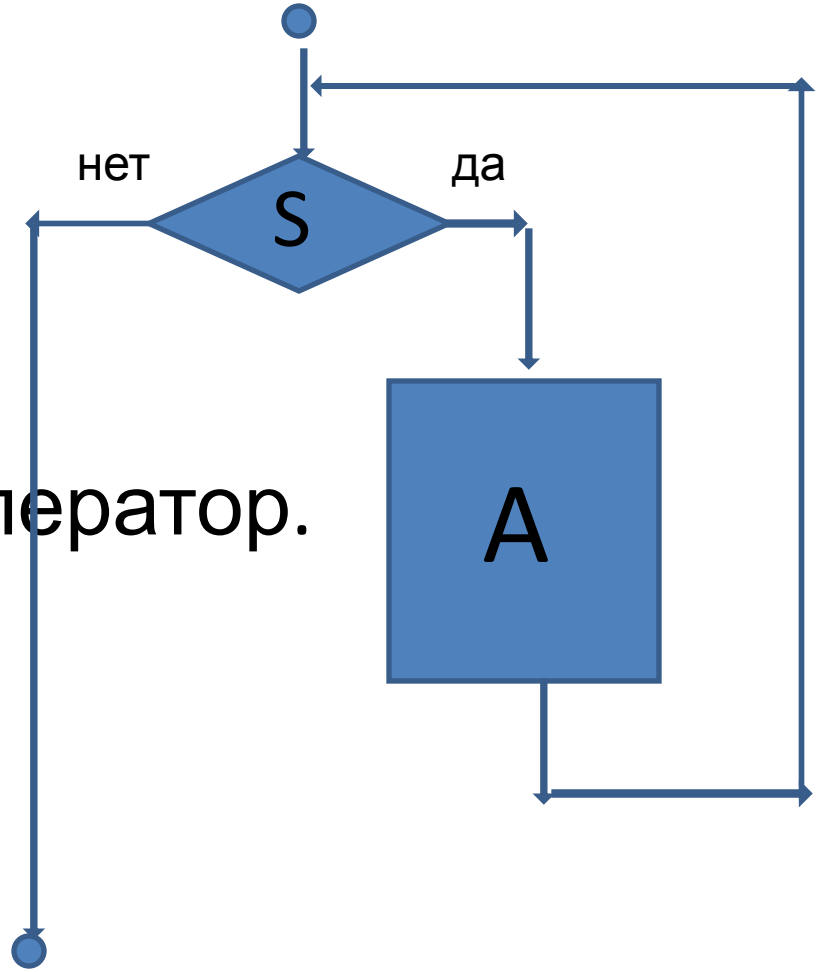
Оператор цикла с предусловием

While S Do A ;



S - логическое выражение;

A - тело цикла, один оператор.

В этом операторе тело цикла будет выполняться до тех пор, пока значение выражения **S** истинно.

Если при входе в цикл значение **S** есть **False**, тело цикла не выполнится ни разу.

Пример.

Задан бесконечный ряд:

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} - \ldots$$

Подсчитать сумму ряда с заданной точностью.

- Для решения этой задачи надо суммировать члены ряда до тех пор, пока абсолютная величина прибавляемого члена не станет меньше значения требуемой точности.
- Полученная при этом сумма есть сумма ряда с заданной точностью.
- Алгоритм решения этой задачи состоит из цикла, для которого заранее не известно число повторений.

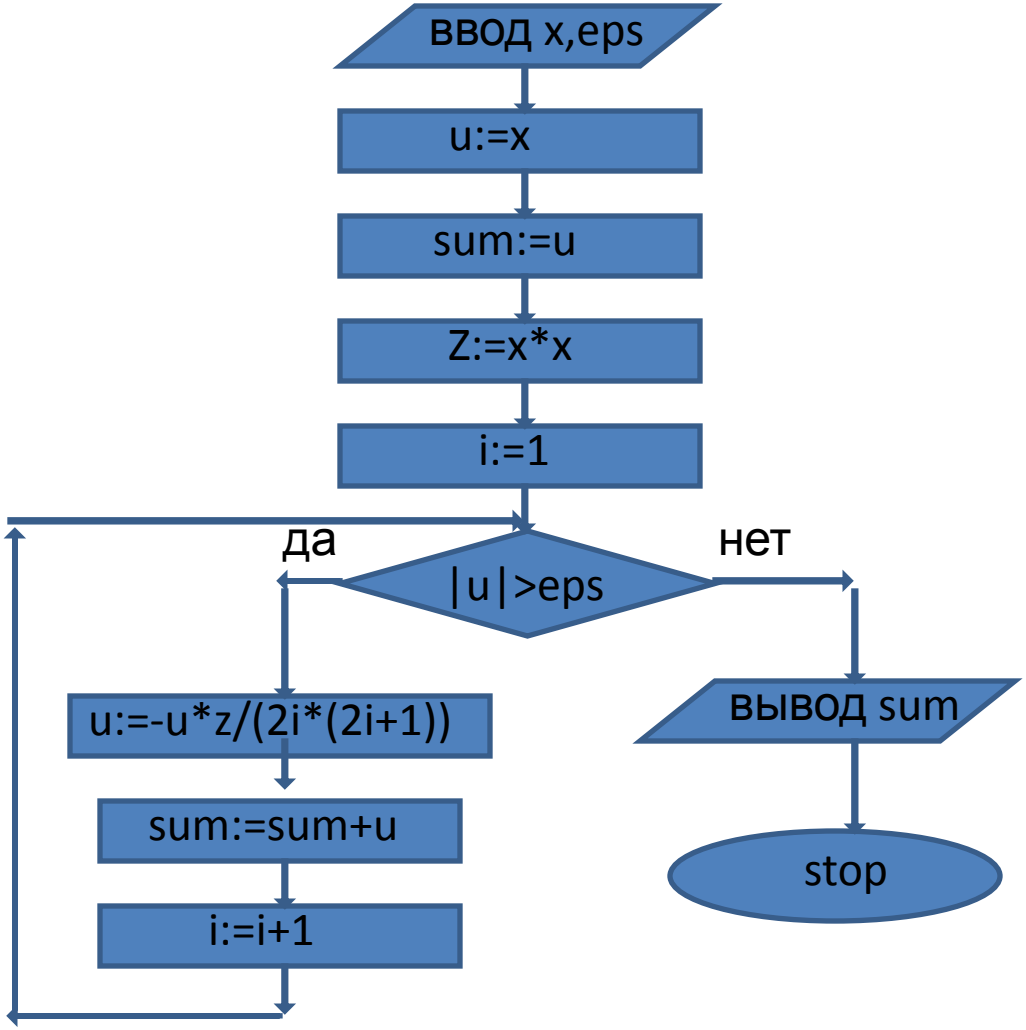
Легко заметить, что, имея значение **(i - 1)** - го члена ряда, можно получить **i** - ый член, используя рекуррентную формулу:

$$u_i = -u_{i-1} * \frac{x^2}{2i * (2i+1)}$$

В этой формуле i – номер члена, u_i – i-ый член ряда.

Эта рекуррентная формула получена делением в общем виде следующего члена ряда на предыдущий.

- Индексы в этой формуле указывают на то, что в правой части стоит предыдущий член, а в левой получаемый из него следующий.
- Т.к. после получения нового значения члена ряда - старое значение больше не нужно, новое значение можно записать на место старого, т.е. использовать для них одну и ту же переменную.
- В программе это выразится в том, что в рекуррентной формуле в левой и правой частях будет присутствовать одна и та же переменная, содержащая значение члена ряда.
- Естественно перед началом вычислений в цикле эта переменная должна получить значение первого члена ряда.


ввод x,eps
u:=x
sum:=u
Z:=x*x
i:=1
да
|u|>eps
нет
u:=-u*z/(2i*(2i+1))
sum:=sum+u
i:=i+1
вывод sum
stop

```
Var x,y,z,sum,u ,eps: real;
     i : integer;
 Begin
   Write('x=');
   Readln(x);            {Ввод значения x};
   Write('eps=');
   Readln(eps);          {Ввод значения eps};
    u := x;
    i := 1;
    sum := u;
    z := x*x;
           {цикл с предусловием для нахождения суммы ряда}
    While  abs( u ) > eps  Do
        Begin
           u := -u * z / ( 2* i * (2* i + 1 ));
           sum := sum + u;
           i := i + 1;
         End;
    Writeln('sum = ', sum)
 end.
```

Цикл, в котором заранее не известно число повторений, называется **итерационным циклом.**

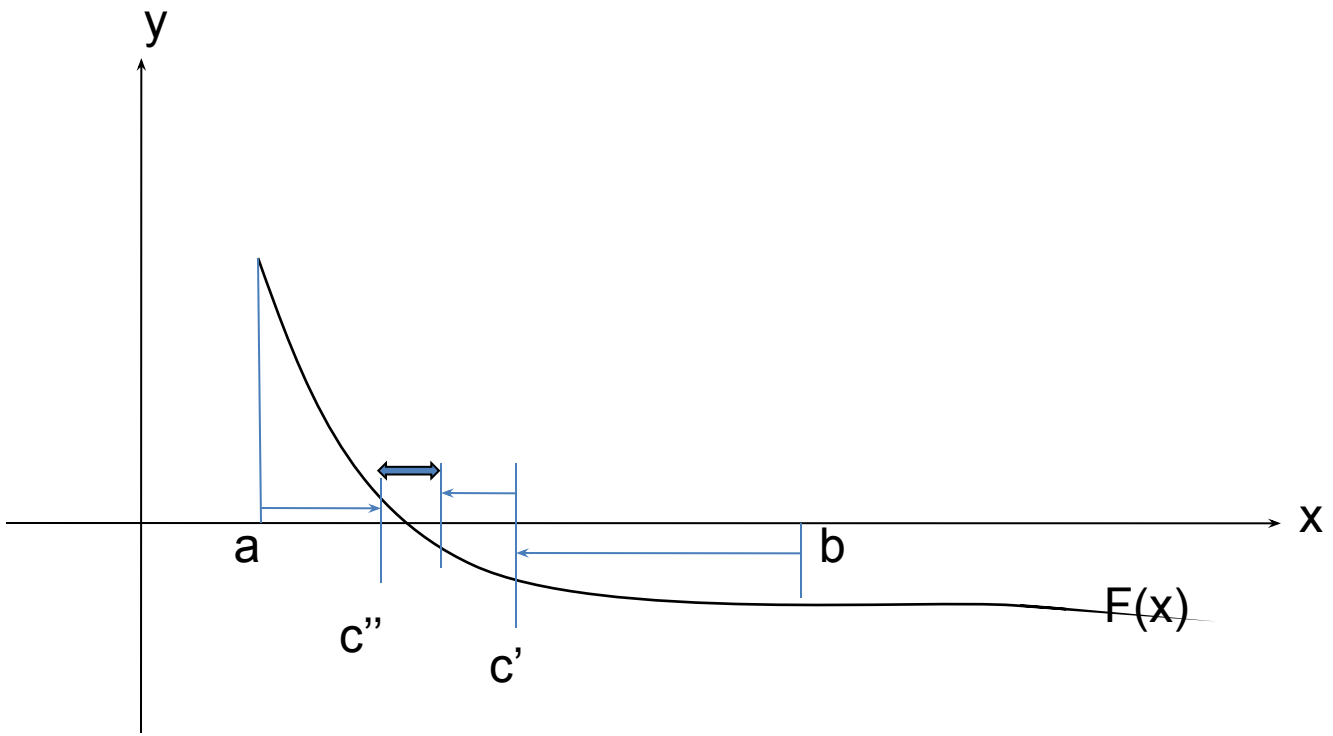
Пример.

Приближённое решение нелинейного уравнения методом бисекции.

Задано уравнение **f(x) = 0** .

Известно, что на отрезке **[a,b]** оно имеет один корень.

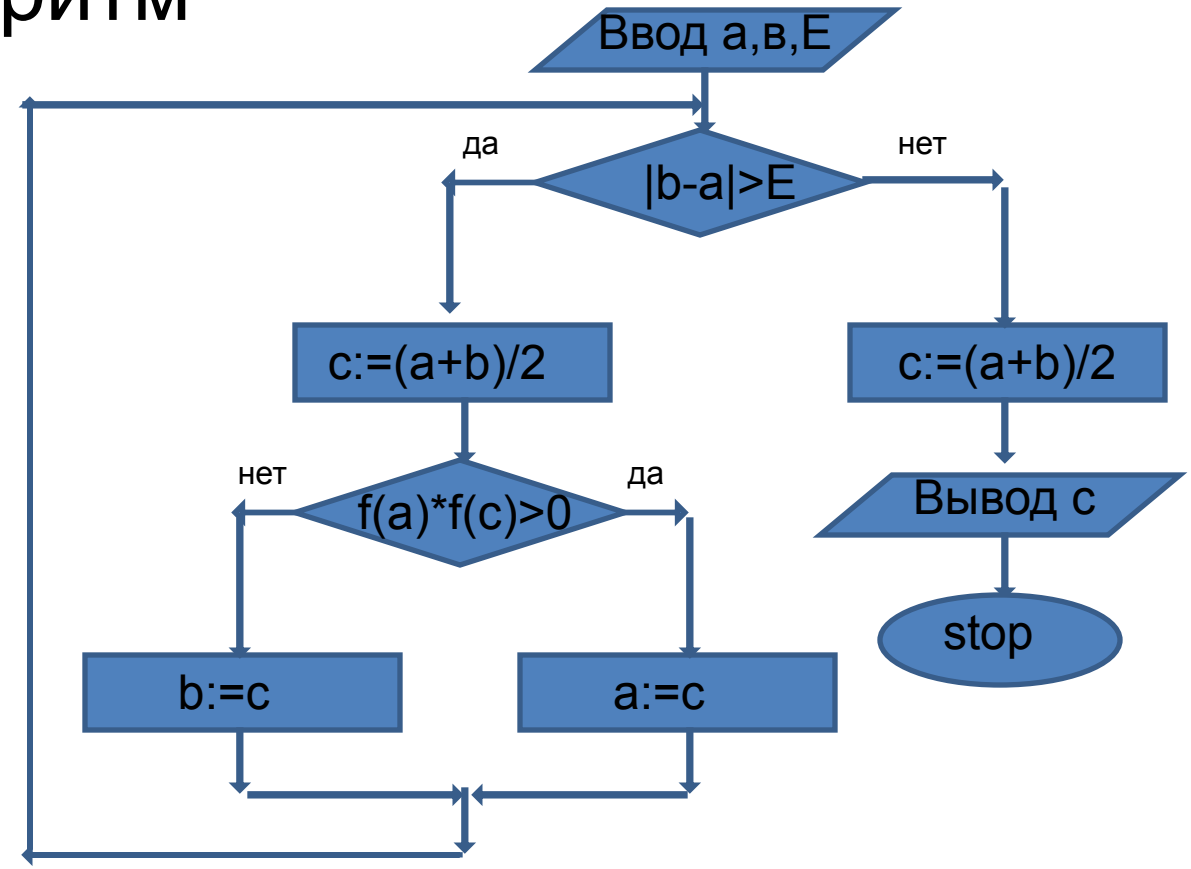
Требуется найти корень с точностью **E**.


y
x
a
b
c”
c’
F(x)

Решение уравнения методом бисекции заключается в следующем.

1. Исходный отрезок делится пополам.
2. Выбирается та половина, на которой есть корень (отрезок уменьшился вдвое).
3. Процесс продолжается до тех пор, пока длина отрезка не станет меньше значения требуемой точности **E**.

Алгоритм



```
Program uravn;
VAR a,b,c,E:REAL;
BEGIN
  WRITE('  a='); {ввод границы отрезка}
  readln(a);
  WRITE('  b=');
  readln(b);
  WRITE('  E='); {ввод значения точности}
  readln(E);
  while abs(b-a) >E do  {цикл поиска корня}
    BEGIN
      c:=(a + b)/2;
      if (sqr(a)-2) * (sqr(c)-2) >0 then a:=c
                                    else b:=c;
    END;
  c:=(a+b)/2;       {результат середина отрезка}
  writeln('  c=',c); {вывод на экран результата}
  readln
end.
```