The background of the slide is a blue-tinted photograph of a chalkboard. On the board, there are faint, light-colored mathematical diagrams, including what appears to be a tree structure or a flowchart. In the foreground, a wooden tray holds several pieces of white and light-colored chalk. The overall scene is slightly out of focus, emphasizing the text overlaid on the image.

Лекция №5

Циклические алгоритмы

План лекции

1. Оператор безусловного перехода
2. Повторяющиеся действия
3. Циклы с предусловием
4. Циклы с постусловием
5. Циклы со счетчиком
6. Сложноциклические структуры
7. Решение задач.

Оператор безусловного перехода

Оператор безусловного перехода – goto.

```
goto <метка>;
```

Для его использования, необходимо описать **метки** в разделе описаний, на которые будет осуществляться переход.

Label

```
metka1, metka2;
```

Меткой может быть число от 1 до 9999, либо последовательность латинских букв и цифр.

Оператор безусловного перехода

Оператор перехода предназначен для указания того, что выполнение программы должно продолжаться с точки программы, обозначенной меткой, значение которой стоит в операторе перехода. Метка в тексте программы располагается непосредственно перед помеченным оператором и отделяется от него двоеточием.

Пример:

```
....  
goto m1;  
....  
....  
m1: write (a,b);  
....
```

Оператор безусловного перехода

Пример использования оператора безусловного перехода:

Label	Var	
m1, m2;		a,b,c : real;
Var	Begin	
a,b,c : real;		read(a,b);
Begin		if b=0 then
read (a,b);		write ('На 0 делить нельзя!')
if b=0 then		else
goto m1;		begin
c:=a/b;		c:=a/b;
write (c);		write (c);
goto m2;		end
m1 : write ('На 0 делить нельзя !');		End.
m2:		
End.		

Из пример очевидна неэффективность использования оператора безусловного перехода.

Циклический алгоритм

Циклический алгоритм реализует повторение некоторых действий. Иными словами циклические алгоритмы включают в себя циклы.

Циклом называется последовательность действий, выполняемых многократно, каждый раз при новых значениях параметров.

Повторяющиеся действия

Повторяющиеся действия можно реализовать с помощью условного оператора и оператора безусловного перехода.

Так как язык Паскаль является структурным языком, использование операторов безусловного перехода считается не совсем уместным в Паскаль-программах, однако для того чтобы рассмотреть организацию циклических алгоритмов с помощью разветвляющейся структуры с безусловным переходом рассмотрим один из примеров.

Повторяющиеся действия

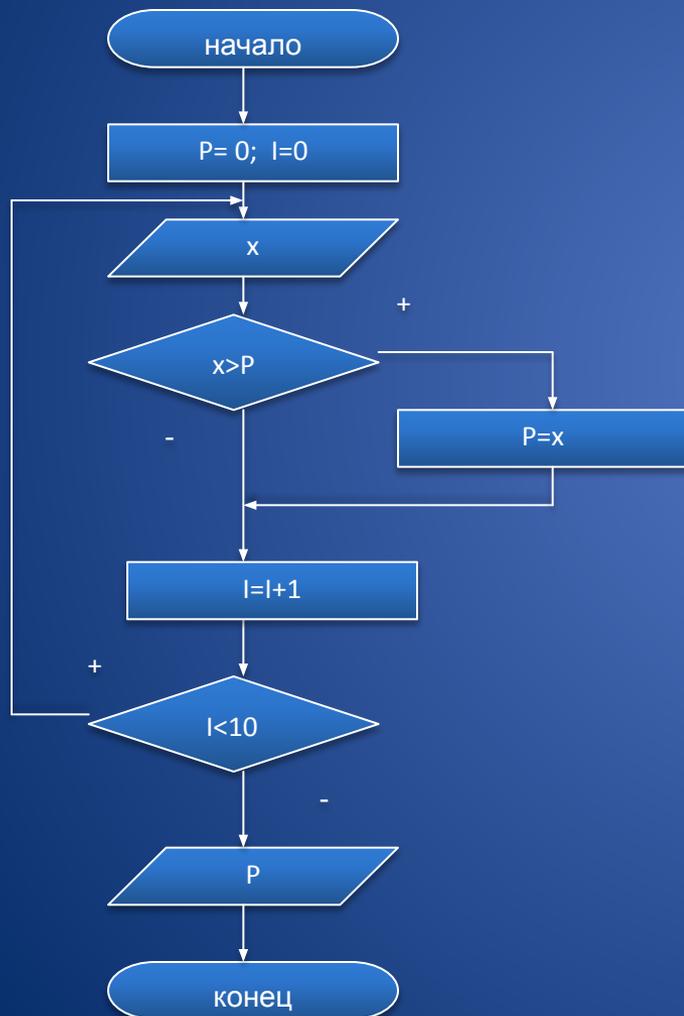
Задача 1.

Найти максимальное число из десяти положительных чисел.

Решение задачи можно построить по следующему алгоритму:

- 1) $i=0$
- 2) $p=0$
- 3) задать очередное значение x
- 4) если $x > p$, то $p=x$
- 5) $i=i+1$
- 6) если $i < 10$ перейти к пункту 3.
- 7) выдать значение p

Повторяющиеся действия



Label

m1;

Var

p, i, x : integer;

Begin

p:=0; i:=0;

m1 : read(x);

if x>p then

 p:=x;

 inc(i);

 if i<10 then

 goto m1;

 writeln (p);

End.

Повторяющиеся действия

В данном примере продемонстрированы повторяющиеся действия (**циклические, цикл**).

Телом цикла называют те операторы, которые повторяются.

```
...
m1 : read(x);
if x>p then
    p:=x;
inc(i);
if i<10 then
    goto m1;
...
```

Управляющей переменной цикла называют переменную, от которой зависит количество повторений.

В нашем примере, управляющей переменной цикла является переменная *i*.

Операторы цикла

На языке Паскаль различают следующие операторы цикла:

- Циклы с предусловием (`while`);
- Циклы с постусловием (`repeat ... until`);
- Циклы со счетчиком (`for`).

Циклы с предусловием

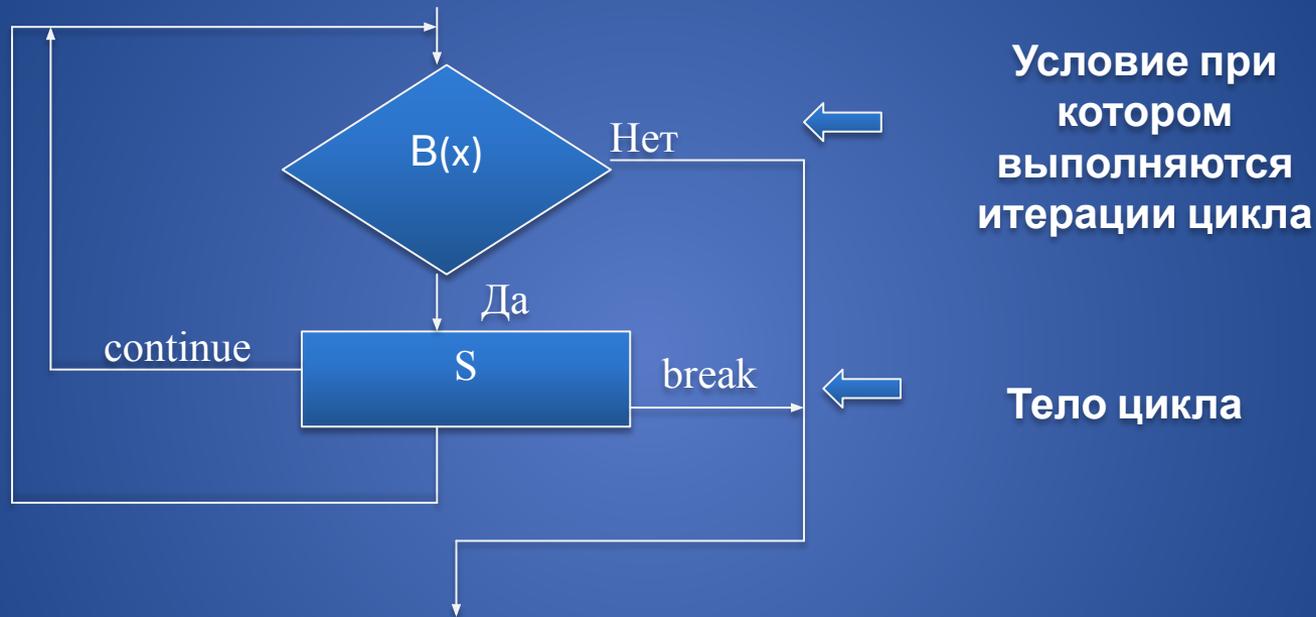
While – это оператор цикла итеративного типа с предусловием, так как в нем анализ конца цикла производится до выполнения операторов тела цикла. Он используется, когда количество повторений операторов тела цикла заранее неизвестно и определяется в процессе выполнения цикла.

По операторам `continue` и `break` можно перейти на анализ условия конца цикла или первый оператор после цикла соответственно.

```
while <логическое выражение> do
    begin
        <тело цикла, состоящее из группы операторов>
    end;
```

Таким образом, организовывать повторяющиеся (циклические) действия в программе будет более правильно без использования операторов безусловного перехода.

Циклы с предусловием



```
while B(x) do  
    S;
```

где $B(x)$ – логическое выражение, в том случае, когда это выражение будет иметь значение Ложь, произойдет выход из цикла;

S – один оператор, простой или составной; он должен включать операторы тела цикла, в том числе оператор изменения операторов логического выражения $B(x)$

Циклы с предусловием

Задача 2.

Лист бумаги делят пополам, полученную половину снова делят пополам и т.д. Определить, какое количество делений потребуется, для того чтобы получить частицу размером с атом. Начальная масса листа 1 грамм, масса атома 10^{-24} грамм.

Var

m,ma : real;

k :integer;

Begin

k:=0;

m:=1;

ma:=1e-24;

while m>ma do

begin

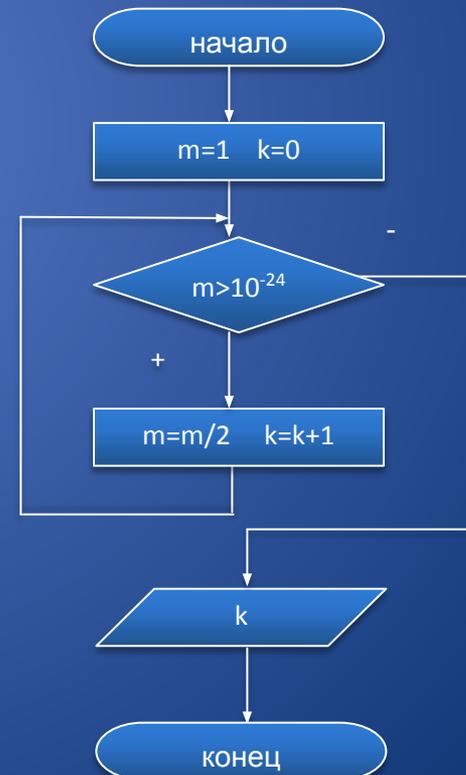
m:=m/2;

inc (k);

end;

writeln (k)

End.



Циклы с предусловием

При использовании цикла с предусловием надо помнить следующее:

- значение условия выполнения цикла должно быть определено до начала цикла;
- если значение условия истинно, то выполняется тело цикла, после чего повторяется проверка условия. Если условие ложно, то происходит выход из цикла;
- хотя бы один из операторов, входящих в тело цикла, должен влиять на значение условия выполнения цикла, иначе цикл будет повторяться бесконечное число раз.

Циклы с предусловием

Задача 2.

Определить значение суммы $S=1/x_1+1/x_2+\dots+1/x_n$, где n – количество слагаемых.

Var

s, x : real;

i, n : integer;

Begin

i:=0; s:=0;

read (n);

while i<n do

begin

inc (i);

read (x);

s:=s+1/x;

end;

writeln (s)

End.

Как будет работать программа, если пользователь введет $x=0$?

Циклы с предусловием

Программа завершит выполнение с сообщением об ошибке (Деление на 0).

```
Var
    s, x : real;
    i, n : integer;
Begin
i:=0; s:=0;
read (n);
while i<n do
    begin
    inc (i);
    read (x);
    if x=0 then
        break;
    s:=s+1/x;
    end;
writeln (s)
End.
```

```
Var
    s, x : real;
    i, n : integer;
Begin
i:=0; s:=0;
read (n);
while i<n do
    begin
    inc (i);
    read (x);
    if x=0 then
        continue;
    s:=s+1/x;
    end;
writeln (s)
End.
```

Циклы с предусловием

В случае использования оператора `break` исполнение программы завершится, т.е. произойдет выход из цикла и вывод накопленной суммы до введенного значения $x=0$.

В случае использования оператора `continue` произойдет переход на выполнение первого оператора тела цикла, и как следствие будут пропущены операторы стоящие после `continue` в теле цикла. Таким образом будет пропущена операция деления на 0.

Операторы `break` и `continue` могут использоваться так же и в других циклах : циклах с постусловием и циклах со счетчиком.

Циклы с постусловием

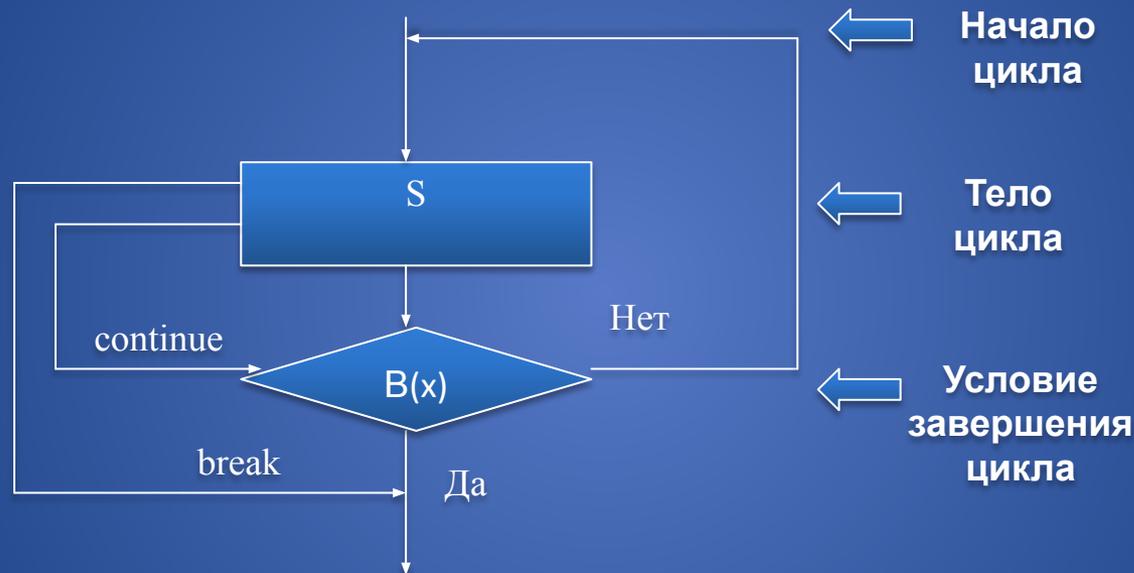
Repeat ... until – это оператор цикла итеративного типа с постусловием, так как в нем анализ конца цикла производится после выполнения операторов тела цикла. Он используется, когда количество повторений операторов тела цикла заранее неизвестно и определяется в процессе выполнения цикла. Операторы тела цикла выполняются хотя бы 1 раз.

repeat

<операторы тела цикла>

until <логическое выражение>

Циклы с постусловием



```
repeat  
S;  
until B(x);
```

где $B(x)$ – логическое выражение, при истинности которого происходит выход из цикла;

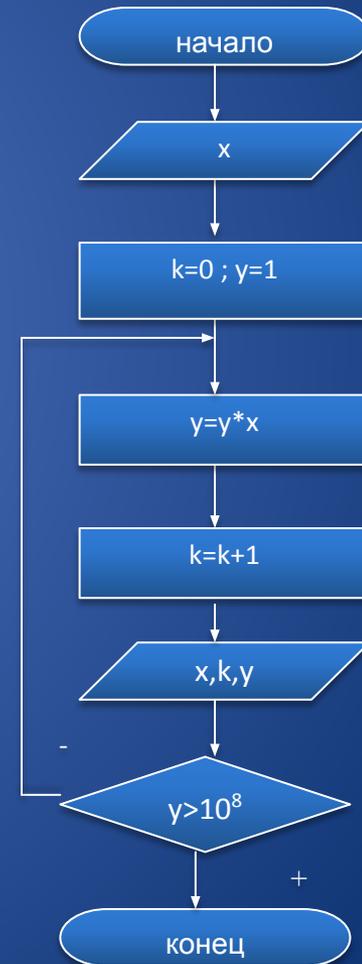
S – один или несколько операторов тела цикла.

Циклы с постусловием

Задача 3.

Дано $x > 1$. Вычислить и вывести степени x .
Вычисления производятся до тех пор, пока
вычисляемое значение не станет более 10^8

```
Var  
    k,x : integer;  
    y : longint;  
Begin  
    read (x);  
    k:=0;  
    y:=1;  
    repeat  
        y:=y*x;  
        inc (k);
```



Сравнение циклов с постусловием и предусловием

Есть небольшое отличие в организации цикла `repeat` по сравнению с `while`: для выполнения в цикле `repeat` нескольких операторов не следует помещать эти операторы в операторные скобки `begin ... end`. Зарезервированные слова `repeat` и `until` действуют как операторные скобки.

Сравнение циклов с постусловием и предусловием

Конструкция `repeat ... until` работает аналогично циклу `while`. Различие заключается в том, что цикл `while` проверяет условие до выполнения действий, в то время как `repeat` проверяет условие после выполнения действий. Это гарантирует хотя бы одно выполнение действий до завершения цикла.

Так же истинность логического выражения в операторе `repeat ... until` свидетельствует о завершении цикла, тогда как в операторе `while` – выполнение тела цикла.

Циклы со счетчиком

Циклы со счетчиком составляют такую конструкцию, в которой выполнение исполнительской части должно повторяться заранее определенное число раз.

Циклы со счетчиком используются довольно часто, и поэтому в языке Паскаль для этих целей имеется специальная конструкция.

```
for <управл. переменная цикла> :=<нач. зн-е> to/downto <кон. зн-е> do  
    <один оператор, являющийся телом цикла>
```

to – используется при шаге изменение управляющей переменной цикла равном 1.

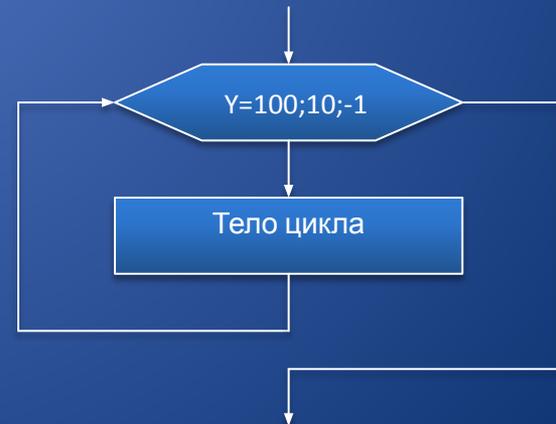
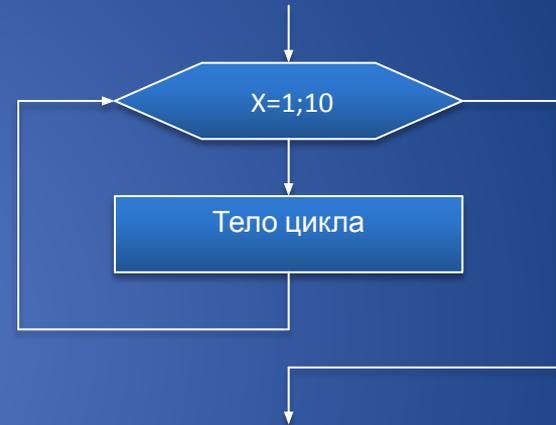
downto – используется при шаге изменение управляющей переменной цикла равном -1.

Циклы со счетчиком

Примеры :

```
for x:=1 to 10 do  
begin  
...  
end;
```

```
for y:=100 downto 10 do  
begin  
...  
end;
```



Циклы со счетчиком

Задача 4.

Найти максимальное число из десяти положительных чисел.

Var

p, i, x : integer;

Begin

p:=0;

for i:=1 to 10 do

begin

read (x);

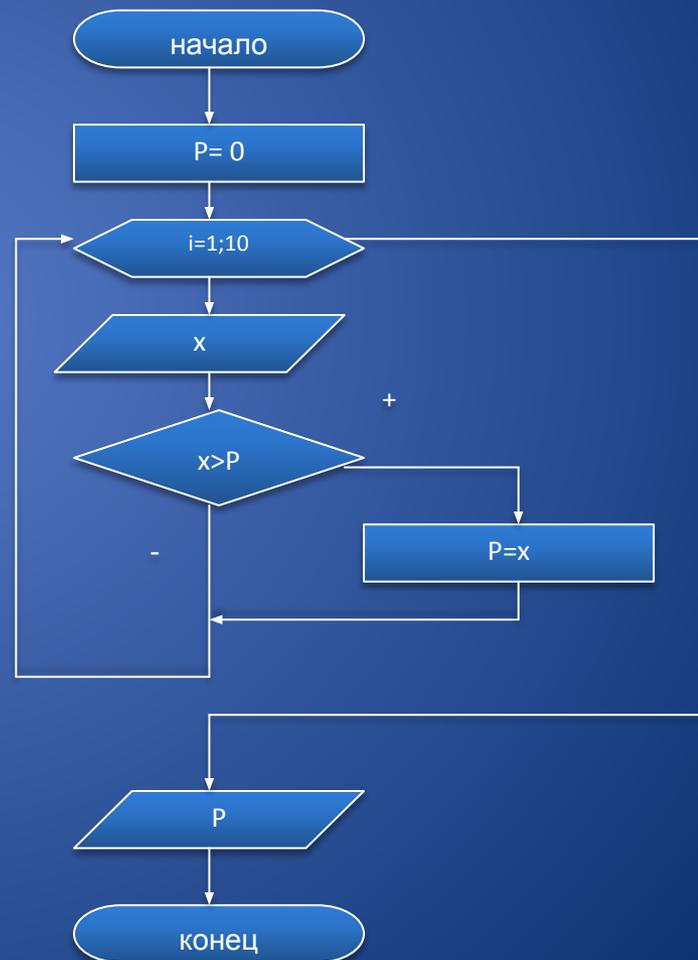
if x>p then

p:=x;

end;

writeln(p);

End.



Циклы со счетчиком

Управляющая переменная цикла со счетчиком не может быть вещественного типа.

В тех случаях, когда тело цикла выполняется заданное, известное количество итераций, но шаг цикла отличен от 1 или -1, то используют циклы `while`, `repeat ... until`.

Пример:

```
...  
x:=0;  
repeat  
    x:=x+2;  
    ...  
until x=10;  
...
```



Ситуации «защивливания»

Рассмотрим несколько примеров циклов :

1) for i:=100 to 10 do

...

2) while true do

...

3) x=5;

repeat

inc (x);

...

until x<2;

4) y:=10;

while y>5 do

begin

...

y:=y+2;

end;

Такие циклы будут выполняться, до тех пор пока не будет прервано исполнение такой «защивленной» программы. Эту ситуацию можно разрешить используя оператор `break` в теле цикла. Однако всегда следует следить за тем, чтобы циклы завершались корректно, т.е. в циклах были установлены такие параметры, которые бы не приводили к «защивливанию».

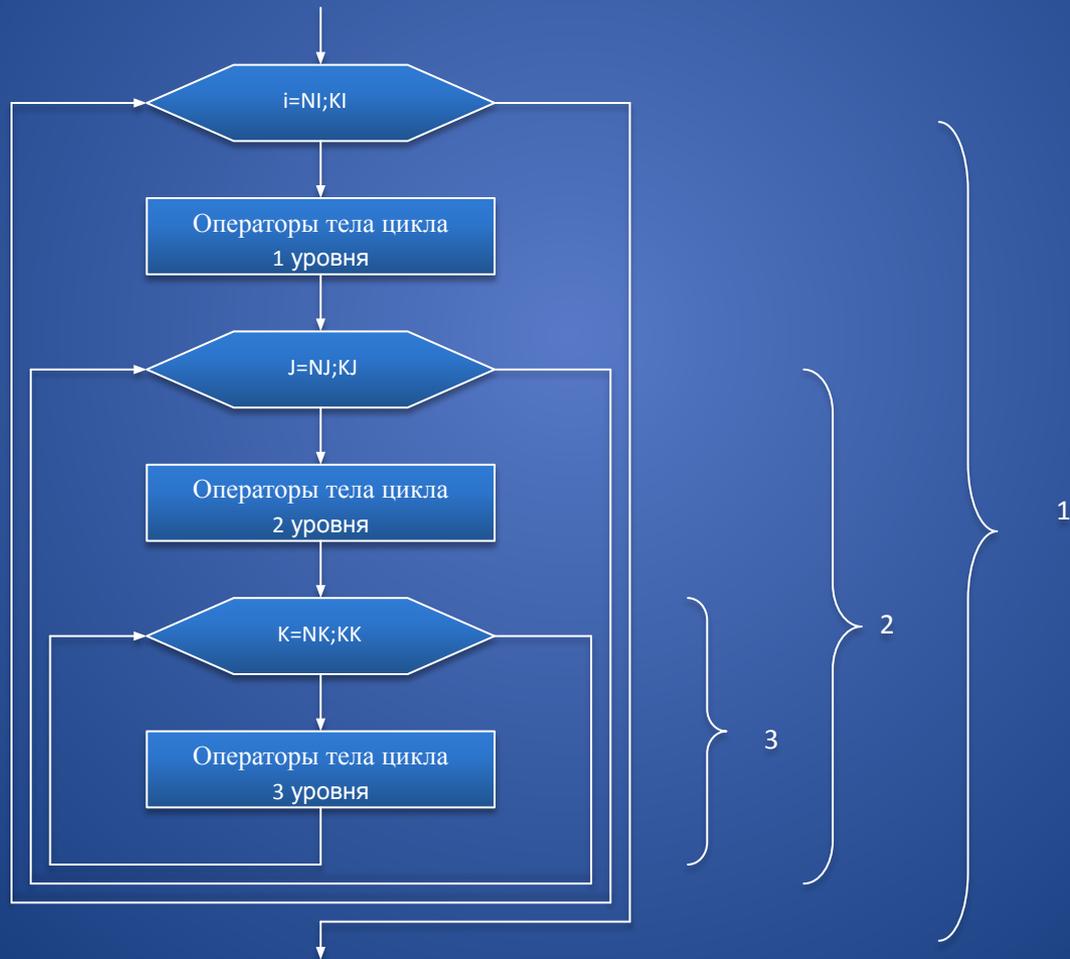
Сложноциклические структуры

Циклы могут быть **простые** и **вложенные** (кратные, циклы в цикле). Для решения многих задач так же используют структуру вложенных циклов, которую и называют **сложноциклической**. Вложенными могут быть циклы любых типов : for, while, repeat ... until.

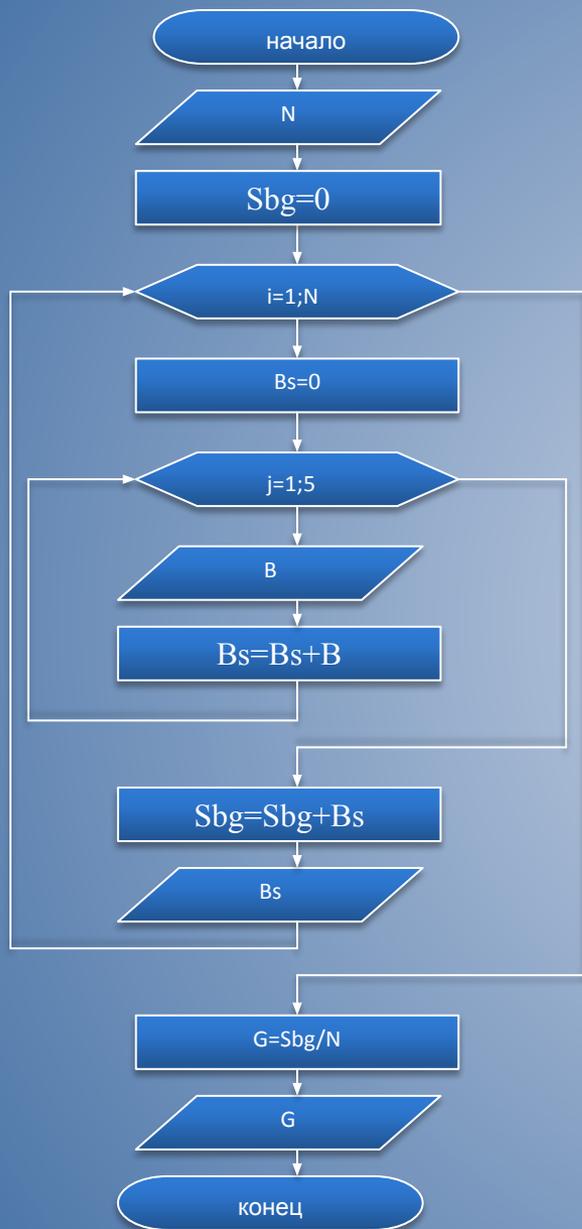
Пример :

```
...
for x:=1 to 10 do
  begin
    ...
    for y:=1 to 5 do
      begin
        ...
        end;
      ...
    end;
  ...
end;
...
```

Сложноциклические структуры



Задача 5. Подсчитать рейтинг (суммарный балл) каждого студента по информатике при решении 5 задач. Определить средний балл группы из N студентов.



Сложноциклические структуры

```
Var
    i, j, n: integer;
    b, bs, sbg, g: real;
Begin
    writeln ('Введите количество учеников : ');
    read (n);
    sbg:=0;
    for i:=1 to n do
        begin
            writeln (' Введите баллы ', i, '-го ученика');
            bs:=0;
            for j:=1 to 5 do
                begin
                    printf (' Количество баллов за решение ', j, '-й задачи: ');
                    read (b);
                    bs:=bs+b;
                    writeln
                        ('-----');
                    writeln (' Суммарный балл ', i, '-го ученика равен ', bs:6:2);
                    writeln
                        ('-----');
                    sbg:=sbg+bs;
                    end;
                g:=sbg/n;
                writeln ('*****');
                writeln (' Средний балл группы равен ', g:6:2, ' балла');
                writeln ('*****');
            End.
```

Решение задач

Задача 6.

Вычислить $S = \sum_{i=1}^n x_i$, где x_i – i -тый член суммы.

Var

s, i, n : integer;

Begin

read (n);

s:=0;

for i:=1 to n do

begin

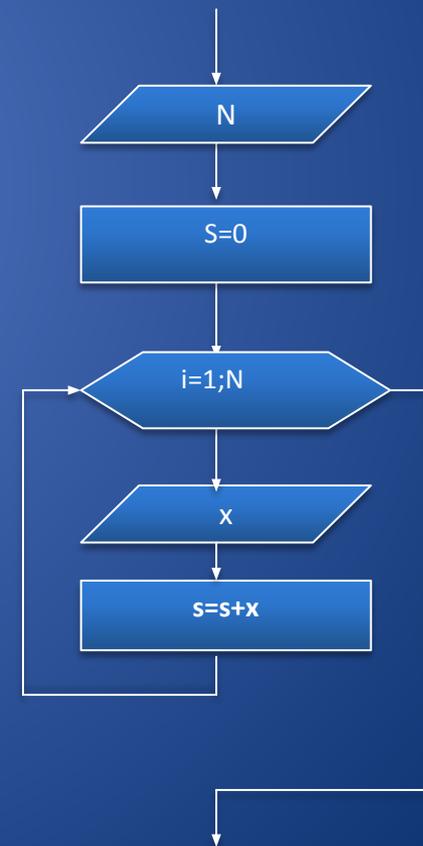
read (x);

s:=s+x;

end;

write (s);

End.



Решение задач

Задача 7.

Вычислить знакопеременную сумму $s = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{18} - \frac{1}{19} + \frac{1}{20}$

Var

i, p : integer;

s: real;

Begin

s:=0;

p:=-1;

for i:=1 to 20 do

begin

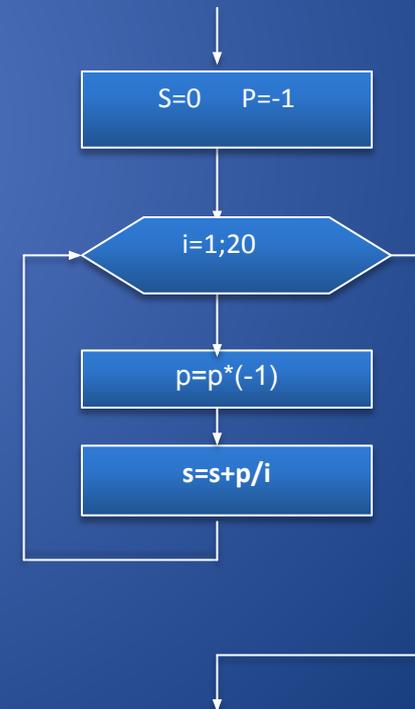
p:=p*(-1);

s:=s+p/i;

end;

write (s);

End.



Решение задач

Задача 8.

Вычислить $P = \prod_{i=1}^n x_i = x_1 * x_2 * \dots * x_n$

Var

p, i, n : integer;

Begin

read (n);

p:=1;

for i:=1 to n do

begin

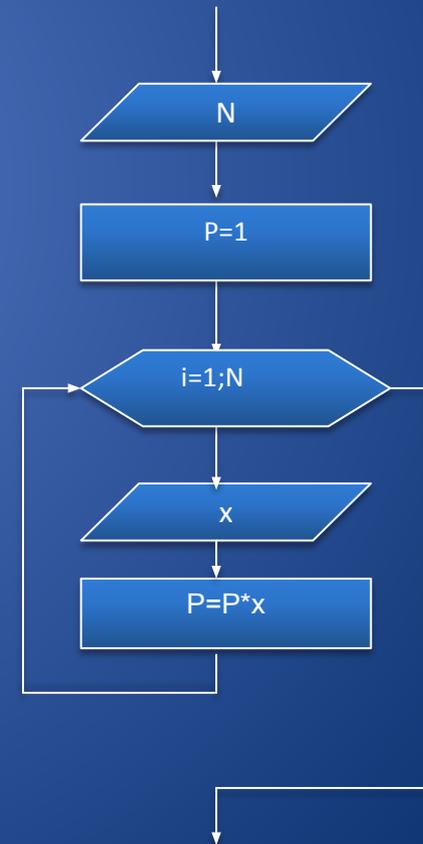
read (x);

p:=p*x;

end;

write (p);

End.



Решение задач

Задача 9.

Вычислить $P = \frac{1}{23} * \frac{2}{21} * \frac{3}{19} * \dots * \frac{10}{5} * \frac{11}{3} * 12$

Var

i, j : integer;

p: real;

Begin

p:=1;

j:=23;

for i:=1 to 12 do

begin

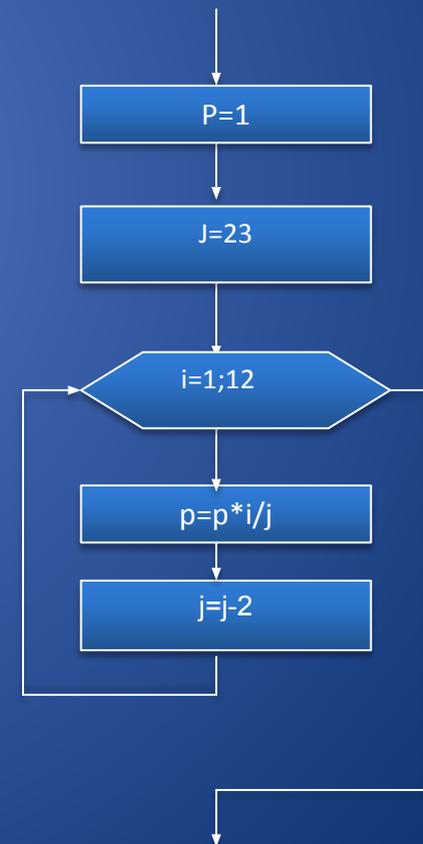
p:=p*i/j;

j:=j-2;

end;

write (p);

End.

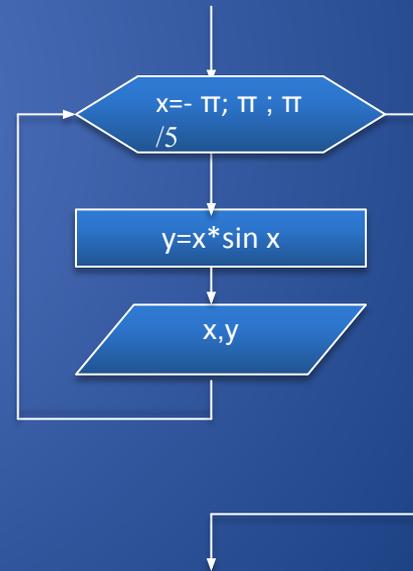


Решение задач

Задача 10.

Напишите программу табулирования функции для получения таблицы функции $y=x*\sin(x)$ при изменении x на отрезке от $-\pi$ до π с шагом $\pi/5$.

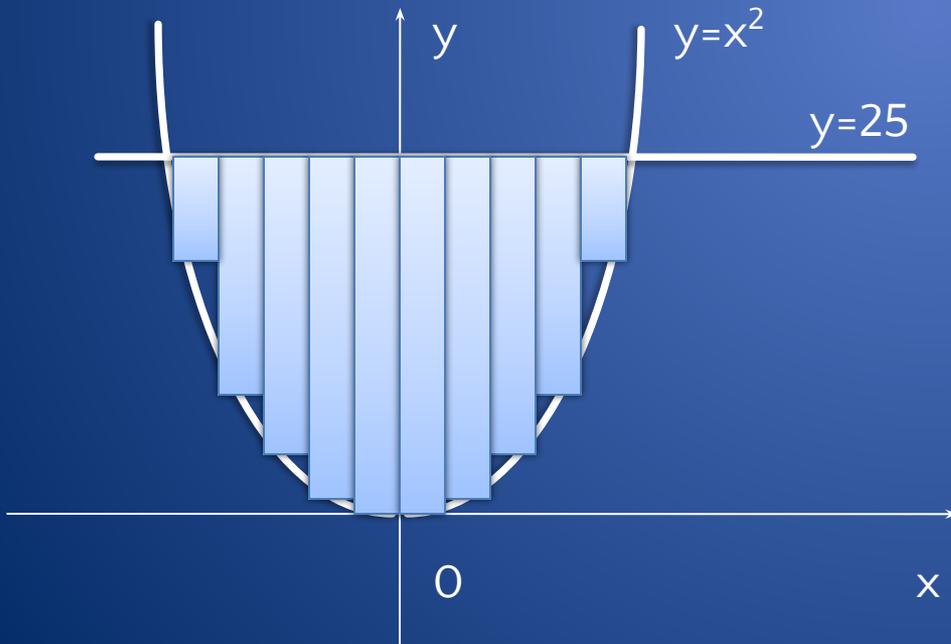
```
Var  
x,y: real;  
Begin  
x:=-3.14;  
repeat  
    y:=x*sin(x);  
    writeln (x:6:2, ' | ', y:6:2);  
    x:=x+pi/5;  
until x>3.14;  
End.
```



Решение задач

Задача 11.

Вычислить приближенно площадь фигуры, ограниченной функцией $y=x^2$ и прямой $y=25$, разбивая отрезок изменения x на 10 частей и суммируя площади прямоугольников с основаниями равными $1/10$ отрезка изменения x , и высотой, определяемой значением функции в середине основания.



$$\begin{cases} y=25 \\ y=x^2 \end{cases} \Rightarrow x=\pm 5$$

Т.к. высота определяется в середине основания прямоугольника, тогда x должен изменяться от -4.5 до 4.5 вследствие того, что $h=1$

$$h=|-5 - 5|/10=1$$

Высота прямоугольника $L=25-x^2$

Площадь прямоугольника

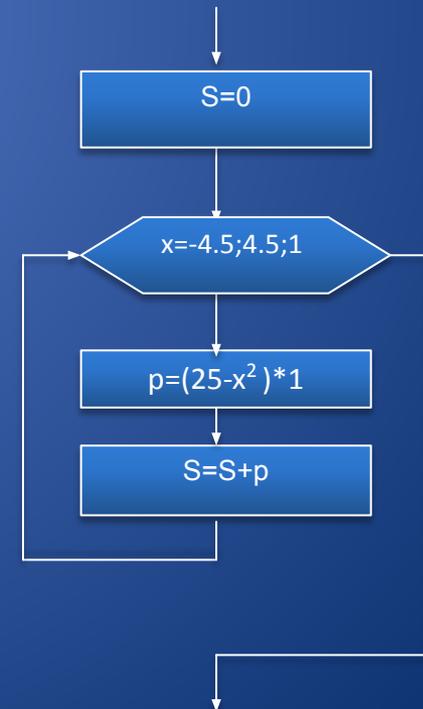
$$P=h*L=(25-x^2)*1$$

Решение задач

Задача 11.

Вычислить приближенно площадь фигуры, ограниченной функцией $y=x^2$ и прямой $y=25$, разбивая отрезок изменения x на 10 частей и суммируя площади прямоугольников с основаниями равными $1/10$ отрезка изменения x , и высотой, определяемой значением функции в середине основания.

```
Var  
    s, x, p : real;  
Begin  
s:=0;  
x:=-4.5;  
while x<=4.5 do  
    begin  
    p:=25-sqr(x);  
    s:=s+p;  
    x:=x+1;  
    end;  
writeln (s);  
End.
```

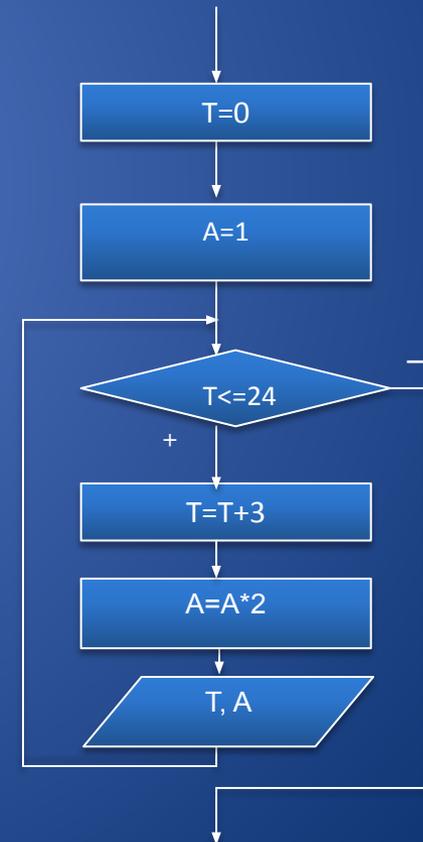


Решение задач

Задача 12.

Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить сколько клеток будет через 3,6,9,12, ... , 24 часа.

```
Var  
a,t : integer;  
Begin  
t:=0;  
a:=1;  
while t<=24 do  
begin  
t:=t+3;  
a:=a*2;  
writeln ('через ',t,'час. будет ',a,'амеб') ;  
end;  
End.
```



Решение задач

Задача 13.

Начав тренировки, спортсмен в первый день пробежал 5 км. Каждый следующий день он увеличивал дневную норму на 10% от нормы предыдущего дня. Через сколько дней он будет пробегать в день более 20 км. ?

Var

d : word;

s : real;

Begin

d:=1;

s:=5;

repeat

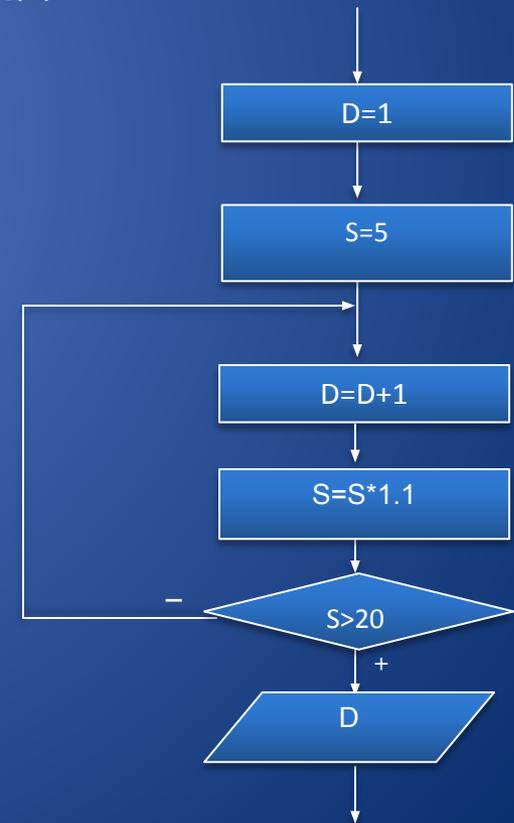
inc (d);

s:=s*1.1;

until s>20;

writeln (d) ;

End.



Решение задач

Задача 14.

Определить m – количество трехзначных натуральных чисел, сумма цифр которых равна n ($1 < n < 27$). Операции деления ($/$, div , mod) не использовать.

Var

m, n, i, j, k : byte;

Begin

 readln (n);

$m := 0$;

 for $i := 1$ to 9 do

 for $j := 0$ to 9 do

 for $k := 0$ to 9 do

 if $(i+j+k) = n$ then

 inc(m);

 writeln (m);

End.

Решение задач

Задача 15.

Вычислить $\prod_{i=1}^n \sum_{j=i}^{2n} \frac{i}{(2j^2 + 1)}$

Var

i, j, n : integer;

p, s : real;

Begin

read (n);

p:=1;

for i:=1 to n do

begin

s:=0;

for j:= 1 to 2*n do

s:=s+i/(2*i*i+1);

p:=p*s;

end;

writeln (p) ;

End.

