),
                                   (-1,  0,  1),
                                   (-4,  3,  4));
```

```
var
i, sG, sD: integer;
OutR: string;
begin
  sG := 0; sD := 0;
    for i := 1 to 3 do
    begin
    sG := sG + a[i,i];
    sD := sD + a[4-i,i];
    end;
  OutR := 'Сумма элементов sG = ' + IntToStr(sG) + ' Сумма элементов sD = ' +
IntToStr(sD) ;
  ShowMessage(OutR);
end;
```

5.2. Индивидуальные задания к лабораторной работе №6

1. Задана таблица чисел.

Таблица 5.2

X/Y	***X1***	***X2***	***X3***	***X4***
Y1	-5,25	-3,25	3,25	0,25
Y2	-3,25	-6,25	1,25	1,25
Y3	1,25	-1,25	6,25	3,25
Y4	0,25	1,25	3,25	5,25

Написать программу вычисления:

№	Задание
1	Вычислить сумму всех положительных значений элементов таблицы (массива)
2	Вычислить произведение всех положительных значений элементов таблицы (массива)

3	Вычислить сумму всех отрицательных значений элементов таблицы (массива)
4	Вычислить произведение всех отрицательных значений элементов таблицы (массива)
5	Вычислить сумму всех значений элементов таблицы (массива), больших (-3)
6	Вычислить произведение всех значений элементов таблицы (массива), больших (-3)
7	Вычислить сумму всех значений элементов таблицы (массива), меньших (3)
8	Вычислить произведение всех значений элементов таблицы (массива), меньших (3)
9	Вычислить среднее арифметическое значение всех положительных элементов таблицы (массива)
10	Вычислить среднее арифметическое значение всех отрицательных элементов таблицы (массива)
11	Вычислить сумму всех положительных значений элементов таблицы (массива)
12	Вычислить сумму всех отрицательных значений элементов таблицы (массива)
13	Вычислить сумму всех значений элементов таблицы (массива), больших (-3)
14	Вычислить сумму всех значений элементов таблицы (массива), меньших (3)
15	Вычислить среднее арифметическое значение всех положительных элементов таблицы (массива)
16	Вычислить среднее арифметическое значение всех отрицательных элементов таблицы (массива)
17	Вычислить произведение всех положительных значений элементов таблицы (массива)
18	Вычислить произведение всех отрицательных значений элементов таблицы (массива)
19	Вычислить произведение всех значений элементов таблицы (массива), больших (-3)

20	Вычислить произведение всех значений элементов таблицы (массива), меньших (3)
21	Вычислить среднее арифметическое значение всех отрицательных элементов таблицы (массива)
22	Вычислить сумму всех положительных значений элементов таблицы (массива)
23	Вычислить сумму всех отрицательных значений элементов таблицы (массива)
24	Вычислить сумму всех значений элементов таблицы (массива), больших (-3)
25	Вычислить сумму всех значений элементов таблицы (массива), меньших (3)

2. Задана таблица чисел.

Таблица 5.3

X/Y	***X1***	***X2***	***X3***	***X4***
Y1	6	12	11	5
Y2	32	33	25	15
Y3	3	14	5	8
Y4	12	-8	-7	4

Написать программу вычисления:

№	**Задание**
1	Вычислить произведение четных значений элементов главной диагонали таблицы (массива)
2	Вычислить сумму нечетных значений элементов дополнительной диагонали таблицы (массива)
3	Вычислить сумму отрицательных значений элементов четвертой строки таблицы (массива)

4	Вычислить произведение положительных значений элементов четвертой строки таблицы (массива)
5	Вычислить произведение суммы элементов главной диагонали и элементов дополнительной диагонали массива
6	Вычислить произведение четных значений элементов второго столбца массива (таблицы)
7	Вычислить произведение нечетных значений элементов четвертого столбца массива (таблицы)
8	Вычислить сумму кратных 5 элементов второй строки таблицы (массива)
9	Вычислить отношение суммы элементов третьего столбца таблицы к сумме элементов четвертого столбца таблицы
10	Вычислить отношение произведения элементов первой строки к сумме значений элементов третьей строки таблицы
11	Вычислить произведение четных значений элементов главной диагонали таблицы (массива)
12	Вычислить сумму отрицательных значений элементов четвертой строки таблицы (массива)
13	Вычислить произведение суммы элементов главной диагонали и элементов дополнительной диагонали массива
14	Вычислить произведение нечетных значений элементов четвертого столбца массива (таблицы)
15	Вычислить отношение суммы элементов третьего столбца таблицы к сумме элементов четвертого столбца таблицы
16	Вычислить сумму нечетных значений элементов дополнительной диагонали таблицы (массива)
17	Вычислить отношение произведения элементов первой строки к сумме значений элементов третьей строки таблицы
18	Вычислить сумму нечетных значений элементов дополнительной диагонали таблицы (массива)
19	Вычислить произведение положительных значений элементов четвертой строки таблицы (массива)
20	Вычислить произведение четных значений элементов второго столбца массива (таблицы)

21	Вычислить сумму кратных 5 элементов второй строки таблицы (массива)
22	Вычислить произведение четных значений элементов главной диагонали таблицы (массива)
23	Вычислить сумму отрицательных значений элементов четвертой строки таблицы (массива)
24	Вычислить произведение суммы элементов главной диагонали и элементов дополнительной диагонали массива
25	Вычислить произведение нечетных значений элементов четвертого столбца массива (таблицы)

ЛИТЕРАТУРА

1. Иващенко В.П., Швачич Г.Г, Овсянников А.В. Основы информационных технологий и программирование в среде Delphi: Учебное пособие. – Днепропетровск: РВА “Дніпро - VAL”, 2008. – 464 с.
2. Хоманченко А., Гофман В., Мещеряков Е., Delphi 7. Полное руководство. – СПб.: БХВ - Петербург, 2010. – 616 с.
3. Культин Н.С. Delphi 7. Программирование на Object Pascal. – СПб.: БХВ - Петербург, 2009 – 342 с.
4. Бобровский С. Delphi 7. Учебный курс. – Ростов на Дону: Феникс, 2008.– 452 с.
5. Марко Кэнту. Delphi 7 для профессионалов. – СПб.: Питер, 2008. – 453 с.

ПРИЛОЖЕНИЕ А

Арифметические операции

Символы операции	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
Div	Целочисленное деление
Mod	Остаток от деления

Операции сравнения

Символы операции	Операция
=	
<>	
>	
<	
>=	
<=	

Логические операции

Символы операции	Операция
And	Логическое умножение (И)
Or	Логическое сложение (ИЛИ)
Xor	Исключающее ИЛИ
Not	Логическое отрицание (НЕ)
Shl	Левый логический сдвиг
Shr	Правый логический сдвиг

ПРИЛОЖЕНИЕ Б

Таблица Б.1

Стандартные арифметические и математические процедуры и функции

Функция	Назначение (действие)	Unit
Abs	Определяет абсолютное значение аргумента	SysUtils
ArcCos	Определяет значение арккосинуса x	Math
ArcCosh	Определяет значение гиперболического арккосинуса аргумента	Math
ArcSin	Определяет значение арксинуса x	Math
ArcSinh	Определяет значение гиперболического арксинуса x	Math
ArcTan	Определяет арктангенс переданного аргумента	Math
ArcTan2	Вычисляет ArcTan(Y/X) и возвращает угол с учётом полученного квадранта	Math
ArcTanh	Определяет значение гиперболического арктангенса аргумента	Math
Cei1	Округляет значение в положительную сторону	Math
Cos	Определяет косинус x	SysUtils
Cosh	Определяет гиперболический косинус x	Math
Cotan	Определяет котангенс x	Math
CycleToRoad	Преобразовывает угол, заданный в количествах оборотов (циклов), в угол, заданный в радианах	Math
Dec	Уменьшает значение переменной	SysUtils
DegToRad	Преобразовывает угол из градусов в радианы	Math
Exp	Вычисляет экспоненциальное значение аргумента	SysUtils
Floor	Округляет переменную в сторону ближайшего меньшего целого числа	Math
FPower	Умножает аргумент на 10 в заданной степени (Val * 10**Pow)	Math
Frac	Возвращает дробную часть аргумента	Math
Frexp	Возвращает мантиссу и экспонент переданного значения	Math

Продолжение табл. Б1

GradToRad	Преобразовывает угол, заданный в десятичных градусах (Grad) в радианы	Math
High	Возвращает наибольшее значение в области значений аргумента	SysUtils
Hypot	Определяет длину гипотенузы прямоугольного треугольника	Math
Inc	Увеличивает значение переменной	SysUtils
Int	Возвращает целую часть передаваемого аргумента	SysUtils
IntPower	Определяет показатель степени исходя из значений основания и целочисленной экспоненты	Math
Ldexp	Определяет значение x*(2P)	Math
Ln	Определяет натуральный логарифм передаваемого аргумента	SysUtils
LnXP1	Определяет натуральный логарифм (x+1)	Math
Log10	Определяет десятичный логарифм	Math
Log2	Определяет двоичный логарифм	Math
LogN	Определяет логарифм с основанием N	Math
Low	Возвращает наименьшее значение в области значений аргумента	SysUtils
Mean	Определяет среднее арифметическое всех значений массива	Math
MaxIntValue	Возвращает наибольшее значение со знаком из ряда содержащихся в массиве целых чисел	Math
MaxValue	Возвращает наибольшее значение со знаком из ряда содержащихся в массиве действительных чисел	Math
MinIntValue	Возвращает наименьшее значение со знаком из ряда содержащихся в массиве целых чисел	Math
MinValue	Возвращает наименьшее значение со знаком из ряда содержащихся в массиве действительных чисел	Math
Norm	Определяет Евклидову норму	Math
Odd	Проверяет, является ли аргумент нечётным числом	SysUtils

Продолжение табл. Б1

Ord	Возвращает порядковый номер значения	SysUtils
Pi	Возвращает значение Пи	SysUtils
Poly	Вычисляет стандартный полином для параметра x	Math
Power	Возводит число в степень	Math
Pred	Возвращает предыдущее значение порядковой переменной	SysUtils
RadToDeg	Преобразовывает угол, указанный в радианах, в угол в градусах	Math
RadToGrad	Преобразовывает угол, указанный в радианах, в угол в десятичных градусах(Grad)	Math
Round	Округляет значение действительного типа до целочисленного значения	SysUtils
Sin	Возвращает синус передаваемого аргумента	SysUtils
SinCos	Определяет синус и косинус угла	Math
Sinh	Возвращает гиперболический синус угла	Math
Sqr	Возводит число в квадрат	SysUtils
Sqrt	Вычисляет корень квадратный	SysUtils
Succ	Возвращает последующее значение порядковой переменной	Math
Sum	Определяет сумму всех элементов массива	Math
SumInt	Определяет сумму всех элементов массива целых чисел	Math
SumOfSquares	Определяет сумму квадратов всех элементов массива	Math
SumAndSquares	Определяет сумму и сумму квадратов всех элементов массива	Math
Tan	Определяет тангенс аргумента	Math
Tanh	Определяет гиперболический тангенс аргумента	Math
Trunc	Отсекает дробную часть действительного числа	Math

СОДЕРЖАНИЕ

Навчальне видання

Швачич Геннадій Григорович
Овсянніков Олександр Васильович
Єфанова Ліна Михайлівна
Іващенко Юрій Сергійович

ОСНОВИ ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ DELPHI

Частина І.

Навчальний посібник

(російською мовою)

Тем. план 2013, поз. 251

Підписано до друку 01.07.2013. Формат 60х84 1/16. 11. Папір друк. Друк плоский. Облік.-вид. арк. 3,52. Умов. друк. арк. 3,48. Тираж 100 пр. Замовлення №

Національна металургійна академія України
49600, м. Дніпропетровськ-5, пр. Гагаріна,4

Редакційно-видавничий відділ НметАУ