Язык программирования Паскаль.

Операторы циклов в Паскале

Циклические процессы.

Циклическими называются программы, содержащие циклы. Цикл — это многократно повторяемый участок программы.

В организации цикла можно выделить следующие ***этапы:***

- подготовка (инициализация) цикла (ПЦ);
- выполнение вычислений цикла (тело цикла) (ТЦ);
- модификация параметров (МП) или подготовка данных (ПД)

Виды циклических алгоритмов

Цикл с предусловием - ПОКА *(while).*

Перед выполнением операторов тела цикла осуществляется проверка условия на продолжение цикла. Если условие справедливо (ветвь «Да»), то цикл повторяется, иначе, происходит выход из цикла.

Особенности данной структуры цикла:

а) число повторений цикла заранее неизвестно;

б) если при первой же проверке условия получается "Нет", то цикл не выполняется ни разу;

в) возможен «бесконечный цикл», когда проверка условия не дает выхода на ветвь «Нет».

Цикл с постусловием – ДО

(repeat)

В блоке "Проверка условия" осуществляется проверка условия на прекращение цикла. Если условие справедливо (ветвь «Да»), то происходит выход из цикла, в противном случае цикл повторяется при новых значениях исходных данных.

Особенности данной структуры цикла:

а) число повторений цикла заранее неизвестно;

б) так как условие проверяется в конце цикла, то тело цикла выполняется как минимум один раз;

в) возможен «бесконечный цикл», когда проверка условия не дает выхода на ветвь «Да».

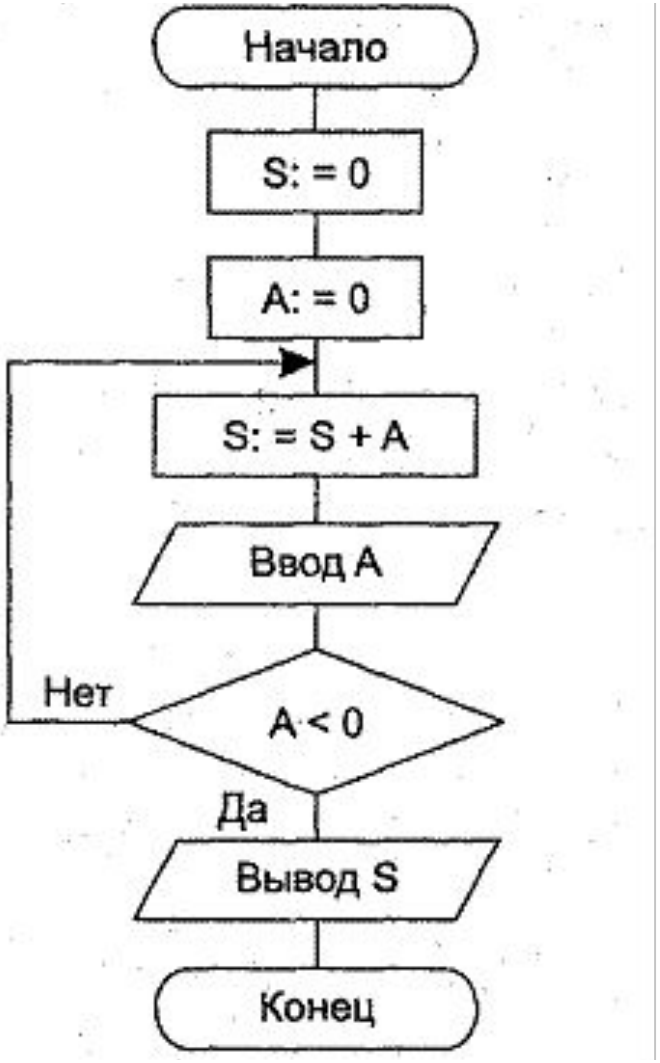
Цикл с параметром (*for*)

Параметр цикла определяет число повторений цикла. Для параметра цикла указывается его начальное значение, конечное значение и шаг изменения. Тело цикла выполняется при каждом значении параметра цикла.

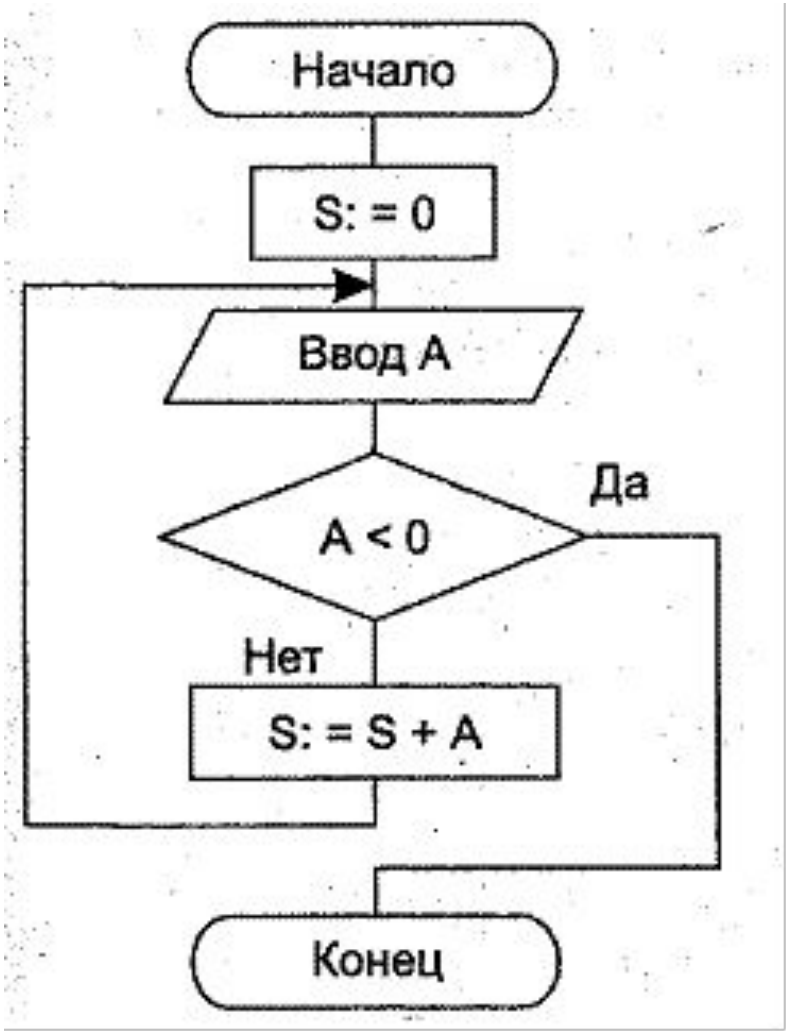
Особенность данной структуры цикла заключается в том, что уже перед началом выполнения цикла известно количество его повторений.

Использование циклов постусловием (repeat) и предусловием (while).

Задача 1: требуется вводить с клавиатуры числа и подсчитывать их сумму. Сумму необходимо подсчитывать до первого введенного отрицательного числа.



цикл с постусловием



цикл с предусловием

1. Цикл с постусловием (Repeat) (итерационный)

repeat <оператор> **until** <условие>

Пример: подсчет суммы натуральных чисел от 1 до 50:

```
var i, sum: integer;
begin
   sum:=0; i:=0;
   repeat
      i:=i+1;
      sum:=sum+i;
   until i=50;
   writeln('Сумма равна: ',sum);
   readln;
end.
```

2. Цикл с предусловием (While) (итерационный)

while <условие> **do** <оператор>.

```
var i, sum: integer;
begin
    sum:=0; i:=0;
    while i<50 do begin
       i:=i+1;
       sum:=sum+i;
    end;
    writeln('Сумма равна: ',sum);
     readln;
end.
```

Счетчик в Паскале вычисляется по рекуррентному выражению:

```
c=c+1;
```

или

```
inc(c);
```

где **c** – накапливаемое количество

1 – шаг увеличения

Сумма в паскале или сумматор

Сумма в Паскале вычисляется по рекуррентному выражению:

S=S+Y

где **S** – накапливаемая сумма

Y – слагаемое

*3. Цикл со счетчиком (For) (*детерминированный)

for <переменная>:=<нач> **to** <кон> **do** <оператор>
for <переменная>:=<нач> **downto** <кон> **do** <оператор>

Пример: вывести на экран таблицу квадратов чисел от 2 до 20.

```
var i: integer;
begin
    for i:=2 to 20 do
        writeln(i,' ',sqr(i));
end.
```

```
for i:=20 downto 2 do
    writeln(i,' ',sqr(i));
```

Пример: рассчитать сумму чисел от 1 до 50:

```
var i, sum: integer;
begin
     sum:=0;
        for i:=1 to 50 do
            sum:=sum+i;
      writeln(sum);
end.
```

Цикл называется **итерационным**, если число повторений тела цикла заранее неизвестно, а зависит от значений параметров (некоторых переменных), участвующих в вычислениях (циклы ***While*** и ***Repeat)***

Цикл называется **детерминированным**, если число повторений тела цикла заранее известно или определено (цикл ***For***).

```
program summ; {Пример: Сортировка массива по
     возрастанию}
var a:array[1..100] of integer;
      s,j,n,i:integer;
begin
write('vvedite chislo elementov: ');
  read(n);
   for i:=1 to n do
  begin
     write('a[i]: ');                    ввод чисел с экрана
      read(a[i]);
   end;
```

```
for i:=1 to n-1 do
 begin
   for j:=i+1 to n do
     begin
      if a[i]>a[j] then
       begin
        s:=a[i];
        a[i]:=a[j];
        a[j]:=s;
       end;
     end;
 end;
  for i:=1 to n do
  write(a[i]);
end.
```

a(5) = (3, 7, 2, 5, 8), n = 5

Циклы по i= 1..4, j= 2..5

I ит.: i=1, j=2

II ит.: i=1, j=3 s=3, a(1)=2, a(3)=3

III ит.: i=1, j=4

IV ит.: i=1, j=5

V ит.: i=2, j=3 s=7, a(2)=3, a(3)=7

…

Язык программирования Паскаль.

Символьные и строковые переменные

1. Символьный тип

Тип данных, переменные которого хранят ровно один символ (букву, цифру, знак препинания и т.п.) называется символьным, в Паскале — **char**.

```
var a, b, chislo, ch: char;
begin
a:='*';  b:='R'; chislo:='3'
ch:='Д';
```

символ 3 отличается от целого числа 3 тем, что символ не может быть использован в арифметических операциях

```
……..
write('Выйти из игры? (Да/Нет)');
readln(ch);
if ch:='Д' then ...{выходить}...  else ...{продолжать}...;
```

Символьные переменные в памяти компьютера **хранятся в виде числовых кодов**!

ASCII (Кодировка символов, которая используется в Паскале)

код пробела – 32,

код ‘A’ — 65, ‘B’ — 66, ‘C’ — 67,

код символа ‘1’ — 48, ‘2’ — 49, ‘.’ — 46 и т. п.

символы (с кодами, меньшими 32) являются *управляющими:*

Символ с кодом 10 - переносит курсор на новую строку,

с кодом 7 — вызывает звуковой сигнал,

с кодом 8 — сдвигает курсор на одну позицию влево.

Под хранение символа выделяется 1 байт (8 бит) → 2^8 =256 символов

ASCII (American Standard Code for Information Interchange)

0 -	16 - ▶	32 -	48 - 0	64 - @	80 - P	96 - '	112 - p
1 - ☺	17 - ◀	33 - !	49 - 1	65 - A	81 - Q	97 - a	113 - q
2 - ☻	18 - ↕	34 - "	50 - 2	66 - B	82 - R	98 - b	114 - r
3 - ♥	19 - ‼	35 - #	51- 3	67 - C	83 - S	99 - c	115 - s
4 - ♦	20 - ¶	36 - $	52 - 4	68 - D	84 - T	100 - d	116 - t
5 - ♣	21 - §	37 - %	53 - 5	69 - E	85 - U	101 - e	117 - u
6 - ♠	22 - ▬	38 -&	54 - 6	70 - F	86 - V	102 - f	118 - v
7 -	23 - ↨	39 - '	55 - 7	71 - G	87 - W	103 - g	119 - w
8 -	24 - ↑	40 - (	56 - 8	72 - H	88 - X	104 - h	120 - x
9 -	25 - ↓	41 -)	57 - 9	73 - I	89 - Y	105 - i	121 - y
10 -	26 - →	42 - *	58 - :	74 - J	90 - Z	106 - j	122 - z
11 -	27 - ←	43 - +	59 - ;	75 - K	91 - [	107 - k	123 - {
12 -	28 - ∟	44 - ,	60 - <	76 - L	92 - \	108 - l	124 - |
13 -	29 - ↔	45 - -	61 - =	77 - M	93 - j	109 - m	125 - }
14 - ♫	30 - ▲	46 - .	62 ->	78 - N	94 - ^	110 - n	126 - ~
15 - ☼	31 - ▼	47 - /	63 - ?	79 - O	95 - ÷	111 - o	127 - ⌂
16 - ▶	32 -	48 -0	64 - @	80 - P	96 -	112 - p	

128 -А	144 - Р	160 - а	176 - ▒	192 - └	208 - ╨	224 -р	240 - Ё
129 - Б	145 - С	161 - б	177 - ▒	193 - ┴	209 - ╤	225 -с	241- ё
130 -В	146 - Т	162 - в	178 - ▓	194 - ┬	210 - ╥	226 -т	242 - Є
131 - Г	147 - У	163 - г	179 - │	195 - ├	211 - ╙	227 -у	243 - є
132 -Д	148 - Ф	164 - д	180 - ┤	196 - ─	212 - ╘	228 -ф	244 - Ї
133 -Е	149 - Х	165 - е	181 - ╡	197 - ┼	213 - ╒	229 - х	245 - ї
134 - Ж	150 - Ц	166 - ж	182 - ╢	198 - ╞	214 - ╓	230 - ц	246 - Ў
135 -З	151 - Ч	167 - з	183 - ╖	199 - ╟	215 - ╫	231 - ч	247 - ў
136 -И	152 -Ш	168 - и	184 - ╕	200 -	216 - ╪	232 - ш	248 - °
137 - Й	153 -Щ	169 - й	185 - ╣	201 - ╔	217 - ┘	233 - щ	249 - ●
138- К	154 - Ъ	170 - к	186 - ║	202 - ╩	218 - ┌	234 - ъ	250 - ∙
139 -Л	155 - Ы	171 - л	187 - ╗	203 - ╦	219 - █	235 - ы	251 - √
140 -М	156 - Ь	172 - м	188 - ╝	204 - ╠	220 - ▄	236 - ь	252- №
141 -Н	157 - Э	173 - н	189 - ╜	205 - ═	221 - ▌	237 - э	253 - ¤
142 -О	158 - Ю	174 - о	190 - ╛	206 - ╬	222 - ▐	238 - ю	254 - ■
143- П	159 - Я	175 - п	191 - ┐	207 - ╧	223 - ▀	239 - я	255 -
144 - Р	160 - а	176 - ▒	192 - └	208 - ╨	224 - р	240 - Ё	

Для данных символьного типа определены следующие стандартные функции:

Функция	Назначение
chr(x)	Преобразует выражение x типа byte в символ и возвращает значение символа
ord(ch)	Преобразует символ ch в его код типа byte и возвращает значение кода.
pred(ch)	Возвращает предыдущий символ.
succ(ch)	Возвращает следующий символ.

Для получения кода символа (символ→код) используется функция **ord( )**:

```
program ASCII;
  var  i: byte; {число, занимающее 1 байт, значения — от 0 до 255}
         ch: char;
    begin
      writeln(‘введите символ);
       readln(ch);
       i:=ord(ch);
       writeln('код символа ', ch, ' — это ', i);
   end.
```

Для перехода от кода к символу (код→символ) используется функция **chr**:

```
program ASCII;
var  ch: char;
begin
  for ch:=#32 to #255 do write(chr(ch),'—>',ch,'  ');
  readln;
end.
```

Сравнение символов:

Также как и числа, символы можно сравнивать на =, <>, <, >, <=, >=.

В этом случае Паскаль сравнивает не сами символы, а их коды.

Все символы упорядочены, т.к. имеют свой личный номер. Важно, что соблюдаются следующие отношения:

'A' < 'B' < 'C' < ... < 'X' < 'Y' < 'Z' '0' < '1' < '2' < ... < '7' < '8' < '9'

Задача. Написать программу, которая считывает две литеры и печатает больше, равна или меньше первая литера второй.

```
Program Sravnenie;
Var    First, Second : char;
Begin
   write ('Введите две литеры: ');
   readln (First, Second);
   write ('Первая литера ');
   if First > Second    then
     write ('больше второй. ')
    else
     if First = Second   then
        write ('равна второй. ')
       else
         write ('меньше второй. ');
End.
```

2. Строковый тип

Строка - последовательностей из символов.

var ***<имя_строки>: string[<длина>]***

```
var   s: string;       {максимальная длина строки — 255 символов}
      s1: string[n];       {максимальная длина — n символов}.
      s2: string[10];    { длина 10 символов}.
```

s:='Hello, world!'

```
program Hello;
var s: string;
begin
    write('Как Вас зовут: ');
    readln(s);
    write('Привет, ',s,'!');
end.
```

ВАЖНО! Если при выполнении программы необходимо ввести значение для нескольких строковых переменных, для каждой из них должен быть указан свой оператор ввода READ(LN). Нельзя записать READLN(a,b,c) или READ(a,b,c)

Хранение строк

В памяти компьютера строка хранится в виде последовательности из символьных переменных, у них нет индивидуальных имён, но есть номера, начинающиеся с 1. Перед первым символом строки имеется ещё и нулевой, в котором хранится символ с кодом, равным длине строки.

```
Var   S: string;        {занимает в памяти 256 байт=1+255}
      S1: string[100];  {занимает в памяти 101 байт }
```

Сравнение строк:
Строки сравниваются последовательно, по символам.
Примеры: 'ананас'<'кокос', 'свинья'>'свинина', ' ' < 'A', 'hell'<'hello'.

Склеивание (конкатенация) строк:
s:= 'abc'+'def'+'ghi'; переменная s будет содержать ‘abcdefghi’.

```
Var
  a,b,c:string;
begin
  writeln(‘Введите фамилию, имя, отчество‘);
  readln(a);    {разные операторы readln}
  readln(b);
  readln(c);
  write(a+b+c);
 end.
```

Процедуры и функции для работы со строками:
length(s: string): integer (после двоеточия записан тип значения, возвращаемого функцией, в нашем случае — целое число). Эта функция возвращает длину строки s.

Пример: Составить программу определяющую, какая из двух фамилий длиннее. Фамилии имеют разную длину.

```
Var   a,b: string;
begin
      writeln(‘Введите две фамилии‘);
      readln(a);
      readln(b);
  if  length(a) > length(b) then write(a) else write(b);
 end.
```

Другие процедуры и функции для работы со строками.

Процедура или функция	***Назначение***	***Пример***
функция Copy(s: string; start: integer; len:integer): string	Возвращает вырезку из строковой переменной s, начиная с символа с номером start, длина которой len	s:=’Бестолковый’; s1:=Copy(s,4,4); *{в s1 станет ‘толк’}*
функция Pos(s1: string; s: string): byte	Ищет подстроку s1 в строке s. Если находит, то возвращает номер символа, с которого начинается первое вхождение s1 в s; если s1 не входит в s, то функция возвращает 0	n:=pos(‘министр’, ‘администратор’); *{=3}* n:=pos(‘abc’, ‘stuvwxyz’);*{=0}*
процедура Insert(s1: string; s: string; start: integer)	Вставляет строку s1 в строковую переменную s начиная с символа с номером start.	S:=‘кот’; insert(‘мпо’,s,3); *{в s станет ‘компот’}*
процедура Delete(s: string; start: integer; len: integer)	Удаляет из строковой переменной s фрагмент, начинающийся с символа с номером start и длиной len	s:= ‘треугольник’; delete(s,4,7); *{в s останется ‘трек’}*

Пример программы. Вывод строки в обратном порядке:

```
var a,b: string;
      i: integer;
begin
    write(‘введите строку: ');
    readln(a);
  for i:=length(a) downto 1 do
              b:=b+a[i];
   writeln(b);
end.
```

Программа формирует строку из 26 символов, содержимым которой является последовательность заглавных букв латинского алфавита.

```
Program stringElements;
Var
  Str : string[26]; {длина строки = 26}
  i : char;
Begin
  Str:=' ';
  for i := 'A' to 'Z' do
    Str := Str + i;
  writeln(Str);
End.
```

Программа показывает автоматическое изменение длины строки после тех или иных операций с нею

```
Program StringLength;
Var
  S : string;    {макс. длина строки = 255}
Begin
  S:='';       {пустая строка}
  writeln (S,' ',SizeOf(S),' ',Length(S));          {размер=256, длина=0}
  S:='Пример длинной строки'; {присваиваем строке некоторое  значение}
  writeln (S,' ',SizeOf(S),' ',Length(S));          {размер=256, длина=21}
  Delete(S,7,8);           {удаляем из строки 8 символов, начиная с 7}
  writeln (S,' ',SizeOf(S),' ',Length(S));          {размер=256, длина=13}
  S:=S+' символов';                          {добавляем к строке строку}
  writeln (S,' ',SizeOf(S),' ',Length(S));          {размер=256, длина=22}
End.
```

Язык программирования Паскаль.

Перечисляемый и ограниченные типы

1. Перечисляемый тип

Перечисляемый тип задаётся перечислением тех значений, которые он может получать. Каждое значение именуется некоторым идентификатором и располагается в списке, обрамлённом круглыми скобками.

type имя = (идентификатор, идентификатор, ..., идентификатор);

```
Type
    Days = (Mon, Tue, Wed, Thu, Fri, Sat, Sun);
    Rainbow  = (Red, Orange, Yellow, Green, Light_blue, Blue, Violet );
var   day: Days;
      color: Rainbow;

day:=Wed;
if day>Fri then writeln('Сегодня выходной');
if day=Mon then writeln('Началась рабочая неделя');
```

Для любого перечислимого типа постулируются следующие аксиомы:

- Все идентификаторы пронумерованы, т.е. каждый идентификатор имеет свой порядковый номер. Первому идентификатору соответствует порядковый номер 0, следующим 1, 2, и т.д.
- Все идентификаторы различны т.е. $\mathbf{w_i \neq w_j}$, если $\mathbf{i \neq j}$.
- Все идентификаторы упорядочены, т.е. $\mathbf{w_i < w_j}$ если $\mathbf{i < j}$.
- Значениями переменной типа T могут быть только идентификаторы $\mathbf{w_i, \ldots w_j}$.
- Не должно быть повторения идентификаторов в других типах.

<u>Задача:</u> Определить число дней в месяце.

```
 Program day_in_month;
Type  Months=(Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov,
Dec);
Var  m:months;
       n: integer;
       c:char;
Begin repeat
       writeln('Введите номер месяца 0..11');
       readln(n);
           if (n>=0) and (n<=11) then m:=months(n)
case m of
     Feb: n:=28;
     Jan, Mar, May, Jul, Aug, Oct, Dec:  n:=31
else n:=30
end;
writeln('В месяце ', n, '  дней')
writeln(‘continue? Y/N’);
readln(c);
      until (c=’n’) or (c=’N’);
End.
```

2. Ограниченный тип

Ограниченный тип данных представляет собой интервал значений порядкового типа, называемого базовым типом. Описание типа задаёт наименьшее и наибольшее значения, входящие в этот интервал.

```
type SmallLatin = 'a'..'z';     {малые латинские буквы}
     Holidays   = Sat..Sun;     {выходные}
Var  sl: SmallLatin;
```

```
Var a: 1..25; ch: 'a' ..'z';
```

переменные *a* и *ch* могут принимать значения только из указанного интервала