Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки кадров

Кафедра «Информатика»

С. А. Чабуркина, Н. С. Емельянченко

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по одноименной дисциплине
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения

Гомель 2013

УДК 004.421+004.43(075.8)
ББК 32.973.26-018.1я73
Ч-12

Рекомендовано кафедрой «Информатика» ГГТУ им. П. О. Сухого
(*протокол № 5 от 27.11.2012 г.*)

Рецензент: канд. физ.-мат. наук, доц. каф. «Информационные технологии»
ГГТУ им. П. О. Сухого *О. А. Кравченко*

Чабуркина, С. А.

Ч-12 Основы алгоритмизации и программирования на языках высокого уровня : лаборатор. практикум по одноим. дисциплине для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заоч. формы обучения / С. А. Чабуркина, Н. С. Емельянченко. – Гомель : ГГТУ им. П. О. Сухого, 2013. – 84 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://library.gstu.by/StartEK/. – Загл. с титул. экрана.

Представлены задания к лабораторным работам, с помощью котрых выполняется разработка и практическая реализация основных видов алгоритмов на языке высокого уровня Delphi.

Рассматриваются три основных типа алгоритмов: линейные, разветвляющиеся и циклические, алгоритмы обработки одномерных и двухмерных статических и динамических массивов, алгоритмы обработки символьной информации и программирование с использованием подпрограмм.

Для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения.

УДК 004.421+004.43(075.8)
ББК 32.973.26-018.1я73

Лабораторная работа № 1
Проектирование интерфейса программ. Создание простого приложения в интегрированной среде разработки (ИСР) Turbo Delphi

Цель работы: Изучить интерфейс ИСР Turbo Delphi, этапы работы при создании приложения в ИСР; научиться создавать интерфейс простого приложения и процедуры обработки событий.

Теоретические сведения:

1. Запуск системы программирования Turbo Delphi.

Запуск Turbo Delphi выполняется с помощью ярлыка на Рабочем столе или через Стартовое меню. После загрузки на экране обычно находятся 6 окон: главное окно (содержит заголовок окна, строку меню, панель инструментов); окно менеджера проекта; палитра ВК (инструментов); окно инспектора объектов; окно структуры и рабочее окно, которое содержит страницу приветствия (при загрузке), форму или окно редактора кода

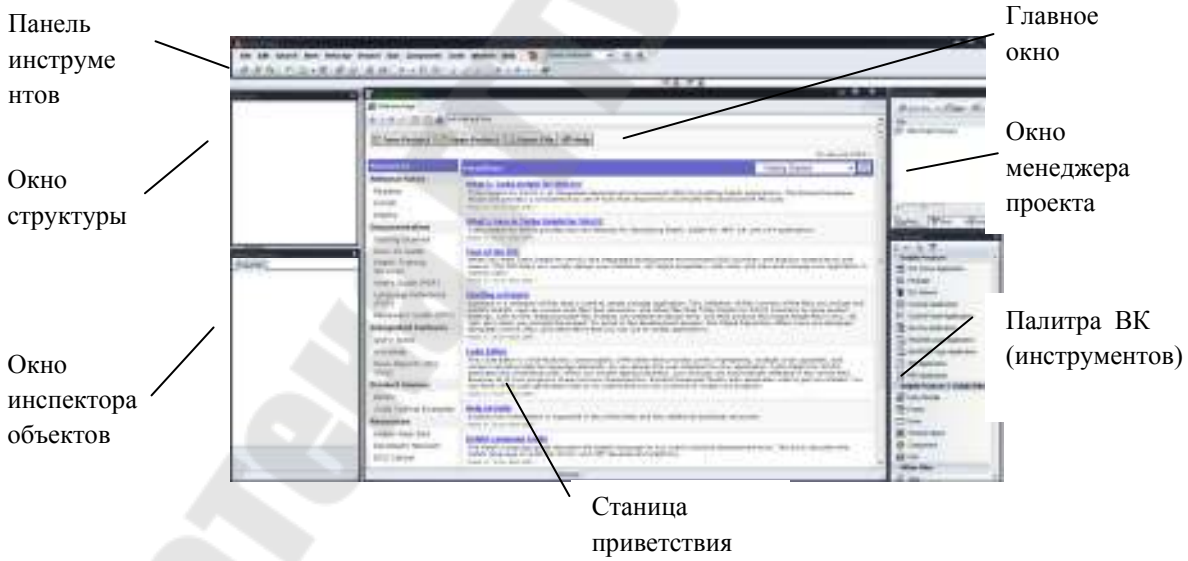


Рис 1.1. Вид окон системы программирования Turbo Delphi

Если какого-то окна нет на экране, то его можно вывести с помощью соответствующих команд меню *View*:

окно менеджера проекта	*Project Manager*	(Ctrl + Alt + F11)
палитра ВК (инструментов)	*Tool Palette*	(Ctrl + Alt + P)

окно инспектора объектов	*Object Inspector*	(F11)
окно структуры	*Structure*	(Shift + Alt + F11)
страница приветствия	*Welcome Page*	

Закрытие главного окна приводит к окончанию работы в ИСР Turbo Delphi.

2. Открытие существующего проекта или создание нового.

Создать новый проект **Project1** можно одним из трех способов:

- выполнить команду *File New – VCL Forms Application – Delphi Win 32;*
- в окне инструментов (Tool Palette) выбрать *VCL Forms Application;*
- на странице приветствия нажать кнопку *New Project* и в появившемся окне выбрать *VCL Forms Application*.

.Для открытия существующего проекта необходимо использовать команду *File – Open Project…*(**Ctrl + F11**) или кнопку **Open Project** на странице приветствия или на Панели Инструментов (ПИ)**.**

После создания или открытия проекта экран имеет следующий вид:

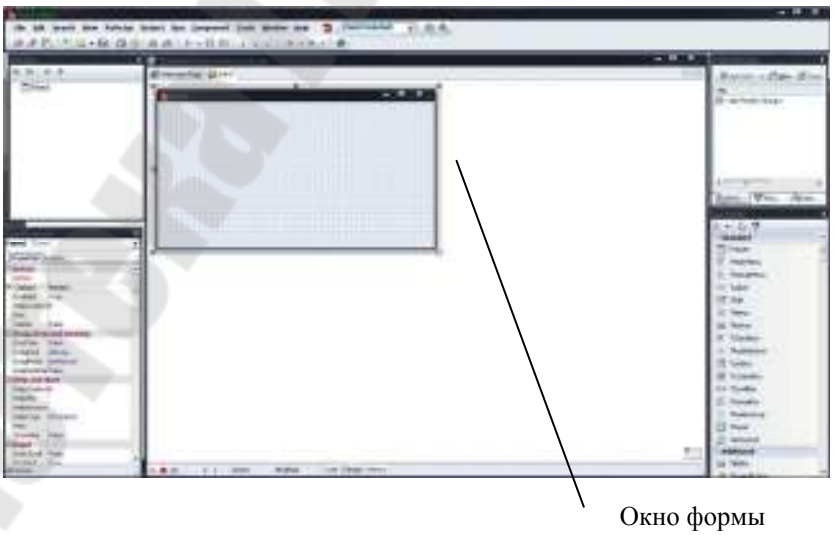


Рис 1.2. Вид окна после создания проекта

Переключение между формой и соответствующим ей модулем выполняется командой *View – Toggle Form/Unit* (**F12**) или с помощью

кнопок в строке состояния **Code** и **Design**.

Файл формы можно просмотреть с помощью команды *View as Text* контекстного меню формы в режиме проектирования. Возврат к форме команда *View as Form* контекстного меню текстового файла формы.

Окно проекта при необходимости можно вывести на экран с помощью команды *Project – View Source*. Закрытие окна редактора кода вызывает закрытие файла проекта, приводит к окончанию работы с ним.

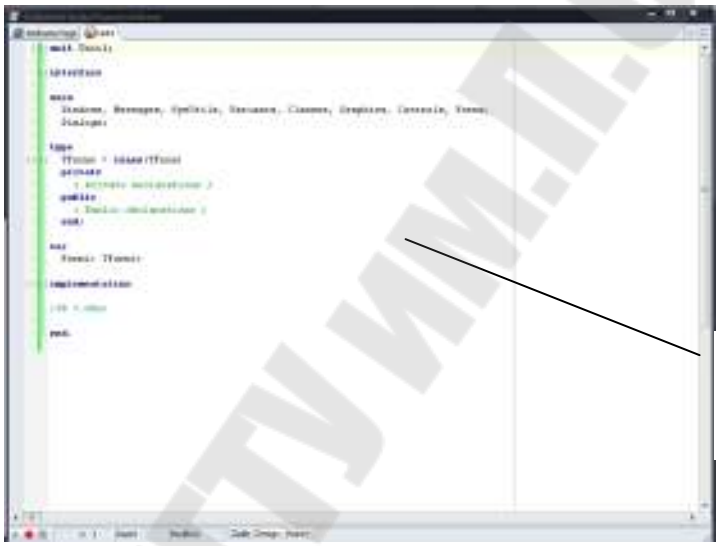


Рис 1.3. Вид окна редактора кода

3. Создание интерфейса приложения.

Разместить на каждой форме визуальные компоненты в соответствии с разработанным интерфейсом проекта.

Для размещения каждого визуального компонента необходимо:

- щелкнуть мышью по визуальному компоненту на палитре ВК (инструментов), а затем щелкнуть мышью в месте его расположения на форме;
- на вкладке *Properties* инспектора объектов найти свойства, которые нужно изменить, и установить для них необходимые значения.

По умолчанию все свойства разделены на категории, но можно вывести свойства в алфавитном порядке без указания категорий. Для этого в контекстном меню вкладки *Properties* выполнить команду *Arrange – by Name.*

Выделенный (щелчком мыши) визуальный компонент можно удалить (клавишей **Delete**), с помощью мыши переместить в другое место на форме, изменить размер компонента или копировать его, используя команды работы с Буфером Обмена меню *Edit*.

4. Сохранение проекта.

Для сохранения модуля и соответствующей ему формы используется команда *File – Save as*…. Проект вместе с его модулем и формой можно сохранить с помощью команды *File – Save All.* Все имена должны быть информативными.

При повторном сохранении проекта с модулем без изменения имен и папки, где находятся файлы, используется команда *File – Save All* или кнопка на ПИ **Save All**, для сохранения модуля и формы команда *File – Save* или кнопка на ПИ **Save**.

5. Создание процедур - обработчиков событий.

Ввести текст всех процедур для обработки событий в соответствии с разработанными алгоритмами решения задачи. Для создания и изменения процедур обработки каждого события необходимо:

- выделить компонент, для которого создается событие;
- найти на вкладке *Events* Инспектора Объектов нужное событие (они, как и свойства) могут быть упорядочены по категориям или именам);
- выполнить двойной щелчок по полю, находящемуся справа от названия события (если процедура только создается, то поле пустое, если процедура изменяется, то поле содержит имя процедуры);
- ввести текст процедуры или изменить ее в окне редактора кода.

Для создания процедуры обработки события по нажатию кнопки (OnClick) можно выполнить двойной щелчок по этой кнопке.

При вводе кода процедуры после набора имени компонента редактор выводит список свойств и методов для этого компонента, а для формы и список всех расположенных на ней компонентов. Можно не набирать свойство или метод на клавиатуре, а выделить его в списке и нажать клавишу **Enter** (или выполнить двойной щелчок по

имени свойства).

6. Выполнение (запуск) проекта.

Выполнить команду Run – Run (**F9**) или использовать кнопку **Run** на ПИ. **Нельзя запустить вторую копию приложения во время работы первой.**

При выполнении приложения Turbo Delphi сначала выполняет компиляцию программы, выявляет ошибки и, если они есть, выводит их в окне сообщений *Messages*, появляющемся обычно в нижней части экрана. При наличии ошибок выполнение программы прекращается. Строка в модуле, в которой найдена ошибка, выделяется красным цветом. Можно выполнить только компиляцию приложения с помощью команды *Project – Compile* (**Ctrl+F9**).

Если ошибок компиляции не найдено, создается исполняемый файл Project1.exe и программа из режима проектирования переходит в режим выполнения. Выводится окно приложения, соответствующее окну главной формы. Вводятся исходные данные, запускаются события, реакция на которые запрограммирована в модуле формы, выводятся результаты. На этом этапе тоже могут быть найдены ошибки, например данные не введены или введены данные несоответствующих типов. В этом случае выводится сообщение об ошибке в окне сообщений, а затем (после нажатия кнопки **ОК** в окне сообщения) окно проекта. В этом случае для завершения этапа выполнения программы и перехода в режим проектирования (как и для выхода из зацикливающейся программы) нужно использовать команду *Run – Program Reset* или кнопку ***Reset*** на ПИ**.**

Для завершения выполнения программы и перехода в режим проектирования необходимо закрыть окно главной формы.

7. Вывод программ и форм на принтер.

В ИСР Turbo Delphi можно распечатать вид формы на этапе проектирования, текст всего модуля или выделенной его части. Для этого необходимо:

- сделать активным модуль или форму;
- выполнить команду *File – Print….* Появится диалоговое окно *Print Selection* для модуля или *Print Form* для формы.
- щелкнуть по кнопке Setup, выбрать принтер; при необходимости установить альбомную ориентацию и щелкнуть по кнопке ОК;

• щелкнуть по кнопке ОК.

Для вывода на принтер окна формы в режиме выполнения необходимо вставить ее в текстовый документ (например, Word), используя Буфер Обмена. Для копирования в БО активной формы используется комбинация клавиш **Alt+PrtScr.**

8. Завершение работы Delphi. Закрыть главное окно Delphi.

Ход работы:

Задание 1.

Разработать приложение, которое позволяет ввести имя и группу пользователя и после щелчка мышью по кнопке Привет вывести приветствие, включающее введенную фамилию и группу.

Интерфейс программы представлен на рисунке 1.4. Поля ввода имени (Edit1) и группы (Edit2) и поле вывода приветствия (Label4) при разработке интерфейса должно быть пустым.

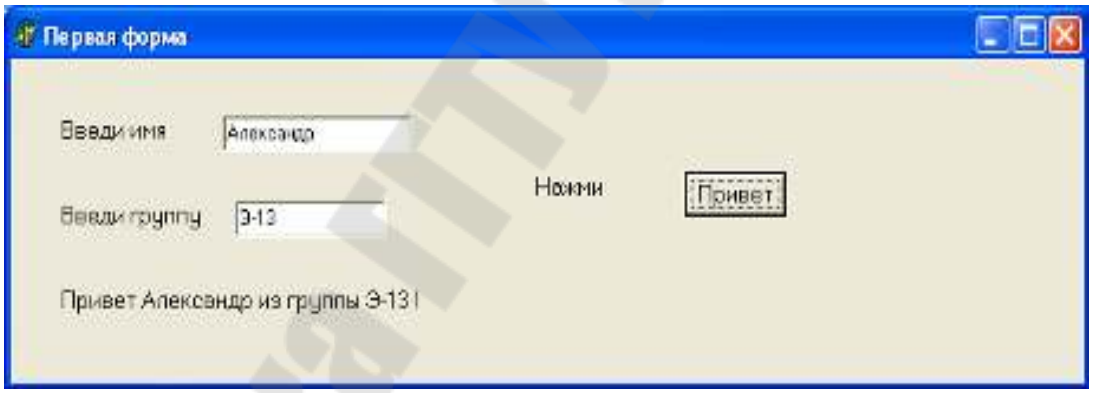


Рис 1.4. Интерфейс программы для задания 1

Используемые компоненты и их свойства, которые устанавливаются на вкладке Properties инспектора объектов (Object Inspector) показаны в таблице 1.1.

Таблица 1.1

Таблица используемых компонентов для задания 1

Элемент интерфейса	*Визуальный компонент*	*Свойство компонента*	*Значение свойства*
Заголовок окна	Form1	Caption	Первая форма
Введи имя	Label1	Caption	Введи имя

Поле ввода имени	Edit1	Text	
Введи группу	Label2	Caption	Введи группу
Поле ввода группы	Edit2	Text	
Нажми	Label3	Caption	Нажми
Кнопка Привет	Button1	Caption	Привет
Поле вывода приветствия	Label4	Caption	

Для всех компонентов можно изменить шрифт с помощью свойства Font.

Процедура обработки события **OnClick** для кнопки Привет (записывается в модуле после двойного щелчка мышью по кнопке Привет или на вкладке **Events** события **OnClick**):

```
procedure TForm1.Button1Click(Sender: TObject);
var
   x,y,z:real;
begin
  x:=StrToFloat(Edit1.Text);
  y:=StrToFloat(Edit2.Text);
  Label4.Caption:='Вы ввели x='+FloatToStr(x)+' y='+FloatToStr(y);
  z:=x+y;
  Edit3.Text:=FloatToStr(z);
end;
```

Задание 2.

Ввести два вещественных числа x и y и вычислить их сумму z. Интерфейс приложения представлен на рисунке 1.5.:

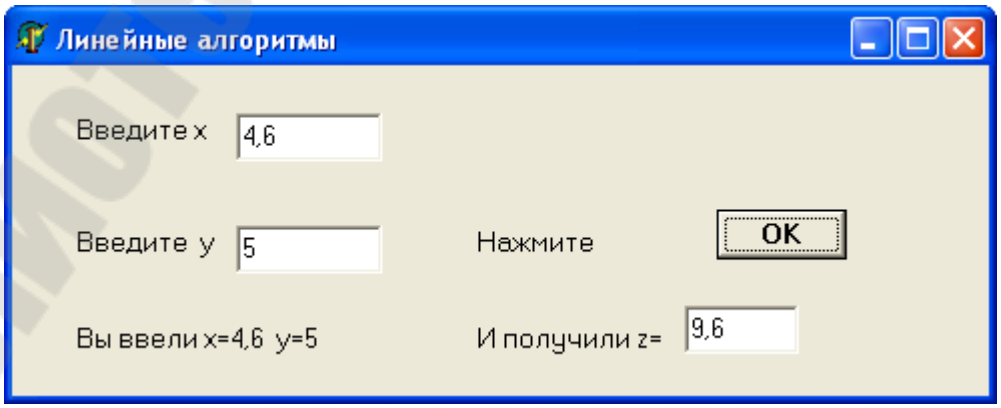


Рис 1.5. Интерфейс программы для задания 2

Используемые компоненты и их свойства, которые устанавливаются на вкладке Properties инспектора объектов (Object Inspector), показаны в таблице 1.2.

Таблица 1.2

Таблица используемых компонентов для задания 2

Элемент интерфейса	*Визуальный компонент*	*Свойство компонента*	*Значение свойства*
Заголовок окна	Form1	Caption	Линейные алгоритмы
Введите x	Label1	Caption	Введите x
Поле ввода x	Edit1	Text	
Введите y	Label2	Caption	Введите y
Поле ввода y	Edit2	Text	
Нажмите	Label3	Caption	Нажмите
Кнопка ОК	Button1	Caption	ОК
Поле вывода x,y	Label4	Caption	
И получили z=	Label5	Caption	И получили z=
Поле вывода z	Edit3	Text	
Поле вывода z	Edit3	ReadOnly	True

Для всех компонентов можно изменить шрифт с помощью свойства Font.

Процедура обработки события OnClick для кнопки ОК (записывается в модуле после двойного щелчка мышью по кнопке ОК или на вкладке Events события OnClick):

```
procedure TForm1.Button1Click(Sender: TObject);
var
  x,y,z:real;
begin
  x:=StrToFloat(Edit1.Text);
  y:=StrToFloat(Edit2.Text);
  Label4.Caption:='Вы ввели x='+FloatToStr(x)+'  y='+FloatToStr(y);
  z:=x+y;
  Edit3.Text:=FloatToStr(z);
end;
```

Задание:

Изучить теоретические сведения и разработать приложения задания 1 и задания 2, описанные в ходе работы. Продемонстрировать преподавателю формы с результатами работы и папки с сохраненными файлами для каждого разработанного приложения.

Контрольные вопросы к лабораторной работе:

1. Какие элементы содержит ИСР Turbo Delphi?
2. Как создать новый проект, открыть существующий?
3. Какую структуру имеет проект Delphi?
4. Назовите основные файлы проекта и их назначение.
5. Как сохранить проект и его отдельные элементы?
6. Как разместить на форме визуальные компоненты и установить их свойства?
7. Как создать процедуры обработки событий?
8. Как выполнить компиляцию и запуск проекта на выполнение?
9. Как вывести на принтер программу и форму?

Лабораторная работа № 2
Алгоритмизация и программирование линейных алгоритмов

Цель работы: Получить навыки разработки графических схем линейных алгоритмов и их реализации в среде Delphi. Научиться составлять тесты для проверки работоспособности программы.

Теоретические сведения:

Линейным называется алгоритм, в котором все указанные действия выполняются один раз в том порядке, в котором они записаны.

Тесты – это наборы исходных данных с известными результатами, с помощью которых выполняется проверка работоспособности (правильности работы) программы.

Описание (объявление) констант:
const
<идентификатор> = <значение>;

Тип константы определяется её значением.

Как значение константы могут использоваться:

- целые числа;
- вещественные числа с использованием десятичной точки (10.7) или в экспоненциальной форме (1.72E02);
- логические константы (true, false);
- символьные константы (‘A’ ≠ ’a’);
- строковые константы (‘ABC’).

Все они могут использоваться в тексте программы непосредственно без описания.

Описание переменных:
var
<идентификатор> : <тип>;
var
a, b: real;
c : byte;

Типы рассматриваются компилятором как образцы для создания констант, переменных, функций. Основные типы данных представлены в таблице 2.1.

Тип данного определяет:

1. диапазон возможных значений;
2. объём выделяемой памяти и форму представления данных в ней;
3. действия, которые можно совершать над данными.

Таблица 2.1

Таблица основных типов данных

Название типа	Выделяемая память (байт)	Диапазон значений	Примечания
Byte	1	0….255	Целые типы

Word	2	0….65535	представляются в памяти точно
Integer	4	2147483648…. ….2147483647	
Real	8	$\pm 10^{-324} \ldots 10^{308}$ 15 – 16 знач. цифр	Вещественные типы представляются в памяти с некоторой степенью точности
Extended	10	$\pm 10^{-4951} \ldots 10^{4932}$ 19 – 20 знач. цифр	
Boolean	1	true, false	Логический тип

Арифметические выражения служат для вычисления целого или вещественного значения и состоит из констант, переменных, функций, знаков арифметических операций и круглых скобок.

Арифметические операции в порядке убывания приоритета:

* , / , div, mod, + , -

Порядок выполнения действий при вычислении значения арифметического выражения определяется приоритетом операций.

Операции div и mod применяется только к целочисленным операндам, результат операции – целое число:

Div – целочисленное деление;

Mod – остаток от целочисленного деления.

В арифметических выражениях могут использоваться математические функции, представленных в таблице 2.2.

Таблица 2.2

Таблица математических функций

Математическая запись	Функция Delphi	Тип аргумента и результата
$\|x\|$	abs(x)	Целый и веществ. тип
x^2	sqr(x)	
$\sqrt{x}$	sqrt(x)	Вещественный тип
ln x	ln(x)	
e^x	exp (x)	
Sin x	sin (x)	
Cos x	cos(x)	
arctg x	arctan (x)	
π	pi	

Все эти функции находятся в модуле System, который подключается к программе автоматически, аргументы функции всегда заключаются в круглые скобки.

Аргументы функций sin(x), cos(x) и результат функции arctan(x) всегда указываются в радианах.

Для вычисления других математических функций можно использовать математические формулы: tg(x)=sin(x)/cos(x); lg(x)=ln(x)/ln(10) и т.д.

В Delphi есть возможность использовать множество математических функций из модуля Math, подключив его к своей программе. Для этого его нужно добавить в список подключаемых модулей предложения uses. В этом случае можно использовать функции: arccos(x), arcsin(x), tan(x), log10(x).

Для возведение числа в степень используются функции модуля Math:

Power(x, y) – возведение вещественного числа x в вещественную степень y. Нельзя возвести отрицательное число в вещественную степень.

Для $\sqrt[3]{x}$ используется функция Power(x, 1/3)

IntPower (x, y) - возведение вещественного числа x в целую степень y.

Оператор присваивания

Синтаксис оператора:

<идентификатор>: = <выражение>

Идентификатор и выражение должны иметь одинаковые типы.

Исключение: вещевой переменной можно присвоить целое значение, но не наоборот.

В операторе присваивания могут использоваться свойства компонентов, как переменные соответствующего типов.

Визуальные компоненты, используемые при создании простых приложений.

Для установки или изменения свойств любого ВК на этапе проектирования интерфейса, его нужно выделить щелчком мыши, если он виден на форме, или найти в списке объектов Инспектора Объектов. Свойства устанавливаются на вкладке Properties ИО.

1. Tlabel – метка вкладка Standard

Используется для размещения в формах различных текстовых надписей. Основное свойство компонента Caption содержит текст надписи.

Если процедура находится в модуле unit1, то имя формы Form1можно не указывать.

2. TEdit - редактируемое однострочное поле (вкладка Standard)

Используется для ввода и вывода редактируемого текста. Текст может занимать только одну строку. Основное свойство Text содержит текст, который находится в окне компонента.

Свойство ReadOnly определяет можно ли редактировать текст на этапе выполнения программы. При значении True - текст редактировать нельзя (для результатов работы), False - текст редактировать можно (для исходных данных).

3. TButton - кнопка (вкладка Standard)

Свойство Caption определяет надпись на кнопке. Щелчок по кнопке во время выполнения программы вызывает процедуру обработки события OnClick.

Заголовок такой процедуры имеет вид:

```
Procedure TForm1.Button1Click (Sender: TObject);
```

Для создания процедуры обработки события OnClick для кнопки нужно на этапе конструирования формы выполнить двойной щелчок по этой кнопке или использовать событие OnClick на вкладке Events Инспектора Объектов.

Ввод данных

Для ввода значений переменных целого и вещественного типов используется компонент Edit. Данные, вводимые с клавиатуры, определяются его свойством Text. Свойство Text имеет строковый тип, поэтому при вводе вещественных и целых чисел необходимо использовать функции преобразование типов StrToFloat (S), StrToInt (S).

Аргумент этих функций имеет строковый тип, результат соответственно вещественный или целый.

При вводе чисел текст, вводимый с клавиатуры, не должен содержать никаких символов кроме цифр, ведущего знака «+» или «-» и десятичной **запятой** для вещественных чисел.

Вывод данных

Данные вещественного и целого типов можно вывести с помощью визуальных компонентов Label, Edit, Memo. Их свойства Caption и Text имеют строковый тип, поэтому необходимо использовать функции IntToStr(x) и FloatToStr (x) для перевода целого или вещественного значения в строку.

```
Edit1.Text:=IntToStr (m);
Label1.Caption:=FloatToStr (b);
```

Если выводимые данные не предназначены для редактирования (результаты), то при их выводе в редактируемое поле Edit нужно запретить возможность редактирования, установив для свойства ReadOnly значение True. Поэтому для вывода результатов лучше использовать визуальный компонент Label, а не Edit.

Для объединения нескольких строк в одну при их выводе используется операция сцепления строк (+).

```
Label1. Caption: = ‘x=’+IntToStr(x);
Memo1.Lines.Add (‘y=’+IntToStr(y));
```

Программным путем можно создать метку, содержащую несколько строк текста.

```
label2.Caption:=’ГГТУ’ + #13+ 'Группа ЭПП-11';
```

Для вывода различных предупреждающих или информационных сообщений (например, сообщений об ошибках) можно использовать процедуру:

ShowMessage (<текст сообщения>);

Задание:

Задание 1

Разработать интерфейс проекта, составить графическую схему алгоритма и программу для вычисления значений переменных в соответствии с условием в таблице 2.3 по своему варианту (выбирается по номеру компьютера). Для ввода исходных данных, вывода исходных данных и результата использовать только одну форму. Исходные данные для отладки программы выбрать самостоятельно.

При разработке интерфейса выполнить следующие задания:

- на форме расположить одну кнопку для выполнения расчета;
- исходные данные вывести в ВК TLabel;
- результаты вывести в ВК TEdit, запретив его редактирование на этапе выполнения программы.

Таблица 2.3

Условие задачи по вариантам для задания 1

Вариант	Вычислить	Расчетные формулы
1	Площадь круга и длину окружности радиуса *r*.	$S = \pi r^2$ $l = 2\pi r$
2	Площадь и угол при основании равнобедренного треугольника с основанием *a* и высотой *h*.	$S = \frac{a\,h}{2}$ $\alpha = arctg(2h/a)$
3	Площадь и периметр прямоугольника со сторонами *a, b*.	$S = ab$ $P = 2(a+b)$
4	Скорость в конце пути и путь, пройденный за время *t* с	$v = at$ $S = \frac{at^2}{2}$

	ускорением a при $v_0=0$.	
5	Сторону и периметр квадрата со стороной а.	$S = a^2$ $P = 4a$
6	Объем и площадь боковой поверхности параллелепипеда со сторонами *a, b, c*.	$V = abc$ $S = 2(a+b)c$
7	Площадь кольца с внешним радиусом *R* и внутренним *r*.	$S=\pi(R^2 - r^2)$
8	Площадь боковой поверхности и объем цилиндра с радиусом основания **r** и высотой *h*.	$S = 2\pi r\, h$ $V=\pi r^2 h$
9	Площадь и периметр прямоугольного треугольника с катетами *a, b* и гипотенузой *c*.	$S = \frac{a\,b}{2}$ $P=a+b+c$
10	Объем и площадь поверхности куба со стороной *a*.	$V = a^3$ $S = 6a^2$
11	Радиус круга, вписанного в треугольник со сторонами *a, b, c*.	$r = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$ $p = \frac{a+b+c}{2}$
12	Площадь основания и объем цилиндра с радиусом основания *r* и высотой *h*.	$S = \pi r^2$ $V=Sh$
13	Объем и площадь основания параллелепипеда со сторонами *a, b, c*.	$V = abc$ $S =ab$
14	Площадь основания и объем конуса с радиусом основания *r* и высотой *h*.	$S = \pi r^2$ $V = \frac{Sh}{3}$
15	Гипотенузу и площадь прямоугольного треугольника с катетами *a, b*.	$c = \sqrt{a^2 + b^2}$ $S = \frac{ab}{2}$
16	Высоту и площадь равнобедренной трапеции с основаниями *a, b* (*b*>*a*) и углом при большем основании α.	$h = \frac{b-a}{2} tg\alpha$ $S = \frac{b+a}{2} h$

Задание 2

Разработать интерфейс проекта, составить графическую схему алгоритма и программу для вычисления функции b=f(x,y,z), заданной в таблице 2.4, в соответствии со своим вариантом. Для ввода исходных данных, вывода исходных данных и результата использовать только одну форму. Исходные данные для отладки программы выбрать самостоятельно.

Обратите внимание на то, что алгоритм создается линейный, без обработки исключительных ситуаций. Поэтому вводите только допустимые исходные данные, т.к. нельзя выполнить проверку корректности ввода.

При разработке интерфейса и программы выполнить следующие задания:

1. На форме расположить три кнопки:
   - для расчета значения функции **без подключения модуля Math**; результат вывести в один ВК TEdit, запретив его редактирование на этапе выполнения программы;
   - для расчета значения функции **с использованием функций модуля Math**; результат вывести в другой ВК TEdit, запретив его редактирование на этапе выполнения программы;
   - для закрытия формы и завершения работы программы (ВК TBitBtn).
2. Установить для всех кнопок всплывающие подсказки.
3. Изменить стандартный шрифт элементов интерфейса на форме
4. Исходные данные вывести в ВК TLabel.
5. Текст условия задачи разместить ВК TMemo исключив возможность его изменения на этапе выполнения.
6. Вид функции вывести как изображение (скопировать формулу в графический редактор Paint и сохранить).

Таблица 2.4

Варианты вида функций для задания 2

Вариант	Вид функции	Вариант	Вид функции
1	$b=\frac{e^{2z}-\sqrt[3]{y\lvert x\rvert}}{arctg2z+1}$	9	$b=x+\frac{\sqrt[3]{zy}}{y+\cos x}+\lg(2x^2+1)$

Вариант	Вид функции	Вариант	Вид функции
2	$b = \frac{\ln^2 \lvert z \rvert}{\sqrt[3]{\lvert x \rvert + \lvert y \rvert}} + e^{\operatorname{tg} 2x}$	10	$b = \lg\left(\sqrt{e^{x-y} + x^{\lvert y \rvert} + z}\right)$
3	$b = \sqrt[3]{\frac{y^3}{x + y^3 \cos^2 z}}$	11	$b = \sqrt[3]{\left\lvert 1 + \frac{x^2 + 1}{3 + e^y} \right\rvert} + \arcsin 2z$
4	$b = \sqrt{x + \sqrt[4]{\lvert y \rvert}} + \arccos^2 z$	12	$b = \sqrt[3]{\lvert \cos^3 x + \arccos y \rvert} + 2\sin^2 z$
5	$b = \frac{\sqrt[3]{e^{\sin x}} \cdot \cos y}{z^2 + 1}$	13	$b = \frac{\ln\left(y^3\right)\left(z^x - x/2\right)}{2\cos^2 x}$
6	$b = \frac{z(\operatorname{tg} y - e^{-(x+3)})}{\lg(\lvert \sqrt[3]{x + 4} \rvert + 1)}$	14	$b = \frac{\sqrt{10\left(\sqrt[3]{z} + x^{(y+2)}\right)}}{\operatorname{arctg}^2 \sqrt{z} + 1}$
7	$b = \frac{e^{\lvert x-y \rvert}\left(\operatorname{tg}^2 z + 1\right)^x}{\arccos x + 1}$	15	$b = \sin^2 z^3 + \sqrt[5]{\lvert x + \arccos y \rvert}$
8	$b = \frac{\sqrt{y + \sqrt[3]{x}} - 1 + 2z}{\lvert z \rvert e^{-(y + x/2)}}$	16	$b = \frac{\lg\left(y + \sqrt[3]{\lvert z + x^2 \rvert}\right)}{\cos^3 y + x^2}$

Контрольные вопросы к лабораторной работе:

1. Какой алгоритм называется линейным?
2. Как создаются и для чего используются тесты?
3. Как в программе описываются константы и переменные?
4. Какие вы знаете типы данных? Что определяет тип данных?
5. Для чего используются и как записываются арифметические выражения?
6. Какие операции используются в арифметических выражениях?
7. Какие функции модулей System и Math используются в арифметических выражениях?
8. Для чего используется и как записывается оператор присваивания?
9. Назовите назначение и основные свойства ВК классов TButton, TEdit, Tlabel.

10. Как ввести значения переменных целого и вещественного типов?
11. Запишите оператор присваивания для вычисления значения функции из таблицы 2 (вариант выбирается по указанию преподавателя).

Лабораторная работа № 3
Алгоритмизация и программирование разветвляющихся алгоритмов

Цель работы: Получить навыки разработки и реализации разветвляющихся алгоритмов в среде Delphi и использования соответствующих визуальных компонентов. Научиться составлять тесты для проверки разветвляющейся программы.

Теоретические сведения:

Разветвляющимся называется алгоритм, в котором некоторые действия (в программе операторы) могут выполняться один раз или не выполняется в зависимости от заданного условия.

Для реализации разветвляющегося алгоритма в Delphi есть два оператора:

If – условный оператор, **Case –** оператор выбора.

Эти операторы влияют на порядок выполнения других операторов программы.

Для записи условий в операторе **if** используются логические выражения.

Логические выражения состоят из арифметических выражений, операций отношения и логических операций.

Логические выражения имеют тип Boolean и могут принимать одно из двух значений: True или False.

Операции отношения (= ; <> ; < ; <= ; > ; >=) выполняют сравнение двух операндов и определяют истинно выражение (его значение true) или ложно (его значение false).

Логические операции (not; and; or; xor**)** используются для образования сложных логических выражений. Операнды логических операций должны иметь логический (булевский) тип.

Not (логическое отрицание).

Синтаксис:

not (<логическое выражение>)

Результат операции истина, если значение операнда ложь и наоборот.

And (логическое И).

Синтаксис:

(<логическое выражение 1>) and (<логическое выражение 2>)

Результат операции истина, если оба операнда имеют значение истина, и ложь в противном случае.

Or (логическое ИЛИ).

Синтаксис:

(<логическое выражение 1>) or (<логическое выражение 2>)

Результат операции истина, если хотя бы один из операндов имеет значение истина; и ложь, если оба операнда имеют значение ложь.

Приоритет логических операций выше, чем операций отношения, поэтому необходимо использовать круглые скобки для указания порядка действий при вычислении значения логического выражения.

Условный оператор If

Имеет две формы: полную и сокращённую.

Полная форма:

```
If  <условие>
        Then
            < оператор1>
         Else
            < оператор2>;
```

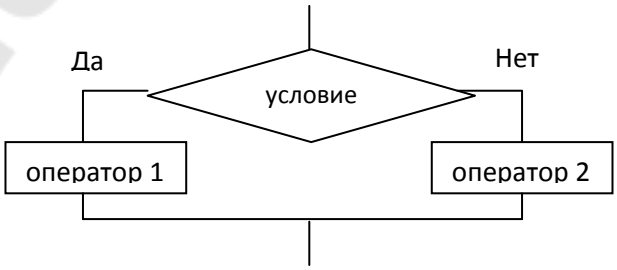


Рис 3.1. Графическая схема для полной формы условного оператора if

Сокращенная форма:

If <условие>

Then

<Оператор1>;

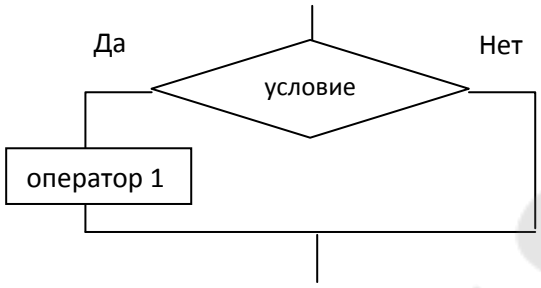


Рис 3.2. Графическая схема для сокращённой формы условного оператора if

Выполнение:

Вычисляется значение логического выражения в условии.

1. Если значение логического выражения true, то выполняется оператор1, если false - оператор 2.
2. Если значение логического выражения true, то выполняется оператор 1, если false – то оператор никаких действий не производит, и программа переходит к выполнению следующего за If оператора.

Операторы If могут быть вложены в другие операторы If.

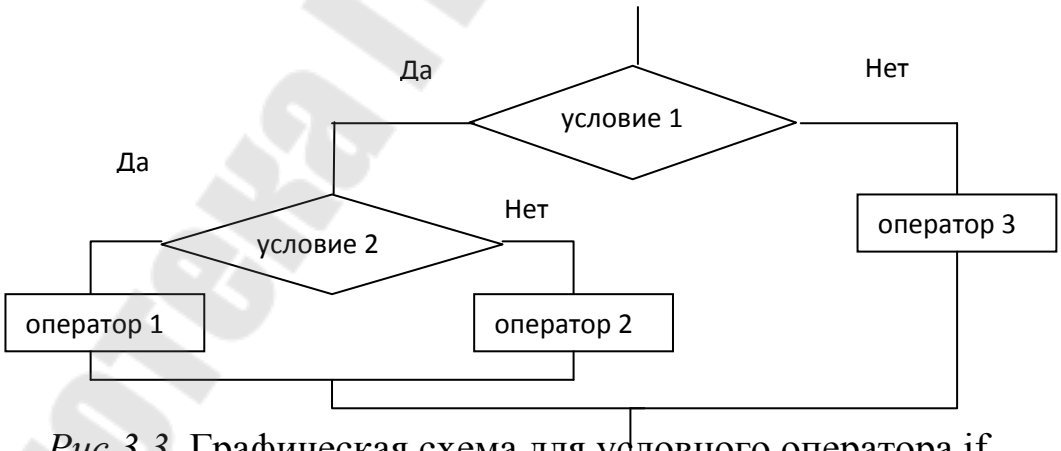


Рис 3.3. Графическая схема для условного оператора if, вложенного в оператор if

If <условие 1>

then

if < условие 2>

then

```
                <оператор 1>
            else
                <оператор 2>
    else
        <оператор 3>;
```

Каждое Else соответствует тому Then, который ему непосредственно предшествует.

Составной оператор

Это последовательность из произвольного числа операторов, заключенных в операторные скобки begin . . . .end.

Синтаксис:

```
begin
    <оператор 1;>
    <оператор 2;>
    ……..
    < оператор N>
end;
```

Составной оператор может использоваться в любом месте программы, где допускается запись оператора. Он применяется, если по какой-либо ветви условия (в программе после слов Then или Else) необходимо записать не один оператор, а несколько.

Синтаксис:

```
If  <условие>
   then<оператор 1>
   else
      begin
      |<оператор 2>
       < оператор 3>
      end;
```

Задание:

На главной форме выбрать нужное задание и ввести пароль. Для выбора номера задания (1, 2, 3) использовать ВК TRadioGroup, для ввода пароля – ВК TEdit (свойство PasswordChar). После

проверки совпадения вводимого пароля с заданным в программе (оператор If) и проверки выбора номера задания (оператор Case) открыть форму, соответствующую заданию.

Задание 1

Разработать интерфейс проекта, составить графическую схему алгоритма и программу вычисления функции ***y=f(x)*** в соответствии с видом функции, приведенном в таблице 3.1.

В программе предусмотреть вывод:

- значения аргумента функции *x*;
- вычисленного значения функции *y*;
- номера формулы, по которой производилось вычисление функции (1, 2 или 3).

Исходные данные для отладки программы выбрать самостоятельно. Выполнить тестирование программы для каждого интервала изменения функции и для каждой точки ветвления функции.

При разработке интерфейса и программы выполнить следующие задания:

7. На форме **Задание 1** создать кнопки «О программе», «Об авторе», «Расчет», «Выход»**,** ВК TEdit для ввода аргумента x и ВК TMemo (для вывода исходных данных и результатов)
8. После щелчка по кнопке «О программе» открыть новую форму в модальном режиме с текстом задания и видом функции. Текст задания разместить в ВК TMemo, исключив возможность его изменения на этапе выполнения. Вид функции разместить в ВК TImage.
9. После щелчка по кнопке «Об авторе» открыть новую форму с необходимыми сведениями об авторе и, возможно, с его фото.
10. После щелчка по кнопке Расчет ввести исходные данные, вывести исходные данные и результаты в ВК TMemo (для всех необходимых тестов вместе).

Проверить корректность ввода аргумента x с помощью защищенного блока.

11. После щелчка по кнопке «Выход» вернуться к главной форме.

Таблица 3.1

Варианты вида функций для задания 1

Вариант	Вид функции
1	$y=\begin{cases}1/x, & \text{если } x\ge -5,\ x\ne 0 \quad (1)\\ x^2, & \text{если } x\le -10 \quad (2)\\ \sqrt{\lvert x+1\rvert} & \text{в ост. случаях} \quad (3)\end{cases}$
2	$y=\begin{cases}x^2, & \text{если } x\le 0,\ x\ne -10 \quad (1)\\ \sqrt{x+1}, & \text{если } x>1 \quad (2)\\ 1/x & \text{в ост. случаях} \quad (3)\end{cases}$
3	$y=\begin{cases}x+e^{2x}, & \text{если } x\le 0,\quad x\ne -1 \quad (1)\\ \cos^2 x, & \text{если } 0<x\le 3{,}14 \quad (2)\\ x & \text{в ост. случаях} \quad (3)\end{cases}$
4	$y=\begin{cases}x^3, & \text{если } x>5,\quad x\ne 20 \quad (1)\\ x^2, & \text{если } -5\le x\le 5 \quad (2)\\ \lg\lvert x\rvert & \text{в ост. случаях} \quad (3)\end{cases}$
5	$y=\begin{cases}\sqrt{x}, & \text{если } x\ge 100,\ x\ne 105 \quad (1)\\ \sqrt[3]{x}, & \text{если } x=20 \text{ или } x=40 \quad (2)\\ x^2+1 & \text{в ост. случаях} \quad (3)\end{cases}$
6	$y=\begin{cases}\sqrt{x-1}, & \text{если } x\ge 10,\ x\ne 20 \quad (1)\\ 1/x+e^{2x}, & \text{если } x<0 \quad (2)\\ \ln(x+1) & \text{в ост. случаях} \quad (3)\end{cases}$
7	$y=\begin{cases}8x+1, & \text{если } x\ge 5,\ x\ne 9 \quad (1)\\ x^2+\lvert x\rvert, & \text{если } x\le 1 \quad (2)\\ x^3+\sqrt{x} & \text{в ост. случаях} \quad (3)\end{cases}$
8	$y=\begin{cases}1-3x, & \text{если } x>0,\ x\ne 8 \quad (1)\\ x^2-\sin x, & \text{если } x\le -1 \quad (2)\\ \cos x & \text{в ост. случаях} \quad (3)\end{cases}$
9	$y=\begin{cases}x^3+1, & \text{если } x\ge 8,\ x\ne 10 \quad (1)\\ 2x^2+\sqrt[3]{\lvert x\rvert}, & \text{если } x\le 1 \quad (2)\\ \sqrt{x} & \text{в ост. случаях} \quad (3)\end{cases}$

Вариант	Вид функции
10	$y=\begin{cases}2x^2, & \text{если } x>0, x\neq 3 \quad (1)\\ \sqrt{x^2+1} & \text{если } x\le -2 \quad (2)\\ \lvert x+5\rvert & \text{в ост. случаях} \quad (3)\end{cases}$
11	$y=\begin{cases}\sqrt{\lvert 2x-x^2-1\rvert}, & \text{если } x\le -1, x\neq -4 \quad (1)\\ \ln(x+3), & \text{если } x>0 \quad (2)\\ x/2 & \text{в ост. случаях} \quad (3)\end{cases}$
12	$y=\begin{cases}\sqrt{x}, & \text{если } x\ge 4, x=1 \quad (1)\\ \ln\lvert x+1\rvert, & \text{если } x\le -2 \quad (2)\\ e^{-x} & \text{в ост. случаях} \quad (3)\end{cases}$
13	$y=\begin{cases}x/3, & \text{если } -3\le x\le 3 \quad (1)\\ \lg(x^2+1), & \text{если } x<-3 \text{ или } x=4 \quad (2)\\ \sqrt{x^3-2} & \text{в ост. случаях} \quad (3)\end{cases}$
14	$y=\begin{cases}\lvert x^3+4\rvert, & \text{если } x\le -1 \text{ или } x=0 \quad (1)\\ \sqrt{x/2}, & \text{если } x\ge 8 \quad (2)\\ x^3 & \text{в ост. случаях} \quad (3)\end{cases}$
15	$y=\begin{cases}\sqrt{x+1}, & \text{если } x\ge 8, x\neq 10 \quad (1)\\ 0{,}6x, & \text{если } 0<x<8 \quad (2)\\ \lg\lvert x\rvert+3 & \text{в ост. случаях} \quad (3)\end{cases}$

Задание 2

Выполнить задание в соответствии со своим вариантом в таблице 3.2. Исходные данные для отладки программы выбрать самостоятельно так, чтобы проверить все возможные варианты выполнения программы. Вывести результаты для всех тестов вместе в ВК Memo. Проверить корректность ввода всех исходных данных с помощью защищенного блока.

Создать кнопки «**О программе», «Об авторе», «Расчет», «Выход»**, действия при щелчке по которым аналогичны реакции на соответствующие события в задании 1.

При выполнении двух заданий можно использовать формы со сведениями об авторе и о программе из задания 1.

Таблица 3.2

Условие задачи по вариантам для задания 2

Вариант	Задание
1	Даны числа a, b, c, d. Если ни одно из чисел a, b, c не равно d, то найти max(d-a, d-b, d-c).
2	Даны числа a, b, c, d. Найти max {min(a, b), min(c, d)}.
3	Даны три целых положительных числа. Если все они четные, каждое число уменьшить в два раза, если хотя бы одно из них четное, увеличить каждое число на 20 , если четных чисел нет, оставить числа без изменения.
4	Даны три целых числа. Найти минимальное из них и прибавить минимальное значение к числам, отличным от минимального.
5	Даны три целых числа. Определить, могут ли они быть сторонами треугольника. Если могут, то определить какой это треугольник: равнобедренный, равносторонний или разносторонний.
6	Даны два угла (в градусах). Определить существует ли треугольник с такими углами. Если да, то будет ли он прямоугольным.
7	Даны три числа a, b, c. Если только одно из них положительное, то найти площадь квадрата со стороной, равной значению положительного числа. В противном случае найти max (a, b, c).
8	Даны числа a, b. Если b=0, то найти min (a, b), если b<0, то найти max (a, b), в противном случае каждое число уменьшить на 20%.
9	Даны три целых положительных числа. Если все они нечетные, каждое число увеличить в два раза, если хотя бы одно из них нечетное, оставить числа без изменения, если нечетных чисел нет, увеличить каждое число на 10.
10	Даны числа a, b, c, d. Найти min {max (a, b), max (c, d)}.
11	Даны два целых числа a, b. Найти вещественные корни уравнения $ax^2+bx+c=0$ или вывести сообщение об их отсутствии.
12	Даны числа a, b, c, d. Если a>b>c>d, то каждое число заменить наибольшим из всех чисел, если a<b<c<d, то каждое число заменить его квадратом, в противном случае оставить числа без изменения.
13	Даны числа a, b, c. Если все они равны нулю, вывести об этом сообщение, если среди чисел нет нулей, найти и

Вариант	Задание
	вывести их произведение, в противном случае нули заменить суммой двух других чисел.
14	Даны числа a, b, c. Вычислить max (a+b+c, abc)*min (a, b, c)
15	Даны два целых числа неравных друг другу. Большее из них увеличить на 50%, меньшее заменить суммой заданных чисел.

Задание 3

Разработать интерфейс проекта, составить графическую схему алгоритма и программу в соответствии с условием, приведенном в таблице 3.3.

Таблица 3.3

Условие задачи по вариантам для задания 3

Вариант	Задание
1	Введите четырехзначное число. Верно ли, что оно содержит 3 одинаковые цифры.
2	Введите четырехзначное число. Является ли оно полиндромом. 2222, 6116, 0440. и т.д.
3	Введите четырехзначное число. Проверьте, больше ли 10 сумма его цифр.
4	Введите четырехзначное число. Проверьте, что больше: сумма первых 2 или последних 2 цифр.
5	Введите четырехзначное число. Верно ли, что все цифры числа различны.
6	Введите четырехзначное число. Найдите последнюю цифру числа. Проверьте, не равна ли она 0.
7	Введите четырехзначное число. Проверьте, есть ли 2 одинаковые цифры в числе.
8	Введите четырехзначное число. Проверьте, равны ли первая и последняя цифра.
9	Введите четырехзначное число. Проверьте, равен ли остаток от деления первой цифры на вторую, остатку от деления третьей цифры на четвертую.
10	Введите четырехзначное число. Проверьте, равна ли целая часть от деления первой цифры на вторую, целой части от деления третьей цифры на четвертую.
11	Введите четырехзначное число. Проверьте, являются ли

	все цифры четными.
12	Введите четырехзначное число. Проверьте, являются ли четными сумма первых двух и последних двух цифр.
13	Введите четырехзначное число. Что больше сумма первых двух цифр или произведение вторых 2 цифр.
14	Введите четырехзначное число. Является ли последняя цифра числа нечетной.
15	Введите четырехзначное число. Является ли остаток от деления первой цифры на последнюю четным.

Контрольные вопросы к лабораторной работе:

1. Какой алгоритм называется разветвляющимся?
2. Как создаются тесты для разветвляющихся алгоритмов?
3. Для чего используется и как записывается логическое выражение?
4. Какой тип и значение имеют логические выражения?
5. Назовите операции отношения. Для чего они используются?
6. Назовите логические операции. От чего зависит результат выполнения логических операций?
7. Как выполняется и как записывается условный оператор if в полной и краткой форме?
8. Как записывается и для чего используется составной оператор?
9. Запишите фрагмент программы, соответствующий графической схеме указанного преподавателем разветвляющегося алгоритма.

Лабораторная работа № 4
Алгоритмизация и программирование циклических алгоритмов. Табулирование функции одной переменной

Цель работы: Получить навыки разработки схем циклических алгоритмов и их реализации в среде Delphi.

Теоретические сведения:

1. Разработка интерфейса приложений, использующих несколько форм

Для создания новой формы и подключения ее к проекту на этапе проектирования используется команда *File – New Form*.

Для работы с несколькими формами используется следующие методы:

Show – показывает форму в немодальном режиме. Если до обращения к методу формы не было на экране, то выводит форму на экран. Если форма была на экране – делает ее активной и передает ей фокус (выводит поверх всех окон). Form2.Show.

ShowModal – показывает форму в модальном режиме. Отличие от Show состоит в том, что никакие действия с другими окнами не возможны, пока модальное окно не будет закрыто. Модальные окна обычно требуют от пользователя принятия каких-либо решений или выводят информационные сообщения для ознакомления. Никакие операторы процедуры не выполняются, пока не закончится выполнения метода ShowModal, и модальное окно не будет закрыто. Form2.ShowModal.

Close – закрывает окно. Для главного окна (формы) завершает работу приложения.

Для вывода на экран закрытого окна (кроме главной формы) используются методы Show и ShowModal.

SetFocus – передает форме или указанному ВК фокус ввода, до использования метода форма должна быть выведена на экран. После использования методов Show и ShowModal фокус ввода передается форме автоматически.

```
Form2.Edit1.SetFocus.
```

Работа с ВК разных форм

При работе с несколькими формами можно в процедуре модуля одной формы открыть вторую форму и обращаться к ВК этой формы.

Для этого необходимо:

1. Подключить к модулю первой формы модуль второй. Для этого в модуле первой формы необходимо указать:

Implementation
uses <имя модуля 2>;

По умолчанию Unit2 или то имя, которое было дано второму модулю при сохранении. Подключение можно сделать автоматически, используя при открытом первом модуле команду *File - Uses Unit,* и выбрать из появившегося списка нужный модуль.

Если это не было сделано, то при компиляции программы Delphi выдаст предупреждающее сообщение и предложит вставить ссылку на модуль.

2. Использовать полные имена для свойств и методов ВК с указанием имени формы, в которой они находятся.

2. Обработка исключительных ситуаций (исключений)

Во время работы программы могут возникнуть такие ситуации, когда программа не может выполняться в соответствии с алгоритмом. Например, данные не введены или введены символы, которые нельзя представить как целые или вещественные числа, в случае деления на 0 и т.д.

Такие ситуации называются исключительными (исключениями), при их возникновении программа завершается аварийно и выдается системное сообщение об ошибке.

В Delphi имеется возможность избежать аварийного завершения программы, если использовать для обработки исключений защищенные блоки.

Для того, что бы эти блоки выполнялись, в среде Delphi7 необходимо выполнить команду *Tools – Debugger Options* и на вкладке Language Exceptions

Снять флажок  в переключателе Stop on Delphi Exceptions.

В Turbo Delphi выполнить команду *Tools – Options.* Найти в списке *Options:*

Debugger Options – Borland Debugger – Language Exceptions.

Снять флажок  в переключателе Notify on language exceptions.

Если флажок снят, но нет обработки исключений с помощью защищенного блока, то программа не завершится аварийно, а только выводится сообщение об ошибке в модальном окне.

Защищенный блок

Для обработки исключений используется защищенный блок вида:

try
  < блок операторов1 >
except
  <блок операторов2 >
end;

Блок операторов 1 – это группа операторов, при выполнении которых могут возникнуть исключения.

Блок операторов 2 – это группа операторов, которые необходимо выполнить, если исключение возникло.

3. Алгоритмизация и программирование циклических алгоритмов

Циклическим называется алгоритм, в котором некоторая последовательность действий может выполняться несколько раз в зависимости от заданного условия. Повторяющаяся последовательность действий (в программе операторов) называется телом цикла.

Условие, от выполнения которого зависит число повторений цикла, включает в себя, по крайней мере, одну переменную, которая называется параметром или переменной цикла. Эта переменная должна обязательно изменяться в теле цикла.

Оператор цикла While

В программе:

```
While <условие>  do        заголовок цикла
   begin
      <тело цикла>;
   end:
```

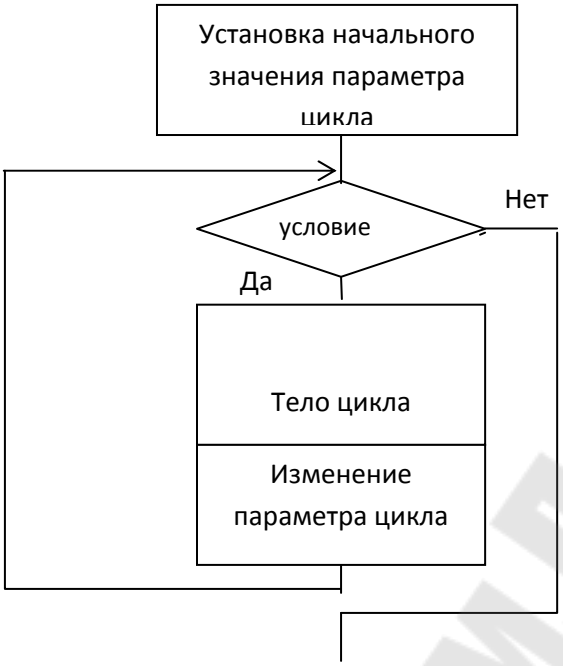


Рис 4.1. Графическое представление оператора цикла While

В условии записывается логическое выражение. Тело цикла – простой или составной оператор. Т.к. в теле цикла обычно находится более одного оператора, то после заголовка цикла записывается begin, а в конце тела цикла – end (составной оператор).

Выполнение оператора:

Вычисляется значение логического выражения в условии. Если это значение true, то выполняются операторы в теле цикла, а затем опять проверяется значение логического выражения. До тех пор, пока его значение будет true, повторяются операторы тела цикла. Как только значение логического выражения будет false, происходит выход из цикла и переход к следующему за циклом оператору. Если перед первым выполнением цикла значение логического выражения false, то тело цикла не выполняется ни разу.

При использовании оператора While необходимо правильно устанавливать начальное значение и изменение параметра цикла, иначе цикл может выполняться бесконечно. Для выхода из зациклившейся программы используется команда *Run – Program Reset*.

4. Табулирование функции

Табулирование функции – это вычисление значений функции для ряда значений аргумента. Аргумент может быть задан в виде

набора произвольных значений (массива) или в виде набора чисел от некоторого начального значения до конечного значения с фиксированным шагом.

Результаты табулирования функции обычно выводятся в таблицу.

5. ВК StringGrid (вкладка Additional) – таблица строк (текстовая таблица)

Основные свойства ВК:

Размер таблицы:

- ColCount (количество столбцов);
- RowCount (количество строк).

Фиксированная часть таблицы

- FixedCols – число столбцов в фиксированной части;
- FixedRows – число строк в фиксированной части;
- FixedColor – цвет фона фиксированной части.
- Col, Row –содержат № столбца и строки ячейки, имеющей фокус ввода;
- DefaultColWidth (по умолчанию 54 пикселя) – ширина ячейки таблицы;
- Option – содержит параметры таблицы.

Значение параметра goEditing=True позволяет изменять данные в ячейках на этапе выполнения программы (для ввода данных из таблицы), goTabs=True позволяет перемещаться по ячейкам таблицы с помощью клавиши Tab.

Основное свойство таблицы Cells [№ столбца, № строки] определяет содержимое ячейки с заданными номерами столбца и строки, нумерация начинается с 0, включает фиксированную и рабочую часть.

6. ВК Chart (вкладка Additional) – диаграмма.

ВК предназначен для графического представления числовых данных. Все свойства компонента устанавливаются в диалоговом окне Editing Chart, которое имеет несколько вкладок. Окно можно открыть с помощью команды *Edit Chart* контекстного меню ВК или использовать вкладку свойств ИО. Свойство SeriesList открывает вкладку Series (серии), свойство Title – вкладку Titles (заголовки).

ВК может содержать несколько диаграмм различных видов. Для построения каждой из них нужно:

- на этапе проектирования на вкладке *Series* с помощью кнопки **Add** добавить пустую серию, которая будет содержать данные, отображаемые с помощью диаграммы, и выбрать ее тип. Каждая серия имеет имя SeriesList [N], где N – порядковый номер серии, начиная с нуля;
- если необходимо, можно очистить серию от старых данных с помощью метода Clear (в противном случае новые точки могут добавляться к старым данным);
- в программе с помощью метода **AddXY** вставить в серию данные.

7. Оператор присоединения With

При установке нескольких свойств одного ВК необходимо каждый раз указывать его имя. Для того, чтобы не указывать несколько раз имя ВК, можно использовать оператор присоединения.

With <имя ВК> *do*
Begin
<операторы>
End;

8. Форматный вывод числовых данных

Функция FloatToStrF имеет следующий синтаксис:

FloatToStrF (x, <формат>, p, q), где

x – это переменная или выражение вещественного типа;

<формат> – название одного из определенных в Delphi форматов.

Если используются форматы ffFixed или ffGeneral, то p - это общее количество десятичных цифр в представлении числа, q – количество цифр в дробной части. Если p или q меньше, чем количество цифр в значении числа, то число округляется. В формате ffGeneral очень большие и очень маленькие числа выводятся в экспоненциальной форме (0.25E-08).

Функция FormatFloat имеет следующий синтаксис:

FormatFloat (<формат>, x), где

x – это переменная или выражение вещественного типа;

<формат> – строковая константа, в которой могут использоваться символы спецификаторы:

0 – определяет поле для цифры; если в данной позиции есть цифра, то она выводится, если нет, то выводится ноль;

– определяет поле для цифры; если в данной позиции есть цифра, то она выводится, если нет, то не выводится ничего, отличается от 0 тем, что начальные и конечные нули не выводятся;

. (точка) – определяет поле для разделителя целой и дробной части числа;

, (запятая) – определяет поле для разделителя тысяч.

Символы, которые выводятся в полях, определяемых символами **.** и **,** устанавливаются в ОС Windows, обычно это запятая и пробел.

Str (X:p:q, S) – преобразование вещественного значения X в строку S по указанному формату, p – это общее количество символов в представлении числа в строке, q – число символов в дробной части числа.

9. ВК RadioGroup (вкладка Standard) – группа переключателей.

Свойства ВК определяют:

Caption – заголовок группы;

Columns: Integer – количество столбцов в группе;

ItemIndex: Integer – содержит номер (значение индекса) установленного переключателя, нумерация начинается с нуля; по умолчанию равен -1, т.е. ни один переключатель не установлен.

Items: TStrings (набор строк) – содержит список строк с поясняющим текстом, находящимся рядом с переключателями; количество строк в списке определяет количество переключателей в группе.

Задание:

Задание 1

В соответствии со своим вариантом вычислить значения функции **y=f(x,a,b)** (для первой или одной, выбранной при выполнении программы из трех функций, приведенных в таблице 4.1)

для значений аргумента x, изменяющегося в интервале от $\mathbf{x_{нач}}$ до $\mathbf{x_{кон}}$ с шагом $\Delta\mathbf{x}$, и заданных констант **a** и **b**.

Исходные данные для отладки программы ($\mathbf{x_{нач}}$, $\mathbf{x_{кон}}$, $\Delta\mathbf{x}$, **a**, **b**), выбрать самостоятельно из интервала значений, где заданные функции определены.

При разработке интерфейса и программы выполнить следующие задания:

1. Для ввода исходных данных использовать главную форму. Исходные данные и результаты вывести на второй форме. Значения аргумента x и функции y вывести в ВК StringGrid, используя форматный вывод, строки таблицы пронумеровать. Таблица должна иметь заголовок и «шапку».
2. Для выбора нужной функции (одной из трех) использовать ВК TComboBox. Вывести динамически (из программы, с помощью метода LoadFromFile) вид выбранной функции в ВК TImage.
3. Выполнить проверку корректности ввода исходных данных (можно всех одновременно) и возможности расчета значения функции для введенных исходных данных, используя защищенный блок.
4. Построить график функции y=f(x,a,b), используя компонент TChart. Необходимость построения графика определить с помощью ВК TCheckBox.

Таблица 4.1

Варианты вида функций к заданию 1

Вариант	Вид функции 1	Вид функции 2	Вид функции 3
1	$y=\frac{arctg\ bx}{1+\sin^2 x}$	$y=\frac{1+\sqrt{bx}}{0{,}5+\sin^2 ax}$	$y=\frac{\ln^2(x-b)}{a\sqrt{x}}$
2	$y=\frac{\sin^2 x+a}{\sqrt{x}+bx}$	$y=\frac{arctg\ (a+x)}{\sqrt{a^3+b^3}}$	$y=\frac{1+\sqrt{bx}}{0{,}5+\sin^2 ax}$
3	$y=\sqrt{\frac{a+bx}{\ln^2 x}}$	$y=\frac{a-\sqrt{bx}}{1+\cos\lvert 2x\rvert}$	$y=\frac{a-e^{bx}}{\ln\lvert 2x\rvert}$
4	$y=\frac{\ln^2(x-b)}{a\sqrt{x}}$	$y=\frac{arctg\ bx}{1+\sqrt[3]{ax}}$	$y=\frac{(a+bx)^2}{1+\cos^3 ax}$
5	$y=\frac{a\ln^2 x}{b+\sqrt{x}}$	$y=\frac{arctg\ bx}{1+\sqrt[3]{ax}}$	$y=\frac{b+\sin^2 ax}{e^{-x/2}}$
6	$y=\frac{e^{ax}+b}{1+\cos^2 x}$	$y=\frac{\sqrt{\lvert a\ln x\rvert}}{1+tg^2 bx}$	$y=\frac{\sin^2 x-a}{bx}$

Вариант	Вид функции 1	Вид функции 2	Вид функции 3
7	$y = \frac{a + \sqrt[3]{x}}{\sin^2 bx}$	$y = \sqrt{\frac{a + bx}{\ln^2 x}}$	$y = \frac{arctg^2 ax}{b + 0{,}5x}$
8	$y = \frac{a\sqrt{\lvert x\rvert} - bx}{\ln^3 x}$	$y = \frac{arctg\ bx}{1 + \sqrt[3]{ax}}$	$y = \frac{\ln(a^2 - x)}{b \sin^2 x}$
9	$y = \frac{\sqrt{ax} - b}{tg^2 x}$	$y = \frac{(a + bx)^2}{1 + \cos^3 ax}$	$y = \frac{a - \sqrt{bx}}{1 + \cos\lvert 2x\rvert}$
10	$y = e^{-x}\frac{a + bx}{\ln^2\lvert x + 1\rvert}$	$y = \frac{arctg\ bx}{1 + \sqrt[3]{ax}}$	$y = \frac{\sin^3 ax}{ax + b}$
11	$y = \frac{tg^2 x - b}{e^{ax}}$	$y = \frac{a + \sqrt[3]{x}}{\sin^2 bx}$	$y = \frac{\sqrt{\lvert a \ln x\rvert}}{1 + tg^2 bx}$
12	$y = \frac{arctg\ bx}{1 + \sqrt[3]{ax}}$	$y = \frac{\ln^2(a + x)}{(b + x)^2}$	$y = \frac{1 + tg^2 x}{b + e^{x/a}}$
13	$y = \frac{\sin^3 ax}{ax + b}$	$y = \frac{arctg\ bx}{1 + \sqrt[3]{ax}}$	$y = \frac{\cos^2 2x + b}{\sqrt{1 + e^{ax}}}$
14	$y = \frac{e^{-ab}}{b + \cos^3 ax}$	$y = \frac{arctg\ (a + x)}{\sqrt{a^3 + b^3}}$	$y = \frac{\sqrt{ax + b}}{\ln^2\lvert x\rvert}$
15	$y = \frac{\ln^2\lvert x\rvert + b}{a\sqrt{x}}$	$y = \frac{arctg\ bx}{1 + \sqrt[3]{ax}}$	$y = \frac{1 + \sin^2 ax}{b^2 + x^2}$

Задание 2

Вычислить и вывести на экран в виде таблицы значения функции по варианту в таблице 4.2, заданной с помощью ряда Тейлора, на интервале от $x_{нач}$ до $x_{кон}$ с шагом Δx с точностью **ε**. Таблица должна иметь заголовок и «шапку». Каждая строка таблицы должна содержать:

- значение аргумента,
- значение функции, полученное с помощью стандартных функций Delphi,
- значение функции, полученное с помощью разложения в ряд Тейлора,
- количество просуммированных членов ряда.

Таблица 4.2

Варианты вида функции для задания 2

Вариант	Разложение функции в ряд Тейлора

1.	$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots\right)$	$\lvert x\rvert>1$
2.	$e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}-\ldots$	$\lvert x\rvert<\infty$
3.	$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-\ \ \ldots$	$\lvert x\rvert<\infty$
4.	$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}\ -\frac{x^4}{4}-\ \ldots$	$-1<x\le 1$
5.	$\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right)$	$\lvert x\rvert<1$
6.	$\ln(1-\mathrm{x})=-\sum_{\mathrm{n=1}}^{\infty}\frac{\mathrm{x^n}}{\mathrm{n}}=-\left(\mathrm{x}+\frac{\mathrm{x}^2}{2}+\frac{\mathrm{x}^3}{3}+\ldots\right)$	$-1\le \mathrm{x}\le 1$
7.	$arcctg\ \ x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}-\ \ \ldots$	$\lvert x\rvert\le 1$
8.	$arctg\ x=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots$	$x>1$
9.	$arctg\ \ x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ \ \ \ldots$	$\lvert x\rvert\le 1$
10.	$Arth\ \ x=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ \ \ \ldots$	$\lvert x\rvert<1$
11.	$Arth\ \ x=\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\ldots$	$\lvert x\rvert>1$
12	$arctg\ \ x=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+\ldots$	
13.	$e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots$	$\lvert x\rvert<\infty$
14.	$\cos\ x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$	$\lvert x\rvert<\infty$
15.	$\frac{\sin\ x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}-\ldots$	$\lvert x\rvert<\infty$

Контрольные вопросы к защите:

1. Какой алгоритм называется циклическим?
2. Как представляется в схеме алгоритма и записывается в программе оператор цикла While?
3. Как выполняется оператор цикла While?
4. Дайте определение табулирования функции.

5. Дайте словесное описание алгоритма табулирования функции для аргумента, заданного интервалом значений с фиксированным шагом.
6. Как выполнить форматирование выводимых данных?
7. Как добавить новую форму к проекту?
8. В чем заключаются особенности использования в проекте нескольких форм?
9. Для чего служат методы Show, ShowModal, Close, SetFocus?
10. Для чего используется ВК StringGrid, Chart, RadioGroup?
11. Назовите основные свойства ВК StringGrid, Chart, RadioGroup и их назначение.
12. Как проверить корректность ввода исходных данных с помощью защищенного блока?

Лабораторная работа № 5
Обработка одномерных массивов

Цель работы: Получить навыки разработки и реализации алгоритмов ввода, вывода и обработки одномерных массивов в среде Delphi.

Теоретические сведения:

Одномерным массивом называется упорядоченная последовательность величин одного типа, имеющих одно имя, но различающихся индексами.

Индекс – это выражение целого типа, определяющее положение отдельной величины в последовательности. Каждая отдельная величина называется элементом массива.

Описание массива:

1 способ (раздел описания переменных):

```
var
   <идентификатор>: array [Nнач..Nкон] of < тип элементов>;
```

Nнач, Nкон – это минимальное и максимальное значение индекса, обязательно константы. Обычно Nнач=1, тогда Nкон это максимальное число элементов в массиве.

2 способ (раздел описания типов и раздел описания переменных):

```
type
   <имя типа>=array [Nнач..Nкон] of  < тип элементов>;
var
    <идентификатор>:<имя типа>;
```

Обращение к элементу массива:

<идентификатор массива> [<индекс>]

Свойства элементов массива:

- все элементы массива имеют один тип;
- номера элементов изменяются от Nнач до Nкон с шагом 1;
- число используемых элементов может быть меньше, чем число элементов в описании массива.

Если работа с одним массивом выполняется в нескольких процедурах одного модуля, то описание массива и числа элементов лучше разместить до всех процедур в разделе реализации модуля (после Implementation), если в разных модулях, то в разделе интерфейса одного из модулей (после Interface вместе с описанием формы).

Ввод массива

Графическая схема представлена на рисунке 5.1.

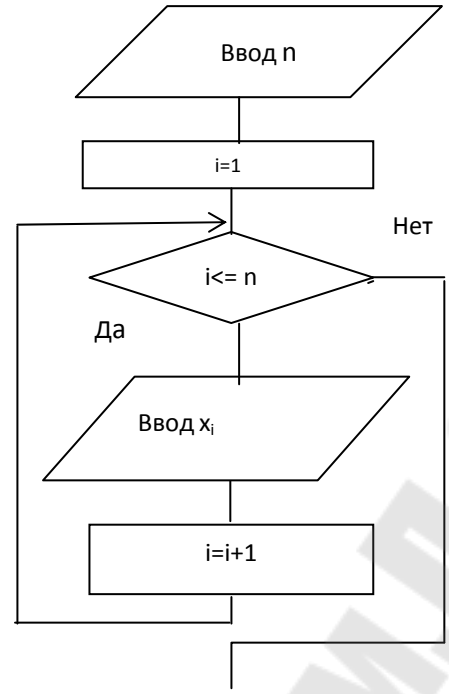


Рис 5.1. Графическое представление ввода массива

I способ:

Рис 5.2. Вид окна проекта

После щелчка по кнопке **Ввод** в процедуре обработки этого события вводятся число элементов и сами элементы. Для ввода элементов используется компонент StringGrid1. Свойства этого ВК, устанавливаемые в ИО:

- RowCount=1;
- ColCount= max числу элементов в описании массива;
- FixedRows=0;
- FixedCols=0;

- Options.GoEditing=True;
- Options.GoTabs=True.

можно установить:

- DefaultColWidth ~34 (по умолчанию 54 пикселя) ширина ячейки таблицы.

Процедура:

```
n:=StrToInt(Edit1.Text);
StringGrid1.ColCount:=n;
for i:=1 to n do
   x[i]:=StrToFloat(StringGrid1.Cells[i-1,1]);
```

II способ:

Рис 5.3. Вид окна проекта

Количество элементов массива вводятся после щелчка по кнопке **ОК**, в соответствующей процедуре обработки события. В этой же процедуре устанавливается число столбцов в StringGrid1 равное количеству элементов массива (ColCount=n) и ВК StringGrid1 делается видимым. Его первую его строку можно сделать фиксированной с заголовками элементов массива, во второй вводить элементы. Ввод элементов в ОЗУ выполняется после щелчка по кнопке **Ввод**.

III способ:

На форме одна кнопка для ввода элементов массива. Для действий, выполняемых с помощью кнопки ОК во втором способе, создается процедура обработки события onChange для ВК Edit1 (TForm1.Edit1Change). В процедуре необходимо проверить, что число элементов введено правильно, с помощью защищенного блока.

```
Procedure TForm1.Edit1Change (Sender: TObject);
var
  i:byte;
begin
  try
    n:=StrToInt(Edit1.Text);
    StringGrid1.ColCount:=n;
    StringGrid1.Visible:=True;
    StringGrid1.Col:=0;
    StringGrid1.Row:=1;
    StringGrid1.SetFocus;
  except
     ShowMessage('Введите n правильно')
  end;
```

Вывод массива

Вывод массива выполняется в ВК StringGrid. Он может находиться на той же форме, где выполнялся ввод, или на новой. Свойства этого ВК, устанавливаемые в ИО:

- RowCount =1; (=2, если с заголовками элементов)
- ColCount = 2, а затем устанавливается в программе равное числу элементов после ввода массива;
- FixedRows=0; (=1, если с заголовками элементов)
- FixedCols=0;
- Visible = False, если массив выводится на той же форме, где был ввод.

Графическая схема представлена на рисунке 5.2.

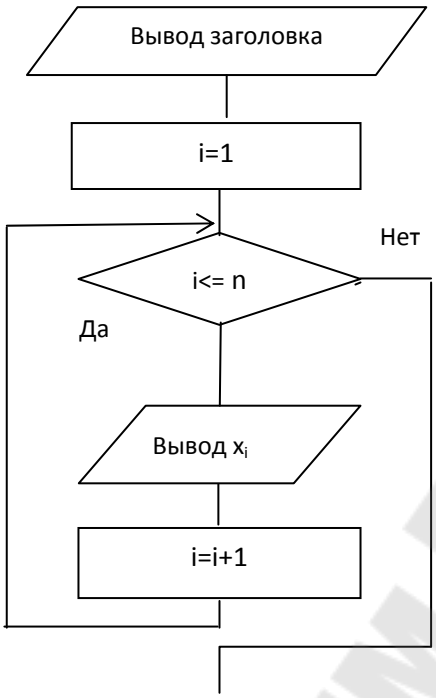


Рис 5.4. Графическое представление вывода массива

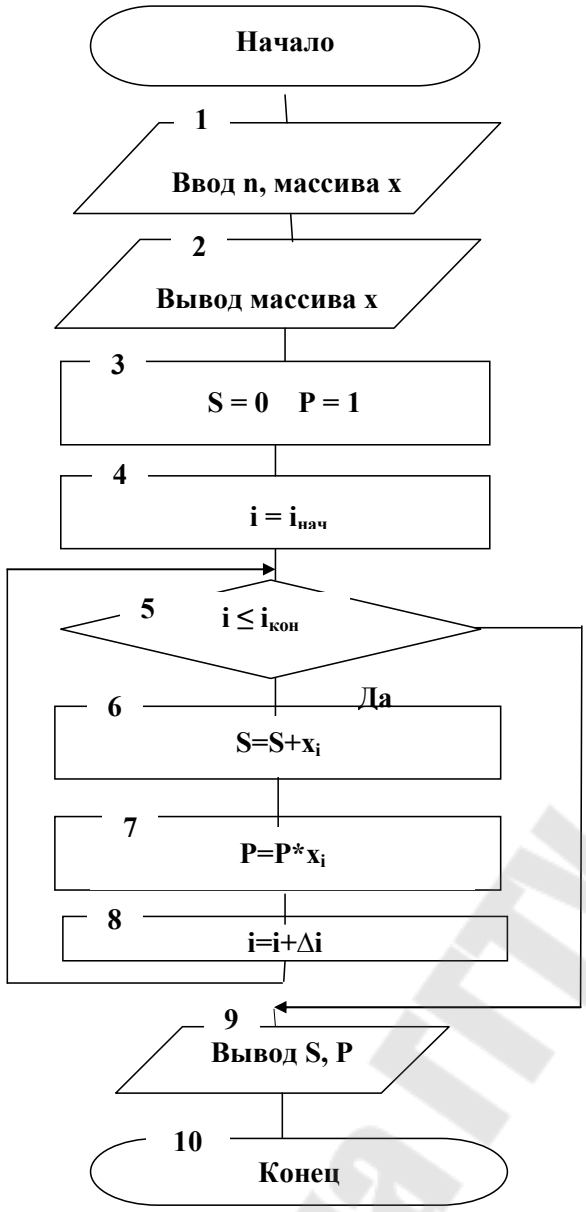
Фрагмент процедуры (после ввода):

```
label3.Caption:=’Исходный массив’;
With StringGrid2 do
 begin
   ColCount=n;
   Visible:=True;    (если на одной форме)
    for i:=1 to n do
      StringGrid2.Cells[i-1,0]:=FloatToStr(x[i]);
 end;
```

Типовые алгоритмы обработки одномерных массивов

1. Вычисление суммы и произведения элементов, находящихся на разных местах в массиве

Вычисление суммы и произведения всех элементов:
S = 0 S = S + x[i] для i = 1 .. n
P = 1 P = P * x[i] для i = 1 .. n



$i_{нач}$ – минимальное значение номера элемента массива

$i_{кон}$ – минимальное значение номера элемента массива

Δi – шаг изменения индекса

Рис 5.5. Общая графическая схема типового алгоритма

2. Вычисление суммы, произведения и количества элементов, удовлетворяющих заданному условию и находящихся на разных местах в массиве

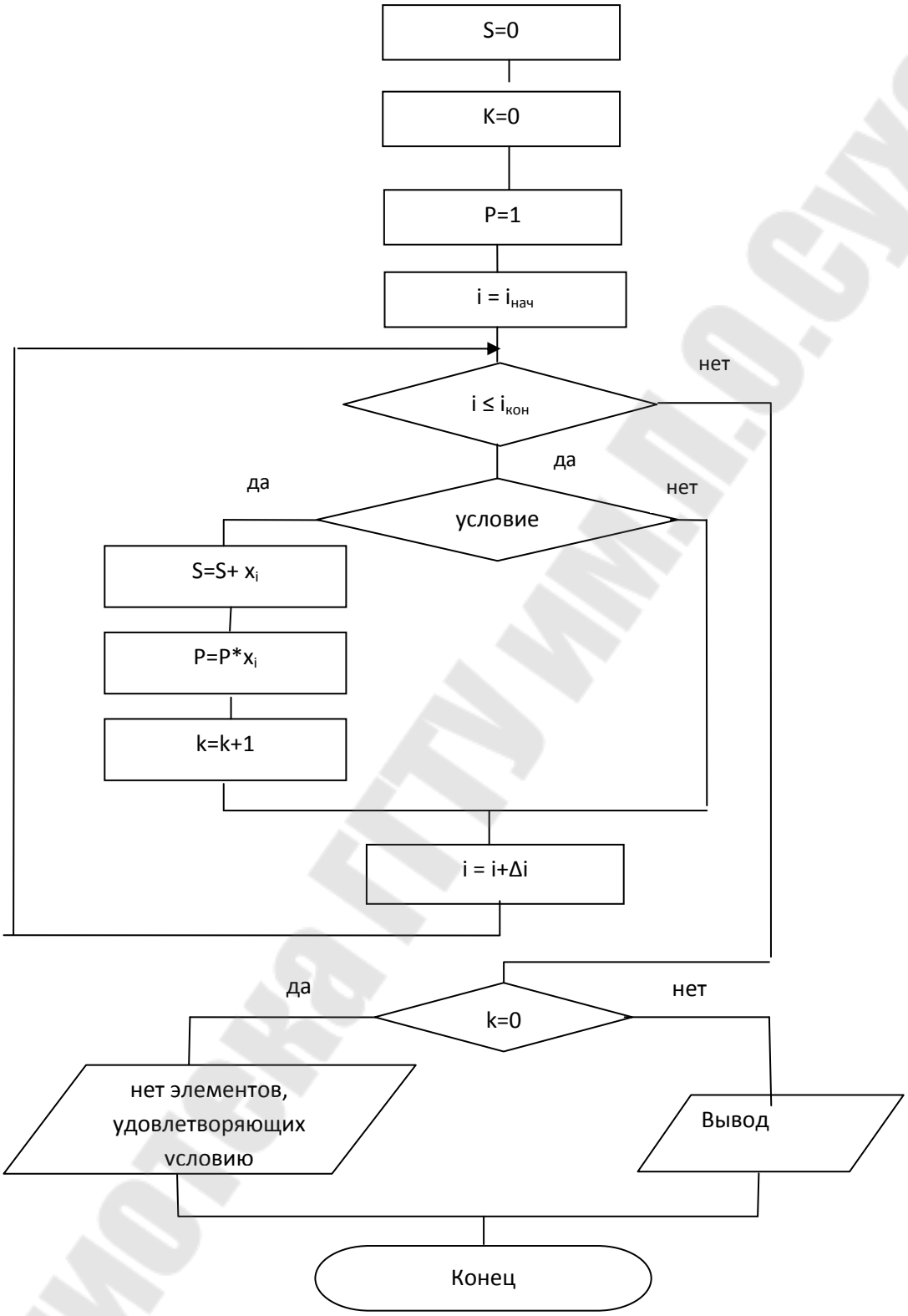


Рис 5.6. Общая графическая схема типового алгоритма

3.

Поиск минимальных и максимальных элементов массива и определение их номеров

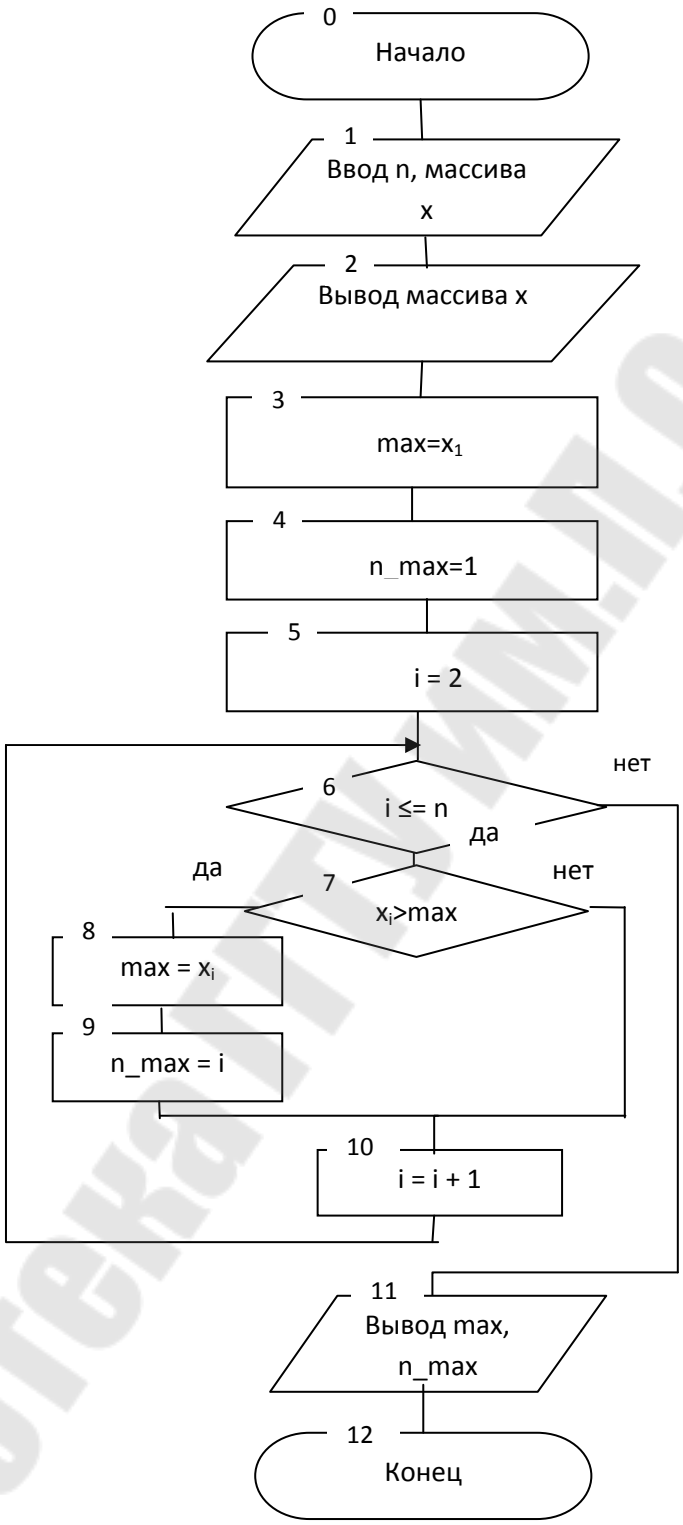
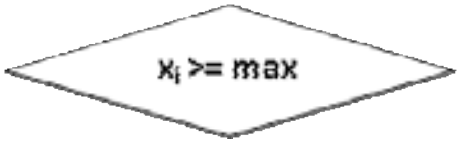


Рис 5.7. Общая графическая схема типового алгоритма

Замечания к алгоритму:

1. При поиске минимального элемента блок **7** должен иметь вид:

2. Для того, чтобы найти номер последнего из равных максимальных элементов, блок **7** должен иметь вид:



Тесты:

1. минимальный элемент находится на первом месте в массиве
2. минимальный элемент находится на последнем месте в массиве
3. минимальный элемент находится в середине массива
4. зависит от конкретных условий задачи

4. Формирование новых массивов

Постановка задачи: из исходного массива перенести в новый массив элементы, удовлетворяющие некоторому условию, не меняя их взаимного расположения.

Тесты:

1. новый массив не сформирован, т.к. в исходном массиве нет элементов, удовлетворяющих заданному условию
2. новый массив сформирован из всех элементы исходного массива, т.к. они все удовлетворяют условию
3. новый массив сформирован из части элементов исходного массива, удовлетворяющих заданному условию

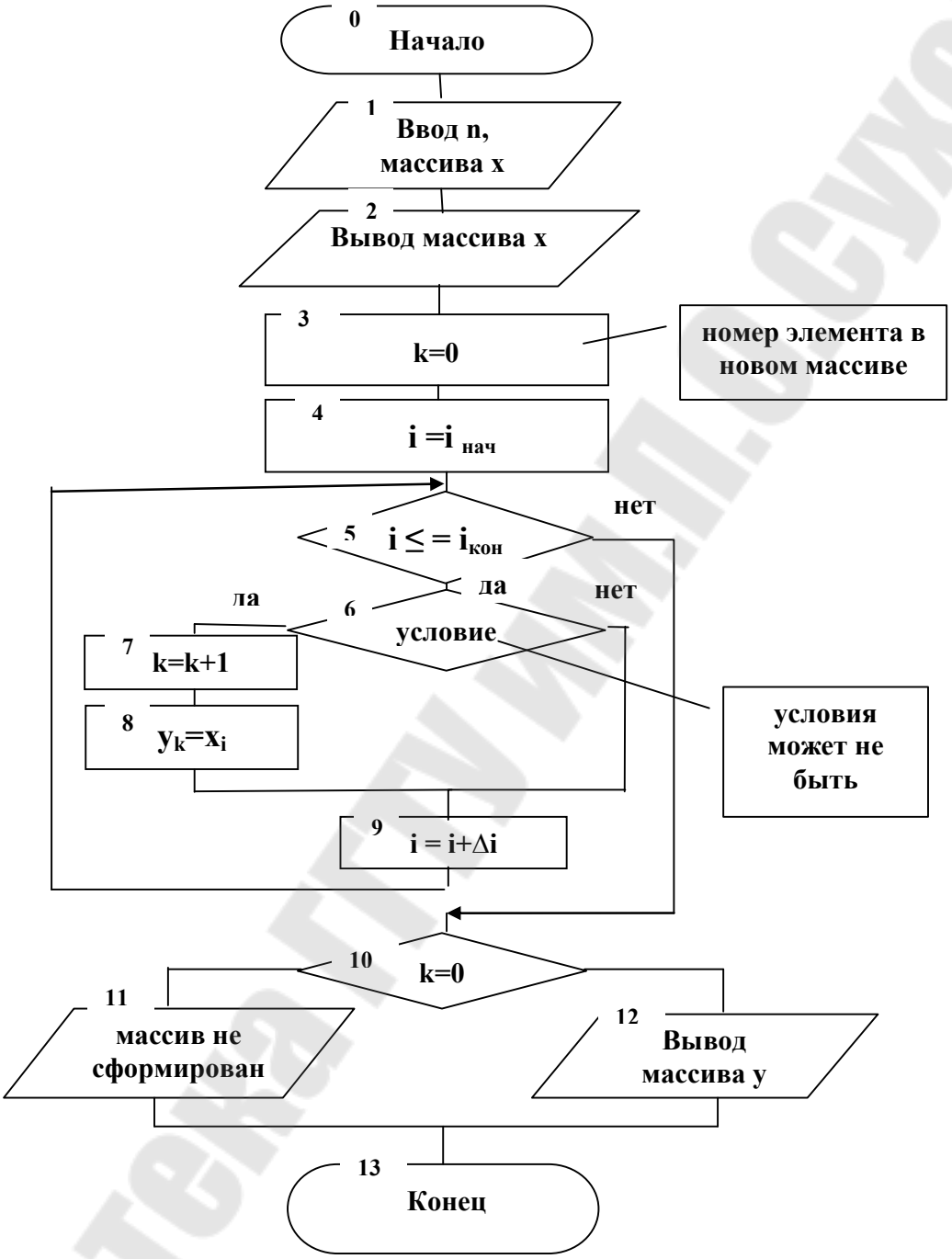


Рис 5.8. Общая графическая схема типового алгоритма

Задание:

При разработке интерфейса и программ выполнить следующие задания:

1. Для ввода размерности исходного массива использовать ВК TEdit, для ввода и вывода элементов массивов – ВК TStringGrid.

2. Динамически установить размер ВК TStringGrid при вводе размерности массива;

3. Для ввода исходных данных использовать главную форму. Исходные данные, промежуточные результаты и результаты вывести на второй форме.

4. Выполнить проверку корректности ввода исходных данных, используя защищенный блок.

Исходные данные для отладки программ выбрать самостоятельно.

Задание 1

В каждой из задач (таблица 5.1) предполагается, что задан массив чисел. Программа должна:

1) вводить размерность и элементы массива;

2) вводить некоторые дополнительные числа;

3) выполнять действия в соответствии с условием задачи;

4) выводить исходные данные и результаты вычислений.

Таблица 5.1

Условие задачи по вариантам для задания 1

Вариант	Задание
1	Найти количество чисел, принадлежащих промежутку [a,b], и сумму чисел, стоящих на местах, кратных 3.
2	Найти сумму чисел, меньших заданного D, и количество чисел, стоящих на четных местах и больших заданного C.
3	Найти произведение всех чисел, стоящих на местах, кратных 4, и количество чисел, небольших заданного A.
4	Найти количество чисел, меньших заданного X, и произведение всех отрицательных чисел, стоящих на нечетных местах.
5	Найти количество чисел, непринадлежащих промежутку (X,Y], и сумму отрицательных чисел, стоящих на четных местах.
6	Найти количество неотрицательных чисел и определить сумму чисел, стоящих на местах, кратных 3, и неравных заданному F.
7	Найти среднее арифметическое отрицательных чисел и определить количество чисел, по величине больших A и стоящих на четных местах.

Вариант	Задание
8	Найти среднее арифметическое положительных чисел, стоящих на нечетных местах, и количество чисел, меньших заданного B.
9	Найти среднее арифметическое чисел, принадлежащих промежутку [A,B), и количество положительных чисел, стоящих на местах, кратных 4.
10	Найти среднее арифметической чисел, неравных заданному C, и произведение неположительных чисел, стоящих на четных местах.
11	Найти среднее арифметическое чисел, больших заданного D и стоящих на нечетных местах, и определить количество чисел, небольших заданного F.
12	Найти среднее арифметическое чисел, непопадающих в промежуток [A,B], и количество положительных чисел, стоящих на местах, кратных 3.
13	Найти среднее арифметическое ненулевых чисел и количество чисел, по величине небольших A и стоящих на четных местах.
14	Вычислить произведение чисел, принадлежащих промежутку (A,B], и количество отрицательных чисел, стоящих на местах, кратных 3.
15	Найти среднее арифметическое положительных чисел, стоящих на нечетных местах, и произведение чисел, меньших заданного C.

Задание 2

В каждой из задач (таблица 5.2) предполагается, что задан массив из N чисел. Программа должна:

1. вводить размерность и элементы исходного массива;
2. выполнять действия в соответствии с условием задачи;
3. выводить:
   - исходный массив;
   - максимальный или минимальный элемент массива и его номер;
   - массив, полученный в результате выполнения задания.

Таблица 5.2

Условие задачи по вариантам для задания 2

Вариант	Задание
1	Найти максимальный элемент и поменять его местами с последним элементом массива.
2	Найти минимальный элемент и поменять его местами с предыдущим элементом массива.
3	Найти минимальный элемент и поменять его местами с последующим элементом массива.
4	Найти максимальный элемент и поменять его местами с шестым элементом массива.
5	Найти максимальный элемент, присвоить его значение последнему элементу массива, а вместо максимального числа записать – 1.
6	Найти минимальный элемент, присвоить его значение первому элементу массива, а вместо минимального элемента записать число 9999.
7	Найти минимальный элемент и поменять его местами с третьим элементом массива.
8	Найти минимальный элемент и заменить его на сумму первого и последнего элементов.
9	Найти максимальный элемент и поменять его местами с предпоследним элементом массива.
10	Найти минимальный элемент, присвоить его значение последнему элементу массива, а вместо минимального элемента записать значение 3N.
11	Найти минимальный элемент и поменять его местами с элементом массива, номер которого задан.
12	Найти максимальный элемент и поменять его местами со вторым элементом массива.
13	Найти минимальный элемент и поменять его местами с последним элементом массива.
14	Найти минимальный элемент и поменять его местами с предпоследним элементом массива.
15	Найти минимальный элемент и поменять его местами с третьим элементом массива.

Задание 3

В каждой из задач (таблица 5.3) предполагается, что задан массив чисел. Программа должна:

1. вводить размерность и элементы исходного массива;
2. вводить некоторые дополнительные числа;
3. выполнять действия в соответствии с условием задачи;
4. выводить:
   - исходные данные;
   - промежуточные результаты (минимум, максимум и т.п.);
   - сформированный массив или сообщения о том, по какой причине его нельзя сформировать.

Таблица 5.3

Условие задачи по вариантам для задания 3

Вариант	Задание
1	Сформировать новый массив из элементов исходного массива, которые больше, чем сумма положительных элементов массива, стоящих за максимальным элементом массива.
2	Сформировать новый массив из отрицательных элементов исходного массива, расположенных между минимальным и максимальным элементами массива, не включая их.
3	Если в массиве нет нулей, вывести сообщение об этом. В противном случае сформировать новый массив из номеров ненулевых элементов, расположенных за первым нулевым элементом до максимального элемента массива включительно.
4	Если в исходном массиве последний элемент максимальный, то заменить каждый нулевой элемент массива на сумму элементов, расположенных в массиве после него.
5	Сформировать новый массив из элементов исходного массива, расположенных в массиве между вторым положительным элементом и минимальным элементом массива, не включая их.
6	Сформировать новый массив из элементов исходного массива, находящихся между первым отрицательным и максимальным элементами массива, не включая их.

Вариант	Задание
7	Сформировать новый массив из элементов исходного массива, расположенных за минимальным элементом массива и которых нет среди элементов, расположенных до минимального элемента.
8	Сформировать новый массив из элементов исходного массива, расположенных между максимальным элементом массива и последним отрицательным элементом, не включая их.
9	Если в массиве все элементы больше заданного числа Z, вывести сообщение об этом. В противном случае сформировать новый массив из номеров ненулевых элементов, начиная с первого элемента небольшего Z и до максимального элемента массива включительно.
10	Сформировать новый массив из номеров положительных элементов исходного массива, расположенных между первым отрицательным элементом и максимальным элементом массива, не включая их.
11	Если в массиве нет нулей, вывести сообщение об этом. В противном случае сформировать новый массив из номеров элементов исходного массива, находящихся между первым нулем и максимальным элементом массива, не включая их.
12	Если в массиве все элементы одинаковые, вывести об этом сообщение. В пртивном случае сформировать новый массив из элементов исходного массива, начиная с первого элемента, не совпадающего с последним элементом массива, и до максимального элемента массива включительно.
13	Если в массиве все элементы нечетные, вывести сообщение об этом, в противном случае заменить каждый нечетный элемент массива на номер максимального из элементов массива, находящихся в массиве за первым четным элементом.
14	Сформировать новый массив из элементов исходного массива, которые больше произведения первого и последнего отрицательных элементов массива и расположены за минимальным элементом массива.

Вариант	Задание
15	Сформировать новый массив из номеров отрицательных элементов исходного массива, расположенных между максимальным и первым положительным элементом массива, следующим за максимальным.

Контрольные вопросы к лабораторной работе:

1. Как выполнить описание одномерного массива? Как обратиться к элементу массива?
2. Опишите массив вещественных чисел, который может содержать не более 10 элементов.
3. Какой визуальный компонент используется для ввода элементов массива? Какие свойства необходимо для него установить в Инспекторе Объектов?
4. Как выполнить ввод массива? Нарисуйте схему алгоритма и напишите фрагмент процедуры ввода.
5. Как выполнить вывод массива? Нарисуйте схему алгоритма и напишите фрагмент процедуры вывода.
6. Дайте словесное описание алгоритма поиска минимального элемента массива и определения его номера в массиве.
7. Нарисуйте схему алгоритма поиска минимального элемента массива и определения его номера в массиве.
8. Дайте словесное описание алгоритма поиска максимального элемента массива и определения его номера в массиве.
9. Нарисуйте схему алгоритма поиска максимального элемента массива и определения его номера в массиве.
10. Как поменять местами два элемента массива?
11. Дайте словесное описание алгоритма формирования нового массива из элементов исходного массива, удовлетворяющих заданному условию.
12. Нарисуйте схему алгоритма формирования нового массива.

Лабораторная работа № 6
Обработка двумерных массивов

Цель работы: Получить навыки разработки и реализации алгоритмов ввода, вывода и обработки двумерных массивов в среде Delphi.

Теоретические сведения:

Описание двумерного массива

Двумерный массив или матрица - это упорядоченная последовательность величин одного типа, имеющих одно имя, но различающихся индексами. Каждой величине или элементу массива соответствуют два целых числа (индекса), определяющие положение элемента в массиве (номер строки и номер столбца). Размер матрицы определяется двумя целыми числами – количеством строк и количеством столбцов.

1 способ (раздел описания переменных):

```
var
   <идентификатор>: array [N1нач..N1кон, N2нач..N2кон] of < тип элементов>;
```

Обычно N1нач и N1кон =1, тогда N1кон и N2кон – это максимально возможное число строк и столбцов в массиве.

Пример 6.1:

```
   x: array [1..10,1..5] of real;
```

2 способ (раздел описания типов и раздел описания переменных):

```
type
   <имя типа>=array [N1нач..N1кон, N2нач..N2кон] of < тип элементов>;
var
```

<идентификатор>:<имя типа>;

Обращение к элементу массива:
<идентификатор массива> [№ строки, № столбца]

Ввод массива

Графическая схема ввода массива представлена на рисунке 6.1.

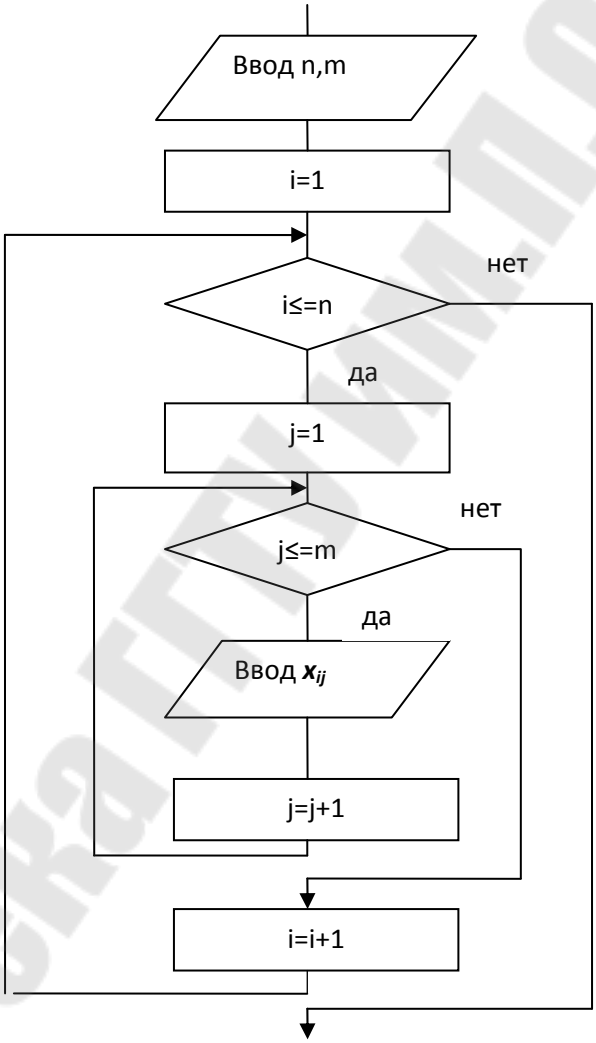


Рис 6.1. Графическое представление ввода двумерного массива

Для ввода элементов массива используется ВК StringGrid.
Свойства ВК StringGrid, которые устанавливаются в ИО:

- RowCount = максимально возможному числу строк и столбцов

- ColCount = в описании массива
- FixedRows = 0
- FixedCols = 0
- Options.GoEditing = True
- Options.GoTabs = True

```
procedure TForm1.Button1Click(Sender: TObject);
var
  n,m,j,i:byte;
  x:array [1..10,1..10] of real;
begin
  n:=StrToInt(Edit1.Text);
  m:=StrToInt(Edit2.Text);
  for i:=1 to n do
   for j:=1 to m do
     x[i,j]:=StrToFloat(StringGrid1.Cells[j-1,i-1]);
     StringGrid1.RowCount:=n;
     StringGrid1.ColCount:=m;
end;
```

Можно использовать для ввода числа строк и столбцов матрицы ВК SpinEdit (вкладка Samples). Для ввода используется оператор <идентификатор>:= SpinEdit_.Value

Свойства MinValue и MaxValue содержат минимальное и максимальное значение диапазона возможных значений.

Вывод массива

Для вывода элементов массива используется ВК StringGrid.
Свойства ВК StringGrid, которые устанавливаются в ИО:

- RowCount = любое, будет установлено в процедуре
- ColCount после ввода размерности массива
- FixedRows = 0 (=1, если с заголовками столбцов)
- FixedCols = 0 (=1, если с заголовками строк)
- Visible = False, если ввод и вывод на одной форме

Графическая схема вывода массива представлена на рисунке 6.1.

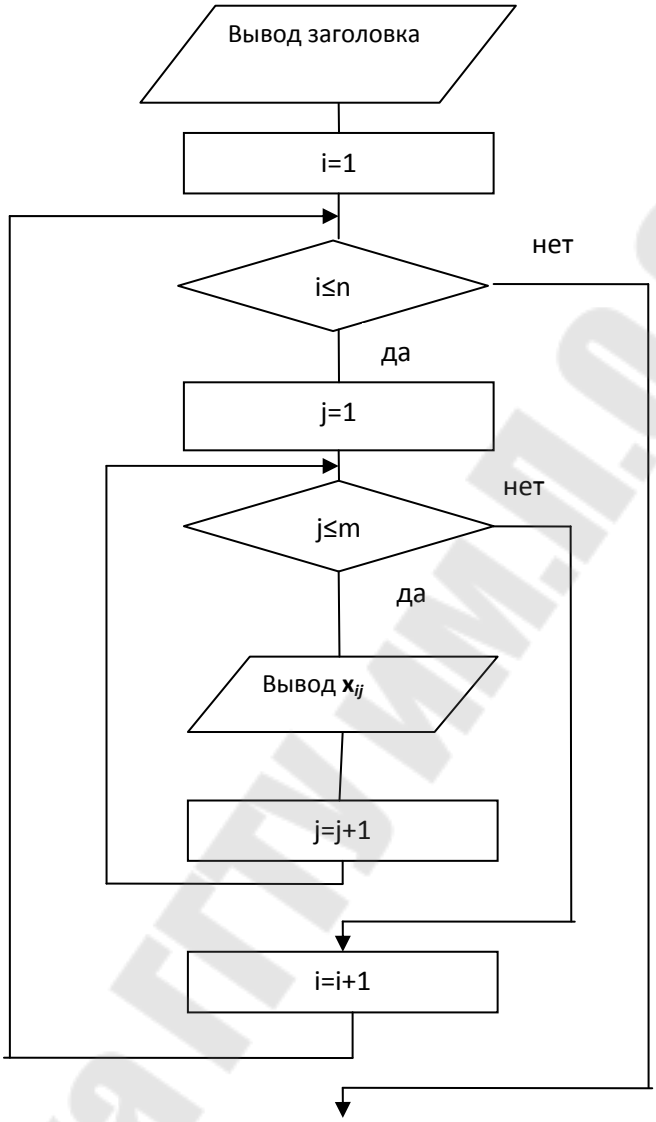


Рис 6.2. Графическое представление вывода двумерного массива

Текст процедуры вывода:

```
With StringGrid2 do
begin
  RowCount:=n;
  ColCount:=m;
  Visible:=True;
  for i:=1 to n do
    for j:=1 to m do
      Cells[j-1, i-1]:=FloatToStr(x[i, j]);
end;
```

Типовые алгоритмы обработки двумерных массивов

1. Вычисление сумм, количеств, произведений элементов матрицы, удовлетворяющих заданному условию

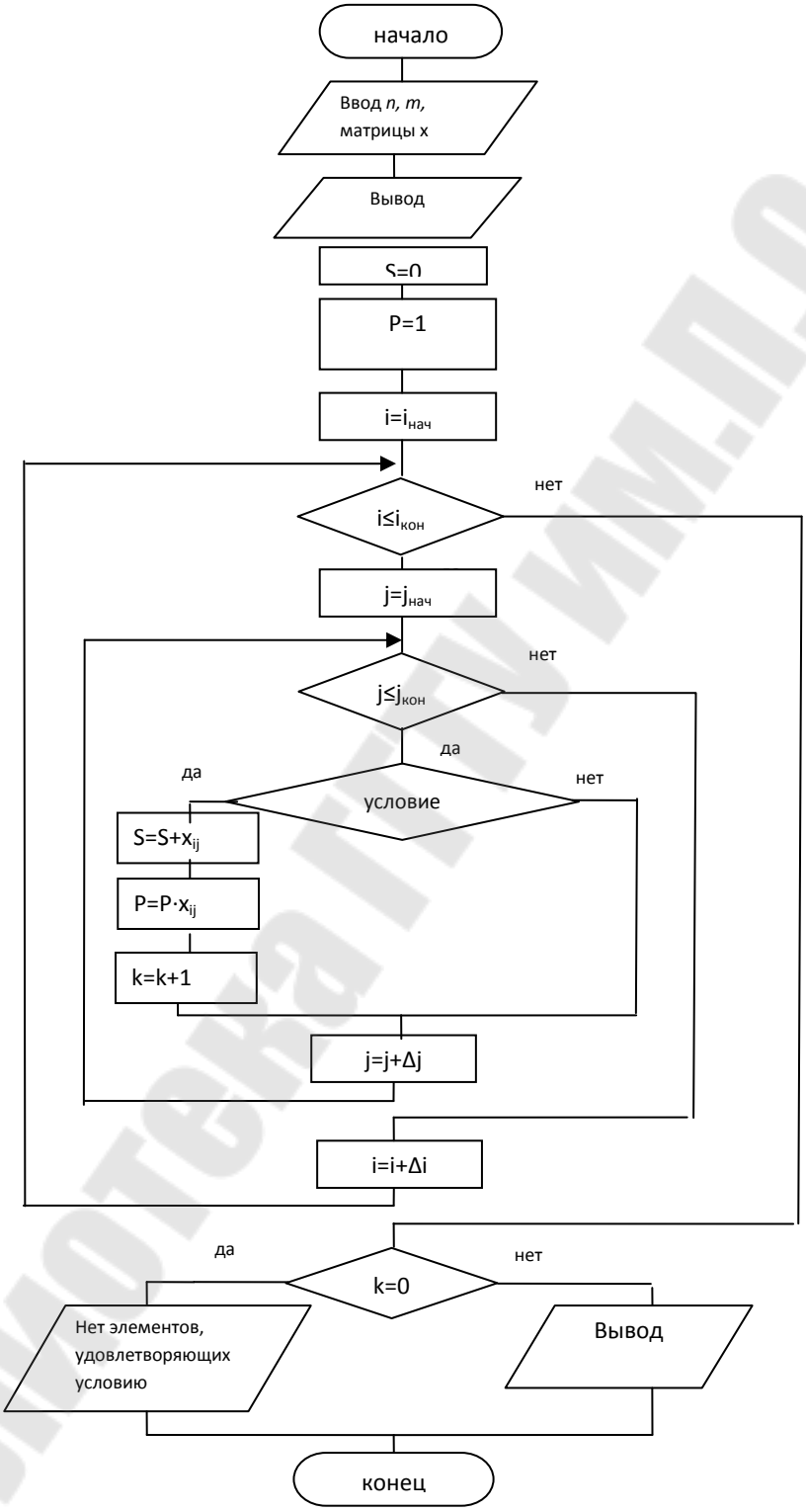


Рис 6.3. Общая графическая схема типового алгоритма

2. Поиск максимальных и минимальных элементов матрицы и определение их местоположения в матрице

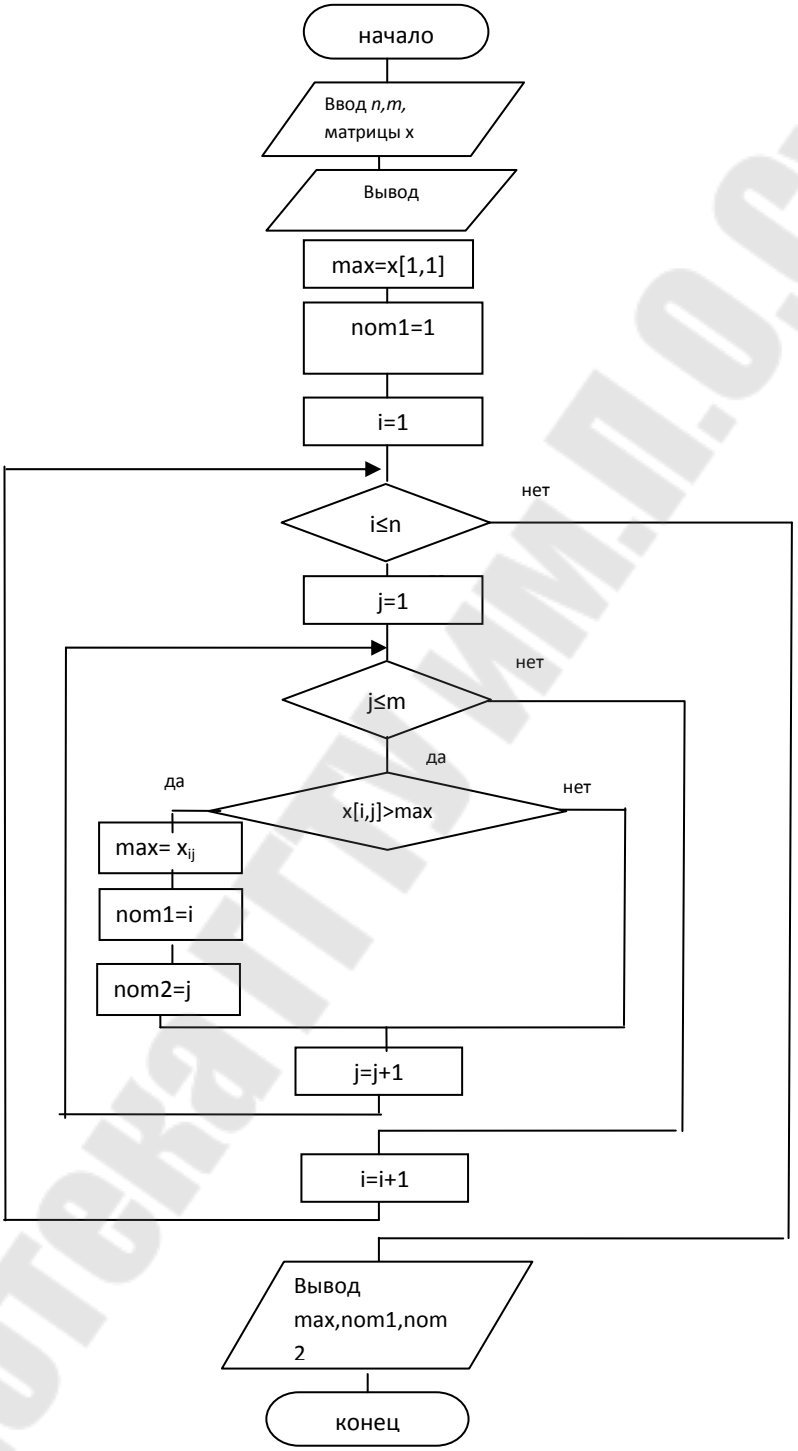


Рис 6.4. Общая графическая схема типового алгоритма

max – максимальный элемент матрицы

nom1 – номер строки, в которой находится максимальный элемент

nom2 – номер столбца, в котором находится максимальный элемент

Программирование циклов с досрочным завершением

Задачи обработки массивов, использующие циклы с досрочным завершением, часто имеют следующие формулировки:

1. Определить есть ли в массиве элемент или группа элементов, удовлетворяющих заданному условию.
2. Определить, все ли элементы или группы элементов удовлетворяют заданному условию.
3. Определить расположение в массиве элемента или группы элементов, удовлетворяющих заданному условию.

При решении таких задач могут возникнуть ситуации, определенные исходным массивом, при которых нет необходимости просматривать весь массив полностью, т.к. в первой и третьей задаче найден элемент или группа элементов, удовлетворяющие заданному условию, а во второй найден элемент или группа элементов, не удовлетворяющие заданному условию. Если в массиве не нужно просматривать все элементы, строки или столбцы, следовательно, нужно завершить цикл просмотра элементов при выполнении некоторого дополнительного условия. Дополнительное условие включается в условие цикла, оно содержит второй параметр цикла.

Параметр цикла – это переменная, значение которой изменяется в теле цикла и от которой зависит число повторений цикла, т.к. она входит в условие цикла.

Для программирования таких алгоритмов всегда используется оператор цикла While, т.к. число повторений цикла нельзя определить до начала цикла, оно зависит от элементов массива.

Использование логических переменных в циклах с досрочным завершением.

Логические переменные могут принимать только одно из двух значений true (истина) или false (ложь). Для их описания используется тип Boolean. В памяти эти переменные занимают 1 байт.

Использование переменных целого и логического типов, соответственно:

```
var                                  var
  k: byte;                             l: boolean;

if k=0 then ……                       if l = true then ……   (if l….)
if k=1 then ……                       if l = false then … (if not l….)

While (i<=n) and (k=0) do            While (i<=n) and l do
    ……..                                 …….. (l=true)
```

Задание:

В каждой задаче (таблица 6.1, 6.2, 6.3) задана матрица размером N x M или N x N (если выполняется обработка элементов на диагоналях матрицы), а в некоторых задачах - дополнительный массив чисел.

Программа должна:

- вводить размерность (ВК TSpinEdit) и элементы матрицы (ВК TStringGrid);
- динамически установить размер ВК TStringGrid при вводе размерности матрицы;
- выполнять действия в соответствии с условием задачи;
- выводить исходные данные, результаты расчета, преобразованную матрицу или сообщение о том, что преобразования не нужны (на второй форме);
- выполнить проверку корректности ввода исходных данных, используя защищенный блок.

Исходные данные для отладки программы выбрать самостоятельно.

Задание 1

Таблица 6.1

Условие задачи по вариантам для задания 1

Вариант	Задание
1	Вычислить сумму положительных элементов, расположенных в столбцах с четными номерами.
2	Вычислить произведение отрицательных элементов, расположенных в строках с нечетными номерами.
3	Вычислить сумму квадратов элементов из интервала [A,B], расположенных в строках с четными номерами.
4	Определить количество элементов, больших заданного A и расположенных в строках с нечетными номерами.
5	Вычислить сумму элементов, меньших заданного B и расположенных в столбцах с номерами, кратными 3.
6	Вычислить произведение положительных элементов, расположенных в строках с номерами, кратными 4.
7	Вычислить сумму квадратов отрицательных элементов, расположенных в столбцах с четными номерами.
8	Определить количество элементов, не принадлежащих промежутку (A,B) и расположенных в столбцах с нечетными номерами.
9	Вычислить сумму элементов, не меньших заданного D и расположенных в строках с четными номерами.
10	Вычислить произведение элементов, небольших заданного F и расположенных в строках с нечетными номерами.
11	Вычислить сумму квадратов положительных элементов, расположенных в столбцах с номерами, кратными 3.
12	Определить количество отрицательных элементов, расположенных в строках с номерами, кратными 3.
13	Вычислить сумму элементов, принадлежащих промежутку [A,B) и расположенных в столбцах с четными номерами.
14	Вычислить произведение элементов, больших заданно го X и расположенных в столбцах с нечетными номерами.
15	Вычислить сумму квадратов элементов, меньших Y и расположенных в строках с нечетными номерами.

Задача 2

Таблица 6.2

Условие задачи по вариантам для задания 2

Вариант	Задание
1	В каждой нечетной по номеру строке матрицы найти минимальный элемент и вычислить произведение этих элементов.
2	В каждом столбце матрицы найти произведение положительных элементов и вычислить сумму этих произведений.
3	Определить количество столбцов матрицы, в которых больше трех положительных элементов.
4	Определить количество строк матрицы, в которых суммы всех элементов отрицательные.
5	В каждой строке матрицы найти самый левый отрицательный элемент и вычислить произведение этих элементов.
6	Определить количество строк матрицы, в которых нет положительных элементов.
7	Заменить в матрице элементы последней строки на произведение элементов соответствующих столбцов.
8	В каждом столбце матрицы найти минимальный элемент и вычислить сумму этих элементов.
9	В каждой строке матрицы найти произведение отрицательных элементов и вычислить сумму этих произведений.
10	Заменить в матрице элементы предпоследней строки на минимальные элементы соответствующих столбцов.
11	Определить количество строк матрицы, в которых произведение положительных элементов больше заданного B.
12	В каждой строке матрицы найти самый правый положительный элемент и вычислить сумму этих элементов.
13	Определить количество столбцов матрицы, в которых нет положительных элементов.
14	Заменить в матрице элементы последнего столбца на суммы элементов соответствующих строк.
15	В каждом столбце матрицы найти максимальный

Вариант	Задание
	элемент и вычислить произведение этих элементов.

Задание 3

Таблица 6.3

Условие задачи по вариантам для задания 3

Вариант	Задание
1	Поменять местами столбец с максимальным элементом матрицы с последним из столбцов, состоящих только из положительных элементов
2	Если максимальный элемент строки находится на первом месте в строке, заменить его суммой элементов этой строки
3	Поменять местами строку с максимальным элементом последнего столбца с первой из строк, состоящих только из отрицательных элементов
4	Поменять местами первый столбец, состоящий только из положительных элементов, со столбцом, в котором находится максимальный элемент матрицы
5	Поменять местами столбец с минимальным элементом первой строки с первым из столбцов, состоящих только из положительных элементов
6	Поменять местами строку с максимальным элементом главной диагонали с первой строкой, состоящей только из ненулевых элементов
7	Если строка начинается с отрицательного элемента, заменить элемент главной диагонали этой строки на максимальный элемент столбца, в котором находится этот элемент главной диагонали
8	Поменять местами строки с максимальным элементом последнего столбца и минимальным элементами матрицы
9	Заменить максимальный элемент матрицы на произведение элементов первого столбца, состоящего только из положительных элементов
10	Поменять местами первый столбец, в котором все элементы одинаковые, со столбцом, в котором находится минимальный элемент последней строки
11	Поменять местами строку с максимальным элементом

Вариант	Задание
	первого столбца с первой строкой, состоящей только из четных чисел
12	Заменить первый из столбцов, в которых два положительных элемента на столбец с минимальным элементом матрицы
13	Поменять местами столбец с минимальным элементом побочной диагонали с последним столбцом, состоящим только из четных элементов
14	Поменять местами первую строку матрицы, в которой произведение элементов отрицательное со строкой, в которой находится максимальный элемент предпоследнего столбца
15	Заменить первый столбец матрицы, в котором все элементы нечетные числа на последний из столбцов, в которых элементы образуют возрастающую последовательность

Контрольные вопросы к лабораторной работе:

1. Как выполнить описание двумерного массива? Как обратиться к элементу массива?
2. Опишите массив вещественных чисел, который может содержать не более 5 строк и 10 столбцов.
3. Какой визуальный компонент используется для ввода элементов массива? Какие свойства необходимо для него установить в Инспекторе Объектов?
4. Как выполнить ввод массива? Нарисуйте схему алгоритма и напишите фрагмент процедуры ввода.
5. Какой визуальный компонент используется для вывода элементов массива? Какие свойства необходимо для него установить в Инспекторе Объектов?
6. Как выполнить вывод массива? Нарисуйте схему алгоритма и напишите фрагмент процедуры вывода.

Лабораторная работа № 7
Обработка символьной информации

Цель работы: Получить навыки разработки и реализации алгоритмов обработки символьной информации в среде Delphi. Научиться использовать при разработке интерфейса управляющее и контекстное меню, страницы вкладок.

Теоретические сведения:

1. Представление символьной информации в памяти ПК

В памяти ПК можно хранить не только числовую, но и символьную информацию. Каждому символу, находящемуся на алфавитно-цифровой части клавиатуры, а также некоторым другим ставится в соответствие некоторый код – целое число, которым символ представлен в памяти ПК.

Для хранения одного символа отводится один байт. Максимальный код, который может храниться в 1 байте, $11111111_2 = 255_{10}$, поэтому всего может рассматриваться 256 символов с кодами от 0 до 255.

Набор кодов всех символов представлен в кодовой таблице. Способ кодировки может быть различным. В ОС DOS использовался код ASCII (American Standard for Information Interchange), в ОС Windows однобайтный код ANSI (по имени American National Standard Institute, предложившего этот код), двухбайтный код Unicode. Первая часть таблицы с кодами 0-127 постоянна для всех таблиц. Во второй части располагаются символы национальных алфавитов, причем эти символы находятся в них на разных местах.

В стандартных Windows Arial, Times New Roman русские буквы от А до Я имеют соответственно коды 192..223, строчные буквы от а до я коды 224..255, выделяются символы Ё (код 168) и ё (код 184). Символы цифр от 0 до 9 имеют коды соответственно 48..57. Символы латинских букв расположены подряд в кодовой таблице, сначала заглавные, затем строчные. Символы с кодами от 0 до 31 относятся к служебным символам. Если они используются в программе, то считаются служебными символами.

Символы можно сравнивать, при этом сравниваются их коды в кодовой таблице.

'a' > 'A', т.к. код 224>192

'0' < '9' <'$A_{лат}$' < '$Z_{лат}$' < 'a'< 'z' < '$A_{рус}$' < 'Я' < '$a_{рус}$' < 'я'

Так будет расположена упорядоченная символьная информация.

2. Описание переменных для хранения символьной информации

В Delphi для хранения символьной информации используются переменные двух типов: символьного и строкового.

Переменная символьного типа может содержать только один символ, в памяти занимает один байт, для ее описания используется тип char. Символьная константа – это один символ, заключенный в апострофы.

Переменная строкового типа может содержать последовательность символов. Строковая константа – это последовательность символов, заключенная в апострофы.

Для описания строк в Delphi есть четыре типа, основные два – это короткие и длинные строки. Для описания короткой строки используется тип String [N], для длинной - тип String.

Общим для всех типов является то, что строка рассматривается как одномерный массив символов, нумерация хранимых символов начинается с 1.

Для короткой строки (при компиляции, статически) в памяти выделяется N+1 байт, ее длина меняется от 0 до N, первый байт используется для хранения длины строки. Максимальная длина такой строки 255 символов. Нулевой символ используется для хранения длины строки.

Длинная строка может содержать от 0 до 2 Гб символов. Память для строки выделяется по мере необходимости во время выполнения программы (динамически) и ограничена только размером доступной программе памяти.

3. Операции над строковыми и символьными переменными.

3.1. Операция сцепления (+) применяется для объединения нескольких строк в одну.

3.2. Операции отношения

< , <= , > , >= , = , <>

Символы можно сравнивать, при этом сравниваются их коды в кодовой таблице.

'a' > 'A', т.к. код 224>192

Строки сравниваются посимвольно слева направо. Две строки считаются равными, если они имеют одинаковую длину и на соответствующих позициях находятся одинаковые символы (их коды совпадают). Та строка считается большей, у которой первый несовпадающий символ имеет больший код или строка имеет большую длину.

4. Стандартные функции для работы с символами

- **Chr (B) –** преобразование числового значения B:Byte в символ, код которого равен B.

Вместо функции можно использовать знак #. #65

Для вывода текста в ВК Label в несколько строк можно использовать символ перехода на новую строку (CR)с кодом 13.

Label1.Caption:=’первая часть строки’ + #13 + ‘вторая часть строки’

- **Ord (Ch)** – преобразование символа Ch в числовое значение типа Byte, соответствующее коду символа в кодовой таблице.
- **UpCase (Ch:Char):Char –** возвращает прописную букву для строчной латинской буквы, для остальных символов возвращает сам символ.

5. Стандартные функции для работы со строками

- **Length (S)** – определение текущей длины строки (числа символов в строке).
- **Copy (S,m,n)** – выделение части строки S длиной n символов, начиная с позиции m. Строка S не изменяется.
- **Pos (S1,S)** – определение номера позиции строки S, начиная с которого строка S1 первый раз входит в строку S. Если S не содержит S1, то функция возвращает ноль.
- **Concat (S1,S2,….SN)** – сцепление строк S1,S2,….SN в одну строку в указанном порядке. Если результат помещается в короткую строку, то его длина должна быть не больше 255.

6. Стандартные процедуры для работы со строками

- **Delete (S, m, n)** – удаление из строки S n символов, начиная с позиции m. Результат остается в строке S, ее длина уменьшается на n символов.
- **Insert (S1, S,m)** – вставка строки S1 в строку S, начиная с позиции m.
- **Str (X,S) –** преобразование числового значения X в строку S. X может иметь формат вывода, в соответствии с которым он преобразуется в S.
- **Val (S, X, Code)** – преобразование значения в строке S в число X целого или вещественного типа. Code равен нулю, если преобразование прошло без ошибок. В противном случае равен номеру первого ошибочного символа, X в этом случае не определено. Строка может содержать ведущие и ведомые пробелы. Разделителем целой и дробной части вещественного числа в строке может быть только «**.**» (точка).

Задание:

При разработке интерфейса использовать управляющее меню, содержащее пункты ***Расчет***, ***Справка***. Меню ***Расчет*** включает пункты ***Задание 1, Задание 2***. Меню ***Справка*** включает пункты ***О программе, Об авторе***.

По желанию можно добавить другие пункты меню и создать контекстное меню для одного из ВК.

Задание 1

В каждой из задач (таблица 7.1) предполагается, что задана строка текста, состоящая не более, чем из 255 символов. Программа должна:

1. вводить строку текста;
2. выполнять вычисления или преобразования в соответствии с условием задачи;
3. выводить:
   - исходную строку и ее длину;
   - результаты вычислений при обработке строки;
   - преобразованную строку и ее длину.

Исходные данные для отладки программы выбрать самостоятельно. Для ввода и вывода строки использовать компоненты TEdit или TMemo.

Таблица 7.1

Условие задачи по вариантам для задания 1

Вариант	Задание
1	Подсчитать общее количество символов '+ ' и '-' и заменить каждый символ ';' на ',' и '.' .
2	После каждого символа ',' вставить пробел и подсчитать количество букв 'A' и 'B' отдельно.
3	Заменить символ '*' на '++' и подсчитать общее количество букв 'F' и 'D'.
4	Подсчитать количество букв 'C' и 'D' отдельно и заменить каждую пару символов '**' на символ ' '.
5	После каждого символа '!' вставить символ 'I' и подсчитать общее количество цифр в строке.
6	Удалить каждую пару символов 'PQ' и подсчитать общее количество символов '.' и ',' в строке.
7	Подсчитать количество пар символов '+ -' и заменить каждый символ '*' на '/-'.
8	После каждой цифры вставить такую же цифру и подсчитать количество пар 'AC' в строке.
9	Удалить каждый символ 'A', стоящий после ',' и подсчитать количество пар 'BC'.
10	Подсчитать количество символов '.' , стоящих перед пробелом, и заменить каждую пару символов 'ST' на символ 'P'.
11	После каждого символа 'A' вставить пробел и подсчитать количество символов 'B', стоящих между знаками '+' и '-'.
12	Удалить каждый символ '?', стоящий после ';', и подсчитать общее количество символов 'o и 'O'.
13	Подсчитать количество символов '+', стоящих между 'A' и 'B', заменить каждый символ '0' на 'OO'.
14	В каждую пару символов 'AB' вставить символ '*', подсчитать, сколько раз в строке символ 'I' стоит перед '2'.
15	Вставить символ ';' после каждого символа 'A' и после

	каждого 'B', подсчитать, сколько раз символ 'C' встречается между символами '*' и '/'.

Задание 2

В каждой из задач (таблица 7.2) предполагается, что задана строка текста, состоящая из последовательности слов. Слова разделяются последовательностью пробелов. Программа должна:

1. вводить строку текста;
2. выполнять вычисления или преобразования в соответствии с условием задачи;
3. выводить:
   - исходную строку и ее длину;
   - результаты вычислений при обработке строки;
   - преобразованную строку и ее длину.

Исходные данные для отладки программы выбрать самостоятельно. Для ввода и вывода строки использовать компоненты TEdit или TMemo.

Таблица 7.2

Условие задачи по вариантам для задания 2

Вариант	Задание
1	После каждого слова поставить запятую. Подсчитать количество слов, в которых есть буква 'п'.
2	Подсчитать количество букв в третьем слове. Поменять местами первое и последнее слова.
3	Во втором слове после каждой буквы вставить пробел. Определить количество слов, которые заканчиваются на 'е'.
4	Перед первой буквой каждого слова вставить символ '*'. Определить количество слов, в которых нет ни одной буквы 'г'.
5	Для первого слова указать, сколько букв 'и' в нем содержится. Переставить первое слово в конец строки.
6	Определить количество слов, начинающихся с буквы 'А'. После каждой буквы предпоследнего слова вставить символ '*'.
7	Подсчитать количество букв во втором слове. Каждое слово заключить в кавычки.
8	Подсчитать количество слов, длина которых больше 5.

	Удалить пробелы, стоящие между первым и вторым словом.
9	Определить количество слов, в которых буква 'и' встречается хотя бы один раз. Поменять местами первое и второе слова.
10	Третье слово строки поставить после первого. Определить количество слов, в которых первая и последняя буквы совпадают.
11	Определить количество слов, вторая буква которых 'р'. Удалить последнюю букву в каждом слове.
12	Подсчитать количество букв в предпоследнем слове. В каждом слове поменять местами первую и последнюю буквы.
13	Перед каждой буквой третьего слова поставить '-'. Определить количество слов, после которых один пробел.
14	После последней буквы каждого слова вставить точку. Для пятого слова указать, сколько букв 'И' в нем содержится.
15	Удалить все пробелы из строки, кроме тех, которые стоят между первым и вторым словом. Определить количество слов, которые по длине меньше 3.

Контрольные вопросы к лабораторной работе:

1. Как представляется символьная информация в памяти компьютера?
2. Какие типы данных используются для хранения символьной информации в памяти компьютера?
3. Для чего служат кодовые таблицы? Приведите примеры кодовых таблиц.
4. Назовите стандартные функции для работы с символами. Приведите примеры их использования.
5. Назовите стандартные функции для работы со строками. Приведите примеры их использования.
6. Назовите стандартные процедуры для работы со строками. Приведите примеры их использования.

Лабораторная работа № 8
Программирование с использованием подпрограмм

Цель работы: изучить типы подпрограмм в Delphi, их структуру, возможности применения, обращение к подпрограммам; получить навыки использования процедур и функций при разработке приложений и размещения их в личном модуле**.**

Теоретические сведения:

Подпрограмма – это логически законченная и специальным образом оформленная часть программы, имеющая собственное имя.

Подпрограммы могут быть стандартными (sin(x), power(x,y)) или подпрограммами пользователя. Все подпрограммы пользователя должны быть описаны в разделе описаний. Подпрограммы пользователя могут быть результатом обработки событий.

В Delphi используются два вида подпрограмм:

- процедуры;
- функции.

Алгоритм, реализуемый подпрограммой, записывается один раз в ее описании, а выполняться может многократно при вызове ее по имени из различных мест других подпрограмм. Любая подпрограмма может содержать в своем разделе описаний описания других подпрограмм. В разделе операторов подпрограммы можно обратиться только к той подпрограмме, которая была описана ранее или к подпрограмме, которая описана в разделе описаний данной подпрограммы.

Любая подпрограмма может быть реализована в виде процедуры. Функции используются в том случае, если результатом работы подпрограммы является одно значение (сумма, максимум и т.д.).

Описание процедуры. Формальные параметры.

Procedure <имя> ([список формальных параметров]); **заголовок процедуры**

<раздел описаний>;
begin
<раздел операторов>
end;

имя - это имя процедуры, формируется по правилам образования идентификаторов и не совпадает с другими идентификаторами в программе.

Формальные параметры – это переменные, которые необходимы для связи процедуры с вызывающей программой. Они являются для процедуры исходными данными или результатом ее работы и используются в операторах тела процедуры. Список формальных параметров может отсутствовать, но заключающие список круглые скобки могут оставаться.

Формальные параметры перечисляются в списке с указанием их типов, разделяются символом "**;**", однотипные параметры можно объединять, а их имена разделять запятыми. Типы формальных параметров должны быть простыми (real, byte, integer) или должны быть указаны в виде определенного ранее идентификатора типа для массивов.

Обращение к процедуре (вызов процедуры). Фактические параметры.

Описание процедуры не вызывает выполнения операторов тела процедуры, а является только формой представления соответствующего алгоритма. Для того, чтобы выполнить процедуру, ее необходимо вызвать (обратиться к ней) из раздела операторов другой процедуры. Для этого используется оператор обращения к процедуре.

<имя процедуры> ([список фактических параметров]);

Список фактических параметров может содержать переменные, выражения, константы. Они разделяются запятыми.

Выполнение оператора обращения к процедуре: в модуле находится описание вызываемой процедуры, формальные параметры заменяются фактическими, и выполняется полученный код процедуры; затем происходит переход к оператору, следующему за оператором вызова процедуры.

Список фактических параметров при обращении к процедуре должен соответствовать списку формальных параметров в описании этой процедуры:

- количество фактических параметров должно совпадать с количеством формальных параметров;
- порядок следования фактических параметров должен совпадать с порядком следования формальных параметров.
-

Delphi проверяет совпадение типов соответствующих параметров, за смысловым соответствием должен следить программист. Никакого соответствия имен нет, они могут совпадать или различаться для соответствующих параметров.

Классы формальных параметров

Формальные параметры разделяются на входные и выходные. Входные являются для процедуры исходными данными, выходные – результатом ее работы.

Формальный параметр является **входным**, если соответствующий ему фактический параметр должен иметь определенное значение перед обращением к процедуре.

Формальный параметр является **выходным**, если соответствующий ему фактический параметр получает новое значение после обращения к процедуре. Перед выходными параметрами необходимо указывать служебное слово **var**.

Использование функций.

Функция – это частный случай подпрограммы, когда подпрограмма имеет единственный выходной параметр простого типа (не массив).

В результате работы функции определяется одно значение. Если подпрограмму можно реализовать в виде функции, то лучше использовать именно этот вид подпрограммы.

Описание функции. Формальные параметры.

Function <имя> ([список формальных параметров]): <тип>;
<тело функции>

Имя функции и список формальных параметров формируется по тем же правилам, что и у процедуры. Список формальных параметров не содержит, как правило, выходных параметров.

тип – это тип результата, обязательно простой.

В теле функции должен быть хотя бы один оператор, обычно выполняемый последним, который присваивает найденное значение имени функции или внутренней переменной result.

result := <выражение>;
или
<имя>:= <выражение>;

Обращение к функции. Фактические параметры.

Не существует специального оператора обращения к функции. Вызов функции происходит с помощью имени функции, которое может использоваться в выражениях аналогично стандартным функциям. После имени функции в круглых скобках должен быть указан список фактических параметров. Он формируется по тем же правилам синтаксиса и соответствия формальным параметрам, что и у процедуры.

Задание 1. Создание собственного модуля. Использование подпрограмм.

1.1. Создать модуль, не связанный с формой, и разместить в нем подпрограммы-функции для вычисления значения трех функций ***y=f(x)***. Вид функций выбрать тот же, что и в задании 1 лабораторной работы 3.

Формальные параметры (входные) подпрограмм-функций:

- значение аргумента ***x***;
- значения заданных констант ***a*** и ***b.***

Результат, возвращаемый подпрограммами-функциями:

- значение функции ***y***.

1.2. Включить в модуль подпрограмму-функцию для вычисления значения функции в зависимости от выбранного вида функции. Использовать подпрограммы-функции, созданные в задании 1.1.

Формальные параметры (входные) подпрограммы-функции:

- значение аргумента ***x***;
- значения заданных констант ***a*** и ***b***;
- номер выбранной функции.

Результат, возвращаемый подпрограммой-функцией:

- значение функции ***y***.

1.3. Подключить созданный модуль к программе в разделе **implementation.**

Вычислить значение функции ***y=f(x)*** для заданного аргумента ***x,*** ***a*** и ***b*** и выбранного вида функции. Для выбора функции использовать ВК TRadioGroup или TComboBox.

Сравнить полученный результат с результатом работы программы задания 1 лабораторной работы 3.

1.4. Включить в модуль процедуру работы со строкой, выполняющую те же функции, что и в задании 1 лабораторной работы 6.

Входные параметры процедуры:

- переменная строкового типа.

Выходные параметры процедуры:

- результат вычислений по заданию;
- преобразованная в соответствии с заданием строка.

1.5. Создать ВК Memo1, заполнить в нем несколько строк текстом на этапе разработки интерфейса. Выполнить обработку текста, используя созданную процедуру для каждой строки. Результат разместить в ВК Memo2.

Задание 2. Использование подпрограмм при обработке массивов

Выполнить задание 1, 2 или 3 лабораторной работы 5 в соответствии со своим вариантом. Все созданные подпрограммы разместить в модуле, связанном с главной формой.

Программа должна:

1. вводить массив, используя процедуру ввода и модальное окно для ввода размерности и элементов массива;
2. выполнять вычисления и преобразования массива в соответствии с условием задачи, используя необходимые процедуры и функции (не менее двух);
3. выводить полученный массив, используя процедуру вывода и форму с ВК TStringGrid или ВК TMemo.

Контрольные вопросы и задания к защите:

1. Для всех подпрограмм в выполненной работе указать их описание и обращение к ним, формальные и фактические параметры этих подпрограмм.
2. Как формируется список фактических параметров подпрограммы? В чем заключается его соответствие списку формальных параметров?
3. Какие формальные параметры называются входными, какие – выходными? В чем отличие в записи этих параметров в списке формальных параметров?
4. В каком случае подпрограмма может быть реализована в виде функции?
5. Как передает результат работы в вызывающую процедуру функция и как процедура?
6. Можно ли те действия, которые выполняет функция, реализовать в виде процедуры? Если нет, то почему? Если да, то написать описание такой процедуры и изменить обращение к функции на обращение к процедуре.
7. Можно ли те действия, которые выполняет процедура, реализовать в виде функции? Если нет, то почему? Если да, то написать описание такой функции и изменить обращение к процедуре на обращение к функции.
8. Написать описание процедуры и функции и обращение к ним для вычисления суммы, произведения, количества, среднего арифметического, определения максимума и минимума, формирования нового массива при обработке одномерных и двумерных массивов.

Литература

1. Delphi. Программирование на языке высокого уровня: Учебник для вузов / В.В. Фаронов. – СПб.: Питер, 2004. – 640 с.

2. Delphi7: Учебный курс / С.И. Бобровский. – СПб.: Питер, 2004. – 735с.

3. Основы программирования в Turbo Delphi / Н.Культин. – СПб.:BHV, 2007 – 384с.

4. Водополова Н.В., Мисюткин В.И., Чабуркина С.А. Основы алгоритмизации. Практическое пособие к лабораторным и контрольным работам по курсам «Информатика» и «Основы информатики и вычислительной техники». – Гомель, 2004, № 2963.

5. Коробейникова Е.В., Токочаков В.И. Работа в интегрированной среде DELPHI. Практическое пособие для студентов всех специальностей дневного и заочного отделений. – Гомель, 2004, № 2910.

6. Коробейникова Е.В., Токочаков В.И. Программирование в среде DELPHI. Практическое пособие по курсу «Информатика» для студентов всех специальностей. – Гомель, 2005, № 2986.

7. Токочаков В.И., Коробейникова Е.В. Практическое пособие по курсу «Информатика» для студентов всех специальностей «Программирование в среде Delphi». – Гомель, 2005, № 3534.

Содержание

Учебное издание

Чабуркина София Абелевна
Емельянченко Наталья Сергеевна

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

**Лабораторный практикум
по одноименной дисциплине
для слушателей специальности 1-40 01 73
«Программное обеспечение информационных систем»
заочной формы обучения**

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 17.0513.
Рег. № 67Е.
http://www.gstu.by