**Алгоритмический язык Паскаль**

Программа на Р записывается в виде последовательности символов:

1. заглавные и прописные латинские буквы;
2. арабские цифры;
3. знаки препинания **( : ; , . )**;
4. знаки операций.

**Арифметические операции правила их записи.**

### Знаки операций: \*, /, -, +, div, mod

(a div b) частное от деления а на b (a mod b) остаток, после деления а на b.

(а,b-целые).

a div b= a mod b =a-((a div b)\*b)



17 div 3 = 5 17 mod 3 = 2 8 div 2 = 4 8 mod 2 = 0 1 div 5 = 0 1 mod 5 = 1

При выполнении арифметических операций соблюдаются следующие правила:

1. Все знаки проставляются a b a\*b;
2. Два знака не могут следовать один за другим n/-2 n/(-2);
3. Соблюдается иерархия выполнения арифметических операций: стандартные функции;div, mod; \*, /; +, -;
4. Изменить иерархию можно только с помощью скобок.

**Типы переменных.**

Переменные на языке Паскаль задаются своими именами (63 символа).

**Целочисленные переменные.**

Shortint - короткие целые числа (1 байт). (-128 127)



Longint - удвоенные целые числа (4 байта). (-231231-1)



Integer - обычные целые числа (2 байта). (-3276832767)



Word - целые положительные числа (2 байта). (065535)



Byte - целые короткие положительные числа (1 байт). (0255)



Верхнее граничное значение целочисленных переменных задано в Паскале как константа и имеет соответствующее имя: Maxint = 32767 (215-1) Maxlongint = 231-1

#### Логический тип.

Boolean - логические переменные (1 байт) могут принимать 2 значения:

1. True (истина)
2. False (ложь)

Логические переменные могут использоваться только в логических выражениях. Над ними можно выполнять только 2 операции:

1. равно (=)
2. не равно (< >)

Для идентификаторов имеет место:

**False< True**

Над логическими переменными возможны следующие операции:

And (и)

Or (или)

Not (не)

Ord (false) = 0 Ord (true) = 1

### Символьный тип.

Chor - служит для хранения одного символа (1 байт)

(буква, цифра, знаки препинания, специальные символы, непосредственно код). Значения символьных переменных задаются в апострофах - “ А”, “9”.

### Строковый тип.

String - строковые переменные (255 байт).

Это строка символов заключенных в апострофах.



### Вещественный тип.

Real - служит для хранения вещественных чисел (6 байт)

#### (11 знаков после запятой).

Могут быть заданы в форме с:

1. фиксированной точкой
2. 0,5; +5,0; -133,15

плавающей точкой 1200 = 120,0 Е+1 = 12,0 Е+2 = 1,2 Е+3 = 12000,0 Е-1 (показатель степени 38)



Используется для изображения очень больших или очень маленьких чисел.

## Стандартные функции.

sin (x) sin x (вещ.)



cos (x) cos x (вещ.)

arctg(x) arctg x (вещ.)

exp (x) ex  (вещ.)

ln (x) ln x (вещ.)

pi (x) 3.14 (вещ.)

abs (x) (вещ.)



sqr (x) x2 (вещ.)

sqrt (x) (вещ.)



trunc (x) целая часть числа,дробная остается без округлен. (цел)

trunc (3.7) = 3 trunc (3.1) = 3 trunc (-3.7) = -3

frag (x) дробная часть числа (вещ.)

int (x) ближайшее наименьшее целое число (цел.)

int (3.4) = 3 int (3.7) = 3 int (-3.4) = -4

round (x) ближайшее целое число (матем округ-е) (цел.)

round (3.14) = 3 round (3.74) = 4 round (-3.14) = -3

random (x) генератор случайных чисел (вещ.)

от 0 до x; если x – отсутствует, диапазон чисел 01



odd (x) возвращает TRUE, если x - число нечетное (лог.)

Аргументом стандартной функции может быть переменная, константа, выражение, стоящее справа от имени в скобках. Воспринимается в радианах для тригонометрических функции.

##### **Структура программы на Паскале.**

# Программа – это последовательность инструкций компьютеру приводящих к конечному результату за конечное число шагов.

Программа, написанная на алгоритмическом языке, переводится на язык машинных команд:

**program**  <имя>**;**

#### <описательная часть>;

<раздел функций и процедур>**;**

begin

<исполнительная часть>**;**

end.

Зарезервированные слова:

1. program – всегда первый;
2. begin – начало;
3. end – конец.

<имя> - присваивается составителем программы (строится по правилам переменных).

#### Описательная часть программы.

1. Все переменные, используемые в программе должны быть описаны.

Описание начинается со служебного слова ***var .***

program ff;

var

i, n: integer; x,y,z: real;

begin;

.........

end.

Список переменных от типа отделяется “**:**”, одно описание от другого – “**;**”, список переменных – “**,**”.

Метки в программе описываются с помощью служебного слова:  **label.**

Константы : **const**.

Пользовательский тип данных: **type**.

#### Исполнительная часть программы.

Отдельные инструкции, входящие в программу, называются операторами. Операторы отделяются один от другого – “**;**”.

Бывают трех типов:

1. пустой оператор;
2. простой оператор;
3. составной оператор.

Составной оператор:

begin

<оператор 1>**;** <оператор 2>**;** …

end**;**

#### Операторы Паскаля.

**Оператор присваивания.**

#### “: =” – знак присваивания.

<переменная> **: =** <выражения>**;**

##### Читается одиноково.



Пример:

;

Паскаль не допускает смешенных выражений. Слева – вещественное, справа - целое выражение (допустимо).

Если в выражении есть хотя бы одна вещественная переменная, все выражение будет вещественным.

**Операторы ввода-вывода.**

*READ* (<список - ввода>)

1. имя оператора ввода;
2. список переменных, разделенных запятыми.

*WRITE* (<список - вывода>)

1. имя оператора вывода;
2. список переменных вывода, разделенных запятыми.

*READLN* (a, b, c) - после вода значений a, b и c курсор перемещается на следующую строку.

*WRITELN* - без списка вывода можно использовать для пропуска строк при оформлении вывода результатов.

В операторе *WRITE* можно использовать формат вывода значений переменных.

Writeln ( ’\_ a = ’ , a:8:3, ’\_ b = ’, b:4);

при a = 341.154, b = 2

\_ a = \_ 341.154 \_ b = \_ 144

при a = 1.3, b = 144

\_ a = \_ \_ \_ 1.300 \_ b = \_ 144

Program \_ prim;

сlrscr - оператор гашения экрана;

var a, b: integer;

x ,y: real;

begin writeln (’ введите a, b ’);

readln ( a, b);

x: = a + b; y: = a/b;

writeln (’ x = ’, x:8:3, ’\_ \_ ’, ’y = ’, y:8:3)

end.

Ключевые слова горят ярче, чем весь текст прогаммы.

**Операторы условия и перехода.**

Строки программы на Паскале не нумеруются. Отдельные строки в программе могут иметь метки, к которым можно переходить.

Метки должны быть описаны с помощью ключевого слова

label N1, N2 … ;

в описательной части программы.

N1, N2, … - идентификатор или целое число (положительное) (09999).



Оператор: **GO TO N;** -передает управление строке с меткой N.

program pr;

label 3;

var

x, y: real;

begin

3: readln (x, y);

go to 3;

end.

Оператор**: IF < условия > THEN P1 [ ELSE P2 ];** если то иначе (не обязательная часть)

<условие> - логическое выражение ;

P1, P2 - простой или составной операторы.

По этому оператору:

если <условие> - "истинно", то выполняется P1 (true);

- " ложно", то выполняется P2 (false).

Если ELSE - отсутствует и <условие> - "ложно", то управление передается следующему оператору.

Распечатать наибольшее из двух чисел:

IF a>b THEN write (a) ELSE write (b).

Вычислить значение функции:

Y =



If x>=0 then y:= sin(x) else y:= - sin(x);

Логические выражения могут быть сложными, составленными с помощью логических операций: AND (и) OR (или) NOT (не).



IF (a>b) and (a>c) THEN writeln (’a = ’, a)

IF a<0 THEN

составной оператор (P1)



ELSE

cоставной оператор (P2)



Паскаль допускает вложенность операторов IF.

IF n>0 THEN

IF ( m div n)>n THEN

m: = m-n

ELSE

m: = m + n;

ELSE - всегда относится к ближайшему оператору IF.

Если n>0 и (m div n)>n будет выполнено m: = m-n.

Если n>0, но (m div n)n будет выполнено m: = m + n.



Если n0 - переход к следующему оператору.



Задача:

Вычислить: y=



Program fun;

var

x, y: real;

begin

writeln (’введите x’); readln (x);

if x>90 then writeln (’функция не определена’)

else begin

*if*  x<0 *then* y: = 0

*else*  y: = SIN (x\*PI/180);

writeln (’y = ’,y:8:3);

end; (составной оператор)

end.

Оператор **CASE … OF;** этот оператор предназначен для замены конструкций из вложенных IF.

Структура:

CASE N of

N1: P1;

N2: P2;

NN: PN;

[else P;] - необязательная часть оператора.

end;

где N - целочисленная переменная, или выражение целочисленного типа.

N1, N2, … NN - возможные значения переменной N.

P, P1, P2, … PN - простые или составные операторы.

По этому оператору :

если значение - N = N1, то выполняется P1 (после чего управление передается оператору следующему за оператором case ... of);

если значение - N = N2, то выполняется P2, иначе P.

Если структура else - отсутствует и N - не принимает ни одного из перечисленных значений, управление передается следующему за case … of оператору.

**Циклические вычислительные процессы.**

Процессы, в которых ряд действий повторяются многократно по одним и тем же математическим зависимостям, называются циклическими.

**Операторы цикла.**

При организации циклов необходимо:

1. определить параметр цикла и его начальное значение;
2. изменять значение параметра цикла на каждом шаге итерации;
3. проверка на выход из цикла.

FOR i: = N TO K DO P

(для) (до) (выполнять)

где i - параметр цикла;

N, K - его начальное и конечное значение;

P - простой или составной оператор;

I, N, K - переменные или константы целого типа.

Шаг изменения i - равен 1.

Если K<N, т.е. имеем дело с циклом с отрицательным шагом: (-1)

FOR i: =N DOWNTO K DO P

Рассмотрим пример.

Вычислить

S= 1+1/2+1/3+ … +1/50

Выделим переменную для накапливания суммы - Sum. Значение этой переменной необходимо предварительно обнулить. Паскаль не производит предварительной, начальной, инициализации переменных. Поэтому сумма может быть искажена без Sum = 0.

Program sum;

Var

i: integer;

sum: real;

begin

sum: =0;

for i: = 1 to 50 do

Sum: = sum + 1/i;

Writeln (' сумма = ', Sum);

end.

Оператор for применяют в тех случаях, когда значения параметра цикла целые и меняются с шагом +1, -1.

**Оператор цикла с постусловием.**

REPEAT

- тело цикла



UNTIL <условие>;

где P1, P2, …PN - любые операторы.

По этому оператору выполняется "тело цикла", а затем проверяется <условие>, если оно не выполнилось, цикл повторяется. И так до тех пор, пока <условие> не будет выполнено.

Необходимо помнить: если <условие> сразу выполнилось, цикл будет пройден один раз.

Вычислить: y = a sin (x ) , x = , =0.2



program fun;

var

y, a, x: real;

begin

x: =0; read (a)

repeat

y: = a \*sin (x); writeln (y, x); x: = x + 0.2;

until x>1;

end.

Действия, которые подчеркнуты, необходимы для организации цикла.

**Оператор цикла с предусловием.**

WHILE <условие> DO P;

Где P - простой или составной оператор.

По этому оператору проверяется <условие> и, если оно выполняется, то выполняется - P, после чего опять проверяется <условие> и т. д.

Итак, P - выполняется до тех пор, пока выполняется <условие>.

Если условие ни разу не выполнилось, P - игнорируется, управление передается следующему оператору.

Вычислить: y = sin x, x, x, - ввести с клавиатуры.



Подсчитать среднее положительное и среднее отрицательное значение функции.

Program cikl;

var

y, x, x k, d x: real;

n, k: integer; SP, n - среднее и количество положительных значений.

SP, S0: real; S0, k - среднее и количество отрицательных значений.

begin

writeln (' введите x - начальное, x - конечное, d x - шаг');

read (x, x k, d x);

SP: =0; S0: = 0; n: =0; k: =0;

While x< = k x do.

P Составной оператор.



if n=0 then writeln ('Отрицательные отсутствуют')

else writeln (' Средние отрицательные = ', SP/n);

if k=0 then writeln (' Положительные отсутствуют ')

else writeln (' Средние положительные = ', S0/k);

end.

###### Массивы

Задача 1:

Составить программу подсчета суммы и произведения элементов одномерного массива.

***Program*** sum-prois;

uses crt;

const

n = 100;

var

a: array [1…n] of real;

n, k, i: integer;

p, s: real;

begin

clrscr; s: = 0; p: =1;

writeln ('введите размер массива'); readln (nk);

writeln ('введите элемент массива');

for i:=1 to nk do

readln (a[i] );

for i:=1 to nk do

begin

s:= s + a[ i ];

p:= p \* a[ i ];

end;

writeln ('Сум. = ', s, 'Произ. =' , p);

end.

Необходимо подготовить ячейки:

1. при накапливании суммы - s=0
2. при подсчете произведения - p=1.

Задача рассчитана на обработку массива с максимальным размером 100 элементов (n=100).

Конкретный размер массива вводится с клавиатуры (nk).

При каждом прохождении через цикл с клавиатуры вводится только один элемент массива. Два цикла for можно было объединить в один.

**Двумерные массивы.**

Двумерный массив можно представить в виде матрицы..



Описание двумерных массивов:

a - имя массива;

n, m - количество строк и столбцов в массиве.

Размер массива - nm.



a[i, j] - элемент стоящий на пересечении i-ой строки и j-го столбца.

Каждый элемент определяется двумя индексами.

a[i, i] - элементы главной диагонали.

a[i, 2] - элементы второго стлбца.

Задача 1.

Составить программу подсчета суммы элементов над главной диагональю в двумерном массиве.

***Program*** matrix;

const

n=10;

m=10;

var

a: **array** [1…n, 1…m] **of** real;

i, j: integer; n, m: integer;

s: real;

begin s:=0;

writeln (' введите размер массива m, n);

readln (n, m);

{Ввод массива:}

for i:=1 to n do

for j:=1 to m do

readln (a[i, j]);

for i:=1 to n do

for j:=**i** to m do

s: s+a[i, j];

writeln('s=', s);

end.

Для ввода элементов массива используются вложенные циклы.

i - параметр внешнего цикла;

j - параметр внутреннего цикла;

i - меняется медленнее j.

Элементы массива необходимо вводить по строкам.