

Циклы по условию на языке Pascal

Учитель информатики:
Корогод В.А



Цель урока

Изучить циклы с предусловием *While... do* и с постусловием *Repeat... until* на примере решения задач с использованием рекуррентных соотношений и бесконечных рядов.



Повторим!

Цикл – процесс многократного повторения каких-либо действий. Язык Паскаль имеет три оператора, с помощью которых можно организовать циклическую структуру:

- ✓ **Цикл с параметром** (счетчиком)
“Для” (*For ...*)
- ✓ **Цикл с предусловием** “Пока” (*While ... do*)
- ✓ **Цикл с постусловием** “До” (*Repeat ... until*)

Поговорим о цикле `For...to...do`

Пусть решается простая задача вывода на экран целых чисел от 1 до 10.

Для этой задачи идеально подходит цикл со счетчиком `For...to...do`.



Программа

```
var i: integer; {счетчик}
Begin
  For i:=1 to 10 do
    Writeln(i);
End.
```



Надо помнить

- ✓ В цикле `For ... to ... do` начальное значение переменной *i* меньше предельного.
- ✓ Шаг изменения *i* по умолчанию равен **+1**.
- ✓ Переменная *i* НИКОГДА не может стоять СЛЕВА от оператора присваивания «**:=**». Тип переменной *i* – любой **скалярный** (*integer, byte, char, др.*) **КРОМЕ вещественного**. *i* НИКОГДА не может быть *real*.



Когда **For...to...do** уступает место

 Если число повторений известно наперед, цикл **For**

 идеален!
Циклы **While... do** и **Repeat... until** используются в целом классе задач, когда повторные вычисления заканчиваются **по заданному наперед условию**:

- ✓ при табулировании графиков функций на заданном интервале **с заданным шагом**;
- ✓ для расчета **с заданной точностью** сумм бесконечных асимптотических рядов, с помощью которых выражаются тригонометрические функции, трансцендентные числа $\pi = 3,1415\dots$ и основание натурального логарифма $e=2,72\dots$;
- ✓ для вычисления квадратного корня из числа методом Герона.



Сравнение циклов While и Repeat

Формат оператора цикла с предусловием:
<присвоение начальных значений переменным, входящим в условие>

While <условие> **do**

begin

<действие 1>

<действие 2>

.....

<действие N>

<изменение условия>

end;

Тело цикла

Формат оператора цикла с постусловием:
< присвоение начальных значений переменным, входящим в условие >

< присвоение начальных значений переменным, входящим в условие >

Repeat

<действие 1>

<действие 2>

.....

<действие N>

<изменение условия>

until <условие>;

Тело цикла

Общее
в этих
циклах

1. **До цикла** задается **начальное значение переменных**, входящих в условие. **Условие** - выражение **булевского** типа.

2. **В теле цикла** значение переменных, входящих в условие, должны **обязательно изменять свое значение**, иначе возникнет ситуация «зависания».

Цикл предусловием **While ... do**

Решение задачи о выводе
10 целых чисел на экран
с использованием цикла
While... do:

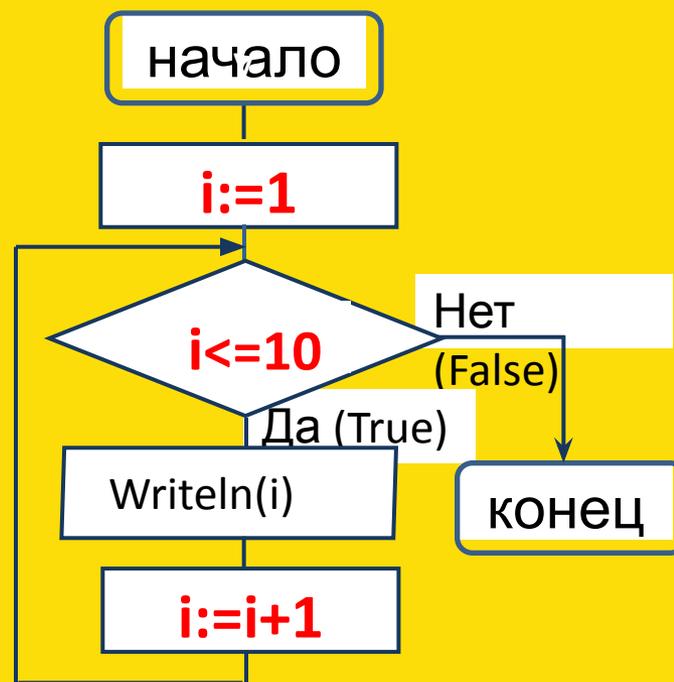


Программа

```
var i: integer; {СЧЕТЧИК}
Begin
  i := 1; {начальное значение}
  While i <= 10 do
    begin
      Writeln(i);
      i := i + 1
    end; {While}
End.
```



Блок-схема алгоритма



Особенности цикла **While...do**

 Так как условие проверяется на входе в цикл, то **при неверном условии цикл не выполняется ни разу**, т.е. не выполняются операторы, стоящие в теле цикла.

 Операторы, входящие в тело цикла, обязательно заключаются в **операторные скобки**, если в теле цикла более одного оператора. В противном случае будет выполняться только первое действие, стоящее под заголовком цикла.

 В теле цикла должно обязательно выполняться **действие, приводящее к изменению условия**, иначе цикл станет бесконечным. Оператор, в котором изменяются переменные, входящие в условие, может стоять не обязательно в конце цикла.



Цикл с постусловием Repeat ... until

Решение задачи о
выводе
10 целых чисел на экран
с использованием цикла

Repeat ... until:

 **Программа**

```
var i: integer; {счетчик}
```

```
Begin
```

```
  i := 1; {начальное значение}
```

```
  Repeat
```

```
    Writeln(i);
```

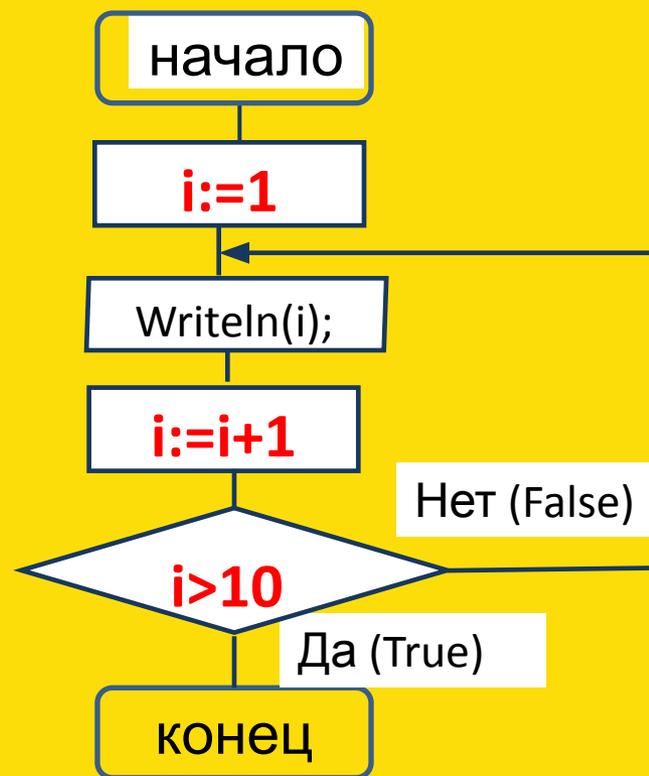
```
    i := i + 1
```

```
  Until i > 10
```

```
End.
```



**Блок-схема
алгоритма**



Особенности цикла Repeat...until

-  Так как условие проверяется на выходе из цикла, то *цикл выполняется хотя бы один раз.*
-  Все операторы, стоящие в теле цикла, выполняются ДО проверки условия, поэтому *операторные скобки не ставятся.*
-  В *теле цикла* должно обязательно выполняться *действие, приводящее к изменению условия*, иначе цикл станет бесконечным. Оператор, в котором изменяются входящие в условие переменные, может стоять не обязательно в конце цикла.



Решаем самостоятельно

Два варианта одной задачи:



Вариант 1

Два игрока А и В бросают кубик N раз, суммируя результаты бросков. Напишите программу, определяющую победителя после N бросков.



Вариант 2

Два игрока А и В бросают кубик и суммируют результаты бросков. Победителем объявляется игрок, набравший первым объявленную сумму очков S . Напишите программу, определяющую победителя.



Какой цикл целесообразно использовать?

Почему?



Задача о рассеянном джентльмене

Некто отправился на работу из дома (пункт А) в офис (пункт В). Расстояние между домом и офисом равно 1 км. Пройдя половину пути, джентльмен вспомнил, что не попрощался с семьей, повернул назад и прошел третью часть расстояния и, боясь опоздать на работу, снова повернул и прошел четверть расстояния. Затем снова повернул и прошел $\frac{1}{5}$ расстояния и т.д.



На каком расстоянии от офиса окажется джентльмен, если продолжит свои метания? Провести вычисления расстояния с точностью до 1 см.

Анализ задачи

- ✓ Расстояние, на котором окажется джентльмен от дома (A), можно записать так:

$$S_A = 1/2 - 1/3 + 1/4 - 1/5 + 1/6 - 1/7 + \dots (-1)^{i+1} / i \dots$$

Так как расстояние AB=1, джентльмен окажется от места работы на расстоянии S:

$$S = 1 - S_A = 1 - [1/2 - 1/3 + 1/4 - 1/5 + 1/6 - 1/7 + \dots (-1)^{i+1} / i \dots]$$

Таким образом, решение задачи сводится к вычислению суммы **гармонического ряда**:

$$S = 1 - 1/2 + 1/3 - 1/4 + 1/5 - \dots (-1)^{i+1} / i - \dots$$

- ✓ Суммирование продолжаем, пока абсолютное значение разности сумм, вычисленных на (i+1)-м шаге и i-м шаге, больше наперед заданного малого числа **eps**, т.е. $|S - S1| > eps$.

Таким образом, ряд вычисляется приближенно с заданной погрешностью.

- ✓ Для решения задачи используем цикл **While**.



Программа для задачи о

ДЖЕНТЛЬМЕНЕ

```
Program harmony_riad; {Вычисление гармонического ряда};
uses crt;
const eps=0.00001; {заданная точность вычисления}
var i: integer; S,S1 : real; p: integer;
Begin
  clrscr; {очистка экрана}
  s1:=0; {начальное значение сумматора}
  s:=1; {суммирование 1-го члена ряда}
  i:=1; {начальное значение для 1-го члена ряда }
  p:= -1; {знак числа отрицательный}
  while abs(s1-s) > eps do
  begin
    s1:=s; {запоминаем сумму, вычисленную на
    предыдущем шаге}
    i:=i+1; {формирование следующего члена ряда
    числа }
    s:=s+p/i; {суммирование знакопеременного ряда}
    p:= - p; {смена знака}
  end; {while}
  writeln('S от офиса=', s:7:5);
  tt
```



Рекуррентные соотношения

! В математике известно понятие **рекуррентной последовательности чисел** (от латинского «*recurrere*» – «возвращаться»).

! Это понятие вводят так: пусть известно k чисел a_1, \dots, a_k , которые являются началом числовой последовательности. Следующие элементы этой последовательности вычисляются так:

$$a_{k+1} = F(a_1, \dots, a_k); \quad a_{k+2} = F(a_1, \dots, a_{k+1}); \quad a_{k+3} = F(a_1, \dots, a_{k+2}); \dots,$$

$$a_{k+i} = F(a_1, \dots, a_{k+i-1})$$

! Величина **рекуррентного соотношения**, в которых очередной член последовательности **глубиной рекурсии** выражен через один или несколько предыдущих.



Примеры рекуррентных соотношений

С помощью **метода рекуррентных соотношений** вычисляют:

- ✓ арифметические и геометрические последовательности;
- ✓ последовательность чисел Фибоначчи;
- ✓ бесконечные последовательности (ряды) для тригонометрических функций;
- ✓ бесконечные последовательности (ряды) для функций e^x , \sqrt{x} , $\ln(1+x)$;
- ✓ выражения вида:

$$\sqrt{x + \sqrt{x + \dots + \sqrt{x}}}$$

N корней

$$x^2 + \frac{x}{x^2 + \frac{2}{x^2 + \frac{4}{x^2 + \frac{8}{x^2 + \frac{16}{x^2 + \frac{32}{x^2 + \frac{64}{x^2 + \frac{128}{x^2 + \frac{256}{x^2}}}}}}}}}}$$



Анализ задачи о вычислении \sqrt{a}

✓ **Задача.** Вычислить квадратный корень целого числа a по рекуррентной формуле *Герона* $X_{i+1} = (X_i + a/X_i)/2$ при заданной точности вычисления eps .

✓ **Алгоритм вычисления.** Зададим X_1 - начальное значение корня из числа a .

Например, $X_1 = a/2$.

Тогда каждое следующее приближение вычисляется через предыдущее:

$$X_2 = (X_1 + a/X_1)/2$$

$$X_3 = (X_2 + a/X_2)/2$$

$$X_{i+1} = (X_i + a/X_i)/2$$

Вычисление продолжаем до тех пор, пока выполнится модуль разницы между X_{i+1} и X_i станет меньше заданной погрешности вычисления eps : $|X_{i+1} - X_i| < eps$

Для решения задачи используем цикл *Repeat... until*.



Program mysqrt

```
program mysqrt;  
  {Вычисление квадратного корня числа по формуле  
  Герона} { $x=(x+a/x)/2$ }  
  uses crt;  
  const eps=0.0001;  
  var   a: integer;   x, x1: real;  
Begin  
  clrscr;  
  write('Введите число a='); readln(a);  
  x:=a/2; {начальное значение корня}  
  repeat  
    x1:=x; {запоминаем предыдущее приближение корня}  
    x:=(x+a/x)/2; {вычисляем (i+1)-е приближение корня}  
  until abs(x-x1)<eps;  
  writeln (' Корень числа ',a,' равен ',x);  
  readln  
End.
```



Задачи с бесконечными рядами

- ✓ **Задача.** Вычислить сумму бесконечного ряда $S = x - x^3/3! + x^5/5! - x^7/7! + \dots$ с заданной точностью *eps*.
- ✓ Будем использовать рекуррентную формулу, с помощью которой каждый последующий член ряда выражается через предыдущий., т.е. справедливо соотношение:
$$u_n = q u_{n-1}$$
- ✓ Определяем величину q , последовательно рассмотрев отношение второго члена к первому, третьего ко второму, четвертого к третьему и т.д.:
$$q_1 = u_2 / u_1 = - (x^3/3!) / x = - x^2 / (2 * 3)$$
$$q_2 = u_3 / u_2 = - (x^5/5!) / (x^3/3!) = - x^2 / (4 * 5)$$
$$q_3 = u_4 / u_3 = - (x^7/7!) / (x^5/5!) = - x^2 / (6 * 7)$$
- ✓ Для произвольного q справедлива рекуррентная формула:
$$q = - x^2 / k / (k+1), \text{ где } k = 2, 4, 6, \dots$$
- ✓ В языках программирования стандартная функция $\sin(x)$ рассчитывается с помощью асимптотического ряда S .



Program mysin

```
Program mysin;  
  const eps=0.00001; {точность вычислений}  
  var   u: real; s: real; k : integer;  
Begin  
  write ('Введите x='); readln(x);  
  s:=0; {обнуление суммы}  
  k:=0; {начальное значение переменной k}  
  u:=x; {первый член ряда}  
  while abs(u) > eps do  
    begin  
      s:=s+u; {суммируем ряд}  
      k:=k+2; {формируем четное число }  
      u:= - u * sqr(x) / k / (k+1) ; {k-член ряда}  
    end;  
  writeln (' сумма ряда S=',S);  
  writeln (' sin x=', sin(x));  
  readln  
End.
```



Арифметическая последовательность

В символьной записи *арифметическую прогрессию* можно представить так:

$$a, a+d, a+2*d, a+3*d, \dots, a+(N-1)*d.$$

Здесь a – первый член последовательности, d – разность между двумя соседними членами, N – число членов последовательности.

Например:

$$1 + 3 + 5 + 7 + 9 + \dots + 99 \quad (a=1, d=2)$$

$$2 + 4 + 6 + 8 + \dots + 100 \quad (a=2, d=2)$$



Два типа задач:

- ✓ Вычислить сумму S при заданном числе членов N (используем цикл *For*).
- ✓ Определить число членов N при достижении заданной суммы S (используем цикл *While*).



Геометрическая прогрессия

Геометрической последовательностью называется последовательность, в которой отношение между ее членом и членом, ему предшествующим, есть величина постоянная.



В символьной записи это можно записать так:

$$a, a*r, a*r^2, a*r^3, \dots, a*r^{(N-1)}$$

Здесь буквой a обозначен первый член последовательности, буквой r – ее знаменатель и буквой N – число членов последовательности.

Например, если $a=4$, $r=0.5$, $N=7$, получаем последовательность

$$4, 2, 1, 0.5, 0.25, 0.125, 0.625$$



Сумма геометрической прогрессии S определяется формулой: $S = a + a*r + a*r^2 + a*r^3 + \dots + a*r^{(N-1)}$



Задача «Изумруды»

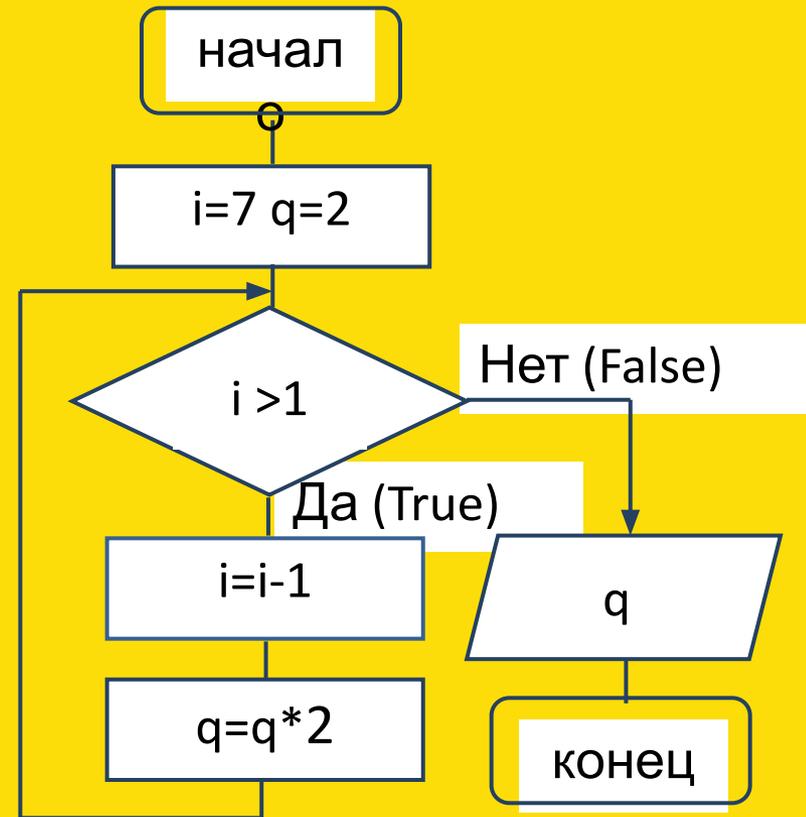


Задача

У царя было семь сыновей. В сундуке лежали изумруды. Пришел первый сын и взял половину того, что было. Пришел второй сын и взял половину того, что осталось и т.д. **Каждый из сыновей приходил и забирал половину того, что осталось.** Наконец, пришел последний, седьмой сын и увидел почти пустой сундук — с двумя изумрудами. Сколько изумрудов лежало в сундуке первоначально? При решении задачи будем использовать цикл **While...do**.



Блок-схема



Программа

```
Program izumrud;  
  Var I, q, r: integer;  
Begin  
  i=7  'номер 7-го сына  
  q=2  'седьмому сыну досталось 2 изумруда  
  r=2  'разность геометрической прогрессии  
  WriteLn('i=', I, ' q=', q)  
  While i>1 do  
  begin  
    q=q*r  'вычисление i-того члена прогрессии  
    i=i-1  'номер следующего сына уменьшается на 1  
    WriteLn('i=', I, ' q=', q)  
  end;  
  WriteLn('всего ',q,' изумрудов')  
End.
```



Задание

1. Не используя стандартные функции (за исключением *abs*), вычислить с разной точностью $\text{eps} > 0$:

a) $Y1 = e^x = 1 + x/1! + x^2/2! + \dots + x^n/n! \dots;$

b) $Y2 = \cos(x) = 1 - x^2/2! + x^4/4! - \dots + (-1)^n x^{2n}/(2n)! + \dots;$

c) $Y3 = \ln(1+x) = x - x^2/2 + x^3/3 - \dots + (-1)^{n-1} x^n/n + \dots$

2. Определить количество итераций (повторений) n в зависимости от *eps*.

3. Занести данные в таблицу:

eps	y1	n	y2	n	y3	n
0.001						
0.0001						
0.00001						



Можно ли вычислить число π ?

- Одним из самых знаменитых чисел в математике, вычисленных приближенно, является число π .
- Число π определяется как отношение длины окружности к ее диаметру.
- Вот значение π до 16-го знака:
3.1415926535897932...
- Числу воздвигают монументы и посвящают стихи.
- Во всех алгоритмических языках есть стандартная функция для вычисления числа π . В языке Паскаль это

*Гордый Рим трубил победу
Над твердыней Сиракуз;
Но трудами Архимеда
Много больше я горжусь.
Надо только постараться
И запомнить все как есть:
Три – четырнадцать – пятнадцать
– Девяносто два и шесть! (С.Бобров)*



Монумент числу π установлен в Сиэтле

← функция Pi .



Число π и бесконечные ряды

У числа π очень интересная история. Еще в 200 г. до нашей эры греческий математик Архимед (тот самый Архимед, который, купаясь в море, вдруг воскликнул «Эврика!» и открыл знаменитый закон Архимеда!) утверждал, что число π меньше, чем $22/7$ и больше, чем $223/71$.

Многие математики выводили формулы для приближенного

вычисления π в виде бесконечных рядов, например:

$$\pi = 4 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + \dots$$

(Готфрид Лейбниц (около 1673 г.)

$$\pi = 2\sqrt{3}[1 - 1/(3 \cdot 3) + 1/(3^2 \cdot 5) - 1/(3^3 \cdot 7) + \dots],$$

(Шарп (около 1699 г.)

$$\pi = \sqrt{6 + 6/1^2 + 6/2^2 + 6/3^2 + 6/4^2 + 6/5^2 \dots}$$

(Эйлер (около 1736 г.)).

Здесь $\sqrt{\quad}$ — обозначение знака квадратного корня из числа.

Домашнее задание

Задача 1. Напишите программу для вычисления n -й степени числа X . Вычисление описать каждым из трех вариантов оператора цикла: *For... to...do*, *While... do*, *Repeat... until*.

Задача 2. Вычислив асимптотический ряд $S = 1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots (-1)^{i+1} (1/(2i+1)) \dots$ с точностью $\text{eps} = 0.0001$, вы узнаете, чему равно число $\pi = 4 * S$. Напишите программу вычисления числа π и сравните со значением π , вычисленным с помощью стандартной функции *Pi*.

Замечание. Здесь удобно использовать такую формулу для нечетного числа:

$$i := i + 2 \quad (i = 1, 2, 3 \dots)$$



Инструкция к демонстрации презентации

Запуск анимационных эффектов осуществляется с помощью

триггеров:
Цель
урока



Общее
в ЭТИХ
циклах

