МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (МАДИ)»
Махачкалинский филиал

КАФЕДРА МиИ

Магомедова З.У.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по программированию

МАХАЧКАЛА - 2014

УДК 519.6

Методические указания к выполнению лабораторных работ по дисциплине «Информатика», Махачкала, 2014 г. – 67стр.

Методические указания предназначены для выполнения лабораторных работ по дисциплине «Информатика» студентами очной и заочной формы обучения следующих специальностей очного и заочного обучения:
080502 Экономика и управление на предприятии (на транспорте);
080502 Экономика и управление на предприятии (в строительстве)
190601 Автомобили и автомобильное хозяйство
190701 Организация перевозок и управление на транспорте
190700 Организация и безопасность движения
080505 Управление персоналом
080100 Бухгалтерский учет, анализ и аудит
190603 Сервис транспортных и технологических машин и оборудования (строительные, дорожные и коммунальные машины).

Указания содержат описания по темам:

- программирование линейных алгоритмов;
- программирование разветвленных алгоритмов;
- программирование циклических алгоритмов;
- Работа с массивами
- Сообщения об ошибках

Составитель:

старший преподаватель кафедры МиИ **Магомедова З.У.**

Рецензент:

кандидат ф.м.н., доцент каф.
прикладной математики ДГУ **Абдурагимов Э.И.**

Печатается согласно Постановлению Ученого совета Махачкалинского филиала МАДИ (ГТУ) от “___” _____ 2014 г.

Содержание

Требования к оформлению лабораторной работы

Контрольная работа должна содержать следующие основные пункты:

- ✓ Титульный лист
- ✓ Введение
- ✓ Цели работы, номер варианта
- ✓ Краткие теоретические сведения
- ✓ Блок-схему
- ✓ Программу
- ✓ Результаты созданной программы
- ✓ Выводы.

Порядок решения задачи на ЭВМ

1) Прочитать внимательно условие задачи.
2) Выделить входные данные, вычислить область определения входных данных.
3) Сформулировать результат решения задачи, выделить выходные данные, вычислить область значений выходных данных.
4) Записать математическое решение задачи, построив логически полную систему решений, выделить и описать особые точки, ввести рабочие переменные.
5) Исследовать все данные, каждому из них назначить имя, выделить константы, переменным назначить тип в соответствии с областью определения или значений. Описать данные.
6) Ввести входные данные и проверить их на корректность.
7) Запрограммировать алгоритм решения задачи, описанный в п.4.
8) Вывести выходных данных на экран.

I. Линейные алгоритмы и программы

1.1. Примеры реализации линейных алгоритмов на языке Паскаль

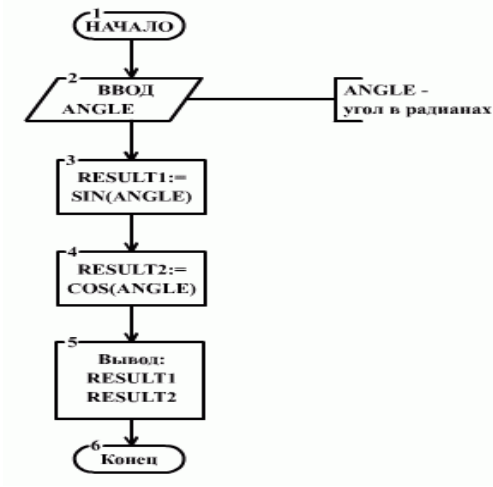
Задача 1

Дан угол в радианах. Вычислить синус и косинус угла.

Словесное описание алгоритма решения этой задачи следующее:

Ввести ANGLE; (угол)
Вычислить RESULT1 по формуле sin(ANGLE);
Вычислить RESULT2 по формуле cos(ANGLE);
Вывести RESULT1, RESULT2

Описание алгоритм в виде блок-схемы:



Программа на языке Паскаль:

```
{Эта программа вычисляет значение синуса (Result1) и косинуса (Result2)
для угла Angle, заданного в радианах}
Var
  Angle : real;                  {Входные данные}
  Result1, Result2: real;        {Выходные данные}
Begin
  Write ( 'Задайте значение угла в радианах ');
   ReadLn (Angle);
   Result1 := sin(Angle);
   Result2 := cos(Angle);
```

```
  WriteLn('синус равен ',Result1:6:3);
  WriteLn('косинус равен ',Result2:6:3);
End.
```

В данной программе значения переменных Result1 и Result2 печатаются в таком формате (:6:3), где 6 позиций отводится на вывод всего числа, включая знак числа и десятичную точку, а 3 позиции - под дробную часть. Форматный вывод можно задать таким образом Result1:0:3. При этом под дробную часть будет отведено 3 позиции, а под целую часть будет выделено столько позиций, сколько цифр в целой части, и одна позиция будет выделена под десятичную точку.

Следует обратить внимание, что текст программы структурирован. Описания данных и операторы сдвинуты вправо относительно зарезервированных слов на одну позицию курсора. Все операторы имеют одинаковый сдвиг. Это значит, что все операторы одинаково «подчиняются» словам BEGIN и END.

Здесь рассмотрена очень простая задача, которая демонстрирует лишь применение математических функций, операторов ввода, вывода и присваивания. Следует обратить внимание на правильный ввод данных. Сначала выдается на экран приглашение или пояснение, что нужно ввести, курсор не переводится на другую строку. Далее идет оператор ввода. Также происходит вывод данных: сначала пояснение, которое можно заменить именем данного, потом - значение. Но в этой задаче приведены все способы описания алгоритма.

Задача 2

Дан радиус круга. Определить длину окружности и площадь круга.

Программа на языке Паскаль:

```
{Эта программа по заданному радиусу вычисляет длину окружности и площадь круга}
    Var
      Radius : Real;
      Square, Len : Real;
    Begin
      Write (' введите радиус круга ');
```

```
  ReadLn (Radius);
  Len := 2*PI*Radius;
  Square := PI * Radius * Radius;
  WriteLn(' Длина окружности =', Len :6:2);
  WriteLn(' Площадь круга =', Square :6:2);
End.
```

Эта задача демонстрирует, что число π специально задавать не нужно, оно уже есть под именем pi или PI.

Задача 3

Вычислить два значения X^3 и X^{10}, используя четыре операции умножения для заданного значения X.

При решении задачи нужно вспомнить, что при умножении чисел, которые являются степенями одного числа, степени складываются.

Словесное описание алгоритма решения этой задачи следующее:

Ввести X;
Вычислить X2=X*X;
Вычислить X3=X*X2;
Вычислить X5=X2*X3;
Вычислить X10=X5*X5;
Выдать на печать X3,X10.

Программа на языке Паскаль:

```
{Эта программа по заданному значению X вычисляет два значения X^3 и
X^10, используя четыре операции умножения}
Var
  x,x2,x3,x5,x10 : real;
Begin
  Write(' Введите значение X =');
  Readln(x);
  x2:=x*x;
  x3:=x2*x;
  x5:=x2*x3;
  x10:=x5*x5;
  writeln('x в 3-ей степени =' ,x3:0:3);
```

```
    writeln('x в 10-ой степени =' ,x10:0:3);
End.
```

Эта задача демонстрирует введение рабочих переменных.

Задача 4

Дано случайное трехзначное число, 100<a<999. Выдать на экран количество десятков в этом числе.

Словесное описание алгоритма:

Из математики нам известно, что всякое трехзначное число можно представить в виде:

a = количество сотен*100 + количество десятков*10 + количество единиц.

Поэтому эту задачу можно решить с помощью операций целочисленного деления.

Количество единиц равно a mod 10, т.е. остаток от целочисленного деления на десять. Количество десятков можно определить двумя операциями. Сначала отделим от трехзначного числа двузначное число, которое содержит первые две цифры трехзначного числа, потом от этого двухзначного числа последнюю цифру, которая и будет количеством десятков. Количество сотен в трехзначном числе определить просто, это целая часть от деления его на 100.

Программа на языке Паскаль:

```
Var
  a, des: integer;
begin
  randomize;
  a:= random(900)+100;
  writeln('a=',a);
  des:=(a div 10)mod 10;
  writeln('des=', des);
end.
```

Задача 5

Дано пятизначное число 10000<a<99999. Переставить местами первую и последнюю цифры.

Разбор Алгоритма. Диапазон возможных значений таков, что само число должно быть типа longint, такой же тип должно иметь и выходное данное – новое число. А каким типом нужно описать промежуточные данные? Для решения задачи мы используем алгоритм, описанный в задаче 3. Представим число в виде: $a=a_4*10000+a_3*1000+a_2*100+a_1*10+a_0$ Тогда решением задачи будет число $b= a_0*10000+a_3*1000+a_2*100+a_1*10+a_4$ При вычислении нового числа b нужно помнить, как работает компьютер. В каждый момент времени делается только одна операция, ее результат записывается в память компьютера. Сколько байт памяти будет выделено для записи этого промежуточного результата? Рассмотрим результат выполнения первого умножения $a_0*10000$. Константа 10000 из диапазона типа integer или word, под нее будет выделено 2 байта. По условию задачи цифра, которая обозначает количество единиц, может иметь значение от 0 до 9. Рассмотрим самый худший случай, путь $a_0=9$, тогда в результате умножения мы получим 90000, а это число должно быть описано как longint. Поэтому все данные этой задачи нужно описать одним типом – longint. Все цифры числа вычислять не обязательно, как показано в программе.

Программа на языке Паскаль:

```
Var
  a, b a0,a4 : longint;
begin
  write('input 10000<a<99999, a=');
  readln(a);
  a0:=a mod 10;
  a4:=a div 10000;
  b:=a-a0+a4-a4*10000+a0*10000;
  writeln('b=', b);
end
```

Задача 6

Вычислить расстояние между двумя точками, заданными координатами (x_1, y_1) и (x_2, y_2), где координаты – целые числа.

Разбор алгоритма.

Это типичная задача, решаемая в графическом режиме на Паскале. Координаты точек на графическом экране могут принимать значения от 0 до 639 (в зависимости от типа монитора) по оси x и от 0 до 479 по оси y.

Алгоритма решения этой задачи очень простой: $l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. А теперь рассмотрим, какого типа должны быть данные, чтобы получить правильный результат. Пусть x_1=0, x_2=500, y_1=0, y_2=500. d_x=x_2-x_1=500, d_y=y_2-y_1=500. Если возвести эти разности в квадрат, то получим числа 250000, это число типа longint. Поэтому типом longint должны быть описаны все целые переменные или d_x и d_y, т.к.в формулу входят квадраты именно этих величин.

Программа на языке Паскаль

```
Var
  x1, y1, x2, y2 : integer;
  dx, dy : longint;
  l : real;
begin
  write('x1='); readln(x1);
  write('y1='); readln(y1);
  write('x2='); readln(x2);
  write('y2='); readln(y2);
  dx:=x2-x1;
  dy:=y2-y1;
  l:=sqrt(sqr(dx)+sqr(dy));
  writeln('l=', l:0:3);
end
```

1.2. Лабораторная работа №1. Программирование лиейных алгоритмов

Цель работы:

1. Построение схемы линейного алгоритма;
2. Изучение структуры программ на языке TurboPascal
3. Использование оператора присваивания.
4. Использование процедур ввода - вывода;

5. Использование стандартных математических функций.

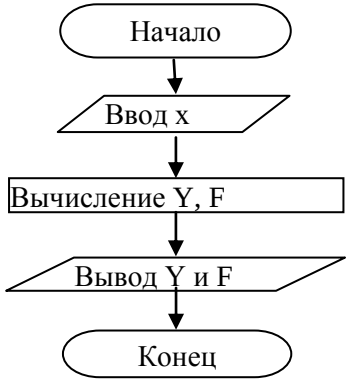
Типовой пример

Составить схему алгоритма и программу для вычисления значений функций Y и F для заданных значений переменной x и постоянных a и b. Значения переменной x>=0. Включить в программу комментарии. Вывести на экран значения F,Y для соответствующих значений x.

$Y = e^{-ax} \cdot (x \cdot Sin(ax+b) - \sqrt{x} \cdot Cos(bx));$

$F = \ln \sqrt{|\, ax^2 + b \,|} - 1.$

Схема линейного алгоритма вычислений



Текст программы

```
Const  a=2.0; b=3.0;
Var  x,Y,F:  real;
{Раздел операторов}
Begin
    Writeln(' Введите x>=0 '); {Запрос на ввод x}
    Readln(x);                 {Ввод значения x}
    {Вычисление значений Y и F}
    Y:= Exp (-a*x)*(x* Sin (a*x+b) -Sqrt (x)* Cos (b*x));
    F:= Ln (Sqrt( Abs (a*x*x-b)))-1;
    {Вывод результатов}
    Writeln('  При x =  ',x,'   Y =  ',Y,'   F =  ',F);
End.
```

1.3. Варианты самостоятельных заданий

1. $Y = \ln|x| \cdot \left(x \cdot arctg(ax) - \sqrt{|x^3|} + \ln(|bx|+3)\right);$
   $F = \sqrt{\left(ax^2 + bx\right)} - e^{-ax}; \quad a = 2{,}5; \quad b = 0{,}5; \quad при\ x = 8\ и\ x = 1.$
2. $Y = x^5 \cdot \left(x \cdot arctg(a+x) - \sqrt{|x-a|} + \ln(|x|+1)^2\right);$
   $F = \sqrt{\left(ax^2 + bx\right)} \cdot e^{-ax} + \ln(|x|+1); \quad a = 0{,}5; b = 1{,}5; при\ x = 9{,}5\ и\ x = 1{,}5.$
3. $Y = (x+a)^5 \cdot \left(arctg\ x - \sqrt{(x-a)^2|} + \ln^2\left(x^2+1\right)\right);$
   $F = \sin x - e^{-ax} + \ln(|x-a|+2); \quad a = 2{,}5; \quad b = 0{,}5; \quad при\ x = 8{,}5\ и\ x = 0{,}5.$
4. $Y = x^5\left(arctg(a+x)^2 - \sqrt{|(x-a)^2|}\right);$
   $F = \sqrt{\left(ax^2 + bx\right)} - e^{-ax} + ax \cdot \ln(x+a); \quad a = 4; \quad b = 0{,}25; \quad при\ x = 8\ и\ x = 1.$
5. $Y = \left(x^5 \cdot Cos(a+x) - \sqrt[3]{|(2x-a)|} + \ln\left((x-2)^2 + 2\right)\right);$
   $F = \sqrt{\left(ax^2 + bx\right)} \cdot e^{-ax} + \ln(|a-x|+3); \quad a = 4; b = 2; при\ x = 3\ и\ x = 1.$
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. $Y = \ln|x-b| \cdot \left(ax \cdot arctg(a+x) - \sqrt{|x^3 - 1|} + \ln(|b+x|+1)\right);$
   $F = 1 + \sqrt{\left(ax^2 + b\right)} - ae^{-ax}; \quad a = 1{,}5; \quad b = 0{,}5; \quad при\ x = 4\ и\ x = 1.$
9. $Y = \left((x-a) \cdot arcctg(a+x) - \sqrt[3]{|(x+a)|} + x \cdot \ln(a+x)\right);$
   $F = \sqrt{\left(ax^2 + x\right)} + ae^{-ax} + \ln(|x|-3); \quad a = 2; b = 0{,}5; при\ x = 3\ и\ x = 1{,}5.$
10. $Y = \left(x \cdot Sin^2(a+x)^3 - \sqrt[3]{(x-a)} + \ln((x-b)+1)\right);$
    $F = \sqrt{\left(ax^2 + x\right)} + a / e^{-ax} + \ln(1-x); \quad a = 2{,}5; b = 0{,}5; при\ x = 0{,}5\ и\ x = 1.$
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. $Y = (x-a)^3(arctg(b+x) - \sqrt{(x-a)^2} + \ln(ax^2+1));$
$F = Sin(ax) - e^{-x} + \ln(x-a+2); \quad a=5; b=2; \quad при\ x=5\ и\ x=1.$

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

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

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

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

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

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

19. $Y = \left(x \cdot Sin^2(a+x)^3 - \sqrt[3]{(x-a)} + \ln((x-b)+1)\right);$
$F = \sqrt{\left(ax^2+x\right)} + a/e^{-ax} + \ln(1-x); \quad a=2{,}5; b=0{,}5; при\ x=0{,}5\ и\ x=1.$

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

II. Разветвляющиеся алгоритмы и программы

Определение. *Разветвляющимся* называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется *ветвью алгоритма.*

Признаком разветвляющегося алгоритма является наличие операций проверки условия. Различают два вида условий – *простые* и *составные*.

Простым условием (отношением) называется выражение, составленное из двух арифметических выражений или двух текстовых величин (иначе их еще называют *операндами*), связанных одним из знаков:

< - меньше, чем...
> - больше, чем...
<= - меньше, чем... или равно
>= - больше, чем... или равно
<> - не равно
= - равно

Условный оператор дает возможность, в зависимости от заданного в нём условия, выполнить то или иное действие, что позволяет разветвлять вычислительный процесс.

Этот оператор имеет следующий вид:

IF *<условие>* ***Then*** *<оператор 1>* ***Else*** *<оператор 2>;*

где *условие* - логическое выражение;

оператор 1, оператор 2 - любые операторы языка ТР.

Условный оператор работает следующим образом: если *<условие>* принимает значение *TRUE* (*истина*), то выполняется *оператор 1*, а *оператор 2* пропускается; если - *FALSE* (*ложь*), то *оператор 1* пропускается, а выполняется *оператор 2*.

На месте *оператора 1* или *оператора 2* может стоять группа операторов, заключенных в операторные скобки (*Begin ... end*).

Часть *Else* (оператор *2*) может быть опущена. Тогда при значении *TRUE* условного выражения выполняется *оператор 1*, в противном случае выполняется оператор, стоящий за *оператором IF*.

Кроме того, ТР обладает средствами безусловного выхода из программных блоков (процедур, функций или основной программы), что позволяет завершать программу или подпрограммы без предварительных переходов по меткам. Для этого используются системные процедуры **EXIT** и **HALT**.

Вызов Exit вызывает завершение работы только того программного блока, в котором он используется.

Процедура Halt завершает выполнение всей программы.

2.1. Примеры реализации развлетвленных алгоритмов на языке Паскаль

Задача 1

Для заданного значения X вычислить значение функции F(X) = Y, которая определяется следующим образом:

$$Y = \begin{cases} 4x, & если \quad x \geq 0; \\ 0, & если \quad x < 0 \end{cases}$$

В этом алгоритме для организации ветвления можно использовать условие X >= 0. Если это условие верно, то надо вычислять Y по формуле Y = 4X, если не верно, то Y = 0.

Блок-схема алгоритма:

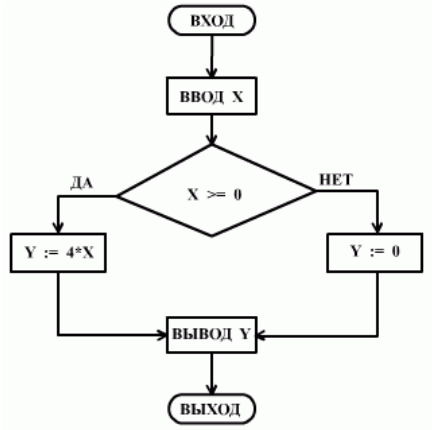


Рисунок 2.4. Блок-схема алгоритма решения задачи 1

На языке Паскаль этот алгоритм может быть записан следующим образом:

```
Var
  x,y : real;
Begin
  Writeln (' Введите значение X ');
  Read(x);
  If  x >= 0 then y := 4 * x
  Else y := 0;
  Writeln('y=', y:6:3);
End.
```

Задача 2.

Даны действительные числа a, b, c. Вычислить корни уравнения $ax^2 + bx + c = 0$.

На языке Паскаль этот алгоритм может быть записан следующим образом:

```
Var
  a,b,c,D,xl,x2 : Real;
Begin
  WriteLn('Введите коэффициенты квадратного уравнения');
  Write(‘a=’); Readln(a);
  Write(‘a=’); Readln(a);
  Write(‘a=’); Readln(a);
  D := b * b - 4 * a * c;
   If  D < 0 Then  WriteLn('При данных коэффициентах квадратное уравнение не имеет  рациональных корней')
  Else
    If  D = 0 Then  begin
          x1:=b / (2 * a);
          WriteLn('При данных коэффициентах квадратное уравнение имеет два одинаковых корня x1=x2: ',x1:6:3);
          end
    Else
      Begin
          xl := (-b + sqrt(D)) / (2 * a);
          x2 := (-b - sqrt(D)) / (2 * a);
          WriteLn('Корни квадратного уравнения: ', xl:6:3, ' ', x2:6:3)
      End;
End.
```

Задача 3.

Чтобы получить одну молекулу серной кислоты H_2SO_4 нужно 2 атома водорода (H), 1 атом серы (S) и 4 атома кислорода (O). В химическом реакторе имеется A атомов водорода, B атомов кислорода и C атомов серы.

Составьте алгоритм и напишите программу, вычисляющую, сколько молекул серной кислоты может получиться из такого количества атомов?

Алгоритм решения задачи:

1) Переменные A,B,C.M1,M2,K : тип - целый;
2) Ввести A,B,C;
3) Если одно из чисел отрицательно, то прекратить решение;
4) Вычислить целую часть от деления A на 2 (водород): M1 = цел.часть (A/2);
5) Вычислить целую часть от деления B на 4 (кислород): M2 = цел.часть (B/4);
6) Найти минимальное из трех чисел C, Ml, M2: K = min(C, Ml, M2);
7) Выдать на экрань: В реакторе образуется К молекул кислоты.

Алгоритм нахождения минимального из трех заданных чисел. Эта подзадача может быть решена несколькими способами. Можно исследовать все возможные соотношения между тремя числами: равны, попарно равны, не равны и т.д. Но можно ввести рабочую переменную min и с ее помощью вычислить количество молекул.

min=минимальное из чисел(a,b)

количество молекул = минимальное из чисел(min, c).

Программа на языке Паскаль.

```
Var
 A,b,c,m1,m2,min:integer;
Begin
 Write('a='); readln(a);{ввод данных}
 Write('b='); readln(b);
 Write('c='); readln(c);
 If (a<=0)or(b<=0)or(c<=0) then begin {проверка на корректность}
  Writeln('Не верно введены данные');
  Halt;
  Readln;  End;
 M1:=a div 2;{вычисление количества комплектов молекул}
 M2:=b div 4;
 If a<b then min:=a else min:=b;{вычисление min из трех чисел}
 If c<min then min:=c;
 Writeln('Количество молекул=',min);
 Readln; {жду нажатия клавиши Enter, не обязательный оператор}
End.
```

Задача 4.

Два прямоугольника, расположенные в первом квадранте, со сторонами, параллельными осям координат, заданы координатами своих левого верхнего и правого нижнего углов. Для первого прямоугольника это точки (xl,yl) и (x2,0), для второго - (x3,y3), (x4,0).

Составьте алгоритм или напишите программу, определяющую, пересекаются ли данные прямоугольники, и вычисляющую общую площадь, если они пересекаются.

Алгоритм решения этой задачи может быть записан следующим образом:

Нарисовать первый квадрант декартовой системы координат и пересекающиеся прямоугольники. Обозначить левую точку пересечения (a, b), правую точку (лежащую на оси X) - (c,d).

Алгоритм решения задачи:

1) Переменные xl,yl,x2,x3,y3,x4,a,b,c,d,S : тип - вещественный;
2) Ввести xl,yl,x2,y2,x3,y3,x4;
3) Если y1 или y3 равны 0, то прекратить решение;
4) Вычислить минимальное из чисел yl и y3 : b = min(yl, y3);
5) Вычислить максимальное из чисел xl и x3 : a = max (xl, x3);
6) Вычислить минимальное из чисел x2 и x4: c = min(x2,x4);
7) Присвоить d = 0 (по условию задачи);
8) Если (a < c) И (b≠0) то начало
9) Выдать на экран : «Прямоугольники пересекаются»;
10) Вычислить общую площадь S = (c - a) / b;
11) Выдать на экран S
12) Конец иначе выдать на экран «Прямоугольники не пересекаются или касаются».

Программа на языке Паскаль.

```
Var
 X1,y1,x2,y2,x3,y3,x4,a,b,c,d,s:real;
Begin
 Write('x1='); readln(x1);{ввод данных}
 Write('y1='); readln(y1);
 Write('x2='); readln(x2);
 Write('x3='); readln(x3);
```

```
 Write('y3='); readln(y3);
 Write('x4='); readln(x4);
 If (y1=0)or(y3=0) then begin
  Writeln('Не верно введены данные');
  Halt;
  Readln;
 End;
 If y1<y3 then b:=y1 else b:=y3;
 If x1>x3 then a:=x1 else a:=x3;
 If x2<x4 then c:=x2 else c;=x4;
 D:=0;
 If a<c then begin
  Writeln('Прямоугольники пересекаются');
  S:=(c-a)/b;
  Writeln('Общая площадь=',s:0:3);
 End;
 Readln;
End.
```

Задача 5.

Дан треугольник ABC с координатами вершин A(Ax,Ay), B(Bx.By), C(Cx.Cy), где числа Ax, Ay, Bx, By, Cx, Cy - целые. Составьте алгоритм, определяющий, является ли этот треугольник равнобедренным. Входные данные задать с помощью датчика случайных чисел из диапазона [0,20].

Алгоритм решения задачи:

1) Ввести координаты вершин Ax, Ay, Bx, By, Cx, Cy;
2) Вычислить квадраты длин сторон AB, BC и AC по формулам

   $AB^2 = (Bx-Ax)^2 + (By-Ay)^2$;

   $BC^2 = (Cx-Bx)^2 + (Cy-By)^2$;

   $AC^2 = (Ax-Cx)^2 + (Ay-Cy)^2$;
3) Если $AB^2 = BC^2$ или $BC^2 = AC^2$ или $AB^2 = AC^2$, то
   Вывести на экран "Треугольник ABC равнобедренный" иначе "Треугольник ABC не равнобедренный".

Программа на языке Паскаль:

```
Const
 Eps=0.00001;
```

```
Var
 Ax,Ay,Bx,By,Cx,Cy : Real;
 AB2,BC2,AC2 : Real; { квадраты сторон }
Begin
 Randomize;
 Writeln ('Координаты точки A ');
 Ax:=random(20)+random;
 Ay:=random(20)+random;
 Writeln('Ax=',Ax:0:2,'Ay',Ay:0:2);
 Writeln ('Координаты точки B ');
 Bx:=random(20)+random;
 By:=random(20)+random;
 Writeln('Bx=',Bx:0:2,'By',By:0:2);
  Writeln ('Координаты точки C ');
 Cx:=random(20)+random;
 Cy:=random(20)+random;
 Writeln('Cx=',Cx:0:2,'Cy',Cy:0:2);
 AB2 := sqr(Ax - Bx) + sqr(Ay - By);
 BC2 := sqr(Cx - Bx) + sqr(Cy - By);
 AC2 := sqr(Ax - Cx) + sqr(Ay - Cy);
 if (abs(AB2 - BC2)<eps) or (abs(AC2 - BC2)<eps) or (abs(AB2 - AC2)<eps)
then
   Writeln ('Треугольник равнобедренный ')
 else
   Writeln (Треугольник не равнобедренный ');
End.
```

Примечание

В этой задаче впервые встретилась задача сравнения вещественных чисел. Вещественные числа нельзя сравнивать с помощью знака «равно», т.к. эти числа представляются в памяти машины с некоторой точностью, которая зависит от типа данных. Поэтому нужно ввести точность, в нашем примере это константа eps, и сравнивать по правилу: если модуль разности чисел меньше точности, то числа равны.

Задача 6.

Известно, что 2004 год 21 века начался в четверг. Какой день недели был в К-тый день этого года?

Алгоритм решения задачи

1) Ввести К
2) Проверить, К на корректность. Число К должно быть больше или равно 1. Если год високосный, то К меньше или равно 366, иначе меньше 366. Год является високосным, если его номер делится на 400 или не делится на 100 и делится на 4. Внутри века условие может быть записано короче, год високосный, если его номер делится на 4.
3) Вычислить остаток от деления К на 7-количество дней в неделю.
4) Выдать на экран день недели, учитывая, что если остаток будет 1, то это будет четверг, 2 – пятница и т.д.

Программа на языке Паскаль

```
Const
 G=2004;
Var
 k,ost:integer;
begin
 wtite('k='); readln(k);
 if (k<1)then {проверка на корректность}
  if ((g mod 4=0)and(k>366))or((g mod 4<>0)and(k>365))then begin
   Writeln('Не верно введены данные');
   Halt;
   Readln;
  end;
 ost:=k mod 7;
 case ost of
  1: writeln('четверг');
  2: writeln('пятница');
  3: writeln('суббота');
  4: writeln('вескресенье');
  5: writeln('понедельник');
  6: writeln('вторник');
```

```
 0: writeln('среда');
end;
 readln
end.
```

2.2. Лабораторная работа №2. Программирование разветвленных алгоритмов

Цель работы:

1. Построение схемы разветвляющегося алгоритма;
2. Изучение структур разветвления (условных операторов) ТР.

Типовой пример

Составить схему алгоритма и программу для вычисления значений функции F для заданных значений переменной x и постоянных a, b.

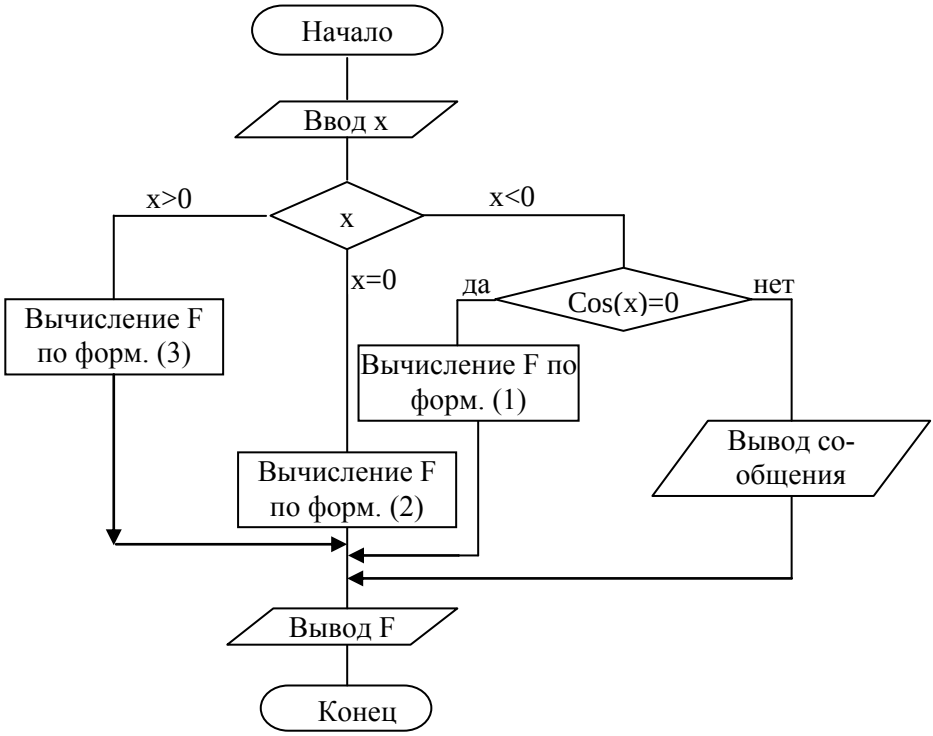
$F = a\ tgx + \sqrt{|x-2|}$, при $x < 2$; (1)

$F = (a^2 - b^2)\ Cos\ \pi$, при $x = 2$; (2)

$F = (x-2)^3\ Sin\ \pi x/2$, при $x > 2$. (3)

Примечание: При составлении алгоритма и программы не забывайте анализировать знаменатель (он не должен равняться нулю), учитывайте область определения математических функций, значения подкоренных выражений и типы данных, участвующих в выражениях.

В рассматриваемом примере значение подкоренного выражения всегда больше 0, арифметические функции Sinx и Cosx определены для любых значений x.



При вычислении значений функции tgx необходимо использовать выражение tgx = sinx /cosx. В связи с этим нужно анализировать значение, которое может принимать знаменатель: cosx не должен равняться 0. Все данные вещественного типа.

Текст программы

с использованием структуры if ... then ...

```
 Const  a=2.0; b=3.0;
 Var  x,F: real;
Begin   {Раздел операторов}
Writeln(' Введите x');      {Запрос на ввод x}
Readln(x);                  {Ввод значения x}
 if  x>2  then  F:= (x-2)*(x-2)*(x-2)* Sin(Pi *x/2);
 if  x=2  then  F:= (a*a-b*b)* Cos (Pi);
 if  x<2  then if Cos (x) <> 0  then
          F:= a*Sin(x)/Cos(x)+Sqrt (Abs(x-2))
           else  begin Writeln(' Знаменатель равен 0');
```

```
                exit {Выход из программы } end;
{Вывод результатов}
      Writeln(' При  x =  ',x:6:2,'  F =  ',F:8:3);  readln;
       End.
```

Текст программы

с использованием структуры if ... then ... else ...;

```
{ Раздел описаний данных}
        Const  a=2.0; b=3.0;
        Var  x,F:  real;
       Begin {Раздел операторов}
      Writeln(' Введите x');      {Запрос на ввод x}
      Readln(x);                  {Ввод значения x}
       if 0 x>2  then  F:= (x-2)*(x-2)*(x-2)* Sin(Pi/2*x)
       else   if  x=2  then  F:= (a*a-b*b)* Cos (Pi)
          else if Cos(x) <>  0 then
              F:= a* Sin(x)/Cos(x)+ Sqrt(Abs(x-2))
             else  begin Writeln(' Знаменатель равен 0');
                exit { Выход из программы } end;
      Writeln(' При x= ',x:6:2,' F= ',F:8:3); readln; {Вывод результатов}
       End.
```

2.3. Варианты самостоятельных заданий

Написать программу на языке Pasal, для реализации разветвляющего алгоритма.

1. $$y=\begin{cases}\left(Cos(a)+6Sin(5bx)\right)^2-\dfrac{2}{7}x^4, & x<-1;\\ Cos(x)-Sin(x), & x>1;\\ Sin(x)+Cos(x), & -1\le x\le 1;\end{cases}$$ a = -1, b = 0.5. ;

2. $$y=\begin{cases}Cos(ax)+3Sin(bx)+10x^3, & x<-1\\ 2x-Sin\ (\dfrac{x}{a+b}), & x>1\\ Sin\ (x)+13x^2, & -1\le x\le 1\end{cases}$$; *a=-2, c=2,5.*

3. $$y=\begin{cases}2a\cdot Sinx\cdot Cosx-\dfrac{2}{7}x^3, & x>10\\ Cos(ax)+Sin(cx), & x<5\\ Sinx^3+Cos^2x^2, & 5\le x\le 10\end{cases}$$; *a=-3, c=-0.5.*

4. $$y=\begin{cases}\sqrt{2x}+Cosx-\dfrac{a}{b}x^3, & x<-0.5\\ Cos(x)+Sin(x), & x>0.5\\ Sinx^3+Cos^2ax^2, & -0.5\le x\le 0.5\end{cases}$$; *a=-1, b=3.*

5. $$y=\begin{cases}\dfrac{15-2x}{(a+b)^2}-\ln|ax|, & x>2\\ \sqrt{2x+Sin^2x}, & 0\le x\le 2\\ \dfrac{a\cdot Cosx}{b}+2x, & x<0\end{cases}$$; *a=-3, b=-0.5.*

6. $$y=\begin{cases}\dfrac{4x^2}{\sqrt{a-b}}, & 0\le x\le 15\\ \dfrac{a}{b}\ln|x|, & x>15\\ 5\cos x, & x<0\end{cases}$$; *a=-0,5, b=3,2.*

7. $$y=\begin{cases}\dfrac{(2c+b^2)x}{(c+b)^3}, & x>10\\ 2c+x\cdot Sin\left(\dfrac{x}{c}\right), & -10<x\le 10\\ (2c-b^2)x, & x\le -10\end{cases}$$; *b=5, c=-0,3*

8. $$y=\begin{cases}e^{ax}+\dfrac{10}{(x+b)^2}, & x<0\\ \ln|x|, & x>0\\ e^x-\dfrac{\sin x}{a-b}, & x=0\end{cases}$$; *a=-3, b=0,5*

9. $y=\begin{cases}\sin^2(x+a)+5, & x<10\\ \cos^2(x-b), & x>12\\ e^{-abx}, & 10\le x\le 12\end{cases}$; *a=-0,3, b=0,7*

10. $y=\begin{cases}3x^2-\dfrac{|a-c|}{x}, & x>1\\ \cos(2\pi)\cdot(x-a)^3 & x<-1\\ \dfrac{Sinx}{x+c}-Cos^2x, & -1\le x\le 1\end{cases}$, *a=-1, c=0.5*

11. $y=\begin{cases}\cos x+\dfrac{(a-b)^2}{x}, & x>1\\ x-\ln(x+3) & x<-1\\ \dfrac{\sqrt{3x-a}}{a^2+b^2}-3x^2, & -1\le x\le 1\end{cases}$; *a=0,1; b=3*

12. $y=\begin{cases}3x^2-c\cdot Sinx^2, & x=0\\ \sqrt{3x+b^2}\cdot Cos\pi x, & x>0\\ 2tgx-b+\dfrac{Sin\pi}{x}, & x<0\end{cases}$; *b=2; c=-0,5*

13. $y=\begin{cases}\dfrac{\sqrt{x}}{Sin\pi}+Cosx, & x=0\\ \dfrac{(a-b)^2}{x}-\ln(2x-1), & x>0\\ 2x(a^3-b)\cdot Sinx, & x<0\end{cases}$; *a=3; b=-2*

14. $y=\begin{cases}(x-2)^2+Sin(ax), & x=1\\ \dfrac{(a-b)}{x}, & x>1\\ 3ab-x^2+\sqrt{a^2+b^2}, & x<1\end{cases}$; *a=7; b=-1*

15. $y=\begin{cases}3x\cdot Cosx, & x<-1\\ \dfrac{x^2}{a-c}, & -1<x<1\\ \ln|x|-(x-a)^2, & x>1\end{cases}$; *a=-4; c=6*

16. $y=\begin{cases}\sin \pi x-4a^2, & x\ge 6\\ \dfrac{\sqrt{\left|x-3x^2+c\right|}}{x-7}, & 1<x<6\\ Sin(3-x)^2+2c^2x, & x\le 1\end{cases}$; *a=-0,5;c=3*

17. $y=\begin{cases}(b-c)^2-Sin^2(\pi+x), & x\ge 1\\ Sinx+Cos^2x-b^2, & x\le 0\\ \dfrac{Sin\,\pi x}{x}+c, & 0<x<1\end{cases}$; *b=-1, c=4*

18. $y=\begin{cases}e^{5x+4}-b^2, & x>1\\ 2x^4-2Sin\,\pi x, & x<-1\\ \dfrac{Sin^2x}{(a+b)^2}, & -1<x<1\end{cases}$; *a=3; b=-1*

19. $y=\begin{cases}x^4+Sinx, & x<0\\ 2x(a^2-c), & 0\le x\le 1\\ Cos(x^2+c), & x>1\end{cases}$; *a=-1; c=-3*

20. $y=\begin{cases}2x+a^2, & x\le 1\\ e^{bx}+\ln|x|, & 1<x<2\\ Cosx\cdot e^{ax}, & x\ge 2\end{cases}$; *a=2,5; b=1,5*

III. Циклические алгоритмы и программы

3.1. Примеры реализации циклических алгоритмов на языке Паскаль

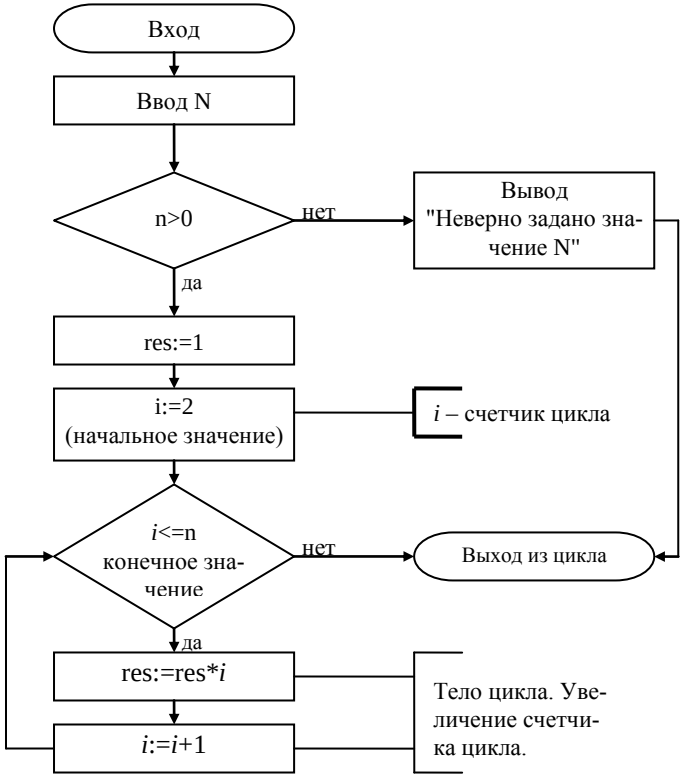
Задача 1.

Дано натуральное число N. Вычислить N! (N-факториал)

Для любого натурального числа N значение N! определяется как произведение всех натуральных чисел от 1 до N: N! = 1*2*3*...*N. (Целое число называется натуральным, если оно больше 0.)

При решении этой задачи удобно использовать цикл со счетчиком. Для вычисления значения N-факториала используется переменная RES, которой предварительно присваивается значение 1. На каждом шаге значение переменной RES умножается на *i*. Счетчик цикла *i* изменяется от 2 до N.

Алгоритм решения данной задачи может быть представлен следующей блок-схемой:



Программа на языке Паскаль:

```
PROGRAM FACTORIAL; {Вычисляет n!}
Var
```

```
    n :Integer;
    i, res :Integer;
 Begin
Write ('Введите  n :');Readln  (n);
if  (n>0)  then
begin
res  :=  1;
For  i  :=  2  to  n  do  res  :=  res * i;
Writeln('N - факториал  = ',  res);
end;
else
Writeln  ('Неверно задано значение N');
Readln;
End.
```

Примечание:

1) Данная задача будет решена правильно, если значение переменной res в процессе вычисления не выйдет из диапазона значений типа Integer. А это возможно только при значениях n, не превосходящих 13.
2) Заголовок программы PROGRAM FACTORIAL; {Вычисляет n!} не является обязательным элементом.
3) Это не единственный алгоритм решения этой задачи. В дальнейшем будут рассмотрены рекурсивный алгоритм и специальный, увеличивающий значение n до 60.

Задача 2.

Дана последовательность N натуральных чисел A_1,..., A_N, элементы которой нужно ввести с клавиатуры. Определить, сколько элементов последовательности являются четными числами.

Программа на языке Паскаль:

```
PROGRAM CH; {Подсчитывает количество четных элементов последовательности}
Var
```

```
n: Integer;
a: Integer; {переменная для хранения элемента последовательности}
 i, kol: Integer;
Begin
Write  ('Введите  n :'); Readln  (n); {ввод количества всех элементов}
 if  (n>0)  then
 begin
 kol  :=  0; {счетчик количества четных элементов}
 For  i  :=  1  to  n  do
 Begin
 Writeln('Введите a[', i, ']  :  ');
 Readln(a);
 if  not  Odd(a)  the  kol := kol + 1;
 {Функция Odd() возвращает значение true, если ее аргумент - нечетное
число, false - если четное}
End;
Writeln('В последовательности ', kol, ' четных чисел');
end
else
Writeln  ('Неверно задано значение N');
Readln;
End.
```

Задача 3.

Представим себе пирамиду из шаров, основанием которой является квадрат со стороной, состоящей из I шаров. Следующий слой состоит из шаров, лежащих в углублениях нижнего слоя, т.е. представляет собой квадрат со стороной из I -1 шаров и т.д. Верхний слой содержит один шар. Требуется определить, сколько шаров потребуется для строительства пирамиды из N слоев.

Программа на языке Паскаль:

```
PROGRAM pir1;
Var
n :Integer;
```

```
i, k :Integer;
Begin
Writeln  ('Введите  число слоев :');
Readln  (n);
if  (n>0)  then
begin
k  :=  1;
For  i  :=  2  to  n  do      k  :=  k + i* i;
Writeln('Для пирамиды из ',  n, ' слоев требуется', k, ' шаров');
End
Else      Writeln  ('Неверно задано значение N');
Readln;
End.
```

Задача 4.

Найти сумму членов вводимой числовой последовательности. Признаком конца последовательности является 0.

Так как в данной задаче длина последовательности заранее не известна, то в программе естественно использовать цикл с предусловием, определяющим условие продолжения цикла. В начале работы программы необходимо задать начальное значение суммы (S:=0) и прочитать первый элемент последовательности для определения значения предусловия на первом шаге работы цикла.

Программа на языке Паскаль:

```
PROGRAM SUMMA1; {Вычисление суммы элементов вводимой последовательности}
Var
S, a :Integer;
Begin
S := 0;
Write  ('Введите  элемент последовательности :');
Read  (a);
While a <> 0 do {условие продолжения цикла}
begin
```

```
S := S + a;
Write  ('Введите  элемент последовательности :');
Readln  (a);
end;
Writeln  ('Сумма  равна =', S:6);
Readln;
End.
```

Задача 5.

Определить, какой максимальной высоты можно построить пирамиду из K шаров. Алгоритм построения пирамиды подробно описан в задаче 3.

Программа на языке Паскаль:

```
PROGRAM pir2;
Var
n :Integer;
k1, k :Integer
Begin
Write  ('Введите  число шаров :');
Readln  (k);
if  (k>0)  then
begin
n := 0;
k1  :=  k
While (k1 >= SQR(n + 1))  do
Begin
n := n + 1;
k1 := k1 - SQR(n);
End;
Writeln('Из ', k , ' шаров можно построить пирамиду из ', n, ' слоев');
Writeln('Останется ',  k1, '  шаров');
end
else  Writeln  ('Неверно задано значение N');
```

```
Readln;
End.
```

Задача 6.

Найти минимальное число элементов последовательности {1/i}: сумма которых 1 + 1/2 + 1/3 +... превышает введенное с клавиатуры значение предельной суммы.

В данной задаче количество вводимых элементов последовательности не известна, его необходимо найти. В нижеприведенной программе предельная сумма вводится в переменную SP. Для вычисления значения этой суммы в этой программе используется переменная S, которой предварительно присваивается нулевое значение (S := 0). На каждом шаге к ранее вычисленному значению S прибавляется i-ый член последовательности (S := S + 1/i), а в переменной i накапливается число рассмотренных элементов последовательности (i := i + 1). Так как суммирование должно выполняться хотя бы один раз, естественно использовать цикл с постусловием.

Программа на языке Паскаль:

```
PROGRAM SUMMA2; {Нахождение минимального числа элементов последовательности 1/i, сумма которых превышает значение предельной суммы}
Var
SP :Real; {предельная сумма}
S :Real; {сумма}
i :Integer; {число введенных элементов}
Begin
S := 0; i := 0;
Writeln  ('Введите предельную сумму:');
Readln  (SP);
Repeat
i := i + 1;
S := S + 1/i;
Until S > SP; {}
Writeln  ('Число элементов  равно =', i:6);
```

```
Writeln  ('Сумма равна =', S:6:3);
Readln;
End.
```

3.2. Лабораторная работа №3 . Циклический алгоритм вычисления значения переменной на промежутке

Цель работы:

1. Построение схем циклических алгоритмов;
2. Использование операторов циклов (повторений):

Типовой пример

Составить 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: $(6x^2\text{-}x^3) > 0$.

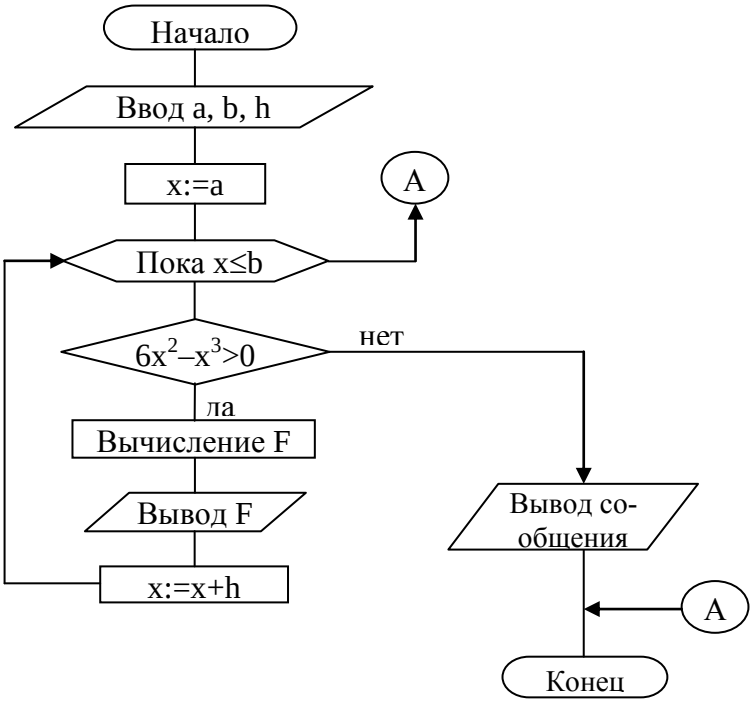
Исходными данными в этой задаче являются значения a, b и h.

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 и программа вычислений зависит от того, какой оператор цикла используется в программе. Рассмотрим три различных варианта решения поставленной задачи.

Вариант 1

Схема алгоритма с использованием цикла While ... Do



Текст программы

{Лабораторная работа 3}

{Программирование циклов с использованием оператора WHILE}

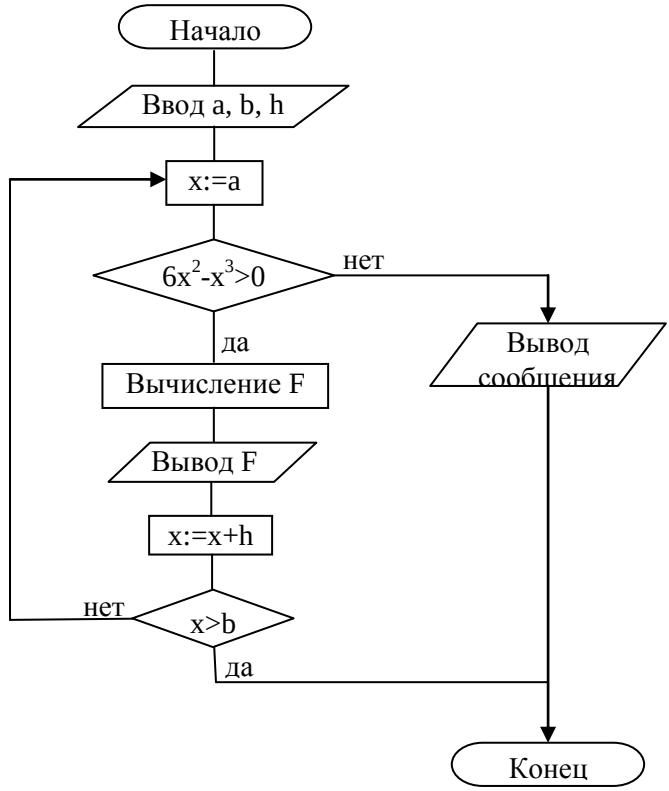
{Студенты гр. ..... Фамилии ...}

```
Uses crt; {Подключение стандартного модуля TP}
Var  x,y,H,A,B: real;
BEGIN
Clrscr; { Очистка экрана}
   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
      Writeln('При x= ',x:5:3,' под знаком логарифма недопустимое
значение')
          Else
```

```
            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; {Конец else}
      x:=x+H;
    End; {Конец while}
Readkey; {Ожидание нажатия любой клавиши}
  END.
```

Вариант 2

Схема алгоритма с использованием цикла Repeat ... Until



Текст программы

```
{Лабораторная работа 3}
{Программирование циклов с использованием конструкции
REPEAT…UNTIL}
{Студенты гр. ... Фамилии ...}
Uses crt; {Подключение стандартного модуля TP}
Var   x,y,A,B,H: real;
BEGIN
   Clrscr; { Очистка экрана}
   Writeln(' Введите A B H');    readln(A,B,H);
   x:=A;
   Repeat
   if  (6*x*x-x*x*x)=<0  then
   Writeln('При x= ',x:5:3, ' под знаком логарифма недопустимое
значение')
   Else  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;
   x:=x+H;
   Until  x>B;
   Readkey;{Ожидание нажатия любой клавиши}
END.
```

Вариант 3

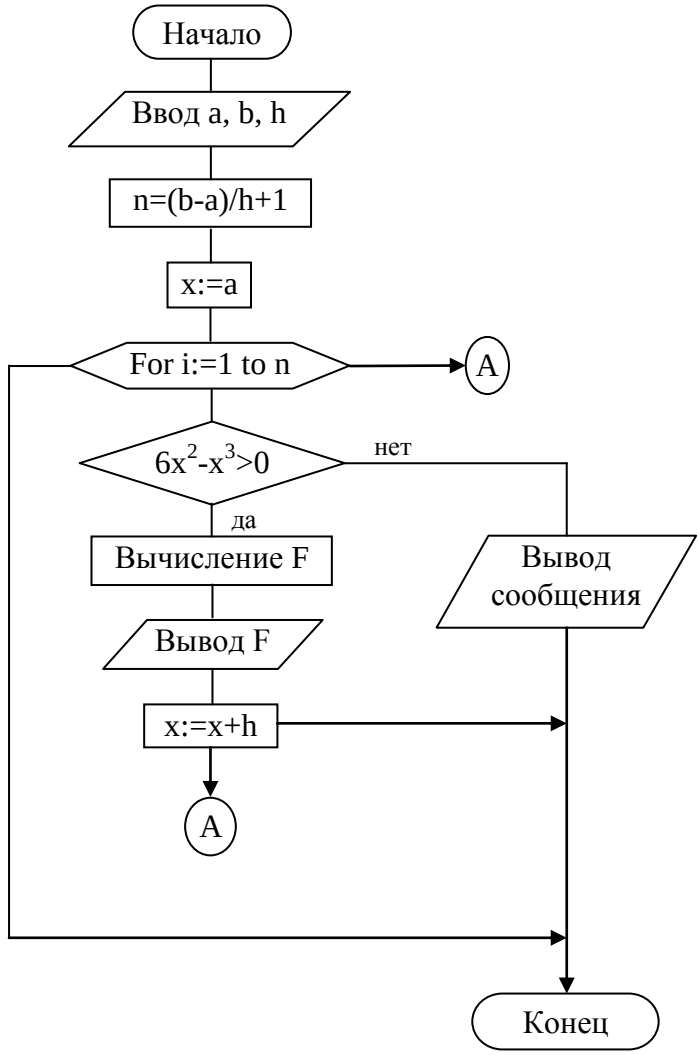
Схема алгоритма с использованием цикла For ... Do

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 результа-

те операции деления. Для этого можно использовать стандартную функцию Round.



Текст программы

{Лабораторная работа 3}

{Программирование циклов с использованием оператора FOR}

{Студенты гр. ....}

Uses crt;

```
Var  x, y, A, B, H:real;
        n, i : integer;
BEGIN
Clrscr;
        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
            Writeln ('При x= ',x:5:3,' под знаком логарифма недопустимое
значение')
            else
                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;
        x:=x+H;
        end;   { Конец For}
  END.
```

3.3. Варианты самостоятельных заданий

1. $Y = xSinx/Cosx$; A = -1.6, B = 1.6, H = 0.2
2. $Y = tgx^2 + ctgx^2$; A = -2, B = 2, H=0.5
3. $Y = Ln(x^2 -9)/x$; A = -6, B = 6, H = 1
4. $Y = xLn(Cosx)$; A = -3, B = 3, H = 0.3
5. $Y = tgx/x$; A = -2, B = 2, H = 0.2
6. $Y = Sinx/(x-5)$; A = -6, B = 6, H = 0.5
7. $Y = x/(1-Cosx)$; A = -4, B = 4, H = 0.5
8. $Y = 1/xtg(x/2)$; A = -3, B = 3, H = 0.5
9. $Y = 1/(x^2 -3)$; A = -4, B = 4, H = 0.5
10. $Y = 1/Ln(x^2/8)$; A = -3, B = 3, H = 0.6
11. $Y = 1/x\ ctg(x/3)$; A = -4, B = 4 , H = 0.5

12. $Y = Ln(16-x^2)/x\ e^x$; A = -6, B = 6, H = 0.5
13. $Y = x\ tg\ x/\ Lnx$; A = -3, B = 3, H = 0.5
14. $Y = x^2 Ln|x|/Sinx$; A = -2, B = 2, H = 0.2
15. $Y = x/Sinx + Ln(x^2 - 4)$; A = - 6, B = 6, H = 0.5
16. $Y = x^2 Cosx/Ln(x^2-1)$; A = -2, B = 2, H = 0.1
17. $Y = xSinx/Ln(x^2-9)$; A = -5, B = 5, H = 0.5
18. $Y = x^2/(Cosx+Ln(x^2-1))$; A = -2, B = 2, H = 0.2
19. $Y = (x^2 + tgx)/Ln(x^2-4)$; A = -4, B = 4, H = 0.8
20. $Y = (x^2 - Cosx)/xLn(x^2-1)$; A = -2, B = 2, H = 0.2

3.4. Лабораторная работа №4. Использование циклических алгоритмов и программ для вычисления суммы членов ряда

Цель работы:

1. Построение схемы итерационного процесса;
2. Приобретение навыков в выборе и использовании операторов цикла.

Типовой пример

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

$$\mathbf{a_n = x^n\ (2n-1)\ /\ n!} \qquad \mathbf{(1)}$$

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

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

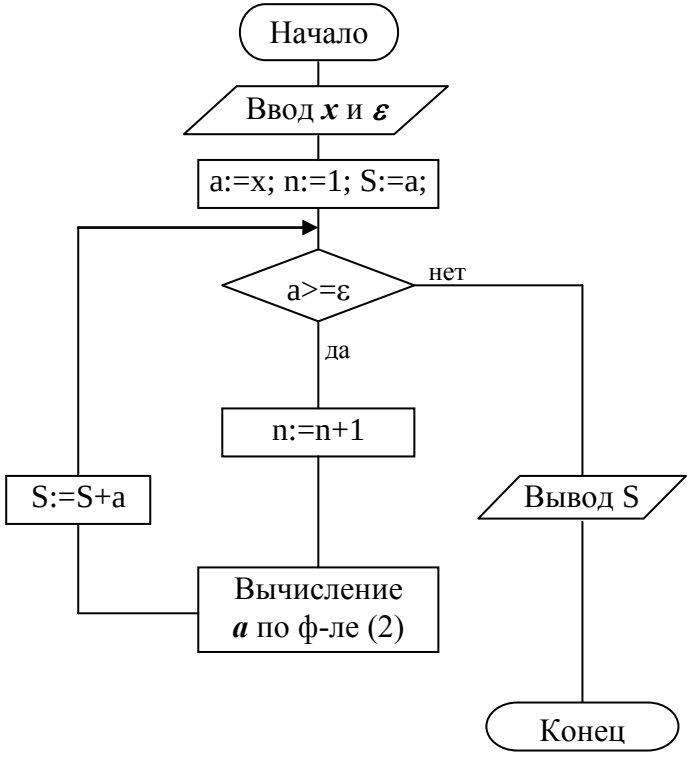
Вычислим a_1 по исходной формуле (1): $\mathbf{a_1 = x(2-1)/1!=x.}$

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).

Схема алгоритма с использованием цикла While ... Do



Текст программы

```
{Лабораторная работа 4}
{Вычисление суммы членов ряда с использованием оператора WHILE}
{Студенты гр. .....  Фамилии…}
        Uses crt; {Подключение стандартного модуля TP}
        Var   x, a, eps, S : real;   n:integer;
        BEGIN
                Clrscr; { Очистка экрана}
```

```
        Writeln(' Введите x,eps');      read(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 (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.

3.5. Варианты самостоятельных заданий

1. $a_n = (2n-1)*x / 2^n$;
2. $a_n = x /((3n-2)(3n+1))$;
3. $a_n = 10n / xn!$;
4. $a_n = xn! /(2n)!$;
5. $a_n = x* 2^n / n!$;
6. $a_n = x (3n * n!) / (3n)!$;
7. $a_n = 3nx / (n+2)!$;
8. $a_n = (2n-1) * x / 2^n$;
9. $a_n = nx / 3^{(n+1)/2}$;
10. $a_n = (x+1) * 5^n / n!$;
11. $a_n = x/n^2$;
12. $a_n = 2nx / (n+1)!$;
13. $a_n = 3nx * 3^n / n!$;
14. $a_n = x(n+1) / (n+2)!$;
15. $a_n = nx / 2^{(n+1)}$;
16. $a_n = 2xn! / (2n)!$;
17. $a_n = xn! / (3n)!$;
18. $a_n = x * 5^n / (n+5)!$;
19. $a_n = x * 2^n /(2n+1)!$;
20. $a_n = x(2n+1)(2n-1)/(2n+1)!$;

IV. Массивы и их описание

Массивы относятся к структурированным типам данных в ТР. Массив состоит из фиксированного числа элементов (компонент) одного типа и характеризуется общим именем. Доступ к отдельным элементам массива осуществляется с помощью общего имени и порядкового номера (индекса или адреса) необходимого элемента.

Имя массива - это любое допустимое в ТР имя, отличное от служебных слов, имен функций и процедур. Массив может быть описан в подразделе Var или в подразделах Var и Type, одновременно.

Первая форма описания - в подразделе Var:

Var Имя_массива: Array[T1] OF [T2];

где Array (Массив) и Of (Из) - служебные слова.

T1 - список типов индексов) в качестве индексов чаще всего используются переменные типа - диапазон, но возможно использование перечислимого типа); T2 - тип элементов массива (любой тип ТР как простой, так и структурированный).

Например:

1) Var Mas: Array[1..10] of integer; - описание массива, состоящего из 10 целых чисел и имеющего имя Mas;
2) Var S: array[1..20] of real; - описание массива, состоящего из 20 вещественных чисел и имеющего имя S;
3) Var Matr:array[1..20,1..20] of byte; - описание массива (матрицы размером 20 на 20), содержащего 400 элементов типа byte.

Если массив объявлен, то к любому его элементу можно обратиться с помощью его имени и индексов. Например:

1) Mas[5]:=21; - пятому элементу массива Mas присвоено значение, равное 21.
2) For i:=1 to 20 do S[i]:=0.0; - присвоение всем 20 элементам массива S значения, равного 0.0 (обнуление массива).
3) For i:=1 to 20 do For j:=1 to 20 do Matr[i,j]:=random(100); - заполнение массива (матрицы) на 400 чисел случайными числами из диапазона от 0 до 100.

Вторая форма описания - в подразделах Var и Type:

Type Имя_типа_массива= Array[T1] OF [T2];
Var Имя_массива: Имя_типа_массива;

Например:

1) Type Tm=array[1..100] of byte; - описание нового типа для массива, состоящего из 100 чисел типа byte
2) Var Mas:Tm; - описание массива с именем Mas.
3) Type Typ_Mas = array[1..100] of char; Var M:Typ_Mas; - описание массива содержащего 100 символов.

1) Type Tmatr = Array[1..25,1..10] of real; Var Matr1,Matr2:Tmatr; - описание матриц с именами Matr1 и Matr2.

Массив с одним индексом называют одномерным, с двумя - двумерным, с тремя - трехмерным и т.д. Число индексов у массива в ТР не ограничивается, но необходимо помнить, что размер массива не должен превышать 64 Кбайт.

Любой двумерный массив представляет собой матрицу: первому индексу можно поставить в соответствие строки, а второму - столбцы матрицы. Кроме того, двумерный массив можно интерпретировать как одномерный, элементами которого является другой одномерный массив.

Описание такого массива имеет вид:

Type tstr=array[1..25] of real;

Var masssiv:array[1..10] of tstr; - это описание равносильно описанию в примере 3, приведенному выше для массивов с именами Matr1 и Matr2.

Оперативная память под элементы массива выделяется на этапе трансляции. В памяти компьютера элементы массива следуют друг за другом.

Если массив двумерный, то память под него выделяется так, что быстрее меняется самый правый индекс. В качестве примера рассмотрим порядок выделения оперативной памяти под массив, описанный следующим образом: Var M:array[1..2,1..4] of byte;

Этот массив будет располагаться в памяти в следующем порядке:

M[1,1]; M[1,2]; M[1,3]; M[1,4]; M[2,1]; M[2,2]; M[2,3]; M[2,4].

В ТР можно одним оператором присваивания передать все элементы одного массива другому массиву того же типа. Например:

```
Var   m1,m2:array[1..10] of word;
        . . .
Begin   . . .
    m1:=m2;   { перезапись из одного массива в другой}
        . . .
End.
```

Для сравнения содержимого двух массивов необходимо использовать оператор цикла с параметром и указываться индексы.

Типизированные константы - массивы

В качестве начального значения используется список констант, отделенных друг от друга запятыми. Список заключается в круглые скобки.

Примеры:

1) Const Mas:array[1..10] of byte=(1,1,1,1,1,1,1,1,1,1); - заполнение массива из 10 целых чисел значением, равным единице.
2) Const massim: array[0..5] of char = ('a','b','c','d','e','f'); - заполнение массива из 6 элементов символами - буквами латинского алфавита.
3) Const Matr: array[1..5,1..2]of byte = ((0,0), (0,0), (0,0), (0,0), (0,0)); - обнуление матрицы из 10 целых чисел.

Замечание: количество переменных в списке констант должно строго соответствовать объявленной длине массива по каждому индексу!

4.1. Примеры реализации одномерных массивов на языке Паскаль

Задача 1.

Заполнить одномерный массив элементами, отвечающими следующему соотношению:

$$a_1=1;\ a_2=1;\ a_i=a_{i-2}+a_{i-1}\ (i = 3, 4, ..., n).$$

Фрагмент программы на языке Паскаль:

```
Read(N); {Ввод количества элементов}
A[1]:= 1;
A[2]:= 1;
FOR I := 3 TO N DO
   A[I] := A[I - 1] + A[I - 2];
```

Другой вариант присваивания значений элементам массива — заполнение значениями, полученными с помощью датчика случайных чисел.

Задача 2.

Заполнить одномерный массив с помощью датчика случайных чисел таким образом, чтобы все его элементы были различны.

Программа на языке Паскаль:

```
Program Create;
```

```
Type Mas = Array[1..100] Of Integer;
Var A : Mas; I, J, N : Byte; Log : Boolean;
Begin
   Write(''); ReadLn(N);
   randomize; A[1] := -32768 + random(65535);
   For I := 2 To N Do
   Begin
      Log := True;
      Repeat
         A[i] := -32768 + random(65535); J := 1;
         While Log and (j <= i - 1) Do
         begin Log := a[i] <> a[j]; j := j + 1 End
      Until Log
   End;
   For i := 1 to N Do Write(a[i]:7); writeln
End.
```

2) ввод значений элементов массива с клавиатуры используется обычно тогда, когда между элементами не наблюдается никакой зависимости. Например, последовательность чисел 1, 2, -5, 6, -111, 0 может быть введена в память следующим образом:

```
Program Vvod;
 Var N, I : Integer;
    A : Array [1..20] Of Integer;
Begin
  Write('Введите количество элементов массива ');  ReadLn(N);
  FOR I := 1 TO N DO
  Begin
   Write('Введите A[', I, '] '); ReadLn(A[I])
End.
```

Над элементами массивами чаще всего выполняются такие действия, как

а) поиск значений;

б) сортировка элементов в порядке возрастания или убывания;

в) подсчет элементов в массиве, удовлетворяющих заданному условию.

Сумму элементов массива можно подсчитать по формуле *S=S+A[I]* первоначально задав *S*=0. Количество элементов массива можно подсчитать по формуле *K=K*+1, первоначально задав *K*=0. Произведение элементов массива можно подсчитать по формуле *P = P * A[I]*, первоначально задав *P* = 1.

Задача 3.

Дан линейный массив целых чисел. Подсчитать, сколько в нем различных чисел. {Подсчет количества различных чисел в линейном массиве. ИДЕЯ РЕШЕНИЯ: заводим вспомогательный массив, элементами которого являются логические величины (False - если элемент уже встречался ранее, True - иначе)}

Программа на языке Паскаль:

```
Program Razlichnye_Elementy;
Var I, N, K, Kol : Integer;
    A : Array [1..50] Of Integer;
    Lo : Array [1..50] Of Boolean;
Begin
  Write('Введите количество элементов массива: '); ReadLn(N);
   FOR I := 1 TO N DO
    Begin
     Write('A[', I, ']='); ReadLn (A[I]);
     Lo[I] := True; {Заполняем вспомогательный массив значениями True}
    End;
  Kol := 0; {переменная, в которой будет храниться количество различных
чисел}
  FOR I := 1 TO N DO
   IF Lo[I] THEN
    Begin
     Kol := Kol + 1;
```

```
  FOR K := I TO N DO {Во вспомогательный массив заносим значение False, если число уже встречалось ранее или совпадает с текущим элементом A[I]}
     Lo[K] := (A[K] <> A[I]) And Lo[K];
   End;
 WriteLn('Количество различных чисел: ', Kol)
END.
```

Тест: $N = 10$; элементы массива - 1, 2, 2, 2, -1, 1, 0, 34, 3, 3. Ответ: 6.

Задача 4.

Дан линейный массив. Упорядочить его элементы в порядке возрастания.

{Сортировка массива выбором (в порядке возрастания). Идея решения: пусть часть массива (по K-й элемент включительно) отсортирована. Нужно найти в неотсортированной части массива минимальный элемент и поменять местами с (K+1)-м}

Программа на языке Паскаль:

```
Program Sortirovka;
Var N, I, J, K, Pr : Integer; A : Array [1..30] Of Integer;
Begin
Write('Введите количество элементов: '); ReadLn(N);
  For I := 1 To N Do
   Begin
    Write('Введите A[', I, '] '); Readln(A[I]);
   End;
  WriteLn;
  For I := 1 To N - 1 Do
   Begin
    K := I;
    For J := I + 1 To N Do If A[J] <= A[K] Then K := J;
    Pr := A[I]; A[I] := A[K]; A[K] := Pr;
   End;
   For I := 1 To N Do Write(A[I], ' ');
```

End.

Тест: $N = 10$; элементы массива - 1, 2, 2, 2, -1, 1, 0, 34, 3, 3.

Ответ: -1, -1, 0, 1, 2, 2, 2, 3, 3, 34.

Если два массива являются массивами эквивалентых типов, то возможно присваивание одного массива другому. При этом все компоненты присваиваемого массива копируются в тот массив,оторому присваивается значение. Типы массивов будут эквивалентными, если эти массивы описываются совместно или описываются идентификатором одного и того же типа. Например, в описании

```
Type Massiv = Array[1..10] Of Real;
Var A, B : Massiv; C, D : Array[1..10] Of Real; E : Array[1..10] Of Real;
```

типы переменных A, B эквивалентны, и поэтому данные переменные совместимы по присваиванию; тип переменных C, D также один и тот же, и поэтому данные переменные также совместны по присваиванию. Но тип переменных C, D не эквивалентен типам переменных A, B, E, поэтому, например, A и D не совместны по присваиванию. Эти особенности необходимо учитывать при работе с массивами.

При решении практических задач часто приходится иметь дело с различными таблицами данных, математическим эквивалентом которых служат матрицы. Такой способ организации данных, при котором каждый элемент определяется номером строки и номером столбца, на пересечении которых он расположен, называется *двумерным массивом* или *таблицей*.

Например, данные о планетах Солнечной системы представлены следующей таблицей:

Планета	**Расстояние до Солнца**	**Относительный объем**	**Относительная масса**
Меркурий	57.9	0.06	0.05
Венера	108.2	0.92	0.81
Земля	149.6	1.00	1.00
Марс	227.9	0.15	0.11
Юпитер	978.3	1345.00	318.40
Сатурн	1429.3	767.00	95.20

Их можно занести в память компьютера, используя понятие двумерного массива. Положение элемента в массиве определяется двумя индексами. Они показывают номер строки и номер столбца. Индексы разделяются запятой. Например: A[7, 6], D[56, 47].

Заполняется двумерный массив аналогично одномерному: с клавиатуры, с помощью оператора присваивания. Например, в результате выполнения программы:

```
Program Vvod2;
 Var I, J : Integer;
    A : Array [1..20, 1..20] Of Integer;
 Begin
  FOR I := 1 TO 3 DO
   FOR J := 1 TO 2 DO A[I, J] := 456 + I
 End.
```

элементы массива примут значения A[1, 1] = 457; A[1, 2] = 457; A[2, 1] = 458; A[2, 2] = 458; A[3, 1] = 459; A[3, 2] = 459.

При описании массива задается требуемый объем памяти под двумерный массив, указываются имя массива и в квадратных скобках диапазоны изменения индексов.

При выполнении инженерных и математических расчетов часто используются переменные более чем с двумя индексами. При решении задач на ЭВМ такие переменные представляются как компоненты соответственно трех-, четырехмерных массивов и т.д.

Однако описание массива в виде многомерной структуры делается лишь из соображений удобства программирования как результат стремления наиболее точно воспроизвести в программе объективно существующие связи между элементами данных решаемой задачи. Что же касается образа массива в памяти ЭВМ, то как одномерные, так и многомерные массивы хранятся в виде линейной последовательности своих компонент, и принципиальной разницы между одномерными и многомерными массивами в памяти ЭВМ нет. Однако порядок, в котором запоминаются элементы многомерных массивов, важно себе представлять. В большинстве алгоритмических языков реализуется общее правило, устанавливающее порядок хранения в памяти элементов массивов: элементы

многомерных массивов хранятся в памяти в последовательности, соответствующей более частому изменению младших индексов.

Задача 5.

Заполнить матрицу порядка *n* по следующему образцу:

1	2	3	...	n-2	n-1	n
2	1	2	...	n-3	n-2	n-1
3	2	1	...	n-4	n-3	n-2
...	...	...	...	...	...	...
n-1	n-2	n-3	...	2	1	2
n	n-1	n-2	...	3	2	1

Программа на языке Паскаль:

```
Program Massiv12;
 Var I, J, K, N : Integer; A : Array [1..10, 1..10] Of Integer;
 Begin
  Write('Введите порядок матрицы: '); ReadLn(N);
   For I := 1 To N Do
    For J := I To N Do
     Begin
      A[I, J] := J - I + 1; A[J, I] := A[I, J];
     End;
    For I := 1 To N Do
     Begin
      WriteLn;
      For J := 1 To N Do Write(A[I, J]:4);
     End
 End.
```

Задача 6.

Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали.

Программа на языке Паскаль:

```
Program Obmen;
 Var N, I, J, Max,Ind, Vsp : Integer;A : Array [1..15, 1..15] Of Integer;
 Begin
  WRITE('Введите количество элементов в массиве: '); READLN(N);
 FOR I := 1 TO N DO
  FOR J := 1 TO N DO
  Begin
   WRITE('A[', I, ',', J, '] '); READLN(A[I, J])
  End;
 FOR I := 1 TO N DO
 Begin
  Max := A[I, 1]; Ind := 1;
   FOR J := 2 TO N DO
      IF A[I, J] > Max THEN
       Begin
        Max := A[I, J]; Ind := J
       End;
   Vsp := A[I, I]; A[I, I] := A[I, Ind]; A[I, Ind] := Vsp
  End;
     FOR I := 1 TO N DO
      Begin
       WriteLn;
       FOR J := 1 TO N Do Write(A[I, J] : 3);
      End; WriteLn
 End.
```

4.2. Лабораторная работа №5. Работа с одномерными массивами

Цель работы:

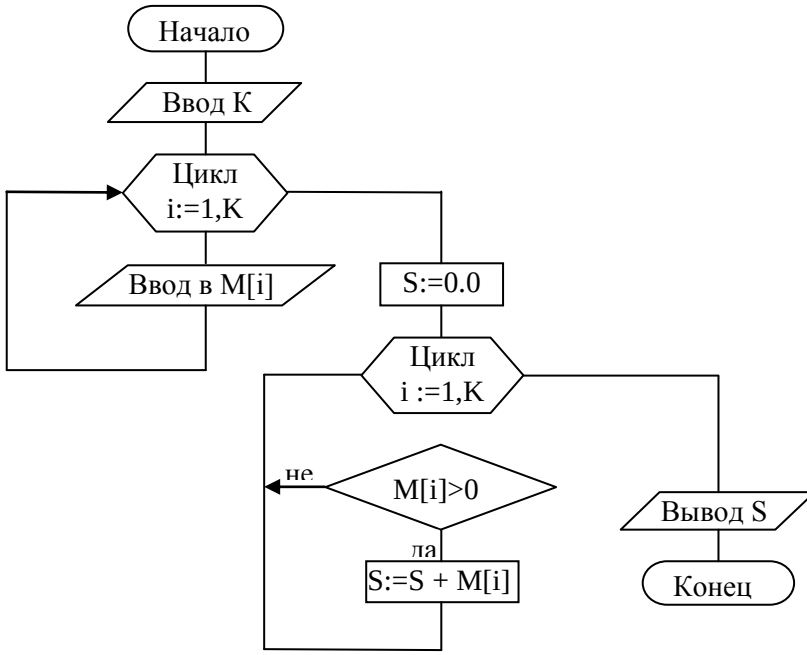
1. Построение схем алгоритмов с использованием массивов;
2. Разработка программ с использованием одномерных массивов

Типовой пример

Составить схему алгоритма и программу определения суммы S всех положительных элементов одномерного массива M, содержащего K ве-

щественных чисел (К≤20). Числа в массив М ввести с экрана дисплея, результат суммирования (S) также вывести на экран.

Схема алгоритма



Текст программы

```
Uses crt; {Подключение стандартного модуля TP}
   Var m:array[1..20] of  real;   {Описание массива вещественных чисел}
   S:real;  {описание переменной для хранения суммы}
   i,K:byte; {описание переменных для счетчика и количества чисел}
   Begin
   Clrscr;   {Очистка экрана}
   Write(' Введите число элементов в массиве К =');
   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:=0.0;   {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.
```

4.3. Варианты самостоятельных заданий

1. Составить схему алгоритма и программу определения суммы наибольшего и наименьшего элементов вещественного массива A={a_i}, i=1,N, N- число элементов массива (N<100). Вывести на экран найденную сумму, а также наибольшее и наименьшее значения и их адреса (индексы) в массиве A.
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 M.
3. Составить схему алгоритма и программу определения суммы всех четных чисел массива K={k_j}, j=1,m, m - число элементов в массиве K (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 T (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 K (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 T (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 К={$к_i$}, 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 К={k_j}, j=1,m, m - число элементов 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.

4.4. Лабораторная работа №6. Работа с двумерными массивами

Цель работы:

1. Построение схем алгоритмов с использованием вложенных циклов;
2. Разработка программ с использованием двумерных массивов

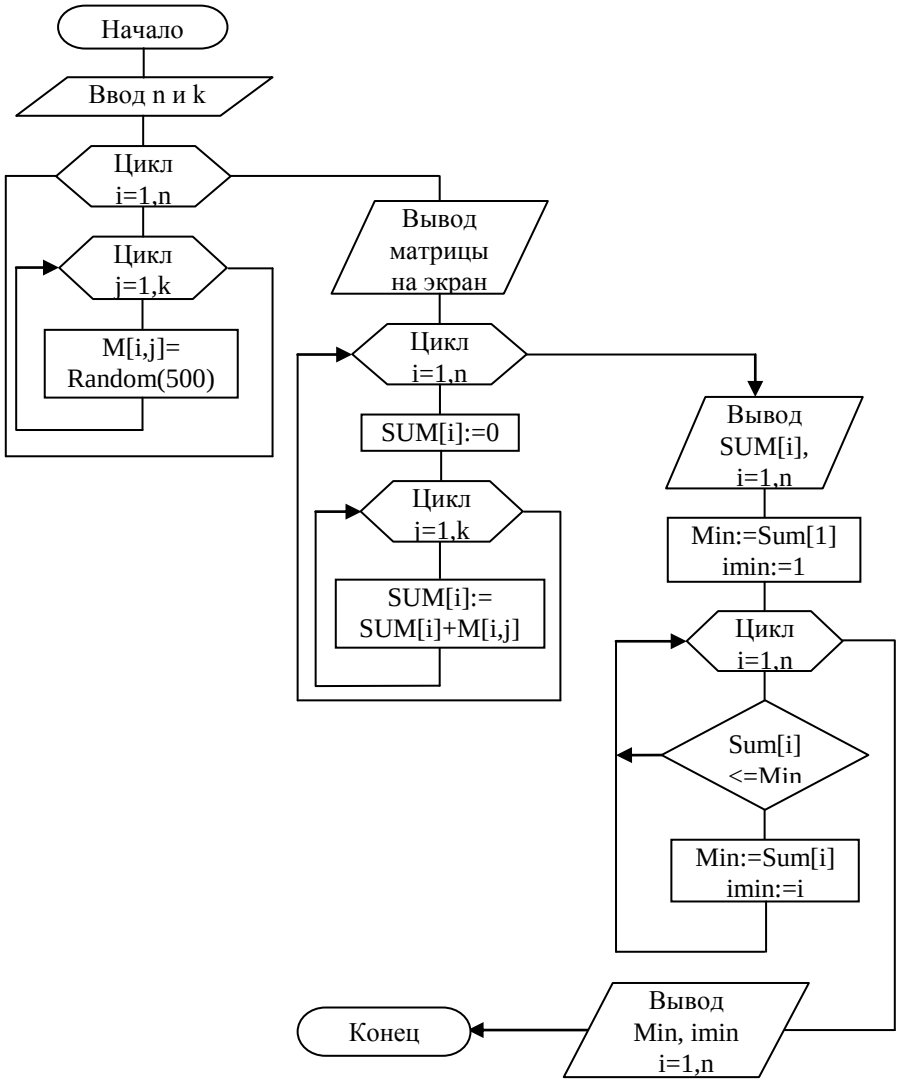
Типовой пример

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.

Схема алгоритма



Текст программы

```
Uses crt;
{ Описание данных}
Var M:array[1..50,1..100]of integer;
    Sum:array[1..50] of longint;
    n,k,i,j,imin:integer;
    Min:longint;
BEGIN
    clrscr;
    {Ввод данных}
    writeln(' Введите число строк и столбцов');
     readln(n,k);
    Randomize;  {Стандартная процедура см. теорию}
    {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);
 readln;
END.
```

4.5. Варианты самостоятельных заданий

1. В прямоугольной матрице размером T*M, имеющей имя MATP содержаться целые числа. T - число строк, M - 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 M*T, имеющей имя MAS содержаться целые числа. M - число строк, T - 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 K*M, имеющей имя MATR содержаться целые числа. K- число строк, M- 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 M*K, , имеющей имя MAT содержаться целые числа. M- число строк, K- число столбцов. Ввести элементы матрицы с клавиатуры. Определить и вывести на экран сумму отрицательных элементов в каждой строке. Определить стро-

ку 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, M- 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 M*K, имеющей имя MAT содержаться целые числа. M- число строк, K- 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 M*K, имеющей имя MA содержаться целые числа. M- число строк, K- 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 M*K, имеющей имя M содержаться целые числа. M- число строк, K- 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 M*K, , имеющей имя MM содержаться целые числа. M- число строк, K- 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 M*K, имеющей имя MAM содержаться целые числа. M- число строк, K- 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.

V. Сообщения об ошибках

5.1. Сообщения об ошибках во время компиляции

1. Out of memory – Выход за границы основной памяти.
2. Identifier expected – Нужен идентификатор.
3. Unknown identifier – Неизвестный идентификатор. Этот идентификатор не объявлен.
4. Duplicate identifier - Повторный идентификатор.
5. Syntax error - Синтаксическая ошибка.
6. Error in real constant - Ошибка в действительной константе.
7. Error in integer constant - Ошибка в целой константе.
8. String constant exceeds line - Строковая константа превышает размеры строки. Вероятно, Вы забыли поставить апостроф в конце строки.
9. Unexpected end of file - Неожиданный конец файла программы или модуля. Возможно:
   - ваш исходный текст закончился до последнего end;
   - вы не закончили комментарий.
10. Line too long - Строка слишком длинная. Допустимая длина - 126 символов.
11. Type identifier expected - Нужен идентификатор типа.
12. Invalid file name - Неверное имя файла.
13. File not found - Файл не найден.
14. Disk full - Диск заполнен.
15. Invalid compiler directive - Ошибочна директива компилятора.
16. Variable identifier expected - Нужен идентификатор переменной.
17. Error in type - Ошибка в определении типа.
18. Structure too large - Слишком большая структура (массив, запись). Допустима структура длиною 65520 байт.
19. Set base type out of range - Базовый тип множества нарушает границы. Количество элементов множества может быть до 255.
20. File components may not be files or objects - Компонентами файла не могут быть файлы или объекты.
21. Invalid string length - Ошибочна длина строки. Допускается 255 символов.
22. Type mismatch - Несовместимость типов. Например:

- переменной слева от знака присваивания и результата выражения в операторе присваивания;
- фактического и формального параметров;
- выражения. Определяющего индекс;
- операндов в выражении.

23. Ordinal type expected - Нужен порядковый тип.
24. Integer constant expected - Нужна целая константа.
25. Constant expected - Нужна константа.
26. Invalid function result type - Ошибочен тип результата функции.
27. Begin expected - Нужен Begin
28. END expected - Нужен END.
29. Error in expression - Ошибка в выражении.
30. Illegal assignment - Неверное присваивание
31. Field identifier expected - Нужен идентификатор поля записи.
32. DO expected -Нужен оператор DO
33. OF expected - Нужен оператор OF
34. THEN expected - Нужен оператор THEN
35. TO or DOWNTO expected - Нужен оператор TO или DOWNTO
36. Undefined forward - Неопределенное опережающее описание.
37. Too many procedures - Слишком много процедур и (или) функций. Допускается не более 512 в одной программе или модуле.
38. Division by zero - Деление на нуль.
39. 1nvalid file type - Ошибочен файловый тип.
40. Cannot read or write variables of this type - Не допускается чтение или запись переменных данного типа:
    - процедуры Read и Readln могут считывать переменные символьного, целого, действительного и строкового типа;
    - процедуры Write и Writeln могут выводить переменные символьного, целого, действительного, булевского и строкового типа.
41. String variable expected - Нужна строковая переменная.
42. String expression expected - Нужно выражение, строкового типа.
43. Unit name mismatch - Ошибочное имя программного модуля.
44. Internal stack overflow - Внутреннее переполнение стека.
45. Constant and case types do not match - Типы констант и тип выражения оператора CASE не соответствуют друг другу.

46. Record or obiect variable expected - Нужна переменная типа запись или объект.
47. Constant out of range - Константа нарушает допустимые границы.
48. File variable expected - Нужна файловая переменная.
49. Integer or real expression expected - Нужно выражение типа real или Integer.
50. ";" expected - Нужна ";".
51. ":" expected - Нужна ":".
52. "(" expected - Нужна "(".
53. ")" expected - Нужна ")".
54. "=" expected - Нужна "=".
55. "=" expected - Нужна ":=".
56. ":=" expected - Нужна ":=".
57. "[" expected - Нужна "[".
58. "]" expected - Нужна "]".
59. "." expected - Нужна ".".
60. ".." expected - Нужна "..".
61. Too many variables - Слишком много переменных; (ОП для них более 64 Кбайт).
62. Invalid FOR control variable - Ошибочен тип переменной параметра цикла оператора FOR.
63. 1nteger variable expected - Нужна переменная целого типа.
64. Files and procedure types are not allowed here - Здесь не допускаются файлы и процедуры.
65. String length mismatch - Ошибочна длина строковой константы.
66. Invalid ordering of fields - Неверный порядок полей в константе типа запись.
67. String constant expected - Нужна константа строкового типа.
68. Overflow in arithmetic operation - Переполнение при выполнении арифметических операций.
69. User break - Компиляция прервана с помощью клавиш Ctrl + Break.
70. CASE constant out of range - константа CASE нарушает допустимыеграницы. Целочисленные константы оператора Case должны быть в диапазоне от 32768 до: 32767.
71. Error in statement - Ошибка в операторе.

72. Invalid variable reference - Ошибочна ссылка на переменную.
73. Too many symbols - Слишком много символов в программе (нужна ОП более 64Кбайт).
74. Statement part too large - Слишком большой раздел операторов (более 24 Кбайт).
75. Files must be var parameters - Файлы должны быть параметрами - переменными.
76. Header does not match previous definition - Заголовок процедуры или функции не соответствует предыдущему определению.
77. Disk error - Критическая ошибка диска: он в состоянии "не готов".
78. Cannot evaluate this expression - Нельзя вычислить данное выражение.
79. Structured variables are not allowed here - Здесь не допускается использование структурной переменной (записи, массива).
80. Invalid floating point operation - Недопустимая операция с плавающий :запятой.
81. Invalid procedure or function reference - Недопустимая ссылка на процедуру или функцию.
82. File access denied - Нужно назначение файла.

5.2. Сообщения об ошибках вовремя выполнения программы

Ошибки, ввода - вывода.

Эти ошибки контролировать с помощью директивы компилятора I и функции IOresult.

Ошибки ввода - вывода системные

1. File not found – Файл не найден.
2. Path not found - Маршрут не найден.
3. File access denied - Невозможен доступ к файлу.
4. Invalid drive nomber - 0шибочен номер дисковода.
5. Cannot remove current directory - Нельзя удалить текущий каталог.
6. Cannot rename across drives - Нельзя при переименовании указывать разные дисководы.

5.3. Ошибки ввода - вывода (программные)

7. Disk read error - Ошибка чтения диска: попытка чтения по концу файла.

8. Disk write error - Ошибка записи на диск: диск заполнен.
9. File not assigned - Файлу не определено имя физического файла с помощью оператора assign.
10. File not open - Файл не открыт.
11. File not Open for input - Файл не открыт для ввода.
12. File not Open for output - Файл не открыт для вывода.
13. Invalid numeric format - Ошибочно арифметическое значение для ввода.

Критические ошибки ввода - вывода

14. Disk is write - protected - Диск защищен от записи.
15. Bad drive request struct length - Неисправно устройство.
16. Drive not ready - Устройство не готово.

5.4. Фатальные ошибки

Эти ошибки всегда приводят к немедленной остановке программы.

17. Division by zero - Деление на нуль.
18. Range check error - Ошибка, при проверке границ (интервальных переменных).
19. Stack overflow error - переполнение стека.
20. Floating point Overflow - Переполнение при операций с плавающей запятой.
21. Invalid floating point operation - Недопустимая операция с плавающей запятой: при преобразовании значения в целое или ошибка в аргументах SQRT или LN.
22. Arithmetic overflow error - Ошибка переполнения при выполнении арифметических операций.

VI. Литература

1. Д.А.Зубок, А.В. Маятин, С.В. Краснов Основы программирования в среде Турбо Паскаль, - СПбГУ ИТМО, 2009
2. Т.А.Павловская Паскаль. Программирование на языке высокого уровня, - Питер, 2007
3. Г.Г.Рапаков, С.Ю. Ржеуцкая Программирование на языке Паскаль, - БХВ-Петербург, 2004
4. В.В.Зелинский Turbo Pascal 7.0, - ScReeW, 2006

Методические указания к выполнению лабораторных работ по программированию

Магомедова З.У.

Компьютерная верстка Гаджиев Г.К.
Подписано в печать. Формат 60*84/16
Гарнитура Times. Печать оперативная. Бумага потребительская.
Усл.печ.л. 4,1. Тираж 100 экз. Заказ №

Отпечатано в типографии Махачкалинского филиала МАДИ
367026, Махачкала, пр. Шамиля, 1