Введение в программирование на языке Pascal

Лекция 2

Структура программы на Pascal

```
{раздел описания программы}
program <название программы>;
    uses  <подключение библиотек и модулей>;
    const  <раздел описания констант>;
    type <раздел описания типов>;
    var <раздел описания переменных>;
    label <раздел описания меток>;
<описание процедур и функций>
   procedure …;
   function …;
{раздел реализации}
    begin
        <тело программы>
    end.
```

Раздел описания констант CONST

- Обычные константы

```
const
Min = 0;
Max = 250;
Centr = (Max-Min) div 2;
```

Константные выражения вычисляются компилятором без выполнения программы на этапе ее создания.

- Типизированные константы

const конст1: тип=значение;

Например:

```
const   nums: integer=10;
  begin
   nums:=nums+5;
  end.
```

Раздел описания типов TYPE

Тип — это множество значений + множество операций, которые можно выполнять над этими значениями, то есть правила манипулирования данными.

```
TYPE
  DAY = 1..31;
   Year = 1900.. 2000; {Интервальный тип}
  LatBukv = ('A','C','G,'H'); {Перечисляемый тип}
  Matr = array[1..12] of real; {Регулярный тип}
```

Раздел описания переменных VAR

```
TYPE
   DAY = 1..31;
VAR
 {явное описание типов}
   A, B: DAY;
   YEAR:1900..2000;
 {неявное описание}
   LES:(LPT, PRN);
{стандартный тип}
   A, B: REAL;
```

Описание процедур и функций

{заголовок процедуры}
PROCEDURE <имя> (<параметры>);
<разделы описания>
{тело процедуры }
BEGIN
<раздел реализации>
END;

```
Procedure Nod(M, N : Integer; Var  K : Integer);
Begin
   While M <> N Do
      If M > N
     Then M := M - N
     Else N := N - M;
    K := M
End;
```

{ заголовок функции}
FUNCTION <имя>(<параметры>): <тип результата>;
<разделы описания>
{тело функции}
BEGIN
<раздел реализации>
END;

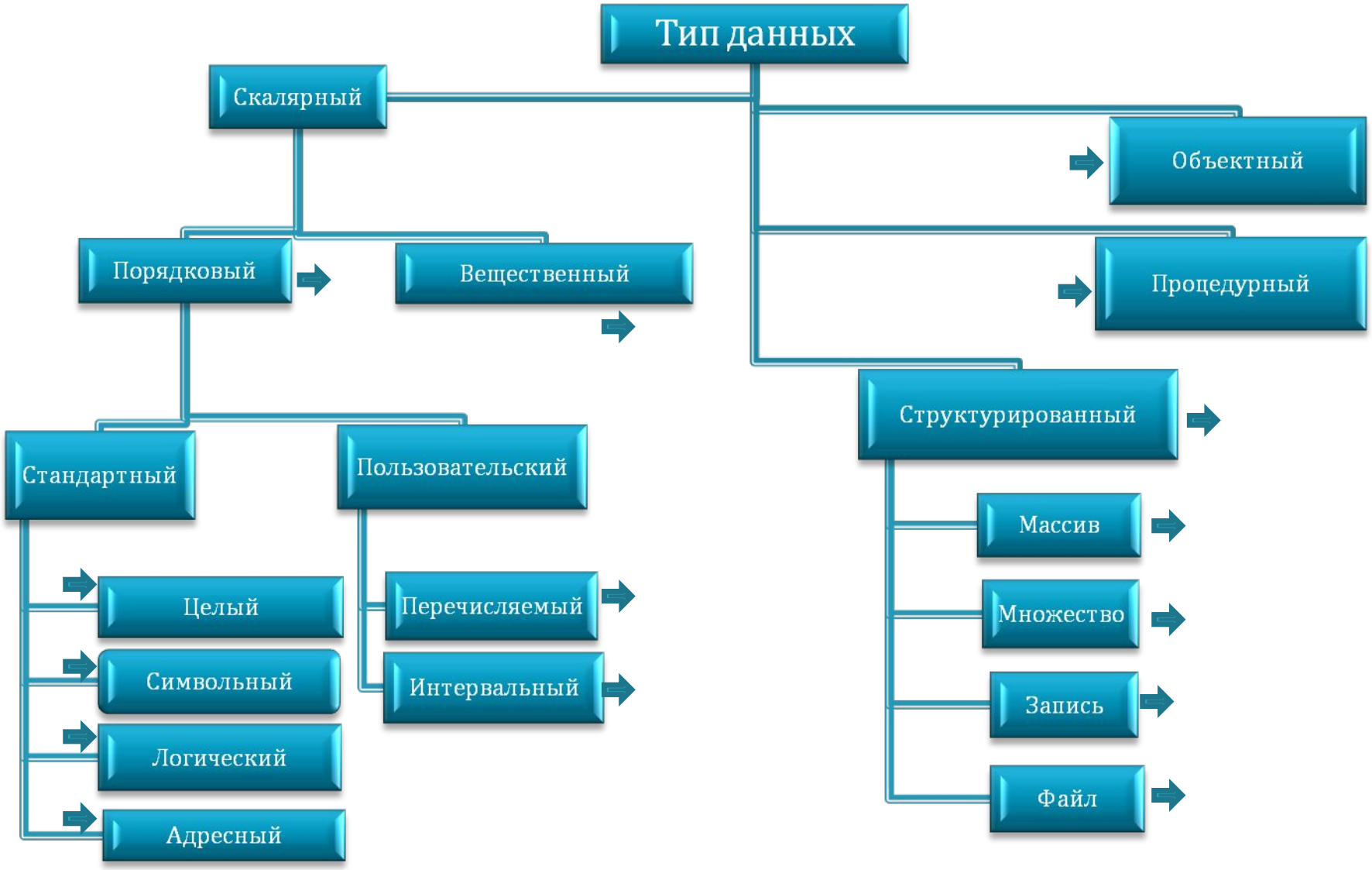
```
Function Nod(M, N : Integer) : Integer;
 Begin
  While M <> N Do
     If M > N
      Then M := M - N
      Else N := N - M;
     Nod := M
End;
```

Концепция типа языка Pascal имеет следующие основные свойства:

- любой тип данных определяет множество значений, которые может принимать переменная, выражение или вырабатывать операция/функция;
- каждая операция или функция требует аргументов фиксированного типа и выдает результат фиксированного типа.

Тип определяет:

- возможные значения переменных, констант, функций, выражений;
- внутреннюю форму представления данных в ЭВМ;
- операции и функции, которые могут выполняться над величинами, принадлежащими к данному типу.


Тип данных
Скалярный
Объектный
Порядковый
Вещественный
Процедурный
Структурированный
Стандартный
Пользовательский
Массив
Целый
Перечисляемый
Множество
Символьный
Интервальный
Запись
Логический
Файл
Адресный

Порядковые типы

Функции для порядковых типов

Ord

По значению ординального типа возвращает *порядковый номер значения.*

Pred

По значению ординального типа возвращает *предшествующее значение.*

Succ

По значению ординального типа возвращает *последующее значение.*

Low

По ординальному типу или переменной ординального типа возвращает *наименьшее значение данного типа.*

High

По ординальному типу или переменной ординального типа возвращает *наибольшее значение данного типа*

Целые типы

Тип	Диапазон допустимых значений	Отводимая память, в байтах
shotint	-128…127	1
integer	-32 768…32 767	2
longint	-2 147 483 648…2 147 483 647	4
byte	0…255	1
word	0…65 535	2
Int64	-2^{63} …2^{63}	8
cardinal	0…4294967295	4

Cardinal это Integer, размер которого не гарантируется. Это основное целое число без знака Обычно **Cardinal** используется в параметре передающемся к функциям типа C.

Логический тип

Булев тип данных (**boolean**) может принимать только два значения (**true** или **false**). Эти величины упорядочены следующим образом: **false < true.** Логические данные могут выступать в роли операндов операции отношения, к ним можно применять функции **ord, succ, pred**, процедуры **inc** и **dec**.

Значение типа **boolean** занимает в памяти 1 байт.

C:\FPC\2.4.0\bin\logic2.pas

```
var
    k, l, m, n: integer;
    b1, b2, b3, b4, b5, b6: boolean;

begin
    write ('Введите четыре числа: ');
    readln (k, l, m, n);

    b1 := (k < l) and (k < n);
    b2 := (k < l) or (k < n);
    b3 := (k = 0) or (l = 0) or (m = 0) or (n = 0);
    b4 := not ( k +_l > m + n);

    writeln ('Первое меньше второго и четвертого. ', b1);
    writeln ('Первое меньше второго или четвертого. ', b2);
    writeln ('Одно из чисел равно нулю. ', b3);
    writeln ('Сумма первой пары не больше суммы второй ', b4);

readln
end.
```

12:20

```
Running "c:\fpc\2.4.0\bin\logic2.exe "
Введите четыре числа: 4 7 6 0
Первое меньше второго и четвертого. FALSE
Первое меньше второго или четвертого. TRUE
Одно из чисел равно нулю. TRUE
Сумма первой пары не больше суммы второй FALSE

Running "c:\fpc\2.4.0\bin\logic2.exe "
Введите четыре числа: 2 6 8 15
Первое меньше второго и четвертого. TRUE
Первое меньше второго или четвертого. TRUE
Одно из чисел равно нулю. FALSE
Сумма первой пары не больше суммы второй TRUE
```

Символьный тип

Множество значений *символьного типа есть* множество символов, упорядоченных в соответствии с их **ASCII**-кодами.

Любое значение символьного типа может быть получено с помощью стандартной функции ***Chr*** *из его кода ASCII.*

Пример:

```
var ch: Char;
…
ch:= ’A’;
ch:= Chr(32); { ch:= ’ ’; }
```

Перечислимый тип

Перечислимый тип определяет упорядоченное множество значений путем перечисления идентификаторов, обозначающих эти значения. Упорядочение значений определяется порядком следования идентификаторов, их определяющих.

Пример:

```
type
Suit = (Spades, Clubs, Diamonds, Hearts);
{ Ord(Spades)=0, Ord(Clubs)=1}
```

Интервальный тип

Интервальный тип представляет собой поддиапазон значений из некоторого порядкового типа, называемого *базовым.*

Примеры:

```
type
TeenAge=13..19;
Day=1..31;
```

Вещественный тип

Вещественный тип имеет множество значений, являющееся подмножеством множества действительных чисел, и которые могут быть представлены в формате с плавающей точкой.

В памяти компьютера представлено в экспоненциальной форме *mE ± p, где*

m – мантисса (целое или дробное число с десятичной точкой), *p – порядок (целое число)*

$$-36.142E + 2 = -36.142 \cdot 102 = -3614.2;$$

$$7.25E - 5 = 7.25 \cdot 10-5 = 0.0000725$$

Тип	Диапазон допустимых значений	Число цифр	Отводимая память, в байтах
Real	$2.9e^{-39}$ … $1.7e^{38}$	11-12	6
Single	$1.5e^{-45}$ … $3.4e^{38}$	7-8	4
Double	$5.0e^{-324}$ …$1.7e^{308}$	15-16	8
Extended	$3.4e^{-4932}$ … $1.1e^{493}$	19-20	10
Comp	$-9.2e^{63}$ … $(9.2e^{63})-1$	19-20	8

Строковый тип

Значение *строкового типа - это последовательность символов* с атрибутом «динамическая длина» (зависящим от фактического количества символов во время выполнения программы) и с атрибутом «размер» от 1 до 255.

Строковому типу, объявленному без указания размера, по умолчанию дается размер в 255 символов.

Текущее значение атрибута «длина» можно получить с помощью стандартной функции **Length.**

Для значений строкового типа определен лексикографический порядок: **'abc'<'ac' , 'ab'<'aba'.**

Символы строки доступны как элементы массива.

```
var                 S:='Turbo Pascal';
S: String[20];      WriteLn('Длина S = ', Length(S)); { 12 }
Ch: Char;           Ch:=S[1]; { 'T' }
…                   Ch:=S[0]; { #12 –длина строки}
```

В Pascal введен тип **pchar**, который описывает так называемые длинные (или ASCIIZ) строки, длина не указывается явно, строка завершается #0.

Структурный тип

Структурный тип данных представляет объекты, содержащие сразу несколько значений, называемых элементами.

Структурный тип характеризуется типом (или типами) элементов, составляющих объект, и способом доступа к элементам.

Элементы объекта структурного типа сами могут иметь структурный тип (многоуровневая структуризация).

Количество уровней структуризации не ограничено.

Массив

Массив содержит фиксированное число элементов одного типа.

В качестве индексного типа допустим любой порядковый тип, кроме LongInt и ограниченных типов, основанных на LongInt. Стандартные функции Low и High, примененные к массиву, выдают нижнюю и верхнюю границы (первого) индекса соответственно

Пример работы с массивом

```
type
TVector=array [1..N] of Real;
TMatrix=array [1..N,1..N] of Real;
{ или =array [1..N] of array [1..N] of Real }
TCube=array[1..N,1..N,1..N] of Real;
Month=(Jan,Feb,Mar,Apr,May,Jun,
                    Jul,Aug,Sep,Oct,Nov,Dec);
DaysInMonth=array [Month] of Byte;

var
V: TVector;
M: TMatrix;
C: TCube;
DM: DaysInMonth;
```

```
for i:=1 to N do
        V[i]:=0;

for i:=1 to N do
  for j:=1 to N do
        M[i,j]:=1;

C[N div 2, 1, 1]:=3;

DM[Jan]:=31;
DM[Feb]:=28;
```

Запись

Запись — структурированный тип данных. Записи являются неоднородными неупорядоченными структурами с прямым доступом к компонентам. Компоненты записи называют **полями записи**

```
Type
TDate=record
        Day: 1..31;
        Month: 1..12;
        Year: Word;
   end;

pol=(m,w);
people=record
          fam:string[20];
          BDay: TDate;
          case mw:pol of
             m: ( voen: boolean; spec: string[15]);
            w: ( merry: boolean; child: byte)
         end;
      end;
```

```
Var
 p1, p2: people;

P1.mw:=m;
p1.voen:=true;
p2.child:=2;

 With p1.Bday do
      begin
        Day:=12;
Month: =1;
 Year:=1994;
      end;
```

Множество

Тип «множество» представляет всевозможные подмножества значений некоторого порядкового типа, называемого базовым.

Базовый тип не может иметь более 256 возможных значений.

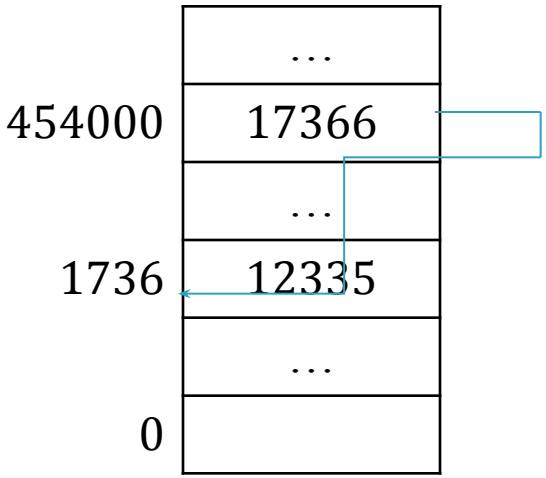
Примеры:

```
type
CharSet=set of Char;
WordSet=set of Word; { Синтаксическая
        ошибка! }
```

Ссылочные типы

Ссылочные типы используются для описания указателей.

Указатель – значение, задающее адрес другого значения в памяти.




```
type
PInteger = ^Integer;

PNode = ^TNode;
TNode = record
Info: Real;
Next: PNode;
end;
```

Файл

Тип «файл» состоит из линейной последовательности компонент некоторого типа.

Тип компонент файла не может быть файловым типом, структурным типом, содержащим элементы файлового типа, и объектным типом. Количество компонент не фиксируется при определении файлового типа.

Процедурный тип

Процедурный тип предоставляет возможность использования переменных-подпрограмм

```
type
TFunc = function(X: Real): Real;
    function tg(X: Real): Real;
       begin
          tg := sin(X)/cos(X);
      end;
procedure PrintFunc(Start, Stop, Step: Real; f: TFunc);
{ Печатает таблицу значений функции f на отрезке [Start;Stop] с шагом Step }
…
PrintFunc(1, 2, 0.01, tg(x));
```

Объектный тип

Объектный тип введен для синтаксической поддержки концепций объектно-ориентированного программирования (ООП).

```
type
TMan = object
Name: String;
Sex: Char;
Age: Integer;
procedure Init(aName: String; aSex: Char;
        aAge:Integer);
end;
```

Эквивалентность

- *структурная эквивалентность* – типы эквивалентны, если эквивалентны их структуры (число составляющих компонентов и их тип);
- *именная эквивалентность* – типы эквивалентны,если эквивалентны идентификаторы их имен.

Типы T1 и T2 эквивалентны, если выполняется одно из следующих условий:

- T1 и T2 совпадают;
- T1 и T2 определены в одном объявлении типа;
- T1 эквивалентен некоторому типу T3, который эквивалентен типу T2

```
type
T1 = Integer;
T2 = T1;
T3 = Integer;
T4 = T2;
T5 = array [1..3] of Integer;
T6 = array [1..3] of Integer;
var
V1, V2: array [1..3] of
Integer;
V3: array [1..3] of Integer;
V4: array [1..3] of Integer;
V5: Integer;
V6: Integer;
```

Эквивалентные типы

{ структурная и именная эквивалентность }

T1, T2, T3, T4 и Integer

переменных V1 и V2

переменных V5 и V6

НЕ эквивалентные типы

{ структурная, но не именная эквивалентность }

T5 и T6

переменных V3 и V4

Совместимость

Два типа T1 и T2 будут совместимыми, если верен хотя бы один вариант из перечисленных ниже:

- T1 и T2 эквивалентны;
- T1 и T2 - оба целочисленные или оба вещественные;
- T1 и T2 являются подмножествами одного типа;
- T1 является некоторым подмножеством T2;
- T1 - строка, а T2 - символ;
- T1 - это нетипизированный указатель, а T2 - типизированный указатель ;
- T1 и T2 - оба процедурные, с одинаковым количеством попарно эквивалентных параметров, а для функций - с эквивалентными типами результатов.

Type	**Совместимые типы**
TNumber=Integer;	Byte, TNumber
TIndex=TNumber;	TIndex, TNumber
CharSet= set of Char;	YesNo, Boolean
YesNo=Boolean;	CharSet, set of Char
ZeroOrOne=0..1;	Suit, RedSuit
Str10=String[10];	String, Str10
Ch10Str=array[1..10] of Char;	**НЕСОВМЕСТИМЫЕ типы**
Suit=(Spades, Clubs,	TArr, THalfArr
Diamonds, Hearts);	TNumber, Real
RedSuit=Diamonds..Hearts;	YesNo, ZeroOrOne
TArr=array[1..N] of Byte;	Str10, Ch10Str
THalfArr=array[1..N div 2] of Byte;	ZeroOrOne, RedSuit

Совместимость по присваиванию

Два типа данных T1 и T2 называются совместимыми по присваиванию, если выполняется хотя бы один вариант из перечисленных ниже:

- T1 и T2 эквивалентны, но не файлы;
- T1 и T2 совместимы, причем T2 - некоторое подмножество в T1;
- T1 – объектный тип, а тип T2 – объектный тип-потомок T1.
- T1 - вещественный тип, а T2 - целый.

```
type
TNumber=Integer;
TTeenAge=13..19;
TIndex=TNumber;
TArr=array [1..N] of Real
var
S: String;
str10: String[10];
n: TNumber;
i: Integer;
Ch: Char;
R: Real;
F1, F2: Text;
Young: TTeenAge;
Idx: TIndex;
B: Byte;
X, Y: TArr;
```

Синтаксически верно

```
S:=str10;
n:=i;
Ch:=S[1];
R:=i;
R:=Young; n:=Young;
Idx:=B;
X:=Y;
```

Синтаксически НЕВЕРНО

```
F2:=F1;
n:=R;
i:=R;
str10:=S;
Ch:=S;
Ch:=str10;
Young:=n; Young:=Idx;
B:=i;
```

Организация ввода-вывода

Read -> файл INPUT	Write -> файл OUTPUT
Read(A1,A2,...AK); ReadLn(A1,A2,...AK); ReadLn;	Write(A1,A2,...AK); WriteLn(A1,A2,...AK); WriteLn; Write(A:K:M)

```
program jh;
uses crt;
var  i:integer;
r:real;
s:string[5];

begin
clrscr;
     write ('введите данные    ');
     readln(i,r,s);
     writeln( 'Это целое  ',i, i:8);
     writeln('Это действительное  ', r, r:10:5);
     writeln ('Это сторока  ',s, s:10);
```

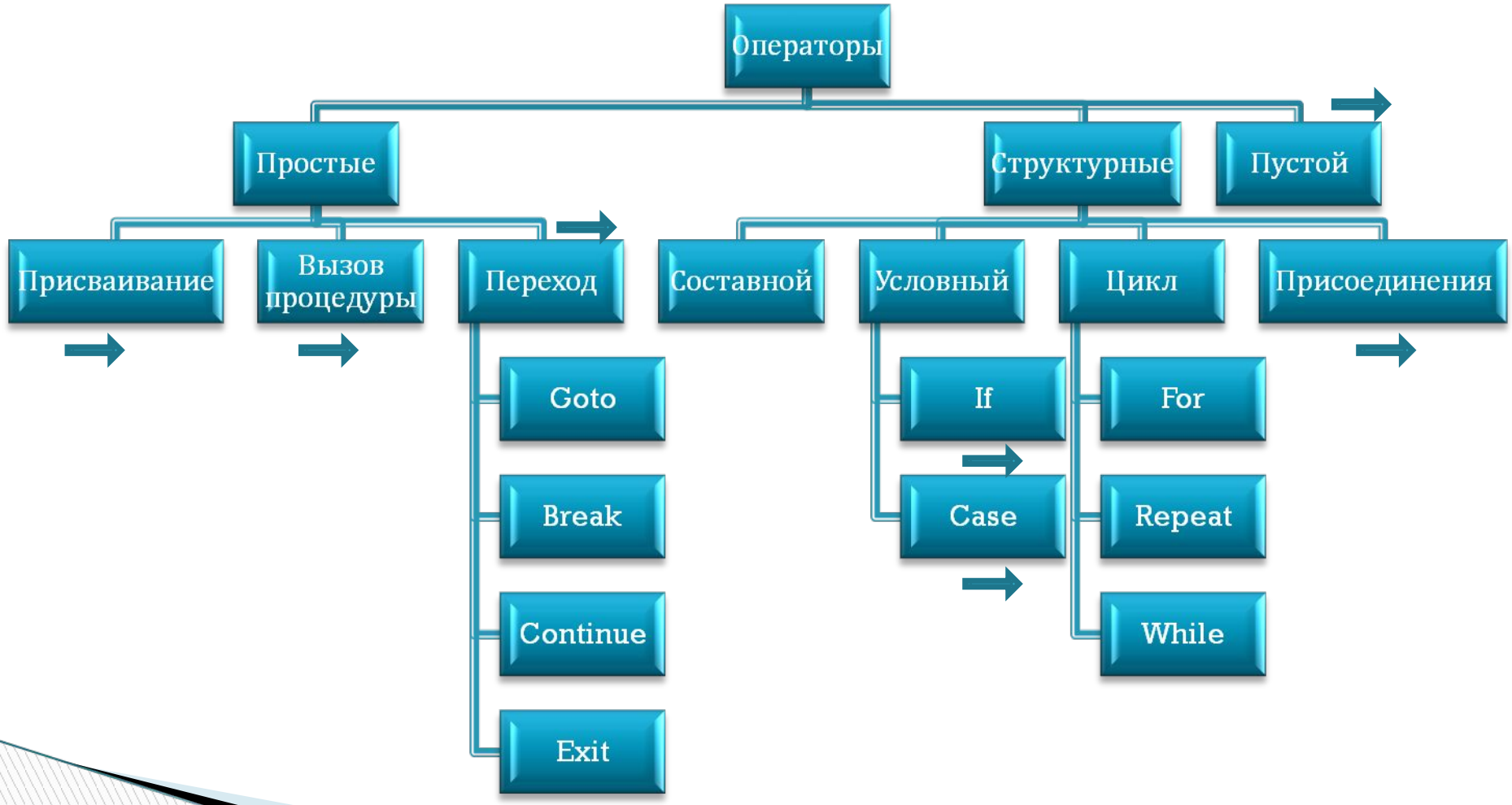
```
Free Pascal
введите данные    123 87.987678 ивт
Это целое  123      123
Это действительное    8.79876780000000E+001   87.98768
Это сторока    ивт         ивт
```

Осуществить расчеты по формуле: $Y = \sqrt[n]{|X^{n+1} + Log_n |X+1||}$

Исходная формула	Формула для программирования	Текст программы
$\sqrt[n]{Z}$, Z>0	$e^{Ln(Z)/n}$	EXP(LN(Z)/N)
Z^{n+1}, Z>0	$e^{(n+1)*Ln(z)}$	EXP((N+1)*LN(Z))
$Log_N\|X+1\|$	$\frac{Ln(\|X+1\|)}{Ln(N)}$	LN(ABS(X+1))/LN(N)

```
PROGRAM PR5;
VAR
X, Y: REAL;
N: INTEGER;
BEGIN
  WRITELN('Введите значения X, N');
  READLN(X, N);
   If  X>=0 THEN
        BEGIN
            Y := EXP(LN(ABS(EXP((N+1)*LN(X))+LN(ABS(X+1))/LN(N)))/N);
                 WRITELN( 'Y =  ', Y:8:4);
        END;
        ELSE     WRITELN ('Ошибочные входные данные')
END.
```

Классификация операторов



Пустой оператор

- Не обозначается и не вызывает никаких действий. Например:

```
I f a>6 then;;;
```

- В целях унификации рекомендуется добавлять пустой оператор как последний оператор составного

```
If a>5 then
 begin
    z:=1;
    x:=c+5; {разделитель не обязателен}
 end;
```

Оператор присваивания

- <Переменная>:=<Выражение>;
  замена текущего значения переменной новым;
  определение значения, возвращаемого функцией.
- Типы переменой и выражения должны быть совместимы по присваиванию.

Оператор вызова процедур

- <Имя процедуры> (<Список фактических параметров>)

Активизирует процедуру с указанным именем, присваивая формальным параметрам значения соответствующих фактических.

Примеры

```
CLRSCR;
Inc(i);
Calculate(a,5,result);
```

Операторы перехода

- **GOTO** < метка>;

Метка- представляет собой целое число без знака, определяется в секции **LABEL** того же блока, что и оператор **Goto.**

- **Break; Continue;**

Досрочно прекращает выполнение цикла и начинает новую итерацию соответственно.

- **Exit;**

Досрочно завершает процедуру/ функцию/ программу

Оператор присоединения

- With <ссылка на запись или объект> DO
  <оператор>

Позволяет ссылаться на поля или методы объекта/записи без указания имени объекта/записи

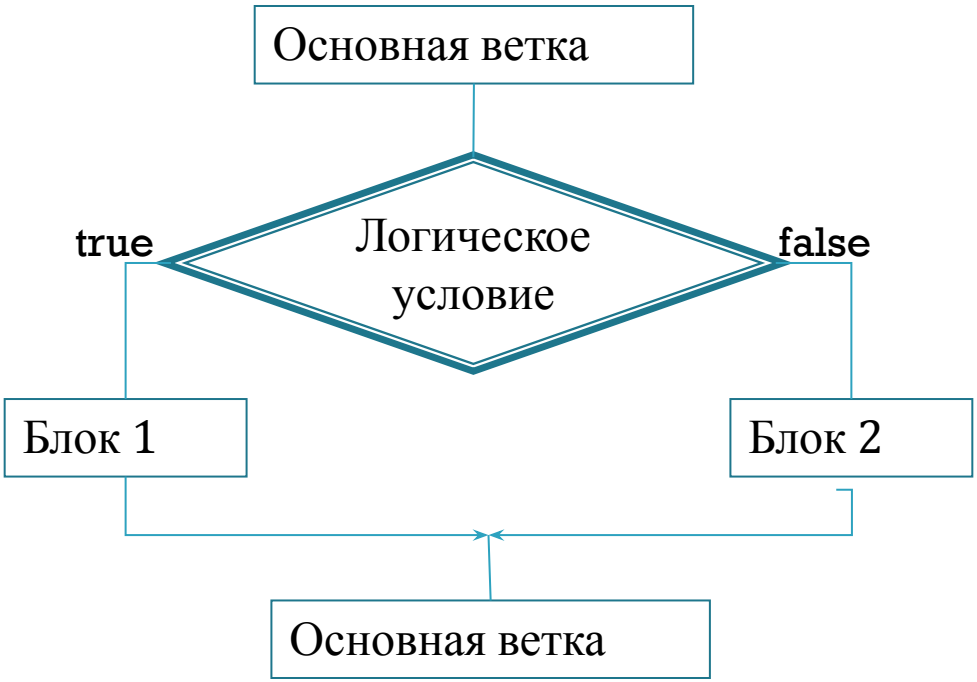
Пример:

```
Type  Tperson = record
        fistname:string[15];
        lastname:string[10];
    end;
Var person:Tperson;
…
                With  person  do
                  begin
                       fistname:=‘Иванов’;
                       Lastname:=‘Иван’;
                   end;
```

Вариант

Person.fistname:=‘Иванов’;
Person.Lastname:=‘Иван’;

Оператор условия if (полная форма)

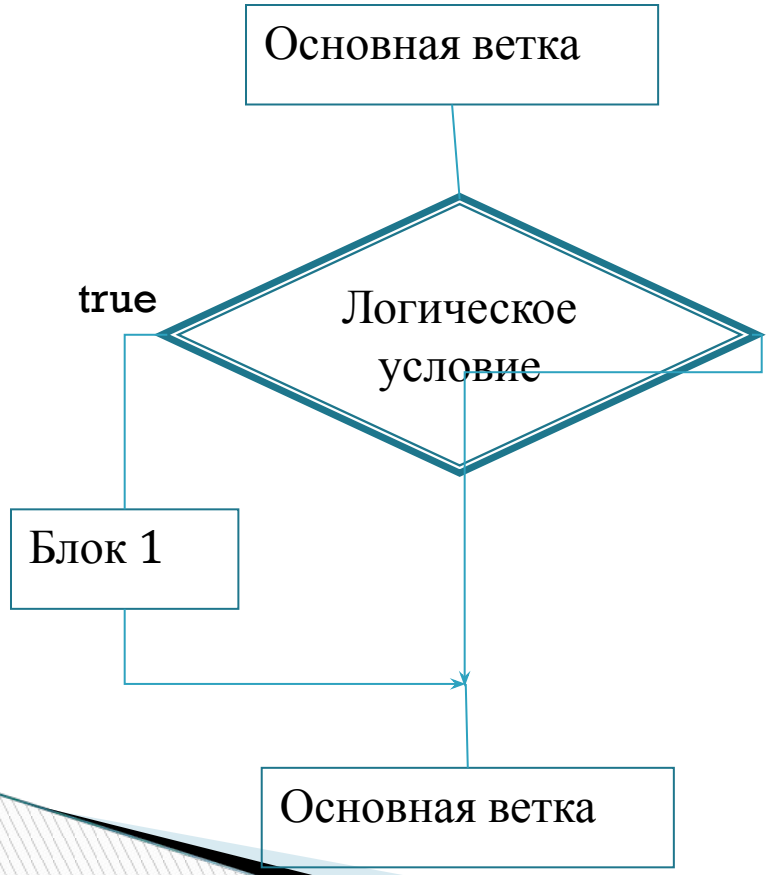



```
If  <логическое условие>
      then
      <оператор>
      else
      <оператор>;
```

Вложенный if

```
If  <логическое условие>
      then
         if <логическое условие>
           then
           <оператор>
           else
           <оператор>;
```

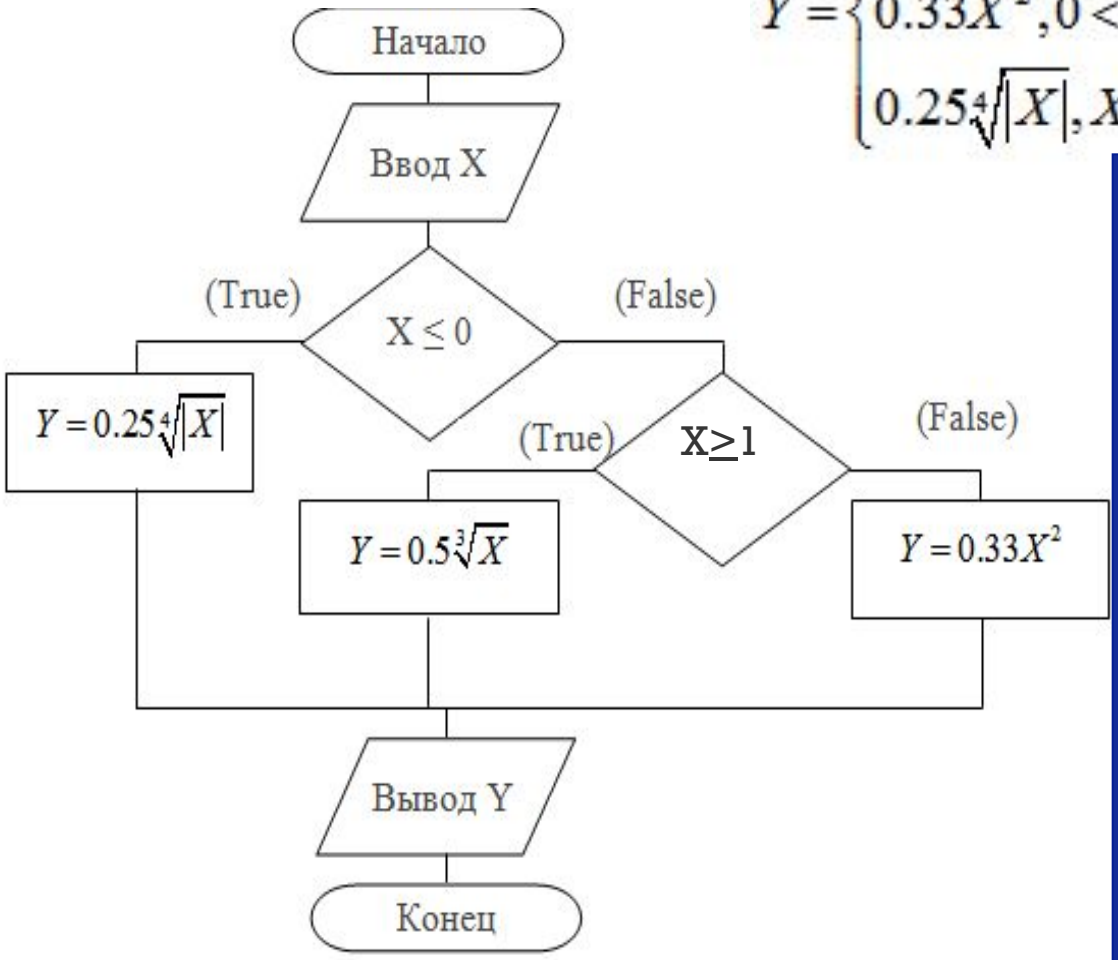
Оператор условия if (сокращенная форма)



If <логическое условие>
then
<оператор>;

Пример . Для заданного с клавиатуры значения X вычислить Y по формуле:

$$Y=\begin{cases}0.5\sqrt[3]{X}, X\geq 1;\\ 0.33X^2, 0<X<1;\\ 0.25\sqrt[4]{|X|}, X\leq 0.\end{cases}$$




```
PROGRAM PR_2;
    VAR X, Y: REAL;
BEGIN
  WRITELN('ВВЕДИТЕ X');
  READLN(X);
IF     X<=0
   THEN
      Y:= EXP(LN(ABS(X))/4)/4
    ELSE    IF X>= 1
            THEN
              Y:= EXP(LN(X)/3)/2
            ELSE
               Y:=SQR(X)*0.33;
  WRITELN('Y=', Y:9:3)
END.
```

Сравнение схем ELSE-IF и THEN-IF

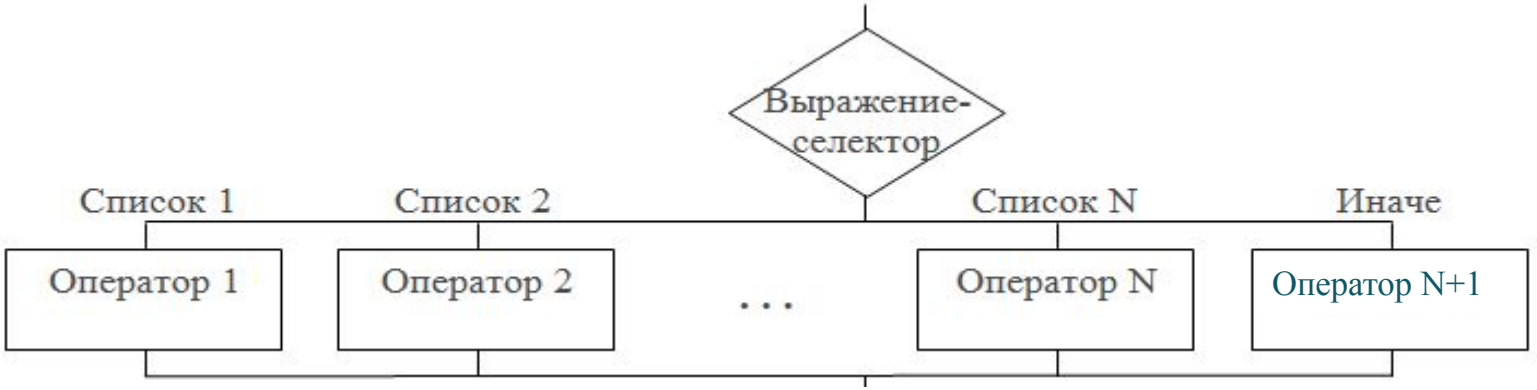
ELSE-IF

```
var
   a: integer;
 begin
   write('Введите целое число: ');
   readln(a);
    if a = 0 then
      writeln('zero')
   else
      if a = 1 then
         writeln('one')
      else
         if a = 2 then
            writeln('two')
         else
            writeln('unknown');

readln
End.
```

THEN-IF

```
var
   a: integer;
 begin
   write('Введите целое число: ');
   readln(a);
    if a <> 0 then
      if a <> 1 then
         if a <> 2 then
            writeln('unknown')
         else
            writeln('two')
      else
         writeln('one')
   else
      writeln('zero');

readln
End.
```

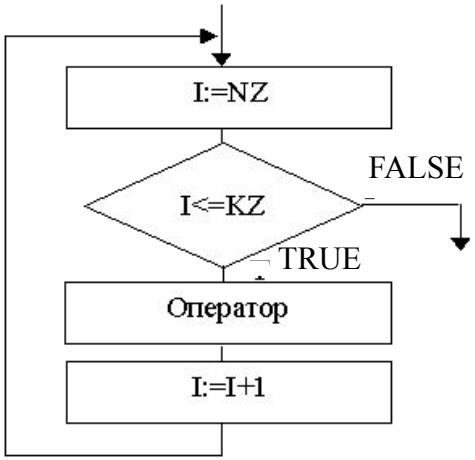
Оператор выбора




```
Case <выражение-селектор> of
      <значение1>:<оператор1>;
            …
      <значение n>:<операторN>
       else <оператор по умолчанию>
end;
```

```
Program PR_3;
var ch: char;
 begin
  write ('Введите символ: ');
  readln (ch);
  case ch of
   '0'..'9':   write ('Это число');
   'a'..'z','A'..'Z':
     write ('Это английская буква');
    'а'..'я','А'..'Я':
     write ('Это русская буква')
   else   write ('Это  спецсимвол')
end;
 readln
end.
```

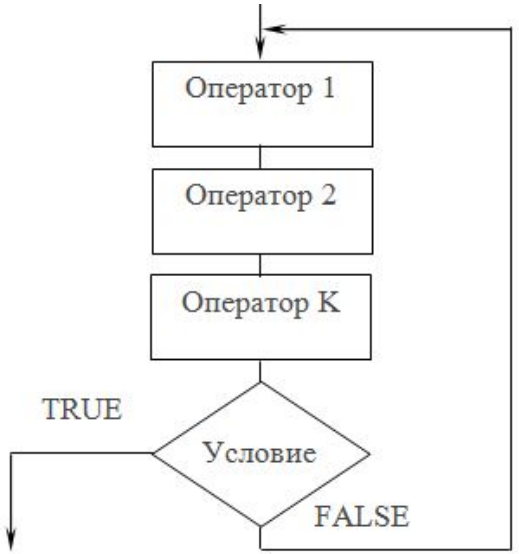
Операторы цикла



Цикл с параметром

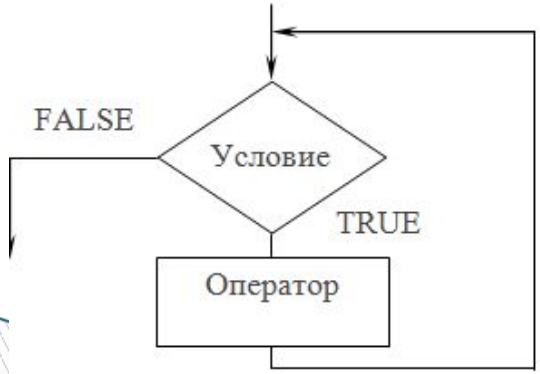
for <Сч>:=<Н_з> **to** <к_з> **do**
<оператор>;

for <Сч>:=<Н_з> **downto** <к_з> **do**
<оператор>;



Цикл с постусловием

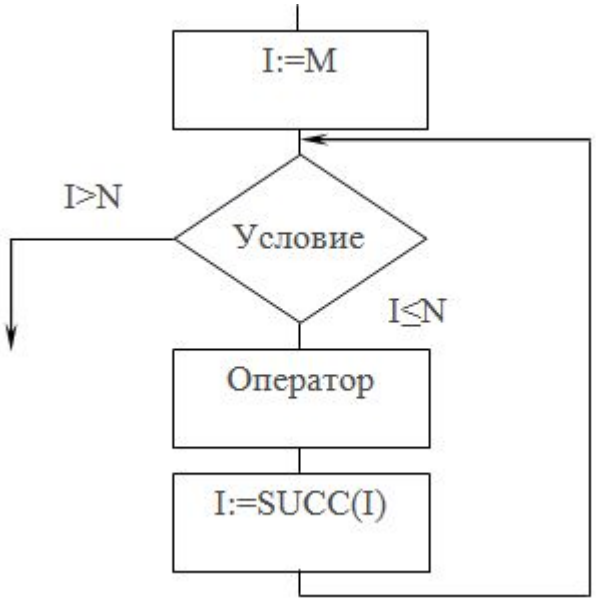
Repeat
<оператор1>;
<оператор2>;
<операторN>;
Until <логическое выражение>;



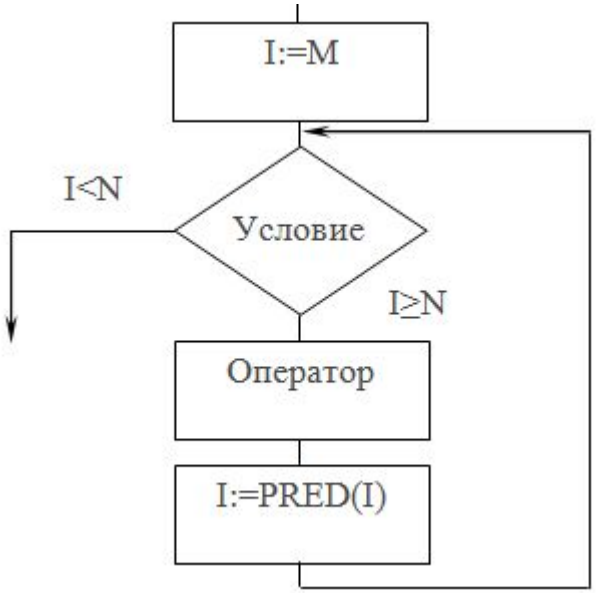
Цикл с предусловием

While <логическое выражение> **do**
<оператор>;

Цикл с параметром

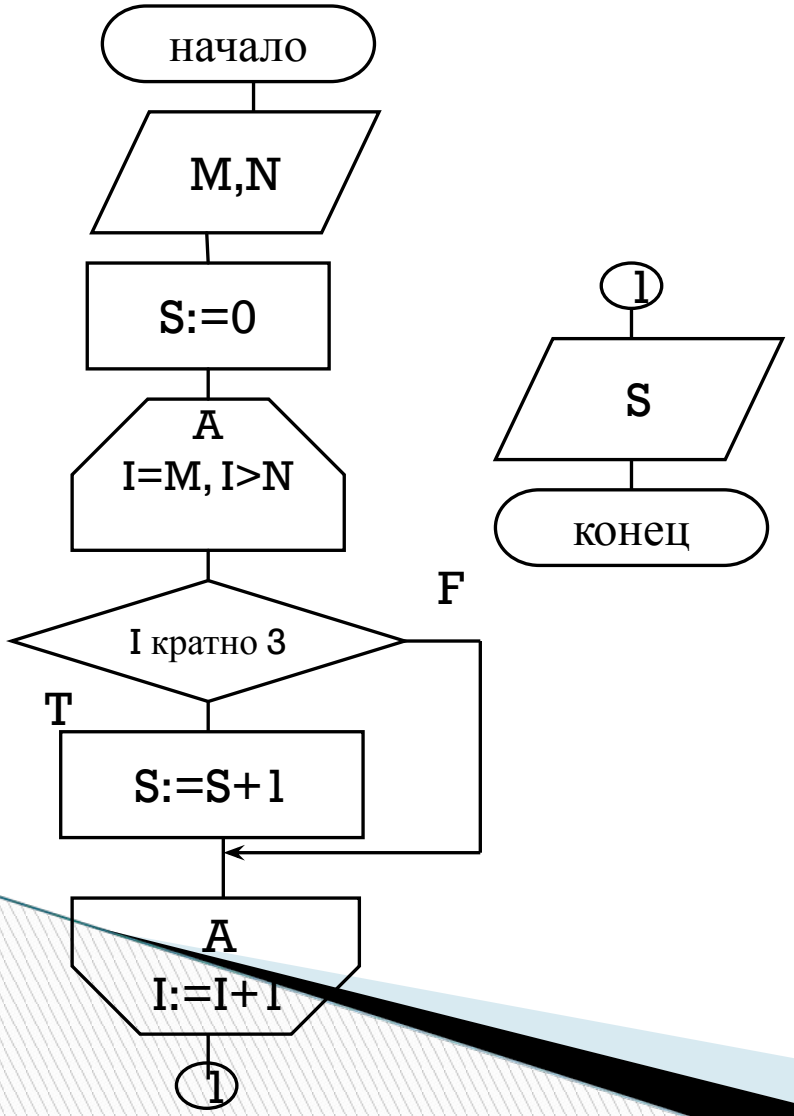


for I:=M **to** N **do**
<оператор>;
для M< N



for I:=M **downto** N **do**
<оператор>;
для M> N

Пример. Найти сумму S всех целых чисел, кратных 3 на отрезке [M, N].

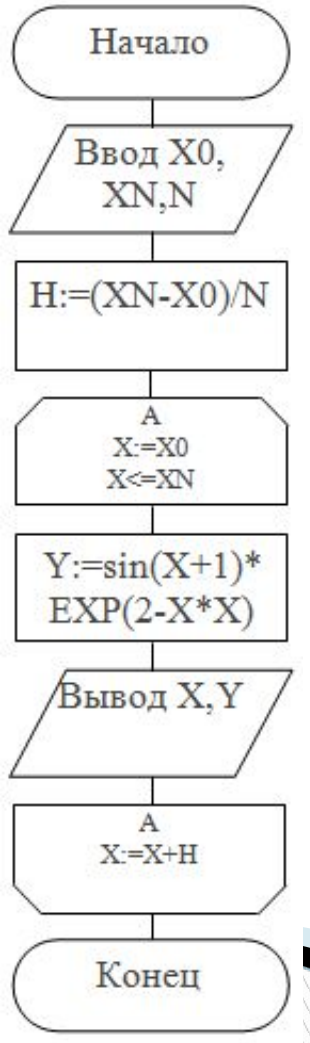



```
PROGRAM    PR;
VAR  X, S: REAL;
        I, M, N: INTEGER;
BEGIN
      WRITELN('ВВЕДИТЕ M И N');
      READLN(M, N);
      S:=0;
     FOR I:=M TO N   DO
         IF I MOD 3 = 0
                  THEN S := S + I;
       WRITELN('S=', S:6:4)
END.
```

Пример.

Табулировать функцию F(X) в N равноотстоящих точках, заданную на промежутк

$$F(x) = \sin(x+1) \cdot e^{-(x^2-2)}$$




```
PROGRAM   PR_while;
VAR     N: INTEGER;
          X, Y: REAL;
          H, X0, XN: REAL;
BEGIN
       WRITELN('ВВЕДИТЕ X0, XN, N');
      READLN(X0, XN, N);
      H := (XN - X0)/N;
      X:=X0;
WHILE X<=XN   DO
     BEGIN
        Y:= SIN(X+1)*EXP(2-X*X);
        X := X + H;
        WRITELN (' ПРИ X  ',  X:4:1,'    Y=',Y:9:6)
      END
END.
```

Вычисление предела последовательности

Последовательность {Xn} определена следующим образом:

$$X_n = \frac{n^2 + 2}{3n^2 - n + 1}, n = 1, 2, 3 \ldots$$

Найти предел последовательности {Xn}, принимая за него такое Xn, при котором |Xn – Xn-1| < ε.

```
PROGRAM LIM;
VAR   X, X1, E: REAL;
        N: INTEGER;
BEGIN
    WRITELN('ВВЕДИТЕ E');
    READLN(E);
    N := 1;
    X := 1;
       REPEAT
         X1 := X;
         X := ( N * N +2)/(3*N * N – N +1);
         N:=N+ 1;
       UNTIL ABS(X – X1) <E;
   WRITELN('Предел последовательности равен', X:12:8)
END.
```

Рекуррентные зависимости

В общем виде формулу для рекуррентных вычислений можно представить так:

$$Y_i = F(Y_{i-1}, Y_{i-2}, \ldots, Y_{i-k}).$$

для вычисления i-го члена последовательности Y_i, где $i > k$, используются k предыдущих членов последовательности $Y_{i-1}, Y_{i-2}, \ldots, Y_{i-k}$

Признаки рекуррентных формул: $(-1)^n$, X^n, $n!$ и подобные этим выражения, присутствующие в формуле общего члена бесконечного ряда.

Пример. Вычислить значение tgx:

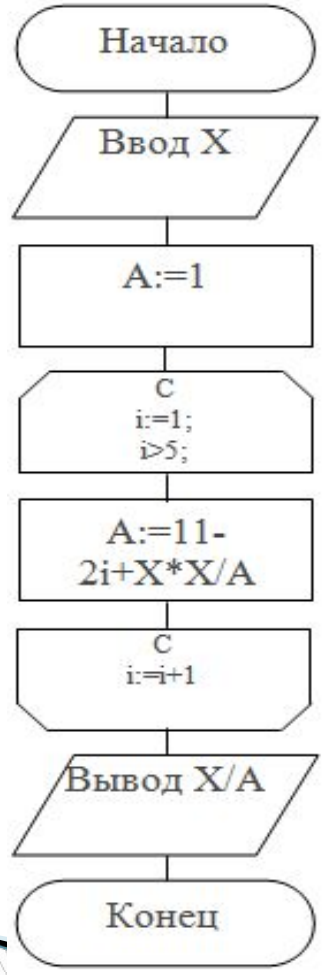
$$tgX = \cfrac{X}{1-\cfrac{X^2}{3-\cfrac{X^2}{5-\cfrac{X^2}{7-\cfrac{X^2}{9-X^2}}}}}$$

$A_i = F(A_{i-1})$

Номер 1	Член последовательности	Величина
0	A_0	1
1	A_1	$9-\frac{X^2}{A_0}$
2	A_2	$7-\frac{X^2}{A_1}$
3	A_3	$5-\frac{X^2}{A_2}$
4	A_4	$3-\frac{X^2}{A_3}$
5	A_5	$1-\frac{X^2}{A_4}$

$$A_i = 11-2i-\frac{X^2}{A_{i-1}}, A_0 = 1, i = 1,2,3,4,5.$$

$$A_i = 11 - 2i - \frac{X^2}{A_{i-1}}, A_0 = 1, i = 1,2,3,4,5.$$




```
PROGRAM    tang;
   VAR    X, A: REAL;
          I: INTEGER;
   BEGIN
     WRITELN('ВВЕДИТЕ X');
     READLN(X);
     A := 1;
     FOR I := 1 TO 5  DO
    A := 11 - 2 * I - X * X/A;
     WRITELN('tgX =   ', X/A:8:5)
   END.
```

Вычисление суммы бесконечного ряда с использованием рекуррентной формулы.

$$Y=\sum_{i-1}^{\infty}(-1)^i\cdot\frac{X^{2i}}{(2i)!}.$$ Вычисления завершить при $$\left|\frac{X^{2i}}{(2i)!}\right|<\varepsilon.$$

1. $$Y=\sum_{i-1}^{n}A_i, A_i=(-1)^i\cdot\frac{X^{2i}}{(2i)!}, |A_i|<\varepsilon.$$

2. $$\frac{A_i}{A_{i-1}}=\frac{(-1)^i\cdot\frac{X^{2i}}{(2i)!}}{(-1)^{i-1}\cdot\frac{X^{2(i-1)}}{(2(i-1))!}}=-\frac{\frac{X^{2i}}{(2i)!}}{\frac{X^{2i-2}}{(2i-2)!}}=-\frac{\frac{X^{2i}}{(2i-2)!\cdot(2i-1)\cdot(2i)}}{\frac{X^{2i}\cdot X^{-2}}{(2i-2)!}}=-\frac{X^2}{(2i)\cdot(2i-1)}.$$

Отсюда находим рекуррентную формулу:

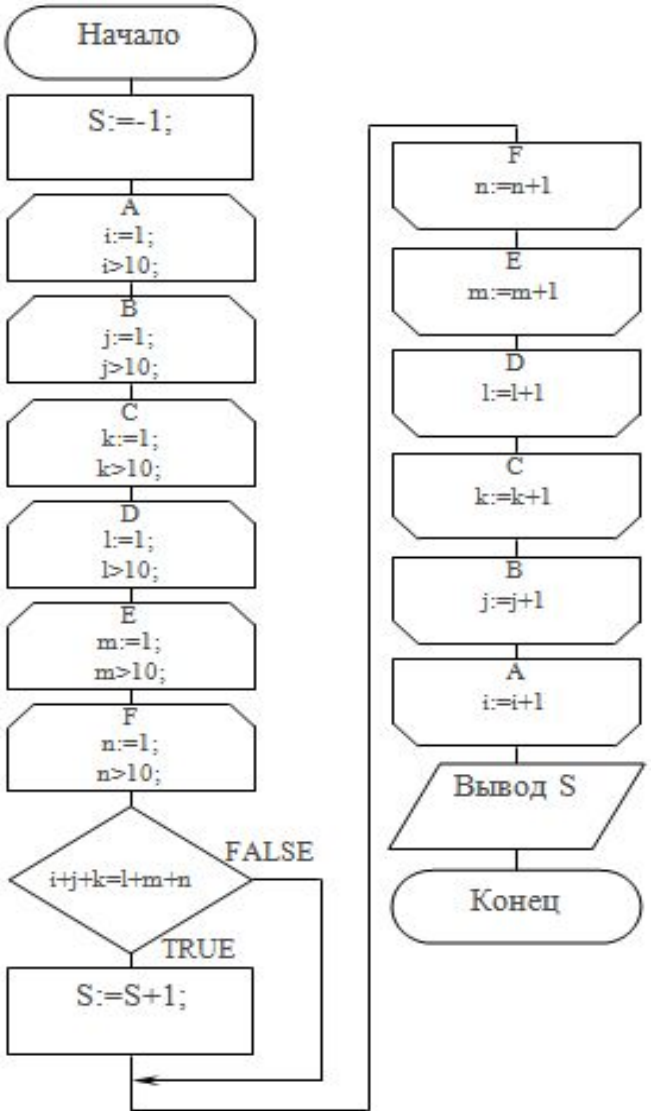
$$A_i=-\frac{X^2\cdot A_{i-1}}{(2i)\cdot(2i-1)}, i=2,3,\ldots,n; A_1=-\frac{X^2}{2}.$$

$$A_i = -\frac{X^2 \cdot A_{i-1}}{(2i)\cdot(2i-1)}, i = 2,3,\ldots,n; A_1 = -\frac{X^2}{2}.$$

```
PROGRAM   RYD;
   VAR      Y, E, A, X: REAL;
        I: INTEGER;
   BEGIN
      WRITELN('Введите X, E');
      READLN(X, E);
      I:= 1;
      A:= -X*X/2;
       Y:=A;
        WHILE ABS(A) >= E   DO
         BEGIN
            I := I+1;
            A:= -X*X/2/I/(2*I - 1)*A;
            Y := Y + A;
        END;
     WRITELN('Y=', Y:10:6)
   END.
```

Вложенный арифметический цикл

Используя вложенный цикл, определить число счастливых билетов S, номера которых меняются от 000001 до 999999.




```
PROGRAM   Happy;
VAR   S,I, N, J, K, L, M: INTEGER;
BEGIN
   S:=-l;
   FOR I := 0   TO  9  DO
     FOR J:=0   TO  9   DO
       FOR  K:=0   TO  9   DO
         FOR  L:=0   TO  9   DO
           FOR  M:=0   TO  9   DO
             FOR N:=0   TO  9   DO
                  IF     I+J + K = L+M + N
                                 THEN S:=S+ 1;
  WRITELN('ЧИСЛО счастливых билетов = ', S:6:0)
END.
```

HALT [(Код)]