МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Кафедра «Вычислительные системы и информационная безопасность»

Основы программирования в среде PascalABC.NET

Практикум
по выполнению лабораторных работ
по дисциплине
«Программирование и алгоритмизация»

г. Ростов-на-Дону
2023 г.

Практикум по выполнению лабораторных работ по дисциплине «Программирование и алгоритмизация». ДГТУ, г. Ростов-на-Дону, 2023 г.
В практикуме изложены краткие теоретические сведения о конструкциях языка PascalABC, приводятся примеры программ и некоторые алгоритмы решения задач, необходимые для успешного выполнения лабораторных работ, индивидуальные задания и контрольные вопросы для самопроверки.
Предназначено для обучающихся по направлению: 15.03.04 «Автоматизация технологических процессов и производств»

ОГЛАВЛЕНИЕ

ЛАБОРАТОРНАЯ РАБОТА № 1. ЛИНЕЙНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ АЛГОРИТМЫ 4

ЛАБОРАТОРНАЯ РАБОТА № 2. ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ 13

ЛАБОРАТОРНАЯ РАБОТА № 3. ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ 21

ЛАБОРАТОРНАЯ РАБОТА № 4. ИСПОЛЬЗОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ И ПРОГРАММ ДЛЯ ВЫЧИСЛЕНИЯ СУММЫ ЧЛЕНОВ РЯДА 29

ЛАБОРАТОРНАЯ РАБОТА № 5. МАССИВЫ 36

ЛАБОРАТОРНАЯ РАБОТА № 6. СИМВОЛЫ И СТРОКИ 46

ЛАБОРАТОРНАЯ РАБОТА № 7. ФАЙЛЫ 51

ЛАБОРАТОРНАЯ РАБОТА № 8. ПРОЦЕДУРЫ И ФУНКЦИИ 57

ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИНФОРМАЦИОННЫХ РЕСУРСОВ 65

ЛАБОРАТОРНАЯ РАБОТА № 1.
ЛИНЕЙНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ АЛГОРИТМЫ

Цель работы: изучение структуры окна системы программирования, организации работы по набору, отладке, компиляции, запуска программы на исполнение; формирование совокупности знаний об операторах программ линейной структуры, умений анализировать исходные данные задачи, навыков разработки и программирования вычислительного процесса линейной структуры, навыков по отладке и тестированию программ, умений оформлять результаты своей деятельности в современном офисном пакете, умений объяснять синтаксис операторов ввода, вывода, присваивания.

Теоретический материал

Алгоритм – это любая конечная последовательность основных математических и логических действий, однозначно определяющих процесс преобразования исходных данных в искомые результаты решения задачи.

Для представления алгоритмов используются несколько способов:

– словесный (описание на естественном человеческом языке);
– с помощью одного из алгоритмических языков программирования;
– графический (на языке блок-схем) и др.

Рассмотрим способ составления алгоритма на языке блок-схем.

Блок-схема – это графическая интерпретация алгоритма, представляющая набор геометрических фигур, каждая из которых изображает какую-либо операцию или действие.

Правила построения алгоритмов

1. Блок-схема строится сверху вниз.
2. В любой блок-схеме имеется только один элемент, соответствующий началу алгоритма, и один элемент, соответствующий концу алгоритма.
3. Должен быть ходя бы один путь их начала блок-схемы любому элементу.
4. Должен быть хотя бы один путь от каждого элемента блок-схемы в конец блок-схемы.

Основные элементы, которые будут использованы при построении блок-схемы решения задачи представлены в табл. 1.1.

Символ	Описание символа
	Процесс – формирование новых значений, выполнение арифметических или логических операций или действий, результаты которых запоминаются в оперативной памяти ЭВМ
	Ветвление – проверка условия: выбор одного из двух направлений выполнения алгоритма в зависимости от некоторого условия
	Модификация – организация циклических конструкций (for…do)
	Предопределенный процесс – вычисление по подпрограмме, использование ранее созданных и отдельно описанных алгоритмов
	Начало или конец программы или вход и выход в подпрограммах
	Ввод или вывод данных
	Комментарий – пояснения, содержание подпрограмм

Алфавит языка

Алфавитом языка называют совокупность всех допустимых символов, которые можно использовать в этом языке.

Алфавит языка Pascal включает в себя следующие символы:

– прописные и строчные буквы латинского алфавита от A до z и символ подчеркивания (_);

– буквы русского алфавита (только для комментариев и текстовых сообщений);

– арабские цифры: 0 1 2 3 4 5 6 7 8 9;

– знаки математических операций: + – * / mod div;

– знаков операций отношений < > <= >= = <>;

– разделители: . , ; : ( ) [ ] { } ' ' ;

– спецсимволов – #, $, ^, &, @, := .

Прописные и строчные буквы в Pascal не различаются. Знаки «возведение в степень» и двойная кавычка в языке отсутствуют.

Идентификаторы

Идентификаторы – имена констант, переменных, меток, типов процедур, функций, модулей, программ и полей в записях. Идентификатор является комбинацией только букв, цифр и знака подчеркивания. Другие символы не допускаются. Должен начинаться с буквы или знака подчеркивания. Длина идентификатора не должна превышать 127символов.

Структура программы

Программа на языке Паскаль имеет следующую структуру:

```
Program <имя программы>;   - Заголовок
 Const      ...; {раздел констант}
 Type       ...; {раздел типов}
 Label      ...; {раздел меток}          Раздел
 Procedure <имя>;  {раздел
 Function <имя>;     подпрограмм};       описаний
 Var  ...; {раздел переменных};

Begin                          Блок
 {раздел операторов};          основных
End.                           операторов
```

Комментарии

Комментариями в Паскале полагается любой текст, заключенный в фигурные скобки { }. Комментарии могут размещаться в любом месте программы. Хороший стиль программирования рекомендует не экономить на комментариях.

Понятие типа

Тип идентификатора определяет:

1) способ его внутреннего представления и множество его допустимых значений;

2) действия, которые можно над ним выполнить.

Некоторые типы приведены в таблице 1.1.

Таблица 1.1 – Некоторые простые типы

Название типа	Обозначение	Допустимые значения	Формат	Примеры
Целочисленный	`Integer`	-32768 ... 32767	2 байта	-25, 1999
Вещественный	`Real`	$\pm 2{,}9 \cdot 10^{-39}$ ... $\pm 1{,}7 \cdot 10^{38}$	6 байтов	-0.32, -0.1 E -7
Символьный	`Char`	Любой из 256 символов расширенного кода ASCII	1 байт	‘A’, ‘!’, ‘щ’
Логический	`Boolean`	false, true	1 байт	False, true
Строковый	`String`	Последовательность символов из набора ASCII, расположенная на одной строке	(n+1) байт, где n – число символов в строке	‘Это строка символов’

Константы

Значения констант (за исключением типизированных констант) не могут меняться в процессе выполнения программы.

Задание констант идентификаторами осуществляется в разделе описаний программы после служебного слова const:

```
Const
<Имя константы 1> = <значение 1>;
<Имя константы 2> = <значение 2>;
...
<Имя константы N> = <значение N>;
```

Например:

```
Const a = 7;
   b1 = 3;
   massa = 15;
```

При этом тип константы автоматически определяется типом её значения в правой части.

Переменные

В отличие от констант переменные могут менять свои значения (в пределах объявленного типа).

Раздел описания переменных программы начинается с зарезервированного слова **var** с последующим списком конкретных переменных:

```
Var
<Список переменных 1>: <тип 1>;
<Список переменных 2>: <тип 2>;
     ....
<Список переменных N>: <тип N>;
```

Например:

```
var k,z:integer; sum,pr:real;
```

Каждый список состоит из перечисления имен переменных одинакового типа.

Рассмотрим следующий пример программы:

```
PROGRAM SURFACE;
const R1 = 1.0;
   R2 = 2.0;
var S:real;
BEGIN
S:=Pi*R1*R1;
writeln('S1=',S);
S:=Pi*R2*R2;
writeln('S2=',S);
END.
```

В программе описывается и используется переменная S вещественного типа, которая вначале принимает значение площади круга радиуса R1, а затем – радиуса R2.

Стандартные функции

В языке Паскаль существует ряд стандартных (заранее разработанных и встроенных в язык) процедур и функций. В данном разделе мы рассмотрим лишь наиболее часто используемые стандартные функции для величин вещественного и целого типов (таблица 1.2.).

Таблица 1.2 – Некоторые стандартные математические функции

Функция	Назначение	Тип результата	Форма записи	Результат
`Abs(x)`	Абсолютное значение x (модуль аргумента)	Совпадает с типом x	`Abs(-5)`	5
`Cos(x)`	Косинус x	Вещественный	`Cos(0)`	1
`Exp(x)`	e^x	Вещественный	`Exp(2)`	7.389
`Frac(x)`	Выделяет дробную часть числа x	Вещественный	`Frac(3.9)`	0.9
`Int(x)`	Выделяет целую часть числа x (ближайшее меньшее целое)	Вещественный	`Int(3.9)` `Int(-1.2)`	3 -2
`Ln(x)`	Натуральный логарифм	Вещественный	`Ln(2)`	0.693
`Pi`	Число π	Вещественный	`Pi`	3,14159
`Round(x)`	Округление вещественного числа x до ближайшего целого	Целый	`Round(3.9)` `Round(-1.2)`	4 -1
`Random`	Генератор случайных чисел `0 ≤ x < 1`	Вещественный	`Random`	0.2
`Random(N)`	Генератор случайных чисел `0 ≤ x < N`	Целый	`Random(10)`	7
`Sin(x)`	Синус x	Вещественный	`Sin(Pi/2)`	1
`Sqr(x)`	Квадрат x	Совпадает с типом x	`Sqr(5)`	25
`Sqrt(x)`	Корень x	Вещественный	`Sqrt(36)`	6

ОПЕРАЦИИ

Математические операции

Математические (арифметические) операции применимы к величинам целых и вещественных типов и приведены в таблице 1.3.

Таблица 1.3 – Арифметические операции для величин целого и вещественного типов

Знак арифметической операции	Наименование	Символ языка Паскаль	Форма записи	Результат
·	Умножение	*	2*3	6
:	Деление	/	8/4	2
+	Сложение	+	2+3	5
-	Вычитание	-	15-7	8
	Целая часть от деления	`Div`	5 div 2 7 div 4	2 1
	Остаток от деления	`Mod`	5 mod 2 7 mod 4	1 3

<u>Пример</u>: Дано трехзначное целое число `abc`, где `a, b, c` – любые цифры в диапазоне 0…9. Найти эти цифры. Решение получим с помощью следующих операторов:

```
a:= abc div 100;
c:= abc mod 10;
b:= abc div 10 mod 10;
```

Следует отметить, что в Паскале отсутствует операция возведения в степень. Для целых степеней её можно заменить умножением, например, вместо X^3 заменить `X*X*X`. Для нецелых степеней можно использовать известное тождество: $a^b = e^{b \cdot \ln a}$

Для типов `char` и `string` определена единственная математическая операция “+” или сцепление. Например, результатом операции `‘My name is ’ + ‘Ivan’` будет строка `‘My name is Ivan’`.

Операции отношения (сравнения)

Данная операция применима к данным типа `real, integer, char, string, boolean`. В операциях участвуют однотипные объекты (исключение составляют типы real и integer); результат сравнения имеет логический тип.

Операции отношения обозначаются:

=	-	равно;	<=	-	меньше или равно;
<>	-	не равно;	>=	-	больше или равно;
<	-	меньше;	>	-	больше;

Тип Real и Integer

Сравнение данных типов соответствует математическим правилам.

Тип Boolean

Для логического типа принято, что `true > false`.

Тип Char

Сравнение символов производится путем сравнивания их ASCII-кодов (целых чисел в диапазоне 0…255).

Тип String

Символы строк сравниваются попарно с учётом их внутреннего кода. Например, операция `‘Яша’ > ‘Яков’ возвратит значение true; ‘Вася’ < ‘Василий’ – false.`

Приоритет операций

Круглые скобки

Выражения, заключённые в круглые скобки, вычисляются в первую очередь. Количество круглых скобок в выражении может быть любым, причем число открывающих скобок должно быть равно числу закрывающих скобок. Выражения в скобках могут быть вложенными друг в друга и не должны пересекаться.

Например, математическое выражение: $\frac{c}{(a-b)^2}$

на Паскале можно записать в виде одной строки: `c/((a-b)*(a-b))`.

Порядок вычисления выражений

Вычисления значений выражений выполняется в определённом порядке. В первую очередь вычисляются выражения, заключённые в круглые скобки. В случае вложенных скобок сначала вычисляются внутренние выражения, затем – внешние. Приоритеты операций, выполняемых при вычислении выражений, приведены в таблице 1.4.

Таблица 1.4 – Приоритеты операций

Приоритет	Операции
1	Вычисление значений функций
2	`*, /, div, mod`
3	`+, –`
4	`=, <>, <, >, <=, >=`

ОПЕРАТОРЫ ПРОГРАММ ЛИНЕЙНОЙ СТРУКТУРЫ

Программа линейной структуры — программа, реализующая алгоритм структуры «Следование» (рисунок. 1.1), в котором блоки выполняются в указанном порядке, последовательно друг за другом. Для организации линейной программы используют операторы ввода, вывода данных и оператор присваивания.

Самым простым действием над переменной является занесение в нее величины соответствующего типа – присвоение переменной конкретного значения.

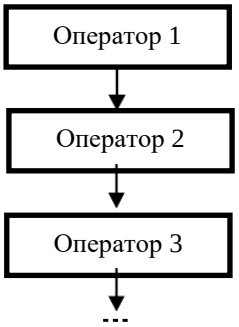


Рисунок 1.1 – Структура «Следование»

Оператор присваивания

Оператор присваивания имеет следующую структуру:

`Переменная:=выражение;`

Он работает следующим образом: вначале вычисляется выражение, а затем его значение присваивается указанной переменной. Рассмотрим, например, оператор

`x:=x + 1;`

Данный оператор нельзя рассматривать как математическое уравнение. Его следует трактовать так: к значению переменной `x` следует прибавить `1`, а результат присвоить этой же переменной `x`.

В операторах присваивания следует соблюдать правило соответствия типов: тип выражения в правой части должен совпадать с типом переменной в левой части. Единственное исключение из данного правила: переменной типа `real` можно присваивать целочисленный результат.

Операторы ввода-вывода

`Read(x1, x2, ..., xn);` осуществляет ввод `x1, x2, ..., xn` с клавиатуры

`Readln(x1, x2, ..., xn);` осуществляет ввод `x1, x2, ..., xn` с клавиатуры и после ввода значения последней переменной обеспечивает переход к началу новой строки.

`Readln;` обеспечивает пропуск одной строки и переход к началу новой строки.

Элементами списка ввода могут быть имена переменных, которые должны быть заполнены значениями, введенными с клавиатуры.

Выполнение операторов ввода происходит так: ход программы приостанавливается, на экран выводится курсор, пользователь с клавиатуры вводит необходимые значения в том порядке, в котором они требуются списком ввода, нажимает Enter (в случае использования read данные при вводе разделяются пробелами). После этого набранные данные попадают в соответствующие им переменные и выполнение программы продолжается.

Операторы вывода

`Write(x1, x2, ..., xn);` выполняет вывод значений, соответствующих перечисленным именам, размещая выводимые значения в одной строке.

`Writeln(x1, x2, ..., xn);` выполняет вывод значений `x1, x2, ..., xn`, соответствующих перечисленным именам, и после вывода последнего значения осуществляет переход к новой строке.

`Writeln;` обеспечивает пропуск одной строки и переход к началу новой строки.

Элементами списка вывода могут являться имена переменных, выражения, константы. Прежде чем вывести на экран, компьютер вычислит значения выражений.

Рассмотрим для примера следующий оператор:

```
Write ('Вывод результатов: ', 'a= ', a, 'b= ', b);
```

Здесь список вывода состоит из пяти элементов:

–строковой константы `'Вывод результатов:'`;

–строковой константы `'a= '`;

–значения переменной `a`;

–строковой константы `'b= '`;

–значения переменной `b`.

В процедурах `Write` или `WriteLn` можно использовать форматированный вывод: любой параметр из списка вывода может иметь форму (квадратные скобки показывают, что заключённые в них параметры могут отсутствовать):

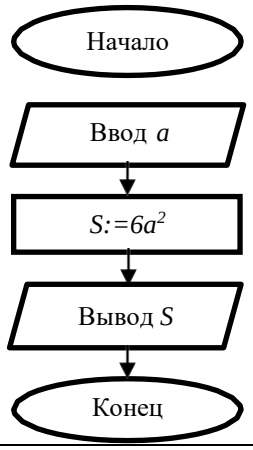
`Par[:<мин. ширина поля>[:<количество десятич. знаков>]],`

где `<мин. ширина поля>` – минимальный размер поля (в символах), в которое будет записываться значение `Par`. Если размер `Par` меньше указанной величины, оно будет дополнено слева пробелами, если больше, то данная установка игнорируется. Параметр `<количество десятич. знаков>` – задаёт количество десятичных знаков в дробной части выводимого числа. Он может использоваться только совместно с подпараметром `<мин. ширина поля>`.

Например, оператор Write (x:7:2) указывает, что на экран выводится содержимое вещественной переменной x. Под число отводится 7 позиций с двумя знаками после запятой. Если переменная x имеет значение 3.14159, то

```
Write(x:7:5) → 3.15159;
Write(x:7:3) → 3.142;
Write(x:7:1) → 3.1.
```

Пример 1. Используя линейную структуру, составить блок-схему и программу для нахождения площади поверхности куба со стороной a.




```
Program Lab_1;
Const a = 4;
Var s:integer;
Begin
 S:=6*a*a;
 Writeln ('Площадь = ', S);
End.
```

Блок-схема алгоритма	Листинг программы

Упражнения

Используя линейную структуру, составить блок-схему к каждому упражнению, затем составить программу, реализующую данный алгоритм.

1. Используя константы, рассчитайте длину окружности, площадь круга и объем шара по заданному радиусу.

2. Составьте программу расчета стоимости товаров и услуг в условиях инфляции по формуле:

$$S = S_0 e^{\frac{inf \cdot t}{100}},$$

где S_0 – начальная стоимость товаров и услуг;

inf – месячный уровень инфляции, %;

t – время, месяцы.

Посчитайте, какая будет цена у Ваших любимых товаров через год при уровне инфляции 0,5 и 1,5 % в месяц.

3. Составьте программу, которая вводила бы Вашу фамилию, а выводила ее же, но с добавкой «Господин», «Товарищ», «Сэр» и т.п. по Вашему усмотрению.

4. Проверьте, что **$\sin^2(x) + \cos^2(x) = 1$** при любых задаваемых значениях x.

5. Заданы действительные числа `a, b, c`. Написать программу расчёта выражений:

а) $y = \dfrac{e^{2a}\sqrt{1+a^2}}{(b+c)^3}$; б) $y = \dfrac{1+\cos(a-2)}{\dfrac{b^4}{2} + \sin^2 c}$.

6. Для целого четырехзначного числа `m` найти составляющие его цифры. Определить число, полученное выписыванием в обратном порядке цифр числа *m*.

7. Написать программу расчета расстояния между двумя точками с координатами (x_1, y_1) и (x_2, y_2), используя формулу: $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

8. Рассчитать значения C по формуле $C = \dfrac{0.427}{(\ln \mathrm{Re} - 0.407)^{2.64}} - \dfrac{d}{\mathrm{Re}}$ при

а) $\mathrm{Re} = 3 \cdot 10^5$, d = 1060 и б) $\mathrm{Re} = 4 \cdot 10^5$, d = 1400.

9. Если корни квадратного уравнения $ax^2 + bx + c = 0$ вещественны, то переменной Mark присвоить значение true, в противном случае – false. При этом условный оператор не использовать.

Варианты самостоятельных заданий

1. $Y = \ln|x| \cdot \left(x \cdot arctg(ax) - \sqrt{|x^3|} + \ln(|bx|+3)\right);$
$F = \sqrt{(ax^2+bx)} - e^{-ax}; \quad a = 2{,}5; \quad b = 0{,}5; \quad при\ x = 8\ и\ x = 1.$

2. $Y = x^5 \cdot \left(x \cdot arctg(a+x) - \sqrt{|x-a|} + \ln(|x|+1)^2\right);$
$F = \sqrt{(ax^2+bx)} \cdot e^{-ax} + \ln(|x|+1); \quad a = 0{,}5; b = 1{,}5; при\ x = 9{,}5\ и\ x = 1{,}5.$

3. $Y = (x+a)^5 \cdot \left(arctg\, x - \sqrt{(x-a)^2|} + \ln^2(x^2+1)\right);$
$F = \sin x - e^{-ax} + \ln(|x-a|+2); \quad a = 2{,}5; \quad b = 0{,}5; \quad при\ x = 8{,}5\ и\ x = 0{,}5.$

4. $Y = x^5\left(arctg(a+x)^2 - \sqrt{|(x-a)^2|}\right);$
$F = \sqrt{(ax^2+bx)} - e^{-ax} + ax \cdot \ln(x+a); \quad a = 4; \quad b = 0{,}25; \quad при\ x = 8\ и\ x = 1.$

5. $Y = \left(x^5 \cdot Cos(a+x) - \sqrt[3]{|(2x-a)|} + \ln\left((x-2)^2+2\right)\right);$
$F = \sqrt{(ax^2+bx)} \cdot e^{-ax} + \ln(|a-x|+3); \quad a = 4; b = 2; при\ x = 3\ и\ x = 1.$

6. $Y = \ln|x + a^2|(ax \cdot arctg\, x - \sqrt{|x|} + \ln(|b+x|+1));$
$F = \sqrt{(ax^2+x)} + e^{-ax}; \quad a = 1{,}5; b = 0{,}5; \quad при\ x = 4\ и\ x = 1.$

7. $Y = ax^5(ax \cdot arctg(a+x) - \sqrt{|(x-a)|} + \ln(|x|+1)^2);$
$F = x\sqrt{(ax^2+bx)} + e^{-x} + \ln(|x-a|+1); \quad a = 3{,}5; b = 1{,}5; \quad при\ x = 5\ и\ x = 1.$

8. $Y = \ln|x-b| \cdot \left(ax \cdot arctg(a+x) - \sqrt{|x^3-1|} + \ln(|b+x|+1)\right);$
$F = 1 + \sqrt{(ax^2+b)} - ae^{-ax}; \quad a = 1{,}5; \quad b = 0{,}5; \quad при\ x = 4\ и\ x = 1.$

9. $Y = \left((x-a)\cdot arcctg(a+x) - \sqrt[3]{|(x+a)|} + x\cdot \ln(a+x)\right);$

$F = \sqrt{(ax^2+x)} + ae^{-ax} + \ln(|x|-3);\quad a=2; b=0{,}5; при\ x=3\ и\ x=1{,}5.$

10. $Y = \left(x\cdot Sin^2(a+x)^3 - \sqrt[3]{(x-a)} + \ln((x-b)+1)\right);$

$F = \sqrt{(ax^2+x)} + a/e^{-ax} + \ln(1-x);\quad a=2{,}5; b=0{,}5; при\ x=0{,}5\ и\ x=1.$

11. $Y = \ln(bx-a^2)(ax\cdot arctg\,x - \sqrt{|x|} + \ln(|b+x|+1));$

$F = \sqrt{(ax^2+x-4)} + e^{-ax};\quad a=1{,}5; b=0{,}5;\quad при\ x=2\ и\ x=1.$

12. $Y = (x-a)^3(arctg(b+x) - \sqrt{(x-a)^2} + \ln(ax^2+1));$

$F = Sin(ax) - e^{-x} + \ln(x-a+2);\quad a=5; b=2;\quad при\ x=5\ и\ x=1.$

Контрольные вопросы

1. Что такое «алгоритм»? Каковы его свойства? Перечислите способы записи алгоритма.
2. Из каких элементов состоит блок-схема? Для чего предназначен каждый блок?
3. Какова структура программы, написанной на языке Паскаль?
4. Что такое «оператор»? Из каких операторов состоит линейная программа?
5. Для чего нужны комментарии в программе?
6. Запишите синтаксис оператора ввода, оператора вывода, оператора присваивания.
7. Для чего нужен раздел объявления переменных? Как объявляются переменные?
8. Что такое «тип данных»? Для чего нужен тип данных? Какие типы данных есть в Паскале?
9. Как записываются арифметические операции в языке Паскаль?
10. Что такое «переменная», «идентификатор»? Каковы правила записи идентификаторов?
11. Что значит «объявить переменную»? Объявите переменную x целочисленного типа данных.
12. Какие встроенные математические функции есть в Паскале?
13. Перечислите правила набора математических выражений на языке Паскаль.
14. Как оформляется оператор вывода на экран?
15. Что можно указывать в качестве элементов списка вывода? Какой символ используется для разделения элементов списка вывода?
16. Как должен быть оформлен оператор вывода, чтобы информация выводилась на экран с новой строки?
17. Как оформляется оператор ввода? Как работает оператор ввода?
18. Почему перед оператором ввода в программе целесообразно записывать оператор вывода?
19. Укажите приоритет выполнения арифметических операций при расчете значения выражения.
20. Как оформляется оператор присваивания? Как он работает?
21. Какого типа данных могут быть переменные s=1,7 и m=8?
22. Для чего нужен формат вывода данных? Как он задается?
23. Для чего предназначены операторы div и mod?
24. Запишите встроенные математические функции языка Паскаль.

ЛАБОРАТОРНАЯ РАБОТА № 2.
ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Цель работы: формирование совокупности знаний об операторах программ разветвляющейся структуры; овладение практическими навыками разработки, программирования вычислительного процесса разветвляющейся структуры, умений анализировать исходные данные задачи, умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их, умений объяснять синтаксис операторов ветвления; получение дальнейших навыков по отладке и тестированию программы.

Теоретический материал

Логические операции

Логические операции применяются к величинам логического типа. В языке Паскаль определены следующие четыре логические операции (таблица 2.1).

Таблица 2.1 – Логические операции

Операция	Наименование
`and`	Логическое И
`or`	Логическое ИЛИ
`xor`	Логическое исключающее ИЛИ
`not`	Логическое НЕ

Результат логической операции тоже имеет логический тип (таблица 2.2).

Таблица 2.2 – Логические операции

X	Y	not X	X and Y	X or Y	X xor y
False	-	True	-	-	-
True	-	False	-	-	-
False	False	-	False	False	False
False	True	-	False	True	True
True	False	-	False	True	True
True	True	-	True	True	False

Порядок вычисления логических операций

Приоритет	Операции
1	`not`
2	`and`
3	`or, xor`
4	`=, <>, <, >, <=, >=`

<u>Пример:</u> Пусть a = false, b = false, c = false, d = true. Тогда выражения, приведённые ниже, дадут следующий результат:

(a = b) and (c < d) → true; (a = b) and not (c < d) → false.

1 3 2 1 4 3 2

Здесь цифрами обозначен общий порядок вычисления выражений.

ОПЕРАТОРЫ ПРОГРАММ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Условный оператор

Условный оператор предусматривает выбор одной из нескольких последовательностей операторов (ветвей) в зависимости от некоторых условий (рисунки 2.1 и 2.2).

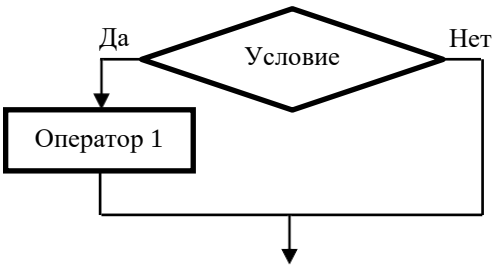


Рисунок 2.1 – Структура «Ветвление» «Если – то»

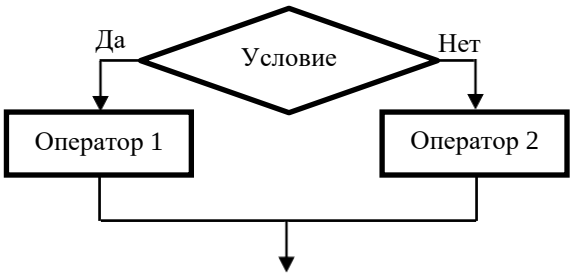


Рисунок 2.2 – Структура «Ветвление» «Если – то – иначе»

Синтаксис условного оператора:

«Если – то» **`if <условие> then <оператор>;`**

«Если – то – иначе» **`If <условие> then <оператор 1> else <оператор 2>;`**

Условный оператор работает следующим образом:

1) вычисляется логическое выражение;

2) если оно имеет значение `true`, то выполняется `<оператор 1>`;

3) если имеет значение `false`, то выполняется `<оператор 2>`.

Составной оператор

Составной оператор – это последовательность произвольных операторов, заключенных в операторные скобки `begin` – `end`:

```
begin
<оператор 1>;
<оператор 2>;
…
<оператор n>;
end;
```

Отдельные операторы внутри составного оператора отделяются друг от друга точкой с запятой.

Составные операторы используются в тех случаях, когда нужно выполнить несколько действий, а по правилам программирования можно использовать только один оператор.

Использование составного оператора (пример):

```
If условие then
 begin
     оператор;
     if выражение then
             оператор
             else
             оператор;
     оператор;
   end
 else
 begin
     оператор;
     оператор;
  end;
```

Пример 1. Написать условный оператор расчета выражения:

$$y = \begin{cases} 1 - \dfrac{x^2}{6}, |x| < 0.1 \\ \dfrac{\sin x}{x}, |x| \geq 0.1 \end{cases}$$

Решение :

```
if abs(x)<0.1 then y:=1-x*x/6 else y:=sin(x)/x;
```

Пример 2. написать программу расчета корней квадратного уравнения $ax^2 + bx + c = 0$, $a \neq 0$.

```
Program roots;{расчет корней квадратного уравнения}
var a,b,c,x1,x2:real;
BEGIN
 readln(a,b,c);{ввод коэффициентов уравнения}
 d:=b*b-4*a*c;{расчет дискриминанта}
 if d<0 then
          {случай мнимых корней}
          writeln('Действительных корней нет!')
      else {случай действительных корней}
          begin
          x1:=(-b-sqrt(d))/(2*a);
          x2:=(-b+sqrt(d))/(2*a);
          writeln('x1=',x1,',x2=',x2);
          end;
END.
```

Оператор выбора Case…of

Оператор выбора обеспечивает выполнение одного оператора из нескольких возможных. Ключ выбора должен иметь порядковый тип.

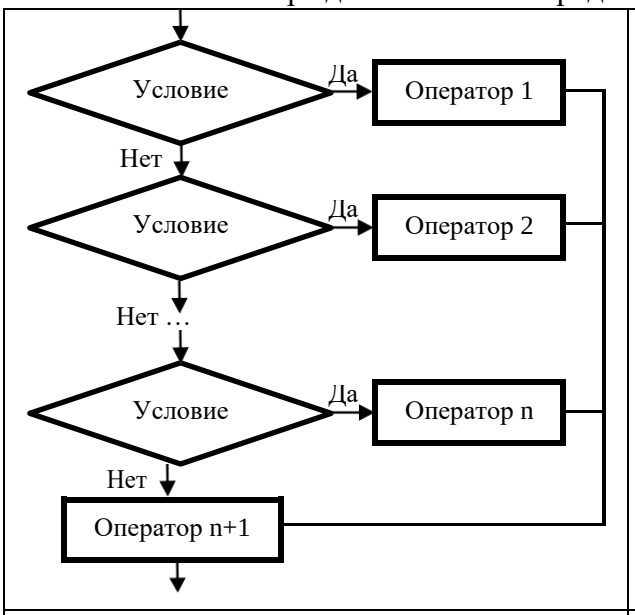


Рисунок 2.3 – Структура «Ветвление» – «Выбор»

```
Case <ключ выбора> of
     C1:<оператор 1>;
     C2:<оператор 2>;
          …
     Cn:<оператор n>;
Else        <оператор>
End;
```

где `<ключ выбора>` – выражение порядкового типа (из рассмотренных нами типов к нему относятся типы `integer`, `char`, `boolean`);
`C1,C2,…, Cn` – константы выбора того же типа, что и `<ключ выбора>`;
`<оператор 1>`, `<оператор 2>`, …, `<оператор n>` – любые операторы

Синтаксис оператора выбора

Оператор выбора работает следующим образом:

1) сначала вычисляется `<ключ выбора>`;

2) если это значение равно какой-либо константе выбора `Cn`, то выполняется соответствующий ей оператор;

3) если такой константы нет, то выполняется оператор, следующий за `Else`.

Примечания: 1) часть конструкции `Else <оператор>` можно опускать;

2) в списке выбора можно использовать несколько констант выбора, разделённых запятой.

Пример 3. ввести с клавиатуры целое число из диапазона 1…7 и напечатать соответствующий ему день недели.

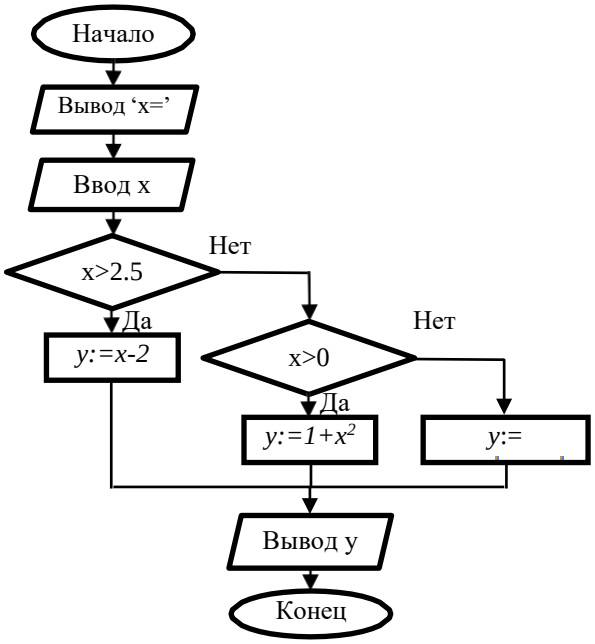
Решение: задачи такого типа удобно решать с помощью оператора выбора. Возможный вариант программы можно записать так:

```
Program Task3;
var i:integer;
BEGIN
 write('Введите число:');
 readln(i);
 case i of
     1:writeln('Понедельник');
     2:writeln('Вторник');
     3:writeln('Среда');
     4:writeln('Четверг');
     5:writeln('Пятница');
     6:writeln('Суббота');
     7:writeln('Воскресенье');
 else writeln('Такого дня недели нет!');
 end;
END.
```

Пример 4. Найти значение функции y.

$$y = \begin{cases} x - 2 & x > 2.5 \\ 1 + x^2 & 0 \le x \le 2.5 \\ x \cdot \ln|\cos x| & x < 0 \end{cases}$$

Как видно из задания, функция вычисляется на трех диапазонах аргумента x по трем различным формулам. Составим блок-схему алгоритма решения данной задачи. Используем вложенный условный оператор.



Текст программы

```
Program Lab_2_variant_03;
Var x,y: real;
Begin
Writeln ('Программа вводит значение аргумента X
и вычисляет значение функции У');
Writeln ('    | x-2       x>2.5');
Writeln ('Y = | 1+x^2     0≤x≤2.5');
Writeln ('    | x*ln|cos(x)|    x<0');
Writeln ('введите x=');
Readln(x);
If x>2.5 then
y:=x-2
  else
     if x>=0 then
             Y:=1+x*x
           else
             Y:=x*ln(abs(cos(x)));
Writeln ('Y= ', y:7:3);
End.
```

Упражнения

1. Даны три вещественных числа `a, b, c`. Напишите программу нахождения минимального и максимального значений этих чисел.

2. Даны вещественные переменные `x, y`. Выяснить принадлежит ли точка (`x, y`) кругу радиуса `r` с центром в начале координат.

3. Заданы три положительных числа `a, b, c`, выражающие длины отрезков. Требуется определить, можно ли из этих отрезков построить треугольник.

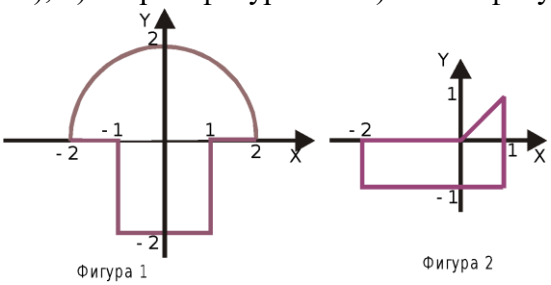
Примечание: из трёх отрезков можно построить треугольник тогда и только тогда, когда сумма длин любых двух отрезков больше третьего, т.е. `(a+b)>c` и `(a+c)>b` и `(b+c)>a`.

4. Если уравнение $ax^2+bx+c=0$ ($a \neq 0$) имеет вещественные корни, то логической переменной L присвоить значение `true`, а переменным `x1, x2` – сами корни, в противном случае L присвоить `false`, а `x1, x2` не вычислять.

5. Ввести с клавиатуры число из интервала 1…12 и, используя оператор `case`, напечатать соответствующий ему месяц года. Если этот месяц – июль, дополнительно спросить: "Где Вы желаете провести отпуск? "

6. Дано целое число в диапазоне от 1 до 5. Вывести строку – словесное описание соответствующей оценки (1 – «плохо», 2 – «неудовлетворительно», 3 – «удовлетворительно», 4 – «хорошо», 5 – «отлично»).

6. Задана точка с координатами (x,y). Выяснить, принадлежит ли данная точка: а) первой фигуре (см. рисунок), б) второй фигуре или в) обеим фигурам одновременно?

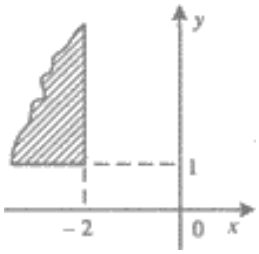
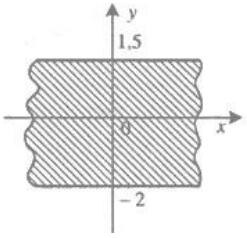
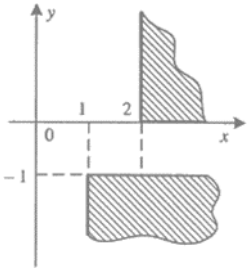
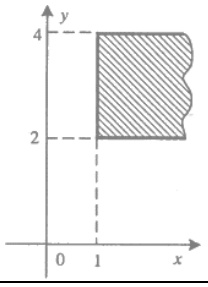
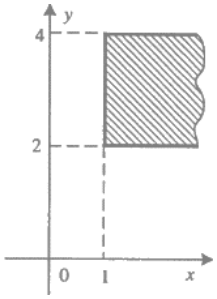
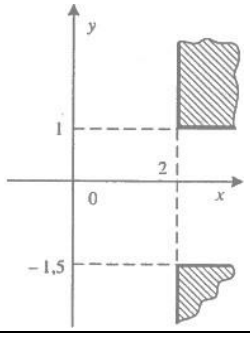
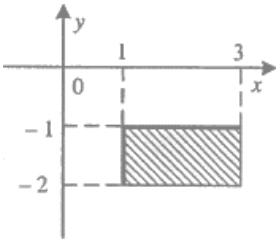
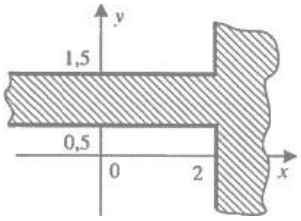
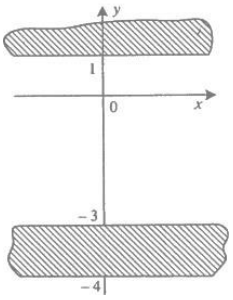
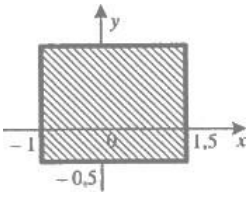
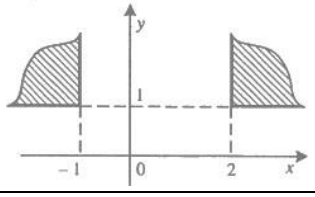
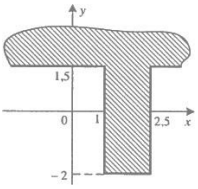


Варианты самостоятельных заданий

Найти значение функции y

№ вар.	Функция	№ вар.	Функция
1.	$y = \begin{cases} x - 2 & x > 2.5 \\ 1 + x^2 & 0 \le x \le 2.5 \\ x \ln\lvert\cos(x)\rvert & x < 0 \end{cases}$	2.	$y = \begin{cases} \sin(2.3x - 1) & x > 2.5 \\ 1 - 3\ln\lvert 1 - x\rvert & 0 <= x <= 2.5 \\ \dfrac{x^2}{2 - x} & x < 0 \end{cases}$
3.	$y = \begin{cases} \sqrt{(tg(x^2 - 1))} & x > 1 \\ -2x & 0 \le x \le 1 \\ e^{\cos(x)} & x < 0 \end{cases}$	4.	$y = \begin{cases} x^2 - 3 + 2.5x^2 & x > 12.5 \\ e^x + 5 + \cos(0.001x) & 0 \le x \le 12.5 \\ x^2 & x < 0 \end{cases}$
5.	$y = \begin{cases} 1 + \sqrt{\lvert\cos(x)\rvert} & x > 1 \\ x + 1 & -0.5 \le x \le 1 \\ 1 - x^2 & x < -0.5 \end{cases}$	6.	$y = \begin{cases} 2.5 \cdot x^3 + 6 \cdot x^2 - 30 & x > 1.5 \\ x + 1 & 0 \le x \le 1.5 \\ x & x < 0 \end{cases}$
7.	$y = \begin{cases} 1 + x & x > 14.5 \\ e^{-x} & 3 \le x \le 14.5 \\ \cos(x) & x < 3 \end{cases}$	8.	$y = \begin{cases} \ln\lvert 1 + x\rvert & x > 3.8 \\ e^{-x} & 2.8 \le x \le 3.8 \\ \cos(x) & x < 2.8 \end{cases}$
9.	$y = \begin{cases} 1 + \sqrt{\cos(x)} & x > 4 \\ x + 1 & 0 \le x \le 4 \\ 1 - x^2 & x < 0 \end{cases}$	10.	$y = \begin{cases} e^{-(x+8)} & x > 3.61 \\ 1 & 0 \le x \le 3.61 \\ \dfrac{x}{5} & x < 0 \end{cases}$
11.	$y = \begin{cases} x & x > 1.5 \\ 2x^2\sqrt{\lvert\cos(2x)\rvert} & 0 \le x \le 1.5 \\ e^{-\cos(3x)} & x < 0 \end{cases}$	12.	$y = \begin{cases} 1 - \sqrt{\cos(2x)} & x > 2.5 \\ x^2 - x & 1 \le x \le 2.5 \\ 1 + x^2 & x < 1 \end{cases}$

1. Записать условие, которое является истинным, когда точка с координатами x, y попадает в заштрихованные участки плоскости:

Вариант		Вариант	
1		7	
2		8	
3		9	
4		10	
5		11	
6		12	

Контрольные вопросы

1. Как записываются логические операции в языке Паскаль?
2. Какие виды условных операторов Вы знаете?

3. В каких случаях в программе используется полный условный оператор? Как он оформляется? Как он работает (что происходит при его выполнении)? Нарисуйте графическую схему выполнения.

4. В каких случаях в программе используется неполный условный оператор? Как он оформляется? Как он работает (что происходит при его выполнении)? Нарисуйте графическую схему выполнения.

5. В каких случаях в программе используется вложенный условный оператор? Как он оформляется?

6. В каких случаях в программе используется оператор выбора? Как он оформляется? Как он работает (что происходит при его выполнении)? Нарисуйте графическую схему выполнения.

7. В каком случае необходимо заключать операторы в операторные скобки begin … end?

ЛАБОРАТОРНАЯ РАБОТА № 3.
ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель работы: формирование совокупности знаний об операторах программ с использованием цикла; овладение практическими навыками разработки, программирования вычислительного процесса циклической структуры, умения объяснять синтаксис и назначение операторов цикла; получение дальнейших навыков по отладке и тестированию программы; умений представлять полученные результаты своей деятельности.

Теоретический материал

Циклы позволяют многократно выполнять отдельный оператор или последовательность операторов. В языке Паскаль имеется три типа циклов: цикл со счетчиком (с параметром), цикл с предусловием, цикл с постусловием.

Цикл со счетчиком (или с параметром)

Цикл со счетчиком (или с параметром) используется тогда, когда заранее известно количество повторов тела цикла. Число повторений здесь задается переменной, называемой параметром цикла, или счетчиком.

Рисунок 3.1 – Структура цикла со счетчиком

Данный оператор может быть представлен в двух форматах:

1) шаг +1: `for <параметр цикла> := <S1> to <S2> do <тело цикла>;`

б) шаг -1: `for <параметр цикла> := <S1> to <S2> downto <тело цикла>;`

где `S1` и `S2` – соответственно начальное и конечное значения параметра цикла.

Цикл с предусловием

Цикл с предусловием используется тогда, когда число повторений заранее неизвестно – надо выполнять цикл, пока не произойдет некоторое событие (пользователь нажмет на кнопку, точность вычислений уложится в заданный порог и т. д.). В операторе while проверка условия выполнения тела цикла производится в самом начале оператора.

Структура данного оператора имеет вид:

```
while <условие продолжения цикла> do
                    <тело цикла>;
```

Условие продолжения цикла – булевское (логическое) выражение, тело цикла – простой или составной оператор. Особенность данного вида цикла состоит в том, что значение выражения условия вычисляется перед каждым выполнением тела цикла. Если результат равен True, тело цикла выполняется и снова вычисляется выражение условия. Если результат равен False, происходят выход из цикла и переход к первому после while оператору.

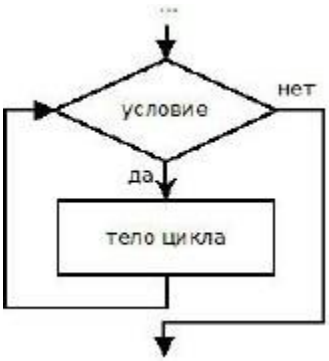


Рисунок 3.2 – Структура цикла с предусловием

Еще один важный момент, о котором нельзя забывать программисту – зацикливание (бесконечное выполнение цикла). Чтобы не происходило зацикливание программы, нужно, чтобы в теле цикла присутствовал оператор, изменяющий значение, проверяемого в условии продолжения цикла.

Цикл с постусловием

Оператор цикла `repeat..until` аналогичен оператору `while`, но отличается от него, во-первых, тем, что условие проверяется после очередного выполнения операторов тела цикла и, таким образом, гарантируется хотя бы однократное выполнение цикла, а во-вторых, цикл завершается, когда условие, стоящее после `until,` становится истинным.

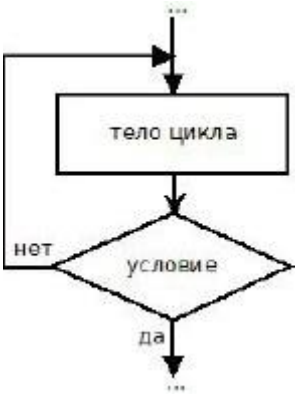


Рисунок 3.3 – Структура цикла постусловием

Структура данного оператора имеет вид:

Структура этого оператора:

```
repeat
<оператор;>
…
<оператор>
until <условие окончания цикла>;
```

Цикл работает следующим образом: выполняется тело цикла (операторы, заключенные между словами repeat и until), затем проверяется условие окончание цикла, если оно имеет значение False, то тело цикла выполняется вновь. Когда условие, наконец, станет истинным, цикла завершится, и далее будет выполняться следующий за циклом оператор.

Следует заметить, что тело цикла в данном случае может содержать как один, так и несколько операторов, однако, операторные скобки здесь не пишутся, т. к. repeat обозначает начало цикла, until – конец цикла.

Так же, как и в случае с циклом while, при программировании операторов тела цикла следует обеспечить влияние, по крайней мере, одного из операторов тела цикла на значение условия, иначе цикл будет выполняться бесконечно.

Пример 1. Составить блок-схему алгоритма и программу для вычисления значений функции Y для значений переменной x, изменяющейся в интервале от a до b с заданным шагом h.

$$Y = \sqrt[3]{(6x^2 - x^3)}$$

Решение

Так как в языке Pascal нет стандартной функции извлечения корня любой степени кроме квадратного, для вычисления функции Y воспользуемся переходом к функциям Exp и Ln.

$$Y = \sqrt[3]{(6x^2 - x^3)} = Exp(1.0/3.0 * Ln(6x^2 - x^3)).$$

При программировании необходимо учесть, что выражение, стоящее под знаком Ln, может быть только положительным: $(6x^2 - x^3) > 0$.

Исходными данными в этой задаче являются значения a, b и h.

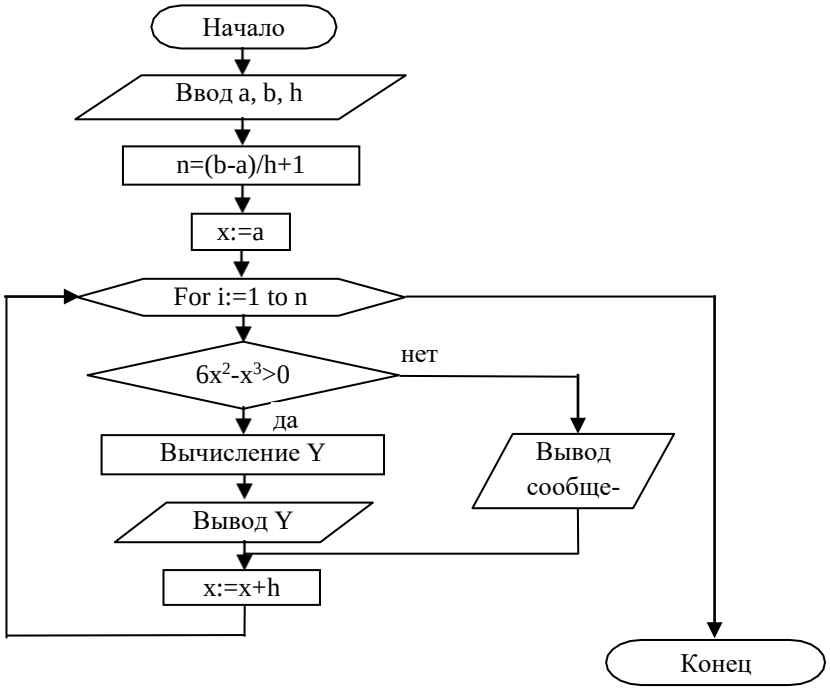
Начальное значение x принимаем равным a (x:=a). Каждое следующее значение x вычисляется по формуле x:= x + h. Вычисления функции Y для новых значений x заканчиваются, когда значение x становится больше, чем заданное значение b.

Схема алгоритма с использованием цикла For ... Do

Для организации цикла For необходимо сначала определить число повторений цикла – n. Это можно сделать с помощью формулы:

n = (b – a) / h + 1.

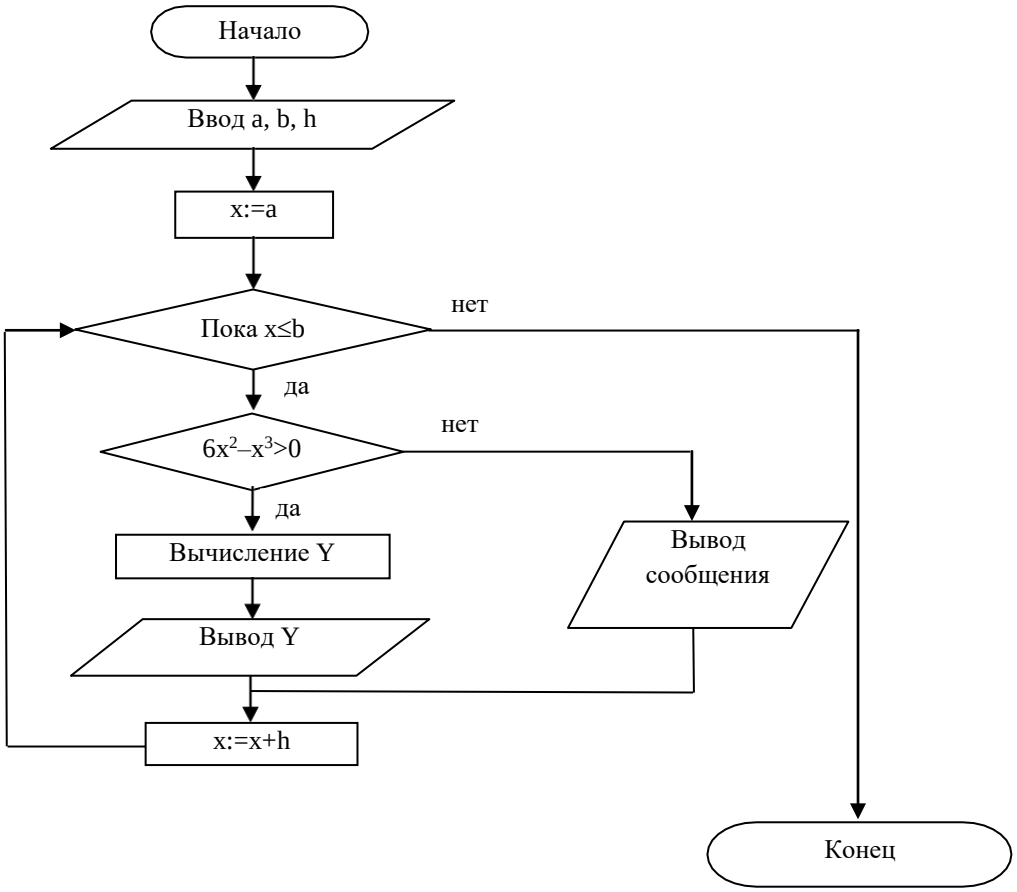
При программировании необходимо учитывать тип переменных, участвующих в выражении. При вычислении значения n в правой части формулы получается результат вещественного типа, а слева стоит переменная целого типа, поэтому необходимо применить операцию округления полученного значения n до целого значения получаемого в результате операции деления. Для этого можно использовать стандартную функцию Round.



Текст программы

```
Program Lab_3_1;
Var x, y, a, b, h:real;
  n, i : integer;
BEGIN
  Writeln(' Введите a b h');
  read(a,b,h);
  n:= Round((b-a)/h)+1;
  x:=a;
   For i:=1 to n do
     begin
       if (6*x*x-x*x*x)>0 then
         begin
            y:= Exp(1.0/3.0*Ln(Abs(6*x*x-x*x*x)));
            Writeln ('x= ',x:8:3,' y= ',y:8:3);
         end
        else
    Writeln ('При x= ', x:5:3,' под знаком
          логарифма недопустимое значение');
    x:=x+h;
  end;  { Конец For}
  END.
```

Схема алгоритма с использованием цикла While ... Do

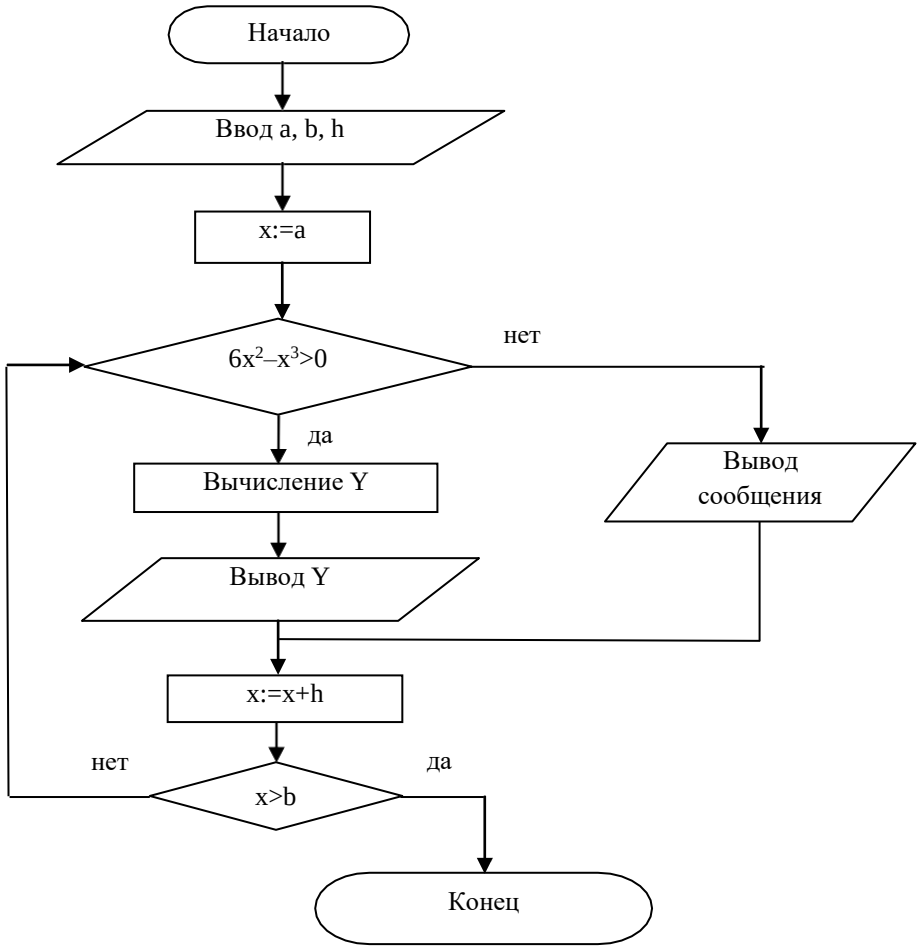


Текст программы

```
Program Lab_3_2;
```

```
Var x,y,h,a,b: real;
BEGIN
  Writeln(' Введите a b h');
  Read(a,b,h);
  x:=A;
  While x<=B Do
  Begin
    if (6*x*x-x*x*x)>0 then
              begin
          y:= Exp(1.0/3.0*Ln(Abs(6*x*x-x*x*x)));
          Writeln('x=',x:8:3,'  y=',y:8:3);
          end
     Else
        Writeln('При x= ',x:5:3,' под знаком логарифма
                              недопустимое значение');
    x:=x+h;
   End; {Конец while}
END.
```

Схема алгоритма с использованием цикла Repeat ... Until



Текст программы

```
Program  Lab_3_2;
Var x,y,h,a,b: real;
BEGIN
  Writeln(' Введите a b h');
```

```
Read(a,b,h);
x:=A;
Repeat
  if (6*x*x-x*x*x)>0 then
            begin
        y:= Exp(1.0/3.0*Ln(Abs(6*x*x-x*x*x)));
        Writeln('x=',x:8:3,'  y=',y:8:3);
        end
   Else
      Writeln('При x= ',x:5:3,' под знаком логарифма
                          недопустимое значение');
  x:=x+h;
Until x>b;
 End;
```

Упражнения

1. Одна штука некоторого товара стоит 120,4 руб. Напечатать таблицу стоимости 2, 3, …, 20 штук этого товара.

2. Напечатать таблицу перевода расстояний в дюймах в сантиметры для значений 10, 11, …, 22 дюйма (1 дюйм = 25,4 мм).

3. Вычислить сумму натуральных нечетных чисел, не превышающих N.

4. Вычислить количество натуральных чисел, кратных трем и не превышающих N.

5. Вводится последовательность ненулевых чисел. Определить, сколько раз в последовательности встречаются четные числа.

6. Вводится последовательность из N произвольных чисел, найти среднее значение положительных элементов последовательности.

7. Вводится последовательность из N произвольных чисел. Определить процент положительных, отрицательных и нулевых элементов.

8. Логической переменной p присвоить значение true, если заданное целое число N>1- простое число (т.е. у N нет других делителей, кроме самого себя) и false в противном случае.

Варианты для самостоятельных заданий

Используя три различных цикла, трижды протабулировать функцию.

1. $f(x)=\sqrt{(x/a+a)}$ на интервале $x\in[-2;\,2.5]$, $\Delta x=0{,}35$, указав значения аргумента, при которых функцию нельзя вычислить.

2. $f(x)=\dfrac{2x+a}{x^3-2x^2-x+2}$ на интервале $x\in[-4;\,5]$, $\Delta x=1$, указав значения аргумента, при которых функцию нельзя вычислить.

3. $f(x)=\sqrt{x^3-6x^2+11x-6}$ на интервале $x\in[0;\,3.5]$, $\Delta x=0{,}4$, указав значения аргумента, при которых функцию нельзя вычислить.

4. $f(k)=\dfrac{\sin(k\cdot\pi/N)}{k\cdot\pi/N}$ на интервале $k\in[-6;3]$, $\Delta k=1$, учтя при этом, что $\dfrac{\sin(0)}{0}=1$.

5. $f(x)=A\cdot\dfrac{\sin(x^3-2x^2-x+2)}{x^3-2x^2-x+2}$ на интервале $x\in[-2;3]$, $\Delta x=0{,}5$, учтя при этом, что $\dfrac{\sin(0)}{0}=1$.

6. $f(x)=\dfrac{2^x+2^{-x}}{x^3+5x^2+2x-8}$ на интервале $x\in[-2;5]$, $\Delta x=0{,}75$, указав значения аргумента, при которых функцию нельзя вычислить.

7. $f(k)=\begin{cases}k^2+1, & \text{при } k \text{ четном}\\ k-a, & \text{при } k \text{ нечетном}\end{cases}$ на интервале $k\in[-2;5]$, $\Delta k=1$.

8. $f(k)=\begin{cases}k^3+a, & \text{при } k \text{ четном}\\ a\cdot k, & \text{при } k \text{ нечетном}\end{cases}$ на интервале $k\in[-3;8]$, $\Delta k=1$.

9. $f(x)=\dfrac{e^{-ax}+e^{ax}}{x^3-7x-6}$ на интервале $x\in[-3;6]$, $\Delta x=1$, указав значения аргумента, при которых функцию нельзя вычислить.

10. $f(x)=\dfrac{ax^2+3}{\sin(\pi x/6)}$ на интервале $x\in[-9;21]$, $\Delta x=3$, указав значения аргумента, при которых функцию нельзя вычислить.

Контрольные вопросы

1. Что такое цикл? Какая программа называется циклической?
2. В каких случаях используется оператор цикла с параметром? Как он работает?
3. Запишите синтаксис оператора цикла с параметром для случая, когда в цикле выполняется 1 оператор. Запишите синтаксис оператора цикла с параметром для случая, когда в цикле выполняется 3 оператора.
4. Что такое «параметр цикла»? Что такое «тело цикла»?
5. С каким шагом перебираются значения параметра цикла? От чего это зависит?
6. Что требуется сделать для того, чтобы в теле цикла выполнялись несколько операторов?
7. Как найти, сколько раз выполняется цикл?
8. Приведите примеры задач, для решения которых используется структура «цикл».
9. В каких случаях используется оператор цикла с предусловием? Как он оформляется? Как он работает?
10. Может ли тело оператора цикла с предусловием:

а) не выполниться ни разу?

б) выполняться бесконечное число раз?

11. В каких случаях используется оператор цикла с постусловием? Как он оформляется? Как он работает?

12. Может ли тело оператора цикла с постусловием:

а) не выполниться ни разу?

б) выполняться бесконечное число раз?

13. Всегда ли можно вместо оператора цикла с параметром использовать оператор цикла с предусловием или постусловием? А наоборот?

ЛАБОРАТОРНАЯ РАБОТА № 4.
ИСПОЛЬЗОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ И ПРОГРАММ ДЛЯ ВЫЧИСЛЕНИЯ СУММЫ ЧЛЕНОВ РЯДА

Цель работы: закрепление навыков в выборе и использовании операторов цикла; овладение практическими навыками разработки и программирования алгоритмов итерационных процессов.

Теоретический материал

Выполнение данной лабораторной работы потребует использования, как и в предыдущей работе, операторов цикла.

Циклы с неизвестным заранее числом повторений, которые выполняются, пока истины определенные условия, называются **итерационными**.

Условиями окончания цикла могут быть: достижение заданной точности вычисления результата, изменение знака определенной величины и т.д.

Вычисление суммы ряда

Расчет суммы N чисел с использованием различных типов операторов цикла приведен в таблице 4.1.

Таблица 4.1 – Расчет суммы N чисел

Цикл со счетчиком	Цикл с предусловием	Цикл с постусловием

`Program Sum;`
`Const N=100;`
`Var i,s: integer;`
`BEGIN`
`S:=0;`
`For i:=1 to N do`
`  S:=S+i;`
`END.`
	`Program Sum;`
	`Const N=100;`
	`Var i,s: integer;`
	`BEGIN`
	`S:=0; i:=1;`
	`While i<=N do`
	`  Begin`
	`  S:=S+i;`
	`  i:=i+1;`
	`  end;`
	`END.`
		`Program Sum;`
		`Const N=100;`
		`Var i,s: integer;`
		`BEGIN`
		`S:=0;i:=1;`
		`Repeat`
		`  S:=S+i;`
		`  i:=i+1;`
		`Until i>N;`
		`END.`

Примечание: Во всех вариантах используется оператор S:=0, который задает начальное значение переменной S, выполняющей роль сумматора. При отсутствии данного оператора, начальное значение S считается неопределенным, а результат суммирования может быть ошибочным.

Вычисление суммы бесконечного ряда с заданной точностью является типичной задачей, использующей итерационный цикл, так как заранее не известно, при каком члене ряда будет достигнута требуемая точность.

Пример 1. Найти сумму знакочередующегося числового ряда: 1-1/2+1/3-1/4…±1/N, где N- некоторое заданное целое число.

Решение:

```
Program Sum;
Const N=10; {длина ряда}
Var i, sign: integer;
     S: real;
BEGIN
  S:=0;{начальное значение}
  Sign:=1;
```

```
  for i:=1 to N do
     begin
       S:=S+sign/i;
       sign:=-sign;
     end;
END.
```

Пример 2. Составить схему алгоритма и программу определения суммы членов ряда с точностью $\varepsilon=10^3$, общий член которого a_n задан формулой:

$$a_n = \frac{x^n \cdot (2 \cdot n - 1)}{n!} \qquad (4.1)$$

Примечание. k! – факториал числа k. 0! = 1; 1! = 1; 2! = 1·2; k! = 1·2·3 ... k.

Решение:

Для сокращения вычислений в программе (при подсчете значения факториала и возведении числа в степень) следует использовать рекуррентную формулу для определения следующего элемента ряда. Для получения рекуррентной формулы вычислим отношение (n+1)-го члена ряда к n-му:

$$\frac{a_{n+1}}{a_n} = \frac{x^{n+1} \cdot (2 \cdot (n+1) - 1) \cdot n!}{x^n \cdot (2 \cdot n - 1) \cdot (n+1)!} = \frac{x \cdot (2 \cdot n + 1)}{(2 \cdot n - 1) \cdot (n+1)}$$

Тогда

$$a_{n+1} = a_n \cdot \frac{x \cdot (2 \cdot n + 1)}{(2 \cdot n - 1) \cdot (n+1)} \qquad (4.2)$$

Вычислим a_1 по формуле (4.1):

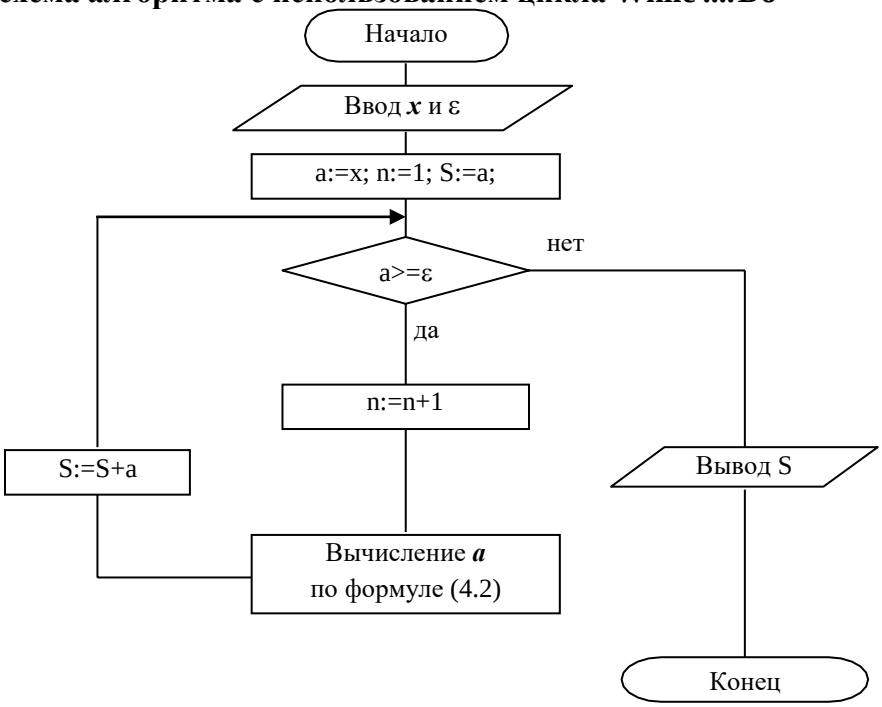
$$a_1 = \frac{x^1 \cdot (2 \cdot 1 - 1)}{1!} = x$$

Воспользуемся этим результатом для задания начального значения ***a*** и суммы членов ряда ***S***.

При составлении алгоритма и программы будем считать, что точность получена, если на очередном шаге вычислений значение $a_n < \varepsilon$.

Входные данные: x и ε (eps – обозначение в программе).

Блок-схема алгоритма с использованием цикла While ....Do



Текст программы

```
Program lab_4_1;
Var x, a, eps, S: real;
        n:integer;
BEGIN
  Writeln('Введите x, eps');
  readln(x, eps);
  {Задание начальных значений переменных}
  a:=x; n:=1; S:=a;
    While a>=eps Do
         Begin
            n:=n+1;
            {Вычисление очередного члена ряда по формуле (4.2)}
            a:=a*x*(2*n+1)/((2*n-1)*(n+1));
            s:=s+a;
         end;
  Writeln('Сумма членов ряда S = ',S:10:4,' Число итераций n =
',n:5);
END.
```

Рекомендация: При составлении программы не забывайте анализировать знаменатель (он не должен равняться нулю) и типы данных участвующих в выражениях.

Вложенные циклы

Табулирование функции и ее разложения в сумму ряда

Пример: Составить блок-схему и программу табулирования двух функций S и Y в заданном диапазоне изменения аргумента x.

Функция $y = 3^x$ разложена в виде суммы ряда $s = 1 + \frac{\ln 3}{1!}x + \frac{(\ln 3)^2}{2!}x^2 + \cdots$(количество слагаемых n = 10). Интервал изменения x = 0, 0.05…π. Результат табулирования вывести в форме следующей таблицы:

```
-------------------------------------------------------------------
!     X        !         y = f(x)            !         S         !
-------------------------------------------------------------------
```

При решении задачи будет использовать два циклических оператора, причем один окажется вложенным в другой, т. е. тело внешнего цикла будет содержать еще один оператор цикла. При этом следует внимательно следить за количеством операторов, составляющих тело цикла (за исключением цикла `repeat...until`). В случае если количество операторов тела цикла больше одного, все операторы, которые необходимо выполнить в цикле, заключаются в операторные скобки (`begin...end`).

Заметим, что значения функций S и Y при одинаковых значениях x будут приблизительно равными.

Составим блок-схему алгоритма. Прежде всего, необходимо найти закономерность вычисления каждого слагаемого ряда. В данном случае эта закономерность определяется формулой:

$$\frac{(\ln 3)^i}{i!}x^i \qquad (4.3)$$

где i – номер слагаемого (в данном случае изменяется от 0 до 10).

По формуле 4.3 рассчитаем нулевое слагаемое:

$$a_0 = \frac{(\ln 3)^0}{0!}x^0 = 1$$

Каждое последующее слагаемое можно выразить через предыдущее следующей формулой:

$$a_i = a_{i-1} \cdot \frac{(ln\ 3)}{i} \cdot x$$

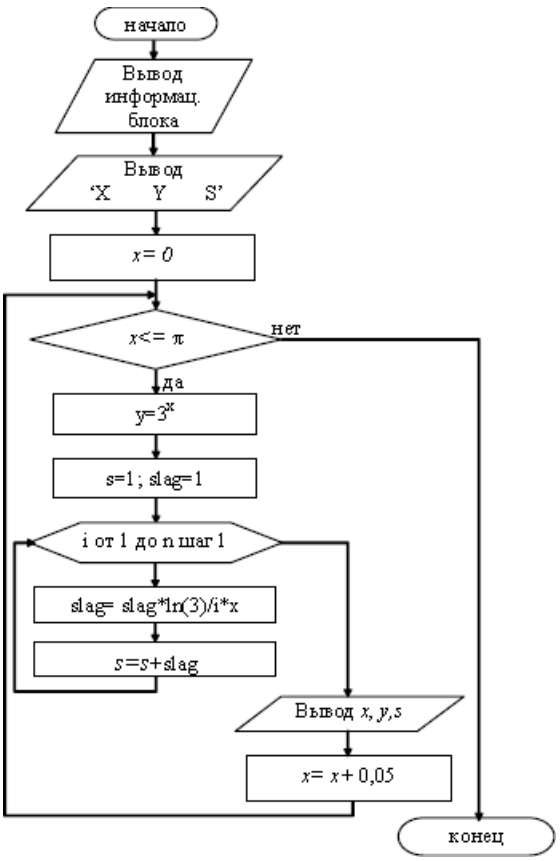
где i – номер слагаемого.

Действительно,

$$a_1 = \frac{(ln\ 3)^1}{1!} x^1 = a_0 \cdot \frac{ln\ 3}{1} \cdot x$$

$$a_2 = \frac{(ln\ 3)^2}{2!} x^2 = \frac{(ln\ 3)^1}{1!} x^1 \cdot \frac{ln\ 3}{2} \cdot x == a_1 \cdot \frac{ln\ 3}{2} \cdot x$$

Последовательно вычисляя и суммируя все слагаемые при определенном значении x, получим требуемую сумму. Для этого организуем цикл, повторяющийся n+1 раз (включая нулевое слагаемое), на каждой итерации которого будет вычисляться значение очередного слагаемого и прибавляться к накапливаемой сумме s.



Текст программы

```
Program lab_4_2;
Const n=10; h=0.05;
Var x, y, s, slag: real;
        i: integer;
BEGIN
  Writeln('!   X   !   Y   !   S   !');
  x:=0;
  While x<1+h do
       Begin
          y:=exp(x*ln(3));
          s:=1; slag:=1;
            for i:=1 to n do
                begin
                  slag:=slag*ln(3)/i*x;
                  s:=s+slag;
                end;
          writeln('!',x:7:2,' !',y:7:3,' !',s:7:3,' !');
          x:=x+h
       end;
END.
```

Упражнения

1. Вычислить сумму $\frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \cdots + \frac{10}{11}$.

2. Вычислить сумму $1 + \frac{1}{3} + \frac{1}{3^2} + \cdots + \frac{1}{3^8}$. Операцию возведения в степень не использовать.

3. Вычислить сумму $1-\frac{1}{2}+\frac{1}{3}-\cdots+(-1)^{n+1}\frac{1}{n}$. Условный оператор и оператор возведения в степень не использовать.

4. Вычислить сумму $x+\frac{x^3}{3}+\frac{x^5}{5}+\cdots+\frac{x^{11}}{11}$ при x = 2.

5. Для заданного целого значения n рассчитать произведение $\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot\Lambda\cdot\frac{2n-1}{2n}$;

6. Известно, что функция exp(x) может быть представлена в виде бесконечного степенного ряда:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^i}{i!} + \ldots$$

7. Для заданного вещественного значения x рассчитать:

а) сумму n членов данного ряда, где n- заданное целое число;

б) сумму ряда с заданной точностью (суммирование прекратить после достижения условия, $\left|\frac{x^i}{i!}\right| < err$, где err – погрешность расчета (малое число).

8. Известно, что функцию cos(x) можно разложить в следующий степенной ряд:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Для заданного вещественного значения x рассчитать сумму этого ряда. Расчет прекратить после достижения условия: $\left|\frac{x^{2i}}{(2i)!}\right| < err$, где *err*- погрешность расчета.

9. Последовательность Фибоначчи образуется так: первый и второй члены последовательности равны 1, каждый следующий равен сумме двух предыдущих (1, 1, 2, 3, 5, 8, 13, …).

а) найти k-й член последовательности Фибоначчи;

б) получить первые *n* членов последовательности Фибоначчи.

Варианты самостоятельных заданий

1. Используя три различных цикла, определить значение

1. Суммы, $S=\sum_{n=1}^{6}\left(n^2+x\right)$.

2. Произведения, $P=\prod_{n=1}^{4}\left(n^2+x\right)$.

3. Суммы, $S=\sum_{n=1}^{5}\left(n^3+n\cdot x\right)$.

4. Произведения, $P=\prod_{n=1}^{8}\left(n+x\right)$.

5. Суммы, $S=\sum_{n=3}^{9}\ln\left(n\right)$.

6. Суммы, $S=\sum_{n=1}^{k}\left(n+1\right)^2$.

7. Произведения, $P=\prod_{n=1}^{k}\left(\frac{n}{2}+x\right)$.

8. Произведения, $P=\prod_{n=1}^{5}\left(\frac{1}{\sqrt{2n}}+x\right)$.

9. Суммы, $S=\sum_{n=1}^{6}(x\cdot n-1)^3$.

10. Произведения, $P=\prod_{n=1}^{5}\left(2x+\sqrt{n}\right)$.

2. Составить блок-схему и программу табулирования двух функций S и Y в заданном диапазоне изменения аргумента x.

Варианты заданий

N вар.	Сумма S	Диапазон изменения x	n	Функция Y
1	$1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots$	[1;2]	15	e^x
2	$\cos x+\frac{\cos 2x}{2}+\frac{\cos 3x}{3}+\ldots$	[-π/5; 9π/5]	40	$-\ln\left\|2\sin\frac{x}{2}\right\|$
3	$x-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots$	[0,1; 1]	10	$\sin x$
4	$\sin x-\frac{\sin 2x}{2}+\frac{\sin 3x}{3}-\ldots$	[-π/5; 4π/5]	40	$\frac{x}{2}$
5	$\cos x+\frac{\cos 3x}{9}+\frac{\cos 5x}{25}+\ldots$	[π/5; π]	40	$\frac{\pi^2}{8}-\frac{\pi}{4}\|x\|$
6	$1+\frac{\cos\frac{\pi}{4}}{1!}x+\frac{\cos 2\frac{\pi}{4}}{2!}x^2+\ldots$	[0,1; 1]	25	$e^{x\cos\frac{\pi}{4}}\cdot\cos(x\sin\frac{\pi}{4})$
7	$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots$	[0,1; 1]	10	$\cos x$
8	$x\sin\frac{\pi}{4}+x^2\sin 2\frac{\pi}{4}+\ldots$	[0,1; 0,8]	40	$\frac{x\sin\frac{\pi}{4}}{1-2x\cos\frac{\pi}{4}+x^2}$
9	$x+\frac{x^5}{5}+\frac{x^9}{9}+\frac{x^{13}}{13}\ldots$	[0,1; 0,8]	30	$\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}arctgx$
10	$1+\frac{\cos x}{1!}+\frac{\cos 2x}{2!}+\ldots$	[0,1; 1]	10	$e^{\cos x}\cdot\cos(\sin x)$

Контрольные вопросы

1. Объясните, как работает формула s:=s+i.
2. Приведите примеры задач, алгоритм решения которых основан на вложенных циклах.
3. Что такое «итерация»? Какие циклы называют итерационными?
4. Должны ли операторы внутреннего цикла полностью располагаться в теле внешнего цикла?
5. Вычислите, сколько раз выполнится тело цикла for i:=1 to 10 do for j:=1 to 10 do s:=s+i*j. Поясните, как работает этот фрагмент программы.
6. С какого цикла начинается выполнение программы: с внешнего или с внутреннего?
7. Нарисуйте блок-схему алгоритма с двумя вложенными друг в друга циклами с параметром.
8. Нарисуйте блок-схему алгоритма, в которой в цикл с параметром вложен цикл с предусловием.

ЛАБОРАТОРНАЯ РАБОТА № 5.
МАССИВЫ

Цель работы: формирование совокупности знаний об особенностях организации работы с массивами; овладение практическими навыками разработки, программирования вычислительного процесса с использованием одномерных массивов; умений анализировать исходные данные задачи; получение дальнейших навыков по отладке и тестированию программы; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их, умений объяснять синтаксис операторов организации работы с массивами.

Теоретический материал

До сих пор мы рассматривали простые типы (исключение составляет тип string). Здесь мы рассмотрим массивы, которые относятся к так называемым структурированным типам. Переменная или константа структурированного типа всегда имеет несколько элементов.

Массив объединяет фиксированное количество элементов одного типа данных.

Описание массива

`<имя массива> : array[<тип индекса >] of <тип>;`

Размерность массива может быть любой, компоненты массива могут быть любого (в том числе и структурированного) типа, индекс (индексы) может (могут) быть только интервального или перечисляемого типа. При употреблении в качестве типа индекса типа Integer или Word можно использовать только его диапазон.

Например, следующие записи

```
V1: array [-10..10] of real;
V2: array [`a`..`z`] of string;
```

описывают два массива: массив с именем V1, содержащий 21 элемент типа real , и массив V2, содержащий 26 (по числу букв английского алфавита) элементов типа string.

Количество индексных типов при описании массива определяет размерность массива. Следовательно, рассмотренные массивы V1 и V2 являются одномерными.

Размерность массива может быть произвольной, ограничение накладывает лишь объем доступной памяти компьютера. Для примера опишем трехмерный массив:

```
V3: array [1..10,`A`..`C`,false..true] of array [0..5] of
integer;
```

Массив V3 содержит 60 (10x3x2) элементов, каждый из которых – одномерный массив из шести целых чисел.

Доступ к элементам массива осуществляется путем указания имени массива, за которым в квадратных скобках помещается значение индексов:

```
<имя массива> [<индексное выражение>],
```

где <индексное выражение>- выражение того же типа, что и при описании массива.

Например, для описанных выше массивов можно записать следующие операторы:

```
V1 [7]:=3.14159;
V2[`c`]:=V2[`b`]+`привет`;
```

Массивы не могут обрабатываться целиком за одним исключением: в случае идентичных массивов можно с помощью оператора присваивания передавать все элементы одного массива другому, например:

```
Var a,b: array [10..30] of char;
BEGIN
. . .
a:=b;
. . .
```

```
END.
```

Одномерные массивы

Массивы в программах удобно использовать совместно с циклами. Здесь мы рассмотрим несколько таких примеров.

Пример 1: Одномерный массив A заполнить N случайными числами в диапазоне 0…1 и вывести элементы полученного массива по к значений в строке.

Решение:

```
Program Task_1;
Const N=10; {количество чисел в массиве}
    K=12; {количество значений в строке}
Var  i: integer;
   A: array [1..N] of real;
BEGIN
for i:=1 to N do A[i]:= random; {заполнение массива}
{вывод массива}
for i:=1 to N do if (i mod k)=0 then writeln (A[i]) else
write (A[i]);
END.
```

Пример 2. Дано целое число n>1 и вещественные числа x1,x2, … , xn. Написать программу расчета значений:

$$M=\frac{\sum_{i=1}^{n} x_i}{n},\quad D=\sqrt{\frac{\sum_{i=1}^{n}(x_i-M)^2}{n-1}}.$$

Решение

```
Program Task_2;
Const n=<…>; {значение n}
Var  i: integer;
  S,M,D: real;
    x: array [1..n] of real;
BEGIN
{ввод значений Xi}
for i:=1 to n do read(x[i]);
{расчет M}
S:=0;
For i:= 1 to n do S:=S+x[i];
M:=S/n;
{расчет D}
S;=0;
For i:=1 to n do S:=S+(x[i]-M)*(x[i]-M);
D:=sqrt(S/(n-1));
END.
```

Пример 3. Заполнить массив D случайными вещественными числами. В полученном массиве провести сортировку его элементов по возрастанию значений.

Решение Для решения используем известный алгоритм сортировки «пузырьков». Его суть: последовательно производится сравнение двух соседних элементов массива и, если предыдущее значение больше последующего, то элементы меняются значениями.

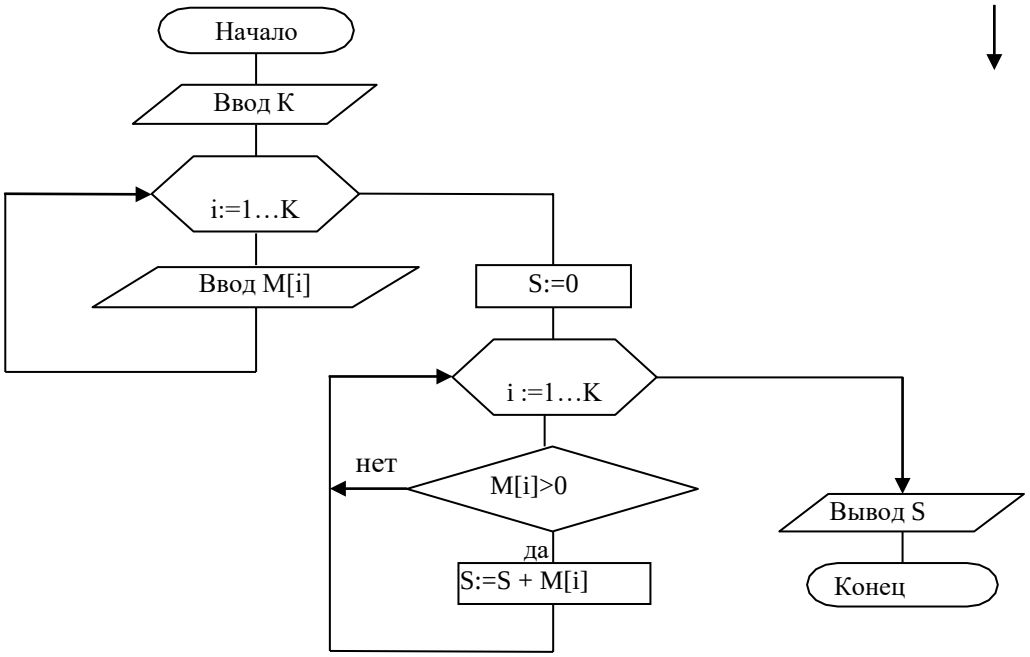
```
Program Sort;
Const n=<…>; {размер массива}
Var  i,j: integer;
    a: real;
  order: boolean;
```

```
    D: array [1..n] of real;
BEGIN
for i:=1 to n do D[i]:=random; {заполнение массива}
{сортировка}
j:=0;
Repeat
  order:=true;
  for i:=1 to n-j-1 do
  if D[i]>D[i+1] then
   begin {перестановка}
   a:=D[i+1];
   D[i+1]:=D[i];
   D[i]:=a;
   order:=false;
   end;
  j:=j+1;
  until order;
END.
```

Логическая переменная order здесь играет следующую роль: если массив еще не упорядочен, то во внутреннем цикле order принимает значение false и сортировка продолжается. Если массив стал упорядоченным, то перестановок элементов массива не происходит, и переменная order сохраняет значение true, что является условием прекращения сортировки.

Пример 4. Составить схему алгоритма и программу определения суммы S всех положительных элементов одномерного массива M, содержащего K вещественных чисел (K≤20). Числа в массив M ввести с экрана дисплея, результат суммирования (S) также вывести на экран.

Блок-схема алгоритма



Текст программы

```
Program summa;
Var m:array[1..20] of real; {Описание массива вещественных чи-
сел}
```

```
S:real; {описание переменной для хранения суммы}
i,K:byte; {описание переменных для счетчика и количества чисел}
Begin
Write(' Введите число элементов в массиве K =');
Readln(K);
Writeln(' Введите ',K,' вещественных чисел через пробел');
For i:=1 to K do read(M[i]); {Ввод вещественных чисел}
writeln;  {Переход на новую строку на экране}
S:=0; {Присвоение начального значения суммы}
For i:=1 to K do
If M[i]>0 then S:=S+M[i]; {Суммирование положительных чисел}
Writeln(' S = ',S); {Вывод полученной суммы}
End.
```

Многомерные массивы

Пример 5. Особенности работы с многомерными массивами рассмотрены на примере следующей задачи: трехмерный массив D[1..L, 1..M, 1..N] заполнить случайными числами и найти максимальное и минимальное значения данного массива.

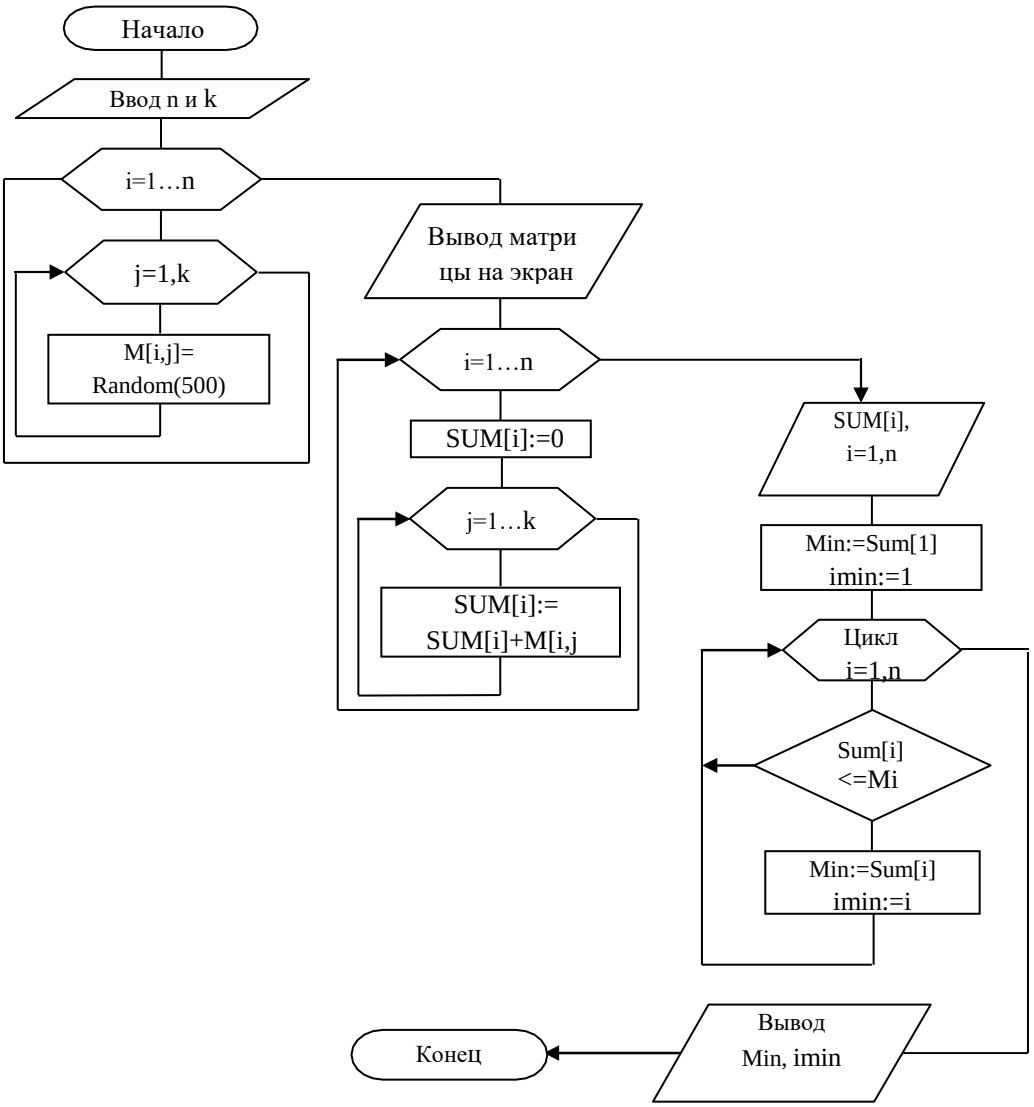
Решение.

```
Program MinMax;
Const L=<...>;
   M=<...>;
   N=<...>;
Var i, j, k: integer;
   Min,Max: real;
      D: array [1..L,1..M,1..N] of real;
BEGIN
{заполнение массива}
   for i:=1 to L do
    for j:=1 to M do
     for k:=1 to N do
     D[i,j,k]:= random;

{нахождение минимального и максимального значений}
   Min:=D[1,1,1]; Max:=D[1,1,1];
   for i:=1 to L do
    for j:=1 to M do
     for k:=1 to N do
     begin
      if Min>D[i,j,k] then Min:=D[i,j,k];
      if Max<D[i,j,k] then Max:=D[i,j,k];
     end;
END.
```

Пример 6. Составить схему алгоритма и программу определения сумм элементов в каждой строке матрицы $M=\{m_{ij}\}$, i=1,n; j=1,k, где n – число строк, k – число столбцов матрицы, m_{ij} – целые числа из диапазона: от 0 до 500. Записать полученные значения сумм в одномерный массив SUM, а затем вывести их на экран дисплея. Числа в массив M занести с помощью функции Random. Определить и вывести на экран номер строки с минимальным значением суммы.

Блок-схема алгоритма



Текст программы

```
Program Summa;
Var M:array[1..50,1..100]of integer;
  Sum:array[1..50] of integer;
  n,k,i,j,imin, Min:integer;
BEGIN
  writeln(' Введите число строк и столбцов');
  readln(n,k);
  Randomize;
  {Заполнение матрицы случайными числами}
  for i:=1 to n do
  for j:=1 to k do
  M[i,j]:=Random(500);
  writeln(' Элементы заполненной матрицы');
  for i:=1 to n do
  begin
     for j:=1 to k do
     write(M[i,j]:4);
     writeln;
  end;
  writeln(' Сумма элементов в каждой строке');
```

```
write(' Номера строк :   ');
for i:=1 to n do
write(i,'  ');writeln;
write(' Сумма в строке : ');
for i:=1 to n do
begin
   Sum[i]:=0;
   for j:=1 to k do
   Sum[i]:=Sum[i]+M[i,j];
   write(Sum[i], ' ');
end;
writeln;
{Поиск минимального значения}
Min:=Sum[1]; imin:=i;
for i:=1 to n do
if Sum[i] <= Min then
   begin
      Min:=Sum[i]; imin:=i;
   end;
 writeln(' Минимальная сумма = ',Min,' в строке ',imin);
 END.
```

Упражнения

1. Задан одномерный массив целых чисел. Элементы массива расположить в обратном порядке.

2. Даны два одномерных массива a и b, содержащих по 15 элементов вещественных чисел. Вычислить

(a[1]+b[15])·(a[2]+b[14]) ... (a[15]+b[1]).

3. Задан двухмерный массив вещественных чисел. Найти его максимальное и минимальное значения.

4. Дана квадратная матрица вещественных чисел A, найти сумму элементов матрицы по ее диагоналям.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

5. Используя функцию Random, заполнить одномерный массив случайными числами. Провести сортировку полученного массива:

а) по возрастанию значений;

б) по убыванию значений.

6. В классе N учеников. Каждому ученику и учителю выдается по $10. Учитель случайным образом выбирает ученика и играет с ним в игру «орел – решка». Стоимость одной игры – $0.5. Долги делать нельзя. Напишите программу, имитирующую эту игру. Игра считается законченной, когда у какого – либо игрока закончатся деньги. Результаты всех туров вывести на экран.

Варианты самостоятельных заданий

Одномерный массив

1. Составить схему алгоритма и программу определения суммы наибольшего и наименьшего элементов вещественного массива A={a_i}, i=1,N, N- число элементов масси-

ва (N<100). Вывести на экран найденную сумму, а также наибольшее и наименьшее значения и их адреса (индексы) в массиве А.

2. Составить схему алгоритма и программу определения разности наибольшего и наименьшего элементов массива вещественных чисел $M=\{m_i\}$, i=1,t, t- число элементов массива (t≤50). Вывести на экран найденную разность, а также наибольшее и наименьшее значения и их адреса в массиве М.

3. Составить схему алгоритма и программу определения суммы всех четных чисел массива $K=\{k_j\}$, j=1,m, m – число элементов в массиве К (m≤60). Вывести на экран найденное значение суммы, а также все четные числа и их адреса.

4. Составить схему алгоритма и программу определения суммы всех нечетных чисел массива $T=\{t_j\}$, j=1,s, s – число элементов в массиве Т (s≤40). Вывести на экран найденное значение суммы, а также все нечетные числа и их адреса.

5. Составить схему алгоритма и программу определения количества четных чисел массива $K=\{k_j\}$, j=1,m, m – число элементов в массиве К (m≤80). Вывести на экран найденное количество, а также все четные числа и их адреса.

6. Составить схему алгоритма и программу определения количества нечетных элементов массива $T=\{t_j\}$, j=1,s, s – число элементов в массиве Т (s≤400). Вывести на экран найденное количество, а также все нечетные числа и их адреса.

7. Составить схему алгоритма и программу определения количества и адресов элементов массива целых чисел $K=\{k_j\}$, j=1,m, m – число элементов в массиве К (m≤55), значения которых не превышают заданного значения K_0. Вывести на экран найденное количество, а также все найденные адреса.

8. Составить схему алгоритма и программу определения количества и адресов элементов массива вещественных чисел $P=\{p_j\}$, j=1,r, r – число элементов в массиве Р (r≤140), значения которых не меньше заданного значения P_0. Вывести на экран найденное количество, а также все найденные адреса.

9. Составить схему алгоритма и программу определения произведения наибольшего и наименьшего элементов массива целых чисел $M=\{m_i\}$, i=1,t, t- число элементов массива (t≤250). Вывести на экран найденное произведение, а также наибольшее и наименьшее значения и их адреса в массиве М.

10. Составить схему алгоритма и программу определения количества и адресов нулевых элементов массива вещественных чисел $K=\{k_j\}$, j=1,m, m – число элементов в массиве К (m≤90). Вывести на экран найденное количество, а также все найденные адреса.

11. Составить схему алгоритма и программу определения сумм всех положительных и отрицательных элементов массива вещественных чисел $M=\{m_i\}$, i=1,t, t- число элементов массива (t≤200). Вывести на экран найденные значения.

12. Составить схему алгоритма и программу определения количества положительных и отрицательных элементов массива вещественных чисел $K=\{\kappa_i\}$, i=1,t, t- число элементов массива (t≤120). Вывести на экран найденные значения.

13. Составить схему алгоритма и программу определения количества и адресов элементов массива вещественных чисел $P=\{p_j\}$, j=1,r, r – число элементов в массиве Р (r≤180), значения которых равны заданному значению P_0. Вывести на экран найденное количество, а также все найденные адреса.

14. Составить схему алгоритма и программу определения суммы и количества положительных элементов массива вещественных чисел $K=\{\kappa_i\}$, i=1,t, t- число элементов массива (t≤170). Вывести на экран найденные значения.

15. Составить схему алгоритма и программу определения суммы и количества отрицательных элементов массива вещественных чисел $K=\{\kappa_i\}$, i=1,t, t- число элементов массива (t≤260). Вывести на экран найденные значения.

16. Составить схему алгоритма и программу определения количества и адресов положительных элементов массива вещественных чисел $K=\{k_j\}$, j=1,m, m – число элемен-

тов в массиве К (m≤190). Вывести на экран найденное количество, а также все найденные адреса.

17. Составить схему алгоритма и программу определения суммы и адресов положительных элементов массива вещественных чисел К={k_j}, j=1,m, m – число элементов в массиве К (m≤900). Вывести на экран найденное количество, а также все найденные адреса.

18. Составить схему алгоритма и программу определения количества и адресов отрицательных элементов массива вещественных чисел К={k_j}, j=1,m, m – число элементов в массиве К (m≤110). Вывести на экран найденное количество, а также все найденные адреса.

19. Составить схему алгоритма и программу определения суммы и адресов отрицательных элементов массива вещественных чисел К={k_j}, j=1,m, m – число элементов в массиве К (m≤450). Вывести на экран найденное количество, а также все найденные адреса.

20. Составить схему алгоритма и программу определения количества и адресов нулевых элементов массива целых чисел К={k_j}, j=1,m, m – число элементов в массиве К (m≤300). Вывести на экран найденное количество, а также все найденные адреса.

Двумерный массив

1. В прямоугольной матрице размером Т*М, имеющей имя MATP содержаться целые числа. Т – число строк, М – число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму элементов в каждой строке. Определить строку с максимальным значением этой суммы и вывести ее номер на экран.

2. В прямоугольной матрице размером М*Т, имеющей имя MAS содержаться целые числа. М – число строк, Т – число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму элементов в каждой строке. Определить строку с минимальным значением этой суммы и вывести ее номер на экран.

3. В прямоугольной матрице размером К*М, имеющей имя MATR содержаться целые числа. К- число строк, М- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму положительных элементов в каждой строке. Определить строку с максимальным значением этой суммы и вывести ее номер на экран.

4. В прямоугольной матрице размером М*К, , имеющей имя MAT содержаться целые числа. М- число строк, К- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму отрицательных элементов в каждой строке. Определить строку с максимальным значением этой суммы и вывести ее номер на экран.

5. В прямоугольной матрице размером L*M, имеющей имя MATR содержаться целые числа. L- число строк, М- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму положительных элементов в каждой строке. Определить строку с минимальным значением этой суммы и вывести ее номер на экран.

6. В прямоугольной матрице размером М*К, имеющей имя MAT содержаться целые числа. М- число строк, К- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму отрицательных элементов в каждой строке. Определить строку с минимальным значением этой суммы и вывести ее номер на экран.

7. В прямоугольной матрице размером М*К, имеющей имя MA содержаться целые числа. М- число строк, К- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму элементов в каждом столбце. Определить столбец с максимальным значением этой суммы и вывести его номер на экран.

8. В прямоугольной матрице размером М*К, имеющей имя М содержаться целые числа. М- число строк, К- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму элементов в каждом столбце. Определить столбец с миниимальным значением этой суммы и вывести его номер на экран.

9. В прямоугольной матрице размером М*К, , имеющей имя ММ содержаться целые числа. М- число строк, К- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму положительных элементов в каждом столбце. Определить столбец с максимальным значением этой суммы и вывести его номер на экран.

10. В прямоугольной матрице размером М*К, имеющей имя МАМ содержаться целые числа. М- число строк, К- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму отрицательных элементов в каждом столбце. Определить столбец с миниимальным значением этой суммы и вывести его номер на экран.

11. В квадратной матрице размером К*К, имеющей имя МА содержаться целые числа. К- число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму положительных элементов в каждом столбце. Определить столбец с минимальным значением этой суммы и вывести его номер на экран.

12. В квадратной матрице размером Т*Т, имеющей имя МКА, содержаться целые числа. Т – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму отрицательных элементов в каждом столбце. Определить столбец с максимальным по модулю значением этой суммы и вывести его номер на экран.

13. В квадратной матрице размером Т*Т, имеющей имя МКА, содержаться целые числа. Т – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран число элементов равных 1 в каждой строке. Определить строку с максимальным числом единиц и вывести ее номер на экран.

14. В квадратной матрице размером Т*Т, имеющей имя МКА, содержаться целые числа. Т – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму отрицательных элементов в каждом столбце. Определить столбец с минимальным по модулю значением этой суммы и вывести его номер на экран.

15. В квадратной матрице размером М*М, имеющей имя МКМ, содержаться целые числа. М – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран число элементов равных 1 в каждой строке. Определить строку с минимальным числом единиц и вывести ее номер на экран.

16. В квадратной матрице размером М*М, имеющей имя МНМ, содержаться целые числа. М – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран число элементов равных 0 в каждой строке. Определить строку с минимальным числом нулевых элементов и вывести ее номер на экран.

17. В квадратной матрице размером К*К, имеющей имя Ма, содержаться целые числа. К – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран число элементов равных 0 в каждой строке. Определить строку с максимальным числом нулевых элементов и вывести ее номер на экран.

18. В квадратной матрице размером М*М, имеющей имя МКМ, содержаться целые числа. М – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран минимальный элемент в каждой строке. Определить строку с максимальным значением из найденных минимумов и вывести ее номер на экран.

19. В квадратной матрице размером М*М, имеющей имя МКМ, содержаться целые числа. М – число строк и столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран максимальный элемент в каждой строке. Определить строку с минимальным значением из найденных максимумов и вывести ее номер на экран.

20. В прямоугольной матрице размером М*Т, имеющей имя МТМ, содержаться целые числа. М – число строк Т- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран минимальный элемент в каждой строке. Определить строку с максимальным значением из найденных минимумов и вывести ее номер на экран.

Контрольные вопросы

1. Что такое «массив», «элемент массива», «индекс элемента массива»?

2. Почему массив является структурированным типом данных?
3. Какого типа могут быть элементы массива?
4. Какого типа могут быть индексы элементов массива?
5. Какие простые типы данных относятся к порядковым?
6. Как объявить одномерный целочисленный массив А из 100 элементов?
7. Как определить количество ячеек памяти, отводимой под массив?
8. Какие существуют способы заполнения одномерного массива?
9. Запишите фрагмент кода программы на языке Паскаль для заполнения массива 50 числами с клавиатуры.
10. Как организовать перебор всех элементов одномерного массива?
11. Запишите фрагмент кода программы на языке Паскаль для перебора всех элементов массива и подсчета их суммы.
12. Запишите синтаксис объявления двумерного массива размера 50x50 вещественного типа данных.
13. Запишите программный блок для заполнения двумерного массива вещественными числами с клавиатуры.
14. Запишите программный блок для заполнения двумерного массива числами из диапазона от -20 до 20 с помощью генератора случайного числа.
15. Запишите программный блок для заполнения двумерного массива числами по формуле:

$$a\,[i\,,j]=(i+j)2\ .$$

16. Запишите программный блок для вывода элементов двумерного массива в виде матрицы.
17. Какой блок позволяет осуществить перебор всех элементов двумерного массива?
18. Можно ли выполнять обход двумерного массива, организовав внешний цикл по столбцам, а внутренний — по строкам?
19. Целесообразно ли использовать вложенные циклы, если совершается обход только главной диагонали квадратной матрицы? Одной строки матрицы? Одного столбца матрицы?
20. При решении каких задач удобно использовать двумерные массивы? Приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА № 6.
СИМВОЛЫ И СТРОКИ

Цель работы: формирование совокупности знаний об особенностях организации работы с символами и строками; овладение практическими навыками разработки, программирования вычислительного процесса с использованием символов и строк; умений анализировать исходные данные задачи; получение дальнейших навыков по отладке и тестированию программы; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их.

Теоретический материал

Символы

Множеством значений стандартного символьного типа char являются символы, упорядоченные согласно кодовой таблице ASCII. Каждому символу приписывается целое число в диапазоне 0 … 255. Это число есть код внутреннего представления символа.

Описание типа

Переменные символьного типа должны быть объявлены в разделе описания с помощью конструкции вида

<имя> : char;

Например:

```
Var
t: char;
s1,s2: char;
```

Символьные константы записываются в апострофах, например `'b','?'`.

Под переменные и константы символьного типа отводится 1 байт памяти.

Операции с символами

Переменные символьного типа могут получать свои значения как в результате операции присваивания, так и получить значение с помощью процедуры ввода Read или Readln. В приведенном ниже примере используются обе эти возможности.

```
Var a,b,otv:char;
BEGIN
a:='a';
b:='b';
 . . .
 Write('Вы хотите завершить работу с программой?');
 Read(otv);
 Writeln('Ваш ответ:',otv);
END.
```

Так как переменная otv объявлена как символьная, достаточно ввести одну букву. Если будет введено несколько символов (например, `'Да'`), то переменная otv получит значение только первого символа – `'Д'`.

Переменные символьного типа можно сравнивать между собой. Сравнение основано на том, что каждому символу поставлено в соответствие номер кода в таблице кодов. Например, ‘0’<’1’<…<’9’< … <’A’<’B’< … <’Z’ < ’a’<’b’< … <’z’ < ’А’<’Б’< … <’Я’ < ‘а’< ’б ’ < … < ’я’ .

Код внутреннего представления символа можно получить с помощью стандартной функции Ord(<символ>), которая по заданному символу возвращает его код. Например, Ord(’b’) вернет значение 98.

И наоборот, если в программе нужно по коду получить символ, то можно использовать встроенную функцию Chr(<код>) или #<код>, где <код> – это номер символа в таблице кодов. Например,

`Chr(98)` или `#98` – это символ b;

`Writeln( 'Mo', #115, #99, Chr(111 ), Chr(119))` –эквивалентен оператору `Writeln('Moscow')`;

Отметим, что символы с кодом 0…31 относятся к служебным. Если эти коды используются в символьном тексте программы, они считаются пробелами. В операторах ввода – вывода они могут иметь самостоятельное значение. Например, оператор `Writeln(#7)` – вызовет выдачу звукового сигнала компьютера.

Массив символов

Также как и числа, символьные данные можно объединять в массивы.

Пусть, например, программа должна вводить с клавиатуры фамилии людей. Это выполняется путем ввода последовательности символов в символьный массив Fam[1..n], длина которого выбирается равной количеству букв самой длинной фамилии. Ввод элементов массива можно выполнить как обычно с помощью оператора цикла со счетчиком, например:

```
For i:=1 to 15 do read(Fam[i]);
```

При этом на клавиатуре надо набирать все 15 символов, даже если это пробелы, пока оператор цикла не завершит свою работу.

Пример 1. Написать программу для ввода фамилии с клавиатуры и хранения символьных данных в массиве.

Решение.

```
PROGRAM FAMILIA;
  Const n=15;
  Var Fam:array[1..n] of char;
  i:integer;
   BEGIN
    i:=0;
    While Fam[i] <> #32 and (i<n) do
    BEGIN
     i:=i+1;
     read(Fam[i]);
    END;
   Writeln; Writeln('длина введенной строки',i)
  END.
```

Здесь предполагается, что длина фамилии не превышает 15 символов. В случае более коротких фамилий ввод можно прервать, если ввести символ пробела (его код – 32).

Строки

Тип string в языке Pascal широко используется для обработки текстов, при создании баз данных, информационных систем и т.д. Он во многом близок к одномерному массиву символов array[0..N] of char. Но есть и существенная особенность: в отличие от массива символов длина строки (т.е. количество символов) может изменяться от 0 до 255.

Таким образом, тип string – последовательность любых символов переменной длины, не превышающей 255 символов.

Описание типа

Строковый тип объявляется с помощью конструкции вида

<имя> : string [N] ,

где <имя> – имя описываемой переменной; N – целое число (0 δ N δ 255) – максимальное количество символов в данной строке. Значение N можно не указывать – в этом случае длина строки полагается максимально возможной, т.е. равной 255.

Пример описания переменной строкового типа:

```
Var
S1: string;
S2: string[25];
```

Значение строковой константы заключается в кавычки. Например, ’Васильев’ – это строковая константа. Ее можно использовать в выражениях. Например, так:

```
S1:= 'Васильев';
Write('эту программу сочинил ', S1);
```

Под строку выделяется N+1 байт памяти. Первый байт служебный, он имеет индекс 0 и содержит символ, порядковый номер которого (код) равен значению текущей длины строки. Первый символ строки занимает второй байт и имеет индекс 1 и т.д.

Операции со строками

Доступ к элементам строки. Строка в языке Pascal трактуется как цепочка символов array [0..N] of char, поэтому к любому символу строки можно обратиться как к элементу такого массива, например:

```
Var St:string;
. . . . .
if St[5]='A' then . . .
```

Текущую (фактическую) длину строки можно определить так: 1) использовать информацию из первого (служебного) байта строки с помощью функции Ord(<строка>[0]); 2) применить стандартную функцию Length(<строка>), которая возвращает целое число – длину соответствующей строки.

Пример 2. Написать программу, в которой с клавиатуры вводится некоторая фамилия, а выводятся ее длина и первая буква.

Решение.

```
Program Family;
Var Fam:string;
   n:integer;
   s:char;
BEGIN
Write('Введите фамилию - ');
Readln(Fam);
N:=Ord(Fam[0]);
s:=Fam[1];
Writeln('Длина фамилии - ',N, ' первая буква - ', s);
END.
```

Операция сложения (сцепления) строк. При выполнении этой операции начало строки, идущей после знака «+», подсоединяется к концу строки, указанной до этого знака. Аналогичный результат возвращает функция `Concat` (<список строк>). Если длина сцепленной строки превышает максимально допустимую длину N, то лишние символы отбрасываются. Ниже в качестве примера приводится фрагмент программы, в котором выполняется операция сложения.

```
A:='програм';
B:='ирова';
C:='ние';
D:=A+B+C;
Write(D); {выведется слово 'программирование'}
```

Операция сравнения строк. Операции сравнения (=, <>, <, >, >=, <=) выполняются над двумя строками посимвольно, слева направо с учетом внутреннего кода символов. Если одна строка меньше другой по длине, недостающие символы короткой строки заменяются значением Chr(0) , т.е. символом с наименьшим кодом. Если в одинаковых позициях сравниваемых строк находятся разные символы, большей считается та строка, у которой в этой позиции находится символ с большим кодом. Например, ’Mariya’ > ’Marina’, а ’Вася’>’Василий’.

Копирование символов из строки. Функция **Copy(<строка>, <номер>, <количество>)** возвращает подстроку заданной <строки>, начинающуюся в позиции, указанной <номером> и содержащую заданное <количество> символов. Например,

```
Write(Copy('Информатика',3,5); {выводится слово "форма"}.
```

Поиск, вставка и удаление в строках. Функция **Pos(<подстрока>, <строка>)** производит поиск <подстроки> в <строке> и возвращает номер символа <строки>, с которого начинается <подстрока>, или ноль – если указанной <подстроки> нет в <строке>. Например,

```
N:=Pos('за','розарий'); {результат: N=3}.
```

Процедура **Delete(<строка>,<номер>,<количество>)** удаляет из <строки> заданное <количество> символов, начиная с позиции, указанной <номером>. Например,

```
Delete('розарий',5,3); {результат процедуры - 'роза'}.
```

Процедура **Insert(<подстрока>,<строка>,<номер>)** вставляет заданную <подстроку> в <строку>, начиная с позиции, заданной <номером>.

```
Insert('Pascal','Turbo ',7);{Результат процедуры - 'Turbo
Pascal'}.
```

В следующем ниже примере в строке S подстрока 'белый орел' заменяется подстрокой 'белый лебедь'.

```
N:=Pos(' ',S);  {определили номер символа подстроки в строке
S}
Delete(S,N+6,4); {удаляем слово 'орел' в сстроке S}
Insert('лебедь',S,N+6);   {вставляем подстроку 'лебедь' в
строку S,начиная с позиции N+6}.
```

Пример 3. Вывести на экран заданную строку в обратном порядке.

Решение.

```
Program Reverse;
Var S:string; {обрабатываемая строка}
   I:byte; {переменная цикла}
 Begin
  Write('введите строку: ');
  Readln(S);
  For i:=Length(S) downto 1 do Write(S[i]);
 End.
```

Упражнения

1. Написать программу, в которой:

а) по заданному символу выводится его внутренний код;

б) по заданному коду выводится соответствующий ему символ.

2. Написать программу, в которой вводятся Ваши фамилия, имя и отчество, подсчитываются длина вводимого текста и сколько раз в нем встречается буква "о".

3. Написать программу, в которой одной строкой вводятся Ваши <фамилия>, <имя> и <отчество>, а выводится сообщение "Здравствуйте, < имя > < отчество >!".

4. Даны два слова. Определить, сколько начальных букв первого слова совпадает с начальными буквами второго слова.

Варианты самостоятельных заданий

1. Дано название футбольного клуба. Напечатать его на экране «столбиком».
2. Составить программу, формирующую строку, состоящую из любого заданного количества любых одинаковых символов.
3. Дано предложение. Напечатать все его буквы и.
4. Дан текст. Подсчитать общее количество «+» и «-», содержащихся в тексте.

5. Дан. Текст. Определить, сколько в нем предложений.
6. Дано предложение. Определить, сколько в нем гласных букв.
7. Дано предложение. Определить, каких букв в нем больше: м или н.
8. Дано предложение. Определить, есть ли в нем буква а. В случае положительного ответа найти также порядковый номер первой из них.
9. Дано предложение. Определить количество букв н, предшествующих первой запятой предложения.
10. Дана последовательность слов. Проверить, правильно ли в ней записаны буквосочетания ча и ща. Исправить ошибки.
11. Дано предложение, в котором имеются одна буква с и одна буква т. Определить, какая из них встречается раньше.
12. Дано предложение. Все буквы е в нем заменить на букву и.
13. Дано предложение. Все буквы, стоящие на четных местах, заменить на «-».
14. Дано предложение. Заменить в нем все вхождения буквосочетания да на нет.
15. Дано слово. Поменять местами его вторую и пятую буквы.
16. Дано предложение. Удалить из него все буквы о, стоящие на четных местах.

Контрольные вопросы

1. Какие операции можно выполнять над символьными строками?
2. Укажите стандартные функции для работы с символьными строками.
3. Для чего предназначены встроенные функции `ord`, `chr`, в Паскале?
4. Какое максимальное количество символов может содержать строка?
5. Как узнать текущую длину строки?
6. Какое значение возвратит функция `pos('name','My name is Ivan')`?
7. Какие существуют функции и процедуры для работы со строками?

ЛАБОРАТОРНАЯ РАБОТА № 7.
ФАЙЛЫ

Цель работы: формирование совокупности знаний об особенностях организации работы с файлами; овладение практическими навыками разработки, программирования вычислительного процесса с использованием файлов; умений анализировать исходные данные задачи; получение дальнейших навыков по отладке и тестированию программы; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их.

Теоретический материал

Под файлами будем понимать именованную область внешней памяти компьютера, предназначенную для долговременного хранения программ, данных и пр. Любой файл имеет три основных особенности:

1) у него есть имя;
2) он содержит компоненты одного типа, который определяет и тип файла;
3) длина файла переменна и ограничена только емкостью устройств внешней памяти.

Файловый тип

Файл в языке Pascal – структурированный тип данных, содержащий последовательность компонентов одного типа. Он во многом похож на массивы, но есть и существенная особенность – число элементов (компонентов) файла не фиксируется, т.е. длина файла переменна.

Элементы файла записываются или считываются последовательно, друг за другом с помощью устройства, называемого указателем файла. В каждый момент доступен для записи или чтения только тот элемент файла, на который установлен указатель.

Файловые переменные также имеют специфическую особенность: над ними нельзя выполнять никакие операции, доступные для других типов данных (например, складывать, сравнивать, присваивать и пр.).

Описание файловой переменной

Файловую переменную можно описать одним из трех способов:

```
<файл. перем.> : file of <тип>;
<файл. перем.> : text;
<файл. перем.> : file;
```

Здесь `<файл. перем.>` – имя файловой переменной, образуемое по правилам языка Pascal;

`<тип>` – тип компонентов файла.

Первый способ определяет так называемый типизированный файл (тип компонентов файла задается программистом), второй способ – текстовый файл (его компонентами являются строки), третий способ – нетипизированный файл (в данной главе мы его рассматривать не будем).

Отметим, что тип файла определяет способ хранения информации в файле.

Пример описания файловых переменных:

```
var
 f1 : file of real; {элементы файла – вещественные
                    числа}
 f2 : text;         {элементами файла являются строки}
```

Некоторые стандартные процедуры и функции для файлов всех типов

Любой внешний файл, находящийся на внешнем носителе, становится доступным программе только после выполнения особой процедуры

Assign(<файл. перем .> , <имя внеш. файла>);

Здесь <файл. перем.> – имя переменной файлового типа, описанной в программе;

<имя внеш. файла> – полное имя файла на внешнем носителе.

Процедура **Assign** устанавливает однозначное соответствие между внешним файлом и файловой переменной в данной программе. Например:

```
var
font : text;
. . .
BEGIN
. . .
Assign(font, 'C:\ Мои документы\ Группа РД-91\ Даруда\
bob.pas ');
. . .
```

После того, как файловая переменная связана с внешним файлом процедурой **Assign**, файл нужно инициировать – указать для данного файла направление передачи данных.

Файл можно инициировать:

- только для чтения;
- только для записи;
- для чтения и записи одновременно.

Инициация для чтения. Осуществляется стандартной процедурой

Reset(<файл. перем.>);

Здесь <файл. перем.> – имя файловой переменной, описанной в программе и связанной с соответствующим внешним файлом при помощи процедуры Assign. Процедура Reset открывает существующий файл для чтения. При этом указатель устанавливается на первый элемент файла.

Инициация для записи. Осуществляется стандартной процедурой

Rewrite(<файл. перем.>);

Здесь <файл. перем.> – то же, что и в предыдущем случае. Процедура открывает новый файл. Если файл с этим именем уже существует, то все его элементы уничтожаются (т.е. внешний файл, указанный в процедуре Assign, становится “пустым”).

Закрытие файла. Файлы, открытые (инициированные) ранее процедурами Reset или Rewrite, после завершения операции чтения или записи следует закрыть с помощью процедуры

Close(<файл. перем.>);

Однако связь с внешним файлом, установленная процедурой Assign, при этом сохраняется.

Конец файла. Как уже отмечалось, операция чтение-запись для файлов проводится поэлементно, при этом указатель файла связан с текущим элементом. Стандартная функция логического типа

EOF(<файл. перем.>) : boolean;

возвращает значение true, если указатель текущего элемента файла находится за его последним элементом и false в остальных случаях.

Примечание. Признаком конца файла является служебный символ, называемый eof и имеющий код 26, т.е. – #26.

Чтение и запись файлов

Чтение файла

Под чтением файла понимается ввод данных из внешнего файла, находящегося на внешнем носителе, в оперативную память компьютера.

Для чтения файла в программе необходимо выполнить следующие действия (здесь и ниже предполагается, что процедура `Assign` уже выполнена):

1) Открыть с помощью процедуры `Reset` файл для чтения;

2) Используя процедуры `Read` или `Readln`, ввести данные для чтения:

```
Read(<файл. перем.>, <сп. ввода>);
Readln(<файл. перем.>, <сп. ввода>);
```

Здесь `<сп. ввода>` – список ввода, содержащий одну или более переменных. Количество переменных в списке ввода не должно превышать количество элементов в файле;

3) Закрыть с помощью процедуры `Close` файл для чтения.

Запись файла

Под записью в файл понимается вывод результатов программы из оперативной памяти компьютера на внешнее устройство.

Для записи файла в программе необходимо выполнить следующие действия:

1) Открыть с помощью процедуры `Rewrite` файл для записи;

2) Используя процедуры `Write` или `Writeln` вывести данные из программы:

```
Write(<файл. перем.>, <сп. вывода>);
Writeln(<файл. перем.>, <сп. вывода>);
```

Здесь `<сп. вывода>` – список вывода, содержащий константы, переменные или выражения, записываемые в файл;

3) Закрыть с помощью процедуры `Close` файл для записи.

Отметим, что в случае типизированного файла тип файла и тип данных в списке ввода или вывода должны совпадать.

Текстовый файл

Текстовые файлы играют особую роль на практике, поэтому рассмотрим их особо. Эти файлы содержат некоторый текст, который состоит из обычных символов, объединенных в строки. Следовательно, текстовый файл трактуется как совокупность строк переменной длины. Каждая строка заканчивается специальным признаком конца строки eol, который представляет собой комбинацию служебных символов #13 и #10. Кроме того, в конце файла, как и для других файлов, ставится признак конца файла `eof` – символ #26.

Напомним, что описание текстового файла имеет вид:

```
<файл. перем .> : text;
```

Чтение файла

Осуществляется с помощью процедур:

```
Read(<файл. перем.>, <сп. ввода>);
Readln(<файл. перем.>, <сп. ввода>);
```

Здесь `<сп. ввода>` – переменные типа `char, integer, real, string.`

Процедуру `Read` удобно использовать для ввода чисел. В этом случае вводимые числа должны быть отделены друг от друга разделителями, в качестве которых можно использовать пробелы, символ `tab, eol` и некоторые другие. В читаемой строке, содержащей числа, выделяется очередная подстрока (ее концом как раз служит разделитель), которая является символьным представлением целого или вещественного числа. Затем она преобразуется во внутренний формат числа и присваивается соответствующей переменной в списке ввода.

Пример 1. Допустим во внешнем файле ’C: \ Мои документы \ m.dat’ записаны вещественные числа. Написать программу, которая их считывает и помещает в массив длиной не более 1000 элементов.

<u>Решение.</u>

```
Program Reading;
Const N=1000;  {количество вводимых чисел}
```

```
Var
  F : text; {текстовый файл}
  m : array[1..N] of real; {массив}
  i : integer;
BEGIN
  Assign(F, 'C: \ Мои документы \ m.dat'); {связь с внеш.
  файлом}
  Reset(F); {открываем файл на чтение}
  i:=1;
  {чтение из файла в массив}
  while not EOF(F) do
            begin
            Read(F,m[i]); {чтение i-го элемента из файла }
            i:=i+1
            end;
  Close(F); { закрываем файл}
END.
```

Процедура `Readln` выполняет те же действия, что и процедура `Read`, но после чтения очередной строки из файла осуществляется переход к началу следующей строки. Ее удобно использовать для чтения именно данных строкового типа. Покажем это на следующем примере.

Пример 2. Написать программу, которая считывает с внешнего файла текст и выводит его построчно на экран.

Решение.

```
Program TextReading;
Var
   F : text; {текстовый файл}
   s : string; {массив}
   name : string[20]; {имя внеш. файла на диске }
BEGIN
   Write('Введите имя файла на диске, с которого читается
   текст');
   Readln(name);
   Assign(F, name); {связь с внешним файлом}
   Reset(F);  {открываем файл на чтение}
   {чтение текста из файла в массив и вывод его на экран}
   while not EOF(F) do
            begin
            Read(F,s); {чтение очередной строки из файла}
            Writeln(s); {вывод строки на экран}
            end;
   Close(F); { закрываем файл}
END.
```

Запись файла

Осуществляется с помощью процедур:

Write(<файл. перем.>, <сп. вывода>);

Writeln(<файл. перем.>, <сп. вывода>);

Здесь <сп. вывода> – выражения типа `char`, `integer`, `real`, `boolean`, `string`.

При работе данных процедур значения выражений из списка вывода записываются в указанный файл. Процедура `Writeln` отличается от `Write` тем, что после записи выражений, перечисленных в списке вывода, записывается еще и признак конца строки `eol` – комбинация служебных символов #13, #10.

Добавление записей в файл

Встроенная процедура

`Append(<файл. перем.>)`

Инициирует запись в уже существующий текстовый файл для добавления данных в его конец. Указатель при этом устанавливается на конец файла. Если текстовый файл ранее уже был открыт с помощью процедур `Reset` или `Rewrite`, то использование процедуры `Append` приведет к закрытию этого файла и открытию его вновь, но уже для добавления записей.

Упражнения

1. Напишите программу, которая считывает свой собственный файл с диска и:

а) выводит его на экран;

б) инвертирует все строки и записывает файл на диск с новым именем (убедитесь, что повторное применение данной программы к этому файлу восстанавливает первоначальный текст).

2. Используя типизированный файл, запишите в него массив из 1000 случайных чисел в диапазоне 0…1, затем считайте из него записанные данные и выведите их на экран.

3. Дан файл целых чисел. Удвоить его, дописав в конец все его элементы в том же порядке.

4. Дан файл, содержащий какой-то набор вещественных чисел. Удалить из него все отрицательные числа.

5. Вводить с клавиатуры строки и записывать их в файл до тех пор, пока не будет введена пустая строка. Считывать их этого файла строки и выводить на экран их длины и первые три символа.

Варианты самостоятельных заданий

1. В существующем типизированном файле, элементами которого являются числа, изменить на заданное число: а) первый элемент; б) пятый элемент; в) k-й элемент. Новое значение вводится с клавиатуры.

2. . В существующем типизированном файле, элементами которого являются отдельные слова, изменить на заданное число: а) первое слово; б) третье слово; в) s-о слово. Новое слово вводится с клавиатуры.

3. Имеется типизированный файл с числами. Изменить все его элементы, порядковый номер которых кратен трем. Новые значение вводятся с клавиатуры.

4. Имеется типизированный файл, элементами которого являются отдельные слова. Если слово в файле начинается на букву к, т заменить ее на К.

5. Имеется типизированный файл с числами. Найти сумму первого и второго чисел файла.

6. Имеется типизированный файл с числами. Найти произведение первого и последнего чисел файла.

7. Имеется типизированный файл с числами. Найти сумму всех чисел файла.

8. Имеется типизированный файл с числами. Найти количество чисел файла, не превышающих числа а.

9. Имеется типизированный файл с числами. Найти среднее арифметическое положительных чисел файла.

10. Имеется типизированный файл с числами. Найти максимальное число, имеющееся в файле.

11. Имеется типизированный файл, элементами которого являются отдельные слова. Найти количество слов, начинающихся на букву м.

12. Имеется типизированный файл, элементами которого являются отдельные слова. Получить предложение, составленное из слов, порядковый номер которых четный.

Контрольные вопросы

1. Что такое файл? В чем заключается особенность текстовых файлов?
2. В чем преимущество использование файлов по сравнению с массивами?
3. Какие операции можно проводить с файлами?
4. Как производится запись в файл?
5. Как производится чтение из файла?
6. Как получить доступ к элементу файла с заданным номером?
7. Как закрыть файл?
8. Может ли файл состоять только из одного элемента?
9. Какое назначение оператора Assign?
10. Если текстовый файл был открыт с помощью оператора Reset, можно ли будет записать в файл какие-либо данные?
11. Как описываются файловые переменные, которые в дальнейшем будут связаны с текстовым файлом?
12. Какая функция возвращает признак конца файла?

ЛАБОРАТОРНАЯ РАБОТА № 8.
ПРОЦЕДУРЫ И ФУНКЦИИ

Цель работы: формирование совокупности знаний об особенностях организации работы с процедурами и функциями; овладение практическими навыками разработки, программирования вычислительного процесса с использованием процедур и функций; умений анализировать исходные данные задачи; получение дальнейших навыков по отладке и тестированию программы; умений оформлять результаты своей деятельности в современном офисном пакете и аргументировать их.

Теоретический материал

При написании программ на языке Паскаль рекомендуется использовать так называемый системный подход. Большая задача(система) разбивается на ряд меньших по сложности и размеру задач, которые, в свою очередь, могут быть также разделены на задачи следующего уровня и т.д. Такие небольшие задачи оформляются как относительно самостоятельные блоки программ – процедуры и функции. Хорошим стилем считается написание относительно небольших процедур и функций, с которыми удобно работать на экране монитора.

Такой системный или блочный подход позволяет:

– повысить наглядность программ;

– повысить надёжность программ за счёт тщательной отладки отдельных блоков;

– облегчить коллективную работу программистов при написании больших программ;

– использовать и создавать библиотеки процедур и функций.

Функции

Структура функции похожа на общую структуру программ. (см. таблицу 8.1).

Таблица 8.1 – Сравнение структуры программы и функции

	Программа	Функция
Заголовок	**Program** <имя>;	**Function** <имя>(<список форм. пар.>):<тип функции>;
Раздел описаний	<раздел описаний глобальных параметров>	<раздел описаний локальных параметров>

Исполняемая часть	**Begin**
	<раздел операторов>
	End.
		Begin
		<раздел операторов функции>
		End;

В заголовке функции:

`Function` – служебное слово, за которым следует имя функции;

`<сп. форм. пар.>` – список формальных параметров, который показывает количество параметров, их последовательность и типы;

`<тип функции>` – тип функции (тип возвращаемого функцией значения).

В разделе описания функции описываются так называемые локальные параметры: константы, переменные, процедуры и функции более низкого уровня и т.п.

В исполняемой части функции приводятся соответствующие операторы функции, среди них обязательно должен быть хотя бы один оператор вида

`<имя>:=<результат>;`

При этом тип результата должен совпадать с типом функции, указанным в её заголовке.

Функция как и любой другой объект ТП, должна быть описана в разделе описаний программы или процедуры (функции) более высокого уровня.

Пример 1. Особенности функций рассмотрим на примере следующей простой задачи. Написать функцию, которая для заданных двух вещественных чисел возвращает их сумму. Использовать заданную функцию в программе.

Решение: Два варианта решения данной задачи приведены в таблице 8.2

Таблица 8.2. – Использование формальных и глобальных параметров для передачи данных функции

Вариант 1: механизм формальных параметров.	Вариант 2: механизм глобальных параметров.

`Program Task1;`
`var a,b:real;`
`Function Sum(x,y:real):real;`
` Begin`
` Sum:=x+y;`
` End;`
`BEGIN`
` Read(a,b);`
` Write(Sum(a,b));`
`END.`
	`Program Task2;`
	`var a,b:real;`
	`Function Sum:real;`
	` Begin`
	` Sum:=a+b;`
	` End;`
	`BEGIN`
	` Read(a,b);`
	` Write(Sum);`
	`END.`

Формальные и фактические параметры

При описании функции Sum в программе Task1, используются формальные параметры x, y. При обращении к данной функции используются фактические параметры a, b.

Замена формальных параметров фактическими делается так: на место первого формального параметра подставляется первый фактический параметр, на место второго формального параметра – второй фактический и т.д. Поэтому количество и типы формальных и фактических параметров должны совпадать.

Локальные и глобальные параметры

Если функция имеет раздел описаний (такой раздел у функции Sum в таблице 7.2 отсутствует), то в нем описываются так называемые локальные параметры, которые имеют смысл только в пределах данной функции.

Если функция использует идентификаторы, не описанные в ней, то такие идентификаторы полагаются глобальными. Предполагается, что они описаны и им присвоены значения в вызывающей программе или подпрограмме.

Глобальные параметры доступны как программе, так и всем её подпрограммам.

В программе Task2 (см. вариант 2 таблицы 7.2) глобальными параметрами для функции Sum являются переменные a и b, описанные в основной программе.

Таким образом, функция может использовать глобальные параметры двумя способами:

1) использовать механизм формальных параметров (см. вариант 1 в таблице 8.2);

2) непосредственно обращаться к глобальным параметрам по их именам (см. вариант 2 в таблице 8.2).

Пример 2. Как уже отмечалось, в языке ТП отсутствует операция возведения в степень. Написать свою функцию: $y(x) = x^n$, $x \neq 0$, n – целое.

Решение:

```
Function Power(x:real;n:integer):real;
{локальные параметры функции}
var i:integer;
  p:real;
begin
p:=1;
if n<>0 then
```

```
        begin
         for i:=1 to abs(n) do p:=p*x;
         if n<0 then p:=1/p;
        end;
    Power:=p;
    End;
```

Пример 3. Написать функцию, которая для заданного целого числа n возвращает значение true, если n – четное, и false – противном случае.

Решение:

```
Function Result(n:integer):boolean;
Begin
Result:=(n mod 2)=0;
End;
```

Процедуры

Структура процедуры похожа на структуру программы (см. таблицу 8.3).

Таблица 8.3 – Сравнение структуры программы и процедуры

	Программа	Процедура
Заголовок	`Program <имя>;`	`Procedure <имя>(<список форм. пар.>);`
Раздел описаний	`<описание глобальных параметров>`	`<описание локальных параметров>`

Исполняемая часть	`BEGIN`
	`<раздел операторов>`
	`END.`
		`Begin`
		`<Раздел операторов процедуры>`
		`end;`

В разделе описаний процедуры описываются локальные объекты: константы, переменные, процедуры и функции следующего уровня и т.п.

Процедура так же, как и функция, должна быть описана в вызывающей программе (подпрограмме более высокого уровня).

Основные различия между процедурами и функциями

Отметим два основных отличия между процедурами и функциями (см. таблицу 8.4).

Таблица 8.4 – Основные отличия процедур и функций

Функция	Процедура
Возвращает одно значение, которое связывается с именем функции.	Может возвращать любое количество значений, которое связывается с именем процедуры.
Может использоваться в выражениях, как обычная переменная, например, `y:=x+ln(x);`	Нельзя использовать в выражениях, например, нельзя написать y:=x+write(x);

Обращение к процедуре

Для вызова процедуры из основной программы или другой подпрограммы следует записать имя процедуры и список фактических параметров, которые должны совпадать по количеству и типам с формальными параметрами.

Пример 4. Для заданных двух вещественных чисел написать процедуру, которая возвращает их сумму и разность. Использовать данную процедуру в программе.

Решение:

```
Program Task4;
{описание глобальных объектов программы}
```

```
var x,y,sp,sm:real;

Procedure Sum_min(x1,x2:real; var s1,s2:real);
 Begin
 s1:=x2+x2;
 s2:=x1-x2;
End;

{раздел операторов программы}
BEGIN
 Read(x,y); {ввод исходных данных}
 Sum_min(x,y,sp,sm); {обращение к процедуре}
 Write(sp,sm); {вывод результатов}
END.
```

Примечания: 1) Процедуру Sum_min можно было написать без списка формальных параметров, т.е. использовать механизм глобальных параметров;
2)В приведённом списке формальных параметров перед переменными s1,s2 стоит служебное слово `var` – его назначение объясняется ниже.

Параметры-значения и параметры-переменные

Формальные параметры, если они приводятся при описании процедуры, можно разделить на несколько категорий. Мы будем рассматривать только две категории:

– параметры-значения;
– параметры-переменные.

Их отличие показано в таблице 8.5 (в качестве примера используется процедура Sum_min из задачи 4).

Таблица 8.5 – Основные отличия параметров-значений и параметров-переменных

Параметры-значения	Параметры-переменные
Признаком является отсутствие служебного слова `var`.	Признаком является наличие служебного слова `var`.
Передаются в процедуру через стек в виде копии их значений. Поэтому процедура не может менять параметры x и y вне процедуры.	Передаются через стек, но напрямую Эти параметры процедура может изменять в основной программе.
В качестве фактических параметров могут выступать любые выражения соответствующего типа.	В качестве фактических параметров должны использоваться только переменные соответствующего типа.

В качестве несколько упрощённого правила можно использовать следующую схему определения категории параметров:

Допустимые значения типов формальных параметров процедур и функций

В заголовках процедур и функций в списке формальных параметров необходимо указывать их тип. Правилами языка Паскаль определено, что можно использовать либо стандартные типы (определёнными разработчиками языка), либо так называемые предопределённые типы (определённые программистом ранее). Вводить новый тип в списке формальных параметров не допускается. Рассмотрим это на следующем примере.

Пример 5. Написать функцию, которая для массива D[1..n] вещественных чисел возвращает его среднеарифметическое значение.

Решение: По правилам языка мы не можем ввести заголовок функции типа следующего:

```
Function Average(D:array [1..n] of real):real;
```

поскольку при этом фактически вводится новый тип. Чтобы правильно написать заголовок, мы можем в основной программе ввести новый соответсвующий тип, а затем его использовать в заголовке функции, как это сделано в приведенном ниже фрагменте программы.

```
Program Task5;
Const n=<…>; {длинна массива}
Type Arrtype=array[1..n] of real;
Var ...

Function Average(D:Arrtype):real;
 Var i:integer;
   s:real;
 Begin
 S:=0;
 For i:=1 to n do s:=s+D[i];
 Average:=s/n;
End;
...

{раздел операторов программы}
BEGIN
...
END.
```

Как видно из приведённого примера, новый тип вводится в разделе описаний программы с помощью следующей конструкции:

```
Type <имя типа>=<тип>;
```

Здесь `Type` – служебное слово, `<имя типа>` – идентификатор вводимого типа; `<тип>` – вводимый тип.

Рассмотрим еще один пример.

Пример 6. Для заданного массива вещественных чисел написать процедуру нахождения минимального и максимального значений.

Решение:

```
Program Task6;
Const k=< … > {длинна массива}
Type NT=array[1..k] of real; {новый тип массив}
Var    d: NT; {переменная нового типа}
 Dmin,Dmax:real;

{описание процедуры}
Procedure MinMax(n:integer; mas: NT; var Min,Max:real);
Var i:integer;
Begin
 Min:=mas[1];
 Max:=mas[1];
 For i:=2 to n do
     Begin
     If Min>mas[i] then Min:=mas[i];
     If Max<mas[i] then Max:=mas[i];
     End;
End;
```

```
...
     {раздел операторов программы}
     BEGIN
     ...
     MinMax(k,D,Dmin,Dmax); {обращение к процедуре}
     ...
     END.
```

Локализация имён

Считается, что все идентификаторы, приведённые в разделе описаний внутри процедуры или функции, локализованы в них. При входе в процедуру (функцию) не только становятся доступными объявленные в них локальные имена, но и сохраняются доступ ко всем именам верхнего уровня.

Для иллюстрации этого правила часто используется следующая аналогия: **прозрачный потолок – непрозрачный пол**, т.е. все объекты сверху видны, снизу нет.

Если имена глобальных и локальных переменных совпадают, то они трактуются как разные переменные: в процедурах и функциях работают только одноимённые локальные переменные, а глобальные считаются недоступными. Для примера рассмотрим следующую задачу.

Пример 7. Какой результат будет получен при работе следующей программы?

```
Program Task7;
{раздел описания программы}
var i:integer;
Procedure Act;
 {раздел описания процедуры}
 var i:integer;
 begin
 write(i);
 end;
{раздел операторов программы}
BEGIN
i:=1;
Act;
END.
```

Решение: В программе и процедуре используется переменная i. Поскольку она описана как в процедуре, так и в программе, то это разные переменные с одинаковым именами. По правилам языка внутри процедуры доступна только одноимённая локальная переменная. Поэтому процедура `write(i)` выведет значение локальной переменной i, которой не присвоено определённое знеачение (в таких случаях говорят, что данная переменная неинициирована, т.е. она имеет некоторое случайное значение).

Рекурсия

Язык Паскаль допускает, чтобы процедуры или функции были рекурсивными.

Рекурсия – это такой способ организации вычислительного процесса, при котором процедура или функция в собственном разделе операторов обращается сама к себе. В ряде случаев этот приём позволяет существенно повысить эффективность программирования. Классическим примером рекурсии является следующая задача.

Пример 8. Написать функцию нахождения факториала: $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$.

Решение:

```
Function Factor(n:integer):integer;
Begin
If n<0 then writeln('ошибка в задании n')
     else {правильное значение n}
```

```
        if n=0 then Factor:=1
             else Factor:=n*Factor(n-1);
End;
```

Следует отметить, что при использовании рекурсии повышается вероятность зацикливания программы.

Упражнения

1. Написать функцию, которая по заданному целому числу n>1 возвращает значение true, если n-простое число и false – в противном случае.

2. Написать процедуру, которая по заданным значениям коэффициентов a,b,c возвращает корни квадратного уравнения $ax^2+bx+c=0$.

3. В языке Паскаль нет операции возведения в степень. Написать специальную функцию Power(a,b) возведения вещественного числа a в нецелую степень b, т.е. $P=a^b$, используя следующий алгоритм:

$$p=\begin{cases}\exp(b\cdot\ln(a)), & a>0\\ \exp(b\cdot\ln(|a|)), & a<0\\ 1, & a=b=0\\ 0, & a=0, b\neq 0\end{cases}$$

4. Написать функцию, которая для заданного значения x рассчитывает сумму ряда:

$$1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^i}{i!}+\ldots.$$

Расчёт суммы прекратить после выполнения условия $\left|\frac{x^i}{i!}\right|<err$, где err – заданная погрешность расчёта.

5. Написать процедуру, которая для заданного целого числа n>1 и массива вещественных чисел $x_1,x_2,\ldots,x_n$ вычисляет следующие значения

$$M=\frac{\sum_{i=1}^{n}x_i}{n};\quad D=\sqrt{\frac{\sum_{i=1}^{n}(x_i-M)^2}{n-1}}.$$

6. Написать процедуру, которая для заданного массива вещественных чисел возвращает отсортированные по возрастанию и убыванию значения данного массива.

Варианты самостоятельных заданий

1. Даны стороны двух треугольников. Найти сумму их периметров и сумму их площадей. Определить процедуру для расчета периметра и площади треугольника по его сторонам.

2. Дан первый элемент и разность арифметической прогрессии. Написать рекурсивную функцию для нахождения n-го элемента прогрессии.

3. Дан первый элемент и разность арифметической прогрессии. Написать рекурсивную функцию для нахождения суммы n элементов прогрессии.

4. Дан первый элемент и знаменатель геометрической прогрессии. Написать рекурсивную функцию для нахождения n-го элемента прогрессии.

5. Дан первый элемент и знаменатель геометрической прогрессии. Написать рекурсивную функцию для нахождения суммы n элементов прогрессии.

6. Составить программу, в результате которой величина a меняется значением c величиной b, а величина c – c величиной d.Определить процедуру, осуществляющую обмен значениями двух переменных величин.

7. Написать рекурсивную функцию для вычисления k-го члена последовательности Фибоначчи.

8. Найти максимальный элемент массива из n элементов, используя рекурсивную функцию.

9. Написать рекурсивную функцию для вычисления индекса максимального элемента массива из n элементов.

10. Определить значение z = max(a, 2b)*max(2a-b),b), использовав функцию.

Контрольные вопросы

1. Что такое процедура?
2. В чем различие между функцией и процедурой?
3. Какие существуют отличия между процедурами и функциями?
4. Зачем перед некоторыми переменными в заголовочной части подпрограмм необходимо ставить `var`?
5. Какие параметры-переменные называются формальными?
6. Что такое фактические параметры процедуры или функции?
7. Что такое рекурсия?
8. Как оформляется рекурсивная функция или процедура?
9. Могут ли в одной программе процедура и функция иметь одно и то же имя?
10. Может ли программист дать функции имя sin?

Перечень использованных информационных ресурсов

1. Осипов, А.В. PascalABC.NET: Введение в современное программирование / А.В. Осипов. – Ростов-на-Дону, 2019 – 572с. (https://pascalabc.net)
2. Фризен, И.Г. Основы алгоритмизации и программирования (среда PascalABC.Net) / И.Г. Фризен. – М.: Форум, 2020. – 392с.
3. Андреева, Т.А. Программирование на языке Pascal / Т.А. Андреева. – М.: ИНТУИТ, 2020. – 277с.
4. Тюльпинова, Н.В. Технология алгоритмизации и программирования на языке Pascal / Н.В. Тюльпинова. – Саратов: Вузовское образование. – 2019. – 244с.
5. Бабушкина, И.А. Практикум по объектно-ориентированному программированию / И.А. Бабушкина, С.М. Окулов, – 3-е изд. (эл.) – М: Бином. Лаборатория знаний, 2012. –366 с.