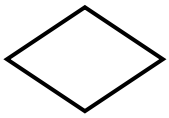
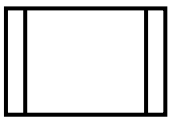
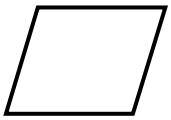
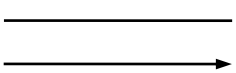
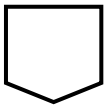
# Использование блок-схем алгоритмов при разработке программ

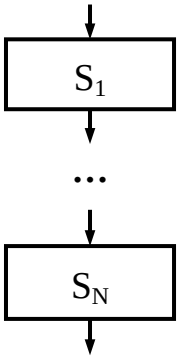
Условные обозначения (ГОСТ 19.003.80)

| Наименование | Обозначение | Функция |
|---|---|---|
| Блок выполнения действия (процесс) |  | Выполнение операции или группы операций, изменяющих значение, форму представления или расположение данных |
| Логический блок (решение) |  | Выбор направления выполнения алгоритма в зависимости от некоторых условий |
| Модификация (заголовок цикла) |  | Выполнение действий, изменяющих команды или группы команд |
| Предопределенный процесс |  | Обращение к вспомогательному алгоритму |
| Блок ввода-Вывода |  | Ввод или вывод данных |
| Пуск- останов |  | Начало или конец алгоритма |
| Линия потока |  | Указание на последовательность связей между блоками |
| Комментарий | ----------[ | Пояснение элемента схемы |
| Соединитель |  | Указатель связи между прерванными линиями потока в пределах одной страницы |
| Межстраничный соединитель |  | Указание связи между частями схемы, расположенными на разных страницах |

**Структура «следование»**

Алгоритмы, в которых все действия записываются последовательно и в том же порядке исполняются, называются линейными. Для их описания используется структура следование.

Она представляет собой последовательность команд, следующих одна за другой. На схеме структура изображается так:




При исполнении алгоритма эти команды выполняются в том порядке, в каком они записаны. На процедурных языках программирования (Бейсик, Паскаль, Си) данная структура реализуется в виде последовательности операторов, следующих один за другим:

<оператор1>
...
<операторN>.

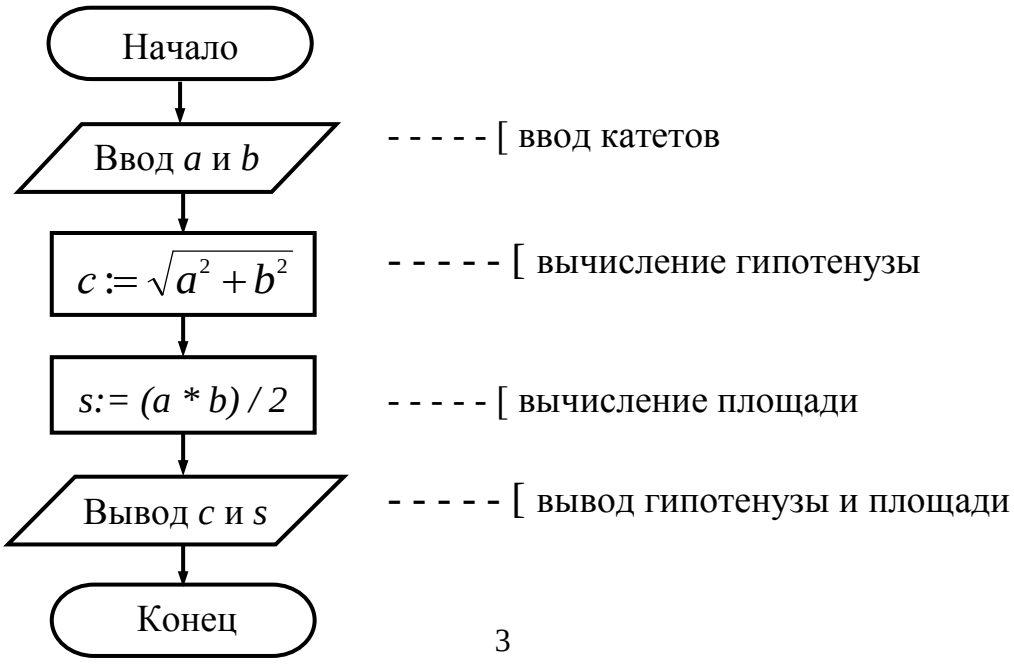
При решении следующей задачи используется структура следования.

Задача.

Известны катеты прямоугольного треугольника. Определить его гипотенузу и площадь.

Решение.

***Блок-схема***



### *Алгоритмический язык*

```
алг  гипотенуза и площадь
нач вещ a,b,c,s
¦ вывод "Введите катеты a, b:"
¦ ввод a,b
¦ c := sqrt ( a**2 + b**2 )
¦ s := 1 / 2 * a * b
¦ вывод "Гипотенуза=", c
¦ вывод "Площадь треугольника=", s
кон
```

### *Бейсик*

```
' Вычисление гипотенузы и площади треугольника
INPUT "Введите катеты a,b"; a, b
c = SQR (a^2 + b ^2)
s = 1 / 2 * a * b
PRINT "Гипотенуза =";c
PRINT "Площадь треугольника =";s
END
```

### *Паскаль*

```
{Вычисление гипотенузы и площади треугольника}
program treugolnik;
var a,b,c,s:real;
begin
   write ( 'Введите катеты a, b: ');
   readln (a, b );
   c := sqrt ( a*a + b*b );
   s := 1 / 2 * a * b;
   writeln ('Гипотенуза=', c);
   writeln ('Площадь треугольника=', s);
end.
```

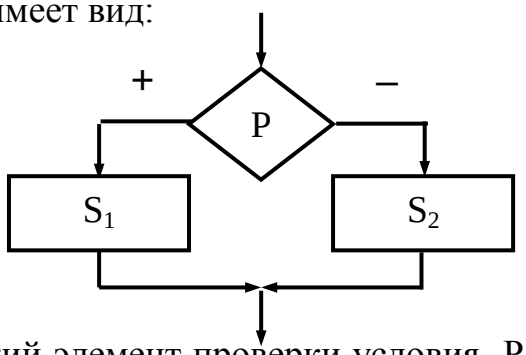
### *Си*

```
#include<stdio.h>
void main()
{
 float a,b,c,s;
 printf("\nВведите катеты a, b: ");
 scanf("%e",&a);
 scanf("%e",&b);
```

```
c = sqrt ( a*a + b*b );
s = 1 / 2 * a * b;
printf("Гипотенуза= %e",c);
printf("Площадь треугольника= %e", s);
 }
```

### 1.3. Структура «развилка»

Алгоритмы, при исполнении которых порядок следования команд определяется в зависимости от результатов проверки некоторых условий, называются ветвящимися. Для их описания используются структуры «развилка» и «выбор». Структура «развилка» имеет вид:




Она содержит логический элемент проверки условия P и два функциональных блока $S_1$ и $S_2$. При исполнении алгоритма сначала вычисляется значение логического выражения (проверяется условие), если оно истинно, то выполняется блок $S_1$, иначе (если ложно) блок $S_2$. Такой вид развилки называется полной условной конструкцией.

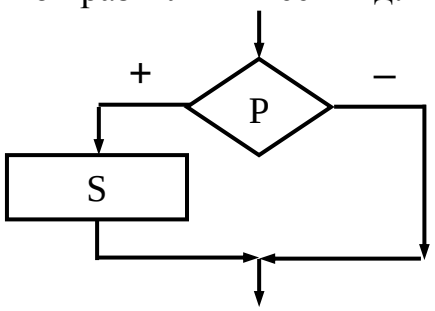
На языках программирования данная структура реализуется так.

| Алгоритмический язык | Бейсик |
|---|---|
| если <условие><br>то <серия1><br>иначе <серия2><br>все | а) линейная форма<br>IF <выражение> THEN <оператор1><br>ELSE <оператор2><br>б) блочная форма<br>IF <выражение> THEN<br><оператор1><br>ELSE<br><оператор2><br>END IF |
| **Паскаль** | **Си** |
| if <выражение><br>then <оператор1><br>else <оператор2>; | if (<выражение>)<br><оператор1>;<br>else <оператор2>; |

Здесь <выражение> – логическое выражение (условие), <оператор> – это либо один оператор, либо группа операторов. В Паскале группа операторов заключается в операторные скобки begin – end, в Си – в фигурные скобки {}.

Структура развилка используется также в неполной форме. В этом случае, если значение логического выражения ложно, никакое действие не выполняется. Такой вид развилки называется неполной условной конструкцией.

Структура неполной развилки имеет вид:



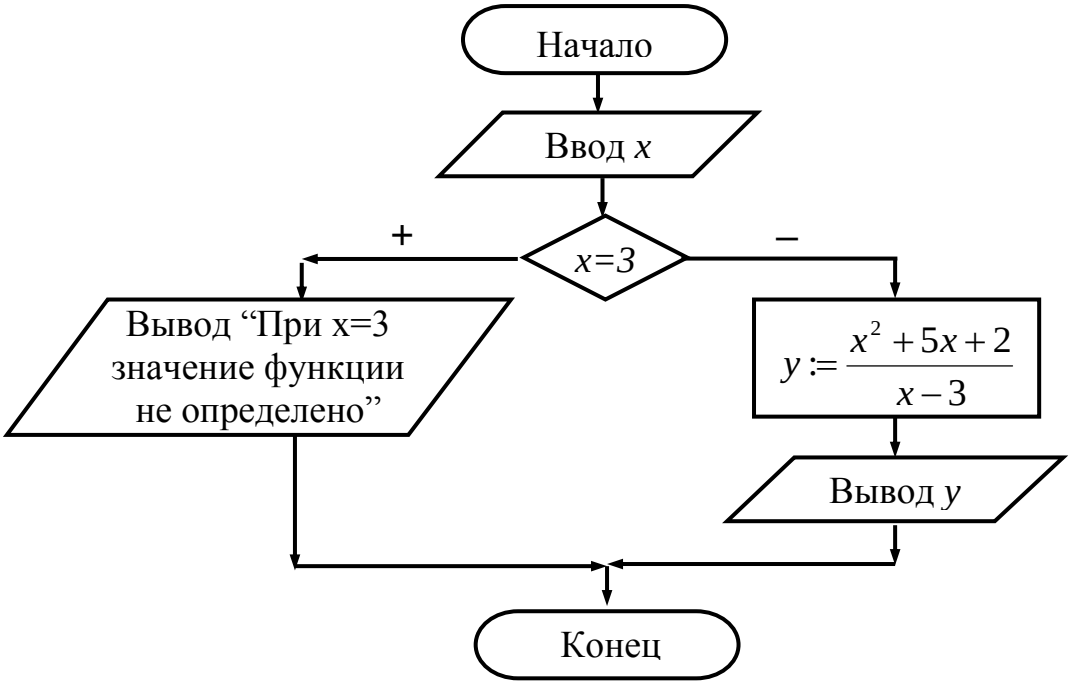

Она реализуется следующим образом:

| Алгоритмический язык | Бейсик |
|---|---|
| если <условие><br>то <серия><br>все | а) линейная форма<br>IF <выражение> THEN <оператор><br>б) блочная форма<br>IF <выражение> THEN<br><оператор><br>END IF |
| **Паскаль** | **Си** |
| if <выражение><br>then <оператор>; | if (<выражение>)<br><оператор>; |

Задача.

Вычислить значение функции $y = \dfrac{x^2 + 5x + 2}{x - 3}$ для заданного $x$.

Решение. ***Блок-схема***



***Алгоритмический язык***

```
алг  выражение
нач вещ x,y
¦ вывод "Введите x:"
¦ ввод x
¦ если x–3=0
¦ ¦то вывод "При x=3 значение функции не определено"
¦ ¦иначе y:=(x**2+5*x+2)/(x–3)
¦ ¦ вывод "y=",y
¦ все
кон
```

***Бейсик***

```
'Выражение
INPUT "Введите x"; x
IF x – 3 = 0 THEN
PRINT " При x=3 значение функции не определено "
ELSE
y = (x^2+5*x+2)/(x–3)
PRINT "y="; y
END IF
END
```

***Паскаль***

```
program vyrazh;
var x, y : real;
begin
   write ('Введите x:'); readln(x);
   if  x– 3 = 0
      then
         write ('При x=3 значение функции не определено')
      else
          begin
              y :=(x*x+5*x+2)/(x–3);
              write ('y=', y)
          end
end.
```

***Си***

```
#include<stdio.h>
void main()
{
 float x,y;
```

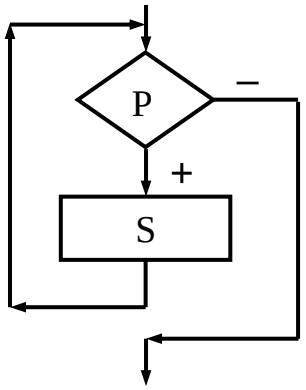
```
printf("\nВведите x: ");
scanf("%e", & x );
if(x–3==0)
   printf("При x=3 значение функции не определено ");
 else
   {y=(x*x+5*x+2)/(x–3);
     printf("y= %e", y); }
 }
```

### 1.4. Структура «цикл»

Алгоритмы, при исполнении которых отдельные команды или серии команд выполняются неоднократно, называются циклическими. Для их описания используется структура «цикл». Структура «цикл» может быть трех видов: «цикл с предусловием» («пока»), «цикл с постусловием» («до»), «цикл с параметром» («для»).

#### 1.4.1. Структура «цикл с предусловием»

Структура цикла с предусловием состоит из логического элемента проверки условия P и функционального блока S, называемого телом цикла. Она имеет вид:




Цикл с предусловием выполняется так: сначала проверяется условие (отсюда название - цикл с предусловием), т.е. вычисляется значение логического выражения. Если оно истинно, то выполняется тело цикла, и снова проверяется условие. Выполнение цикла завершается, когда значение логического выражения становится ложным. Для этого необходимо, чтобы в теле цикла существовала команда, которая влияла бы на условие.

На языках программирования структура реализуется так:

| Алгоритмический язык | Бейсик[1] |
|---|---|
| нц пока <условие><br><серия><br>кц | DO WHILE <выражение><br><оператор><br>LOOP |
| **Паскаль** | **Си** |
| while <выражение><br>do <оператор>; | while (<выражение>)<br><оператор>; |

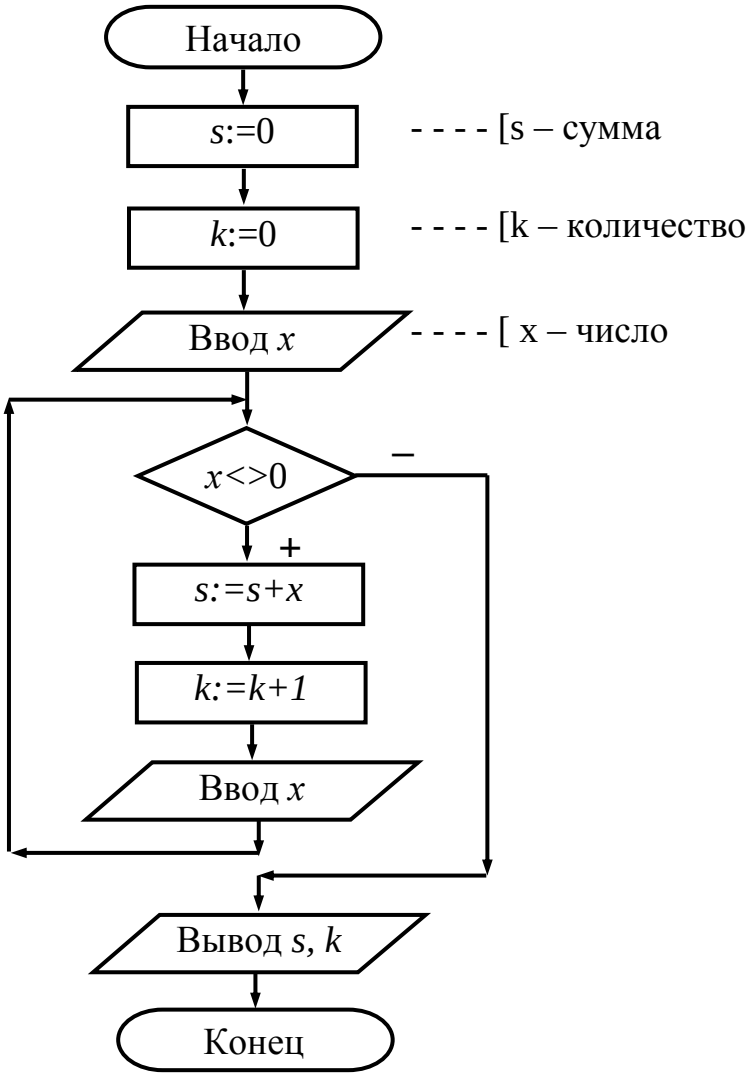
При решении следующей задачи используется структура цикла с предусловием.

Задача.

Вводить числа, пока не встретится 0. Определить сумму и количество введенных чисел.

Решение.

***Блок-схема***




[1] В Бейсике имеется несколько операторов цикла с предусловием. Оператор
WHILE <выражение>
<оператор>
WEND также реализует данную структуру,
оператор DO UNTIL <выражение>
<оператор>
LOOP нет.

***Алгоритмический язык***

```
алг  сумма
нач вещ s,x, цел k
¦ вывод "Введите число:"
¦ ввод x
¦ s:=0
¦ k:=0
¦ нц пока x<>0
¦ ¦ s:=s+x
¦ ¦ k:=k+1
¦ ¦ вывод "Введите число:"
¦ ¦ ввод x
¦ кц
¦ вывод "Сумма чисел=",s,"их количество=",k
кон
```

***Бейсик***

```
'Сумма
s = 0: k=0
INPUT "Введите число:"; x
DO WHILE x <> 0
s = s+ x
k=k+1
INPUT "Введите число:"; x
LOOP
PRINT "Сумма чисел="; s; "их количество=",k
END
```

***Паскаль***

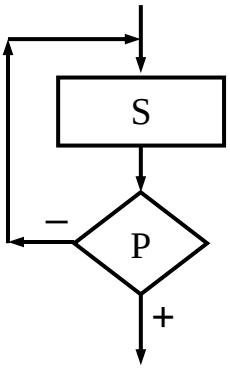
```
program summa;
var s,x:real; k:integer;
begin
   s:=0;k:=0;
   write('Введите число:');
   readln(x);
   while x<>0 do
     begin
       s:=s+x;
        k:=k+1;
       write('Введите число:');
       readln(x);
     end;
   writeln('Сумма чисел=',s,' их количество=', k);
end.
```

***Си***

```
#include<stdio.h>
void main()
{
 float x,s; int k;
 s = 0;
 k=0;
 printf("\nВведите число: ");
 scanf("%e", & x );
 while(x==0)
     {
      s = s+ x
      k=k+1
      printf("\nВведите число: ");
      scanf("%e", & x );
       }
 printf("Сумма чисел= %e", s);
 printf("их количество= %e",k);
}
```

### 1.4.2. Структура «цикл с постусловием» (до)

Структура цикла с постусловием также состоит из логического элемента проверки условия P и функционального блока S – тела цикла.




Цикл с постусловием выполняется так: сначала выполняется команда (команды) в теле цикла, затем проверяется условие, т.е. вычисляется значение логического выражения. Если оно ложно, то снова выполняются команды в теле цикла, и так до тех пор, пока значение логического выражения не примет значение истина, после чего выполнение цикла завершается. Необходимо, чтобы в теле цикла существовала команда, влияющая на условие.

Различие между циклами не только в том, что один с постусловием, а другой с предусловием, но и в том, что в цикле с предусловием функциональный блок S может ни разу не выполниться, если условие P при первой проверке окажется ложным. В цикле с постусловием функциональный блок всегда хотя бы один раз выполнится

На языках программирования структура реализуется так.

| Бейсик | Паскаль |
|---|---|
| DO<br><оператор><br>LOOP UNTIL <выражение> | repeat < оператор ><br>until < выражение > ; |

Структура цикла с постусловием является дополнительной. Поэтому на некоторых языках программирования для ее реализации нет соответствующего оператора. В частности, нет команды цикла с постусловием в школьном алгоритмическом языке, хотя в других версиях алгоритмического языка данная команда есть.

В языке Си также нет оператора, реализующего данную структуру. Для реализации ее можно использовать оператор:

```
do S;
while(!P);
```

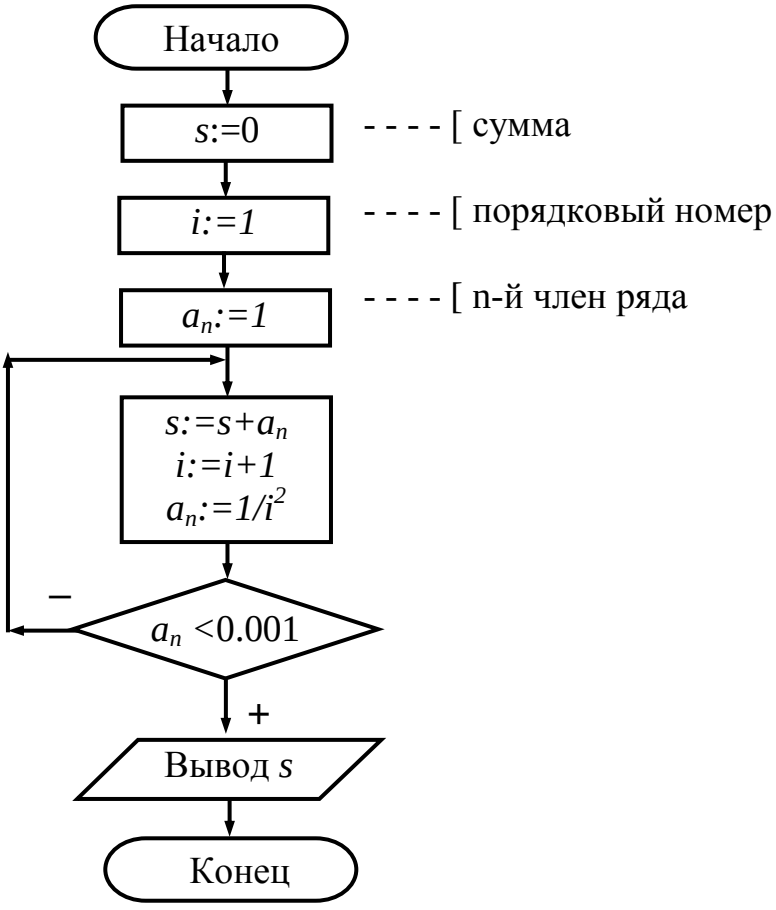
где !P означает отрицание условия P.

Рассмотрим задачу, для решения которой можно использовать цикл с постусловием.

Задача. Вычислить сумму ряда $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+...$ с точностью 0.001.

Решение.

***Блок-схема***



***Бейсик***

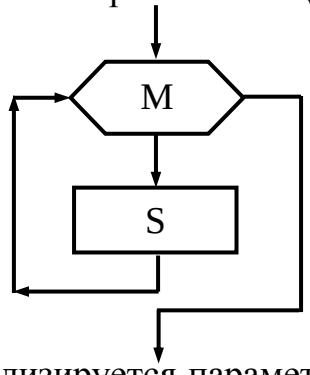
```
'Вычисление суммы ряда
s = 0
i = 1
an = 1
DO
s = s + an
i = i + 1
an = 1 / i^2
LOOP UNTIL an < .001
PRINT "Сумма ряда="; s
END
```

***Паскаль***

```
program summa;
{Вычисление суммы ряда}
var i:integer;
    an,s:real;
begin
    s:=0;
    i:=1;
    an:=1;
    repeat
        s:=s+an;
        i:=i+1;
        an:=1/(i*i)
    until an<0.001;
    writeln('Сумма ряда=',s:6:3);
end.
```

### 1.4.3. Структура «цикл с параметром»

Она состоит из блока модификации M (заголовок цикла) и функционального блока S (тела цикла). Данную структуру рекомендуется использовать, когда заранее известно число повторений тела цикла.




В заголовке цикла инициализируется параметр цикла, то есть ему присваивается начальное значение, указывается конечное значение параметра цикла, до до-

стижения которого тело цикла будет повторяться, и шаг, который показывает, на сколько изменится параметр цикла после каждого выполнения тела цикла.

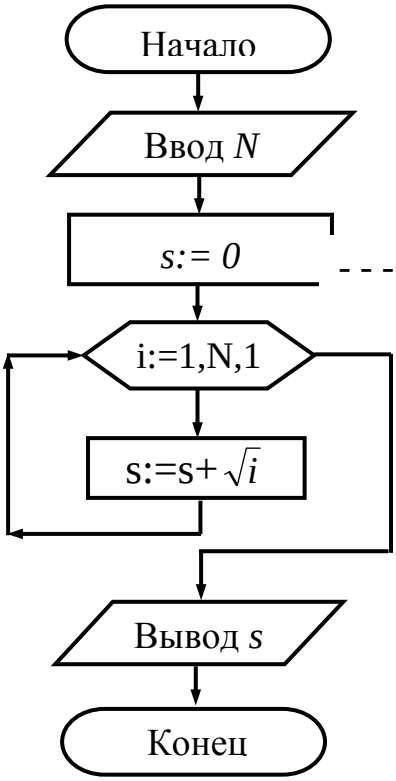
| | |
|---|---|
| Алгоритмический Язык | нц для <параметр цикла> от <начальное значение параметра цикла> до <конечное значение параметра цикла> шаг [<шаг>]<br><серия><br>кц |
| Бейсик | FOR <параметр цикла> = <начальное значение параметра цикла> TO <конечное значение параметра цикла> [STEP <шаг>]<br><оператор><br>NEXT [<параметр цикла>] |
| Паскаль | for <параметр цикла> := <начальное значение параметра цикла><br>to <конечное значение параметра цикла><br>do <оператор>;<br>или<br>for <параметр цикла> := <начальное значение параметра цикла><br>downto <конечное значение параметра цикла><br>do <оператор>; |
| Си | for (<параметр цикла> = <начальное значение параметра цикла>; <условие выполнения цикла>; [<параметр цикла> = <параметр цикла>+<шаг>] )<br><оператор>; |

Если шаг равен 1, то на алгоритмическом языке, в Бейсике и Си его можно не указывать.

Задача. Вычислить сумму ряда $1+\sqrt{2}+\sqrt{3}+...+\sqrt{N}$.

Решение.

***Блок-схема***




***Алгоритмический язык***

```
алг Сумма
нач цел i,n;вещ s
¦ вывод "Введите n:"
¦ ввод n
¦ s:=0
¦ нц для i от 1 до n шаг 1
¦ ¦ s:=s+sqrt(i)
¦ кц
¦ вывод "Сумма ряда=",s
кон
```

***Бейсик***

```
'Сумма ряда
INPUT "Введите N:"; N
s = 0
FOR i = 1 TO N
s = s + SQR(i)
NEXT i
PRINT "Сумма ряда="; s
END
```

***Паскаль***

```
var i,n:integer;
    s:real;
begin
    write('Введите n:');
    readln(n);
    s:=0;
    for i:=1 to n
     do s:=s+sqrt(i);
    writeln('Сумма ряда=',s:6:3);
end.
```

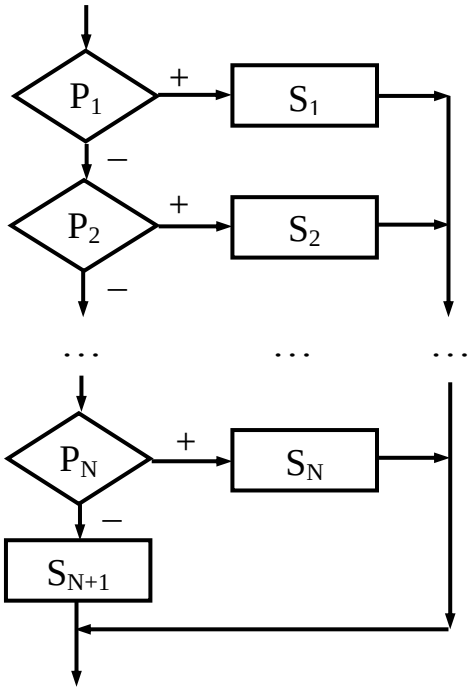
***Си***

```
#include<math.h>
#include<stdio.h>
void main()
{
 int n,i;
 float s;
 printf("\nВведите n: ");
 scanf("%e",&n);
 s=0;
 for (i=1;i<=n; i=i+1)
 s=s+sqrt(i);
```

```
printf("Элемент равен: ");
printf("%e",s);
}
```

### 1.5. Структура «выбор»

Структура выбора является развитием структуры «развилка». В отличие от структуры «развилка» здесь имеется возможность выбора более двух действий. Она имеет вид:




где $P_1$, …, $P_N$ – логические элементы проверки условия; $S_1$, ... $S_{N+1}$ – функциональные блоки.

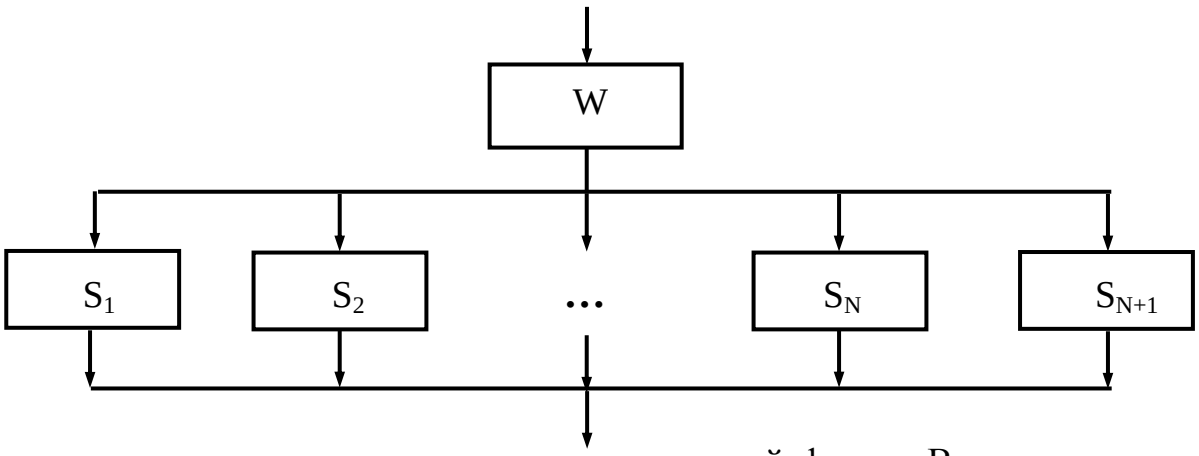
Выполняется следующим образом: проверяется условие, т.е. вычисляется значение выражения, и в зависимости от значения выполняются либо функциональный блок $S_1$, либо $S_2$, и т.д. $S_N$. Если ни одно из условий не выполняется, то выполняется функциональный блок $S_{N+1}$.

На рассматриваемых языках данная структура реализуется так:

| | |
|---|---|
| Алгоритмический язык | Выбор<br>при <условии1>: <серия1><br>при <условии2>:<серия2><br>…<br>при <условииN>:<серияN><br>иначе < серияN+1><br>все |

| | |
|---|---|
| Бейсик | SELECT CASE <выражение><br>CASE <список выражений1><br><оператор1><br>…<br>CASE <список выраженийN><br><операторN><br>CASE ELSE<br><операторN+1><br>END SELECT |
| Паскаль | case <выражение> of<br><список констант1> : <оператор1> ;<br>. . .<br><список константN> : <операторN> ;<br>else <операторN+1><br>end ; |
| Си | switch (<выражение>)<br>case <константа1> : <оператор1> ; break;<br>. . .<br>case <константаN> : <операторN> ; break;<br>default : <операторN+1> ; break |

Для Паскаля, Си, Бейсика больше подходит такая схема:




Данная структура используется также в неполной форме. В этом случае она реализуется так.

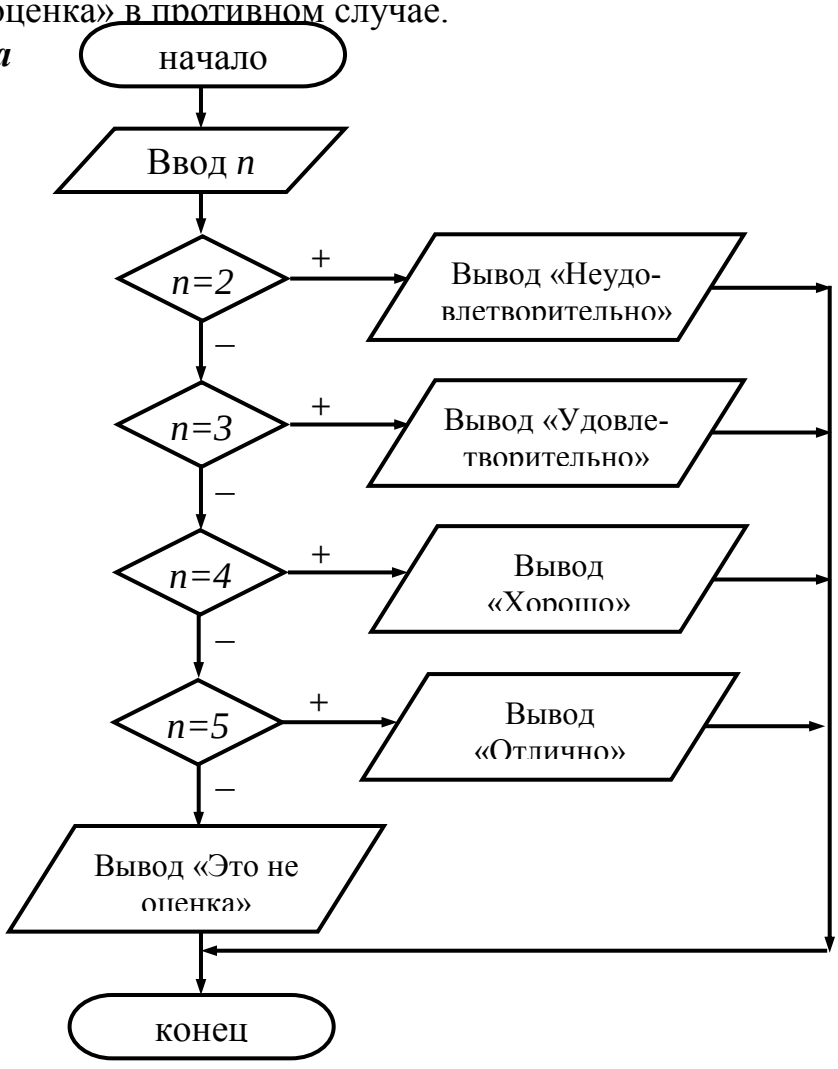
| | |
|---|---|
| Алгоритмический язык | выбор<br>при <условии1>: <серия1><br>при <условии2>:<серия2><br>…<br>при <условииN>:<серияN><br>все |

| | |
|---|---|
| Бейсик | SELECT CASE <выражение><br>CASE < список выражений1><br><оператор1><br>…<br>CASE < список выраженийN><br><операторN><br>END SELECT |
| Паскаль | case <выражение> of<br><список констант1> : <оператор1> ;<br>. . .<br><список константN> : <операторN> ;<br>end ; |
| Си | switch (<выражение>)<br>case <константа1> : <оператор1> ; break;<br>. . .<br>case <константаN> : <операторN> ; break; |

Задача.

Ввести число. Вывести «неудовлетворительно», если введено 2, «удовлетворительно», если введено число 3, «хорошо», если 4, «отлично», если 5, и вывести «это не оценка» в противном случае.

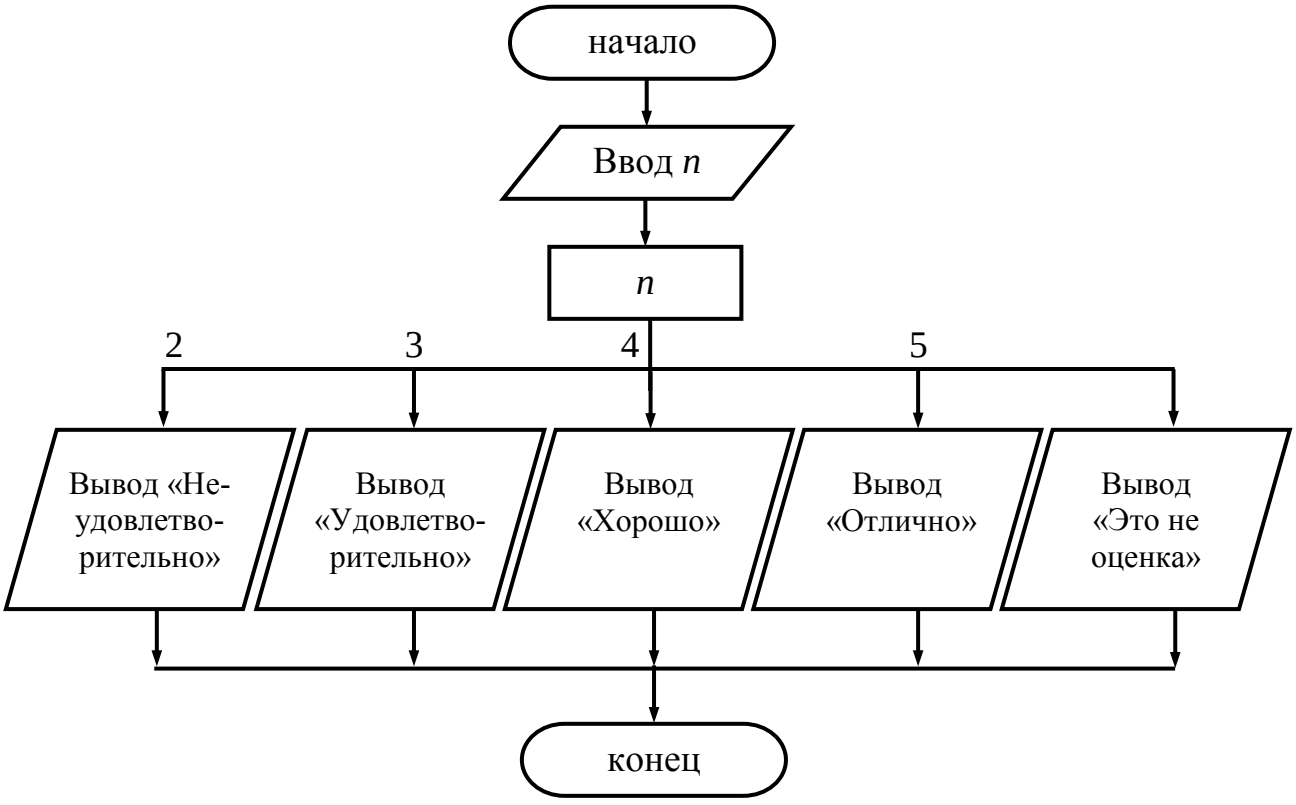
***Блок-схема***



***Алгоритмический язык***

```
алг оценка
нач цел n
| вывод "введите n"
| ввод n
| выбор
| |при n=2:вывод "Неудовлетворительно"
| |при n=3:вывод  "Удовлетворительно"
| |при n=4:вывод "Хорошо"
| |при n=5:вывод "Отлично"
| |иначе вывод "Это не оценка"
| все
кон
```

***Блок-схема***




***Бейсик***

```
'Оценка
INPUT "Введите n:"; n
SELECT CASE n
CASE 2
PRINT"Неудовлетворительно"
CASE 3
PRINT "Удовлетворительно"
```

```
CASE 4
PRINT "Хорошо"
CASE 5
PRINT "Отлично"
CASE ELSE
PRINT "Это не оценка"
END SELECT
END
```

***Паскаль***

```
program ocenka;
var n:integer;
begin
     write('Введите n: ');readln(n);
     case n of
          2:writeln('Неудовлетворительно');
          3:writeln('Удовлетворительно');
          4:writeln('Хорошо');
          5:writeln('Отлично');
    else writeln('Это не оценка')
    end;
end.
```

***Си***

```
#include<stdio.h>
void main()
{
int n;
printf("\nВведите n: ");
scanf("%d",&n);
switch (n)
 {
 case 2: printf("Неудовлетворительно");break;
 case 3: printf("Удовлетворительно");break;
 case 4: printf("Хорошо");break;
 case 5: printf("Отлично");break;
 default: printf("Это не оценка");break;
 }
}
```

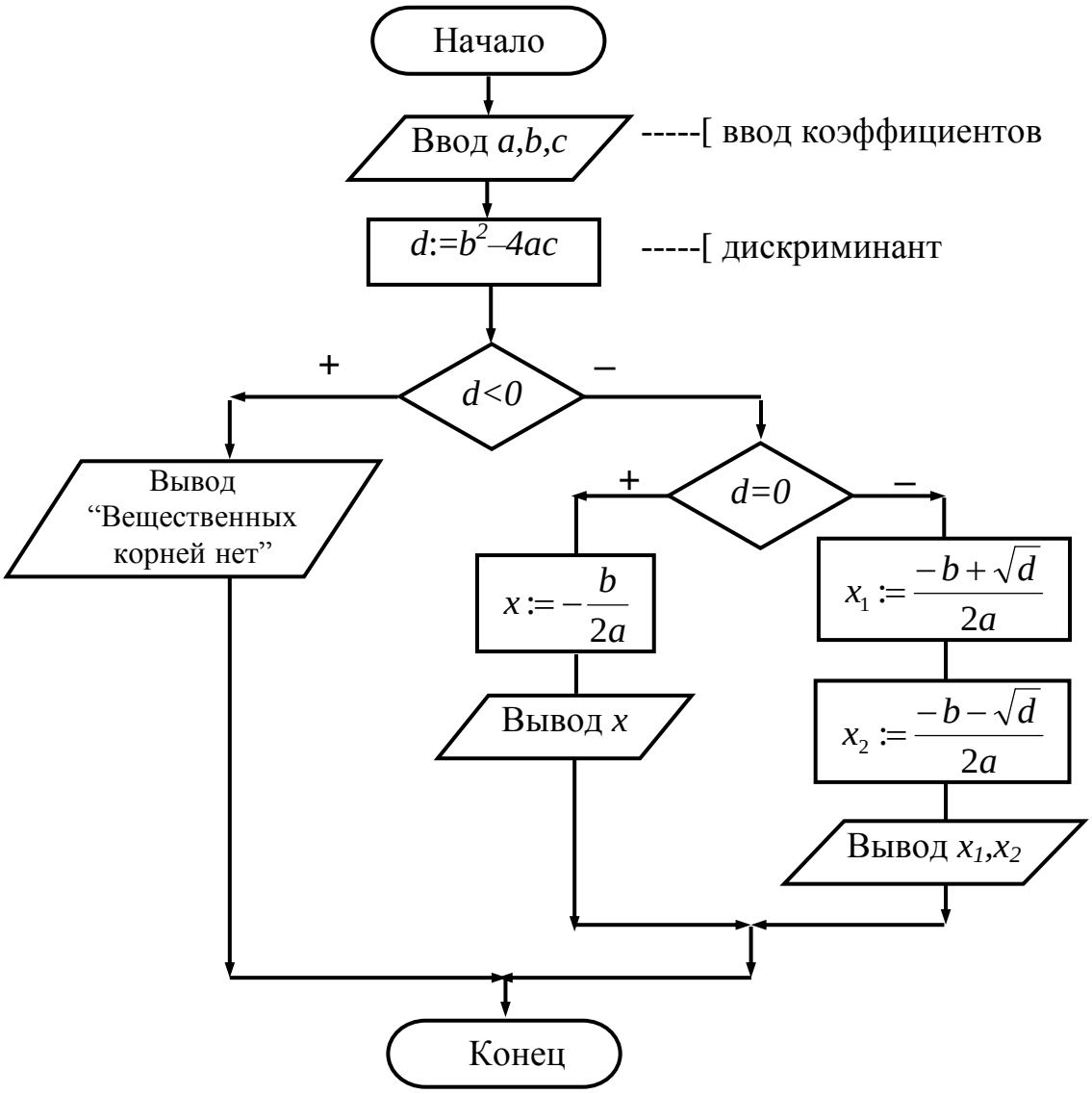
## 2. Примеры задач, в которых используются вложенные структуры

Следующая задача решается с помощью вложенной структуры «развилка в развилке».

Задача. Решить квадратное уравнение $ax^2+bx+c=0$ ($a\neq0$).

Решение

***Блок-схема***




***Алгоритмический язык***

```
алг  квадратное уравнение
нач вещ a,b,c,d,x,x1,x2
¦ вывод "Введите a,b,c"
¦ ввод a,b,c
¦ d:=b**2-4*a*c
¦ если d<0
¦ ¦то вывод "Вещественных корней нет"
¦ ¦иначе
¦ ¦ если d=0
```

```
¦ ¦ ¦то x:=-b/(2*a);
¦ ¦ ¦ вывод "x=",x
¦ ¦ ¦иначе x1:=(-b+sqrt(d))/2/a;
¦ ¦ ¦ x2:=(-b-sqrt(d))/2/a;
¦ ¦ ¦ вывод "x1=",x1;
¦ ¦ ¦ вывод "x2=",x2
¦ ¦ все
¦ все
кон
```

***Бейсик***

```
' Квадратное уравнение
INPUT "Введите коэффциенты a, b, c"; a, b, c
d = b ^ 2 - 4 * a * c
IF d < 0 THEN
   PRINT "Вещественных корней нет"
ELSEIF d = 0 THEN
   x = -b / (2 * a)
   PRINT "x= "; x
ELSE
   x1 = (-b + SQR(d)) / 2 / a
   x2 = (-b - SQR(d)) / 2 / a
   PRINT "x1="; x1
   PRINT "x2="; x2
END IF
END
```

***Паскаль***

```
program kv_uravn;
var x,x1,x2,a,b,c,d:real;
begin
  write('Введите коэффициенты a,b,c:');
  readln(a,b,c);
  d:=b*b-4*a*c;
  if d<0
      then writeln('Вещественных корней нет')
      else if d=0
              then
                begin
                  x:=-b/2/a;
                  writeln('x=',x:6:2)
                end
              else
```

```
            begin
              x1:=(-b+sqrt(d))/2/a;
              x2:=(-b-sqrt(d))/2/a;
              writeln('x1=',x1:6:2);
              writeln('x2=',x2:6:2)
            end;
end.
```

***Cи***

```
#include<stdio.h>
void main()
{
float x,x1,x2,a,b,c,d;
printf("\nВведите коэффициенты a,b,c: ");
scanf("%e", & a );
scanf("%e", & b );
scanf("%e", & c );
d=b*b-4*a*c;
if (d<0)
      printf("Вещественных корней нет ");
      else if (d==0)
            {
              x=-b/2/a;
              printf("x= %e", x);
            }
          else
            {
              x1=(-b+sqrt(d))/2/a;
              x2=(-b-sqrt(d))/2/a;
              printf("x1= %e", x1);
              printf("x2= %e", x2);
            }
}
```

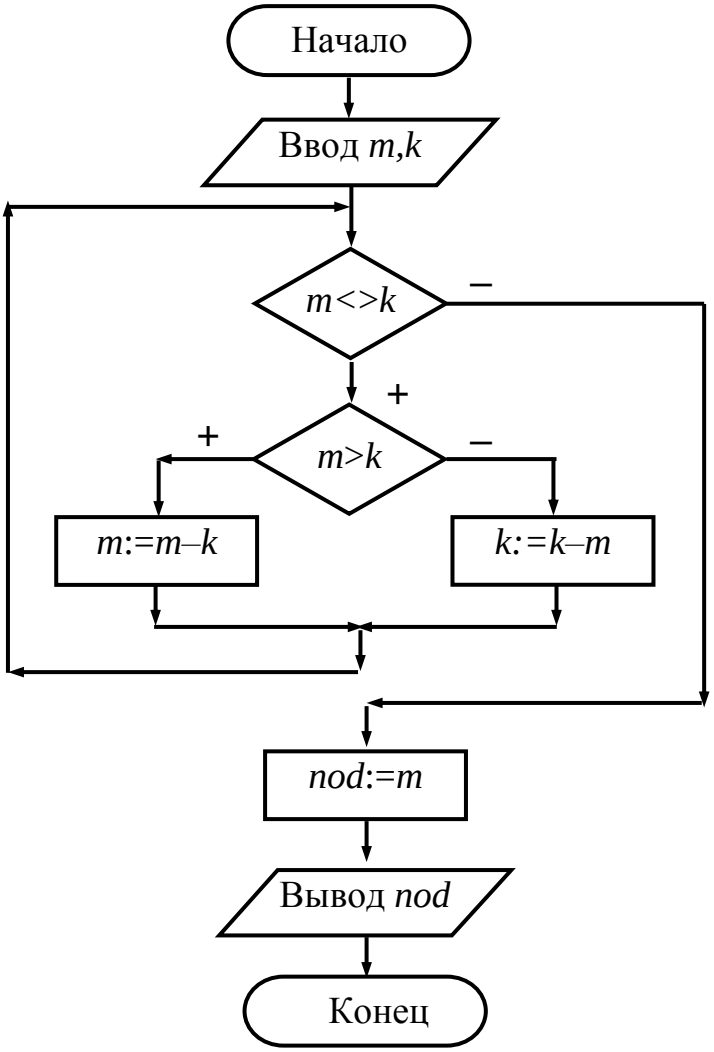
Следующая задача решается с помощью вложенной структуры «развилка в цикле».

Задача (алгоритм Евклида).

Вычислить наибольший общий делитель двух чисел

Решение.

***Блок-схема***




***Алгоритмический язык***

```
алг  нод
нач  цел m, k, nod
¦ вывод  "Введите m, k"
¦ ввод m, k
¦ нц пока m <> k
¦ ¦ если m > k
¦ ¦ ¦то m := m - k
¦ ¦ ¦иначе k := k - m
¦ ¦ все
```

```
¦ кц
¦ nod := m
¦ вывод нс, "НОД=", nod
кон
```

***Бейсик***

```
'Определение  НОД
DEFINT m,k,nod
INPUT " Введите m  и k"; m, k
DO WHILE  m <> k
      IF m > k  THEN  m = m – k  ELSE  k = k – m
LOOP
nod = m
PRINT  " НОД ="; nod
END
```

***Паскаль***

```
{ Определение НОД}
program pr_nod;
var m, k, nod : integer;
begin
   write(‘ Введите m и k’); readln(m,k);
   while m <> k
      do  if  m > k
                then   m := m - k
                else    k := k - m;
   nod := m;
   writeln (‘НОД = ‘, nod);
end.
```

***Си***

```
#include<stdio.h>
void main()
{
 int m,k,nod;
 printf("\nВведите m и k: ");
 scanf("%d",&m);
 scanf("%d",&k);
  while(m!=k)
   if(m>k) m=m-k; else k=k-m;
 nod=m;
 printf("НОД = %d",nod);
 }
```

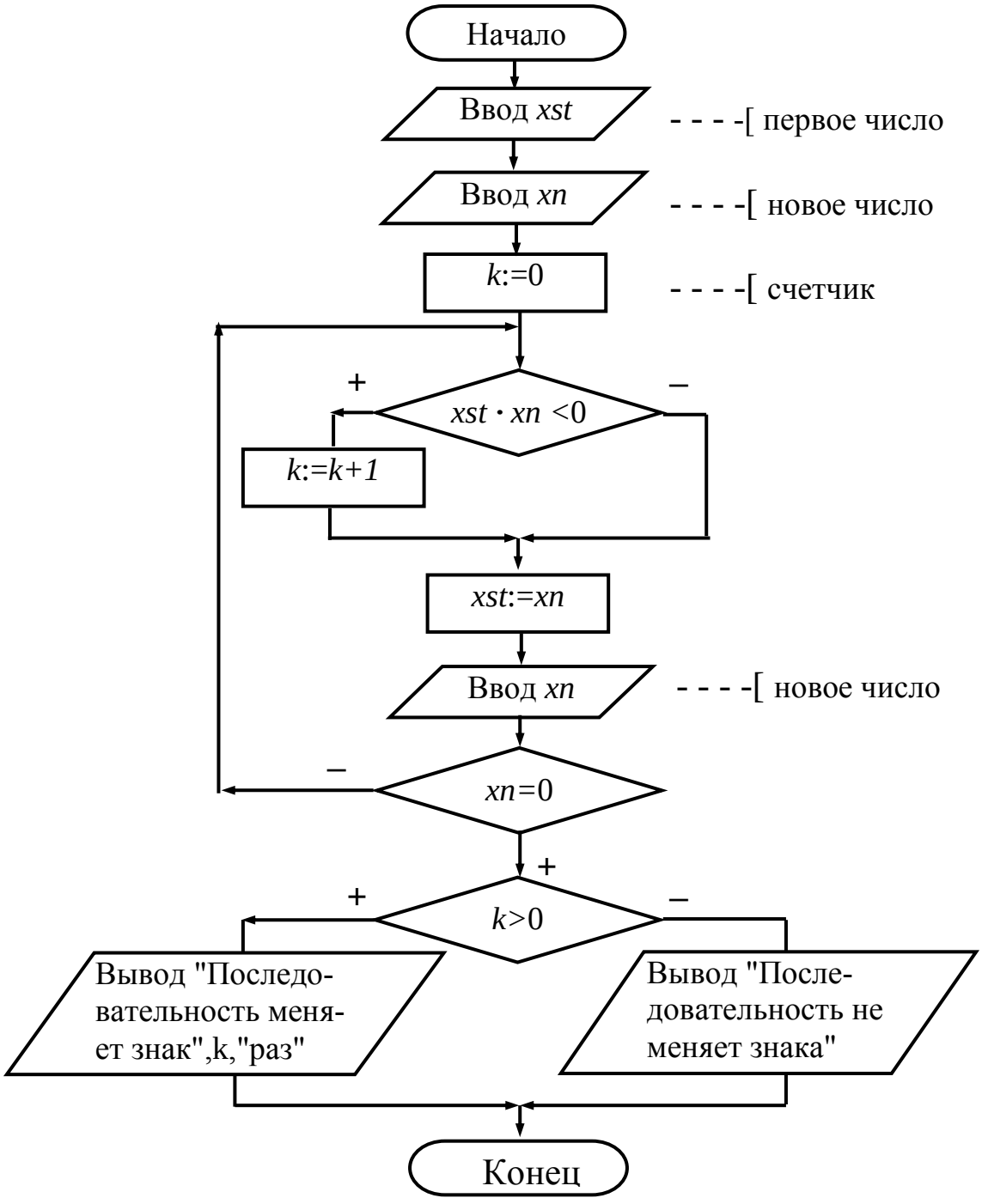
При решении следующей задачи используются последовательно друг за другом структуры цикла с постусловием и развилки, причем, в цикле с постусловием содержится вложенная структура развилки в неполной форме.

Задача.

Вводится последовательность ненулевых чисел (не менее 2), 0 – конец последовательности. Определить, сколько раз последовательность меняет знак.

Решение.

***Блок-схема***




***Бейсик***

```
'Последовательность
INPUT "Введите число:"; xst
INPUT "Введите число:"; xn
k = 0
DO
  IF xst * xn < 0 THEN k = k + 1
  xst = xn
INPUT "Введите число:"; xn
LOOP UNTIL xn = 0
IF k > 0 THEN
PRINT "Последовательность меняет знак "; k; " раз"
ELSE
PRINT "Последовательность не меняет знака"
END IF
END
```

***Паскаль***

```
program znak;
var xst,xn:real;
    k:integer;
begin
     write('Введите число:');
     readln(xst);
     write('Введите число:');
     readln(xn);
     k:=0;
     repeat
        if xst*xn<0
           then k:=k+1;
        xst:=xn;
        write('Введите число:');
        readln(xn)
     until xn=0;
     if k>0
        then writeln('Последовательность меняет знак ',k,' раз')
        else writeln('Последовательность не меняет знака')
end.
```

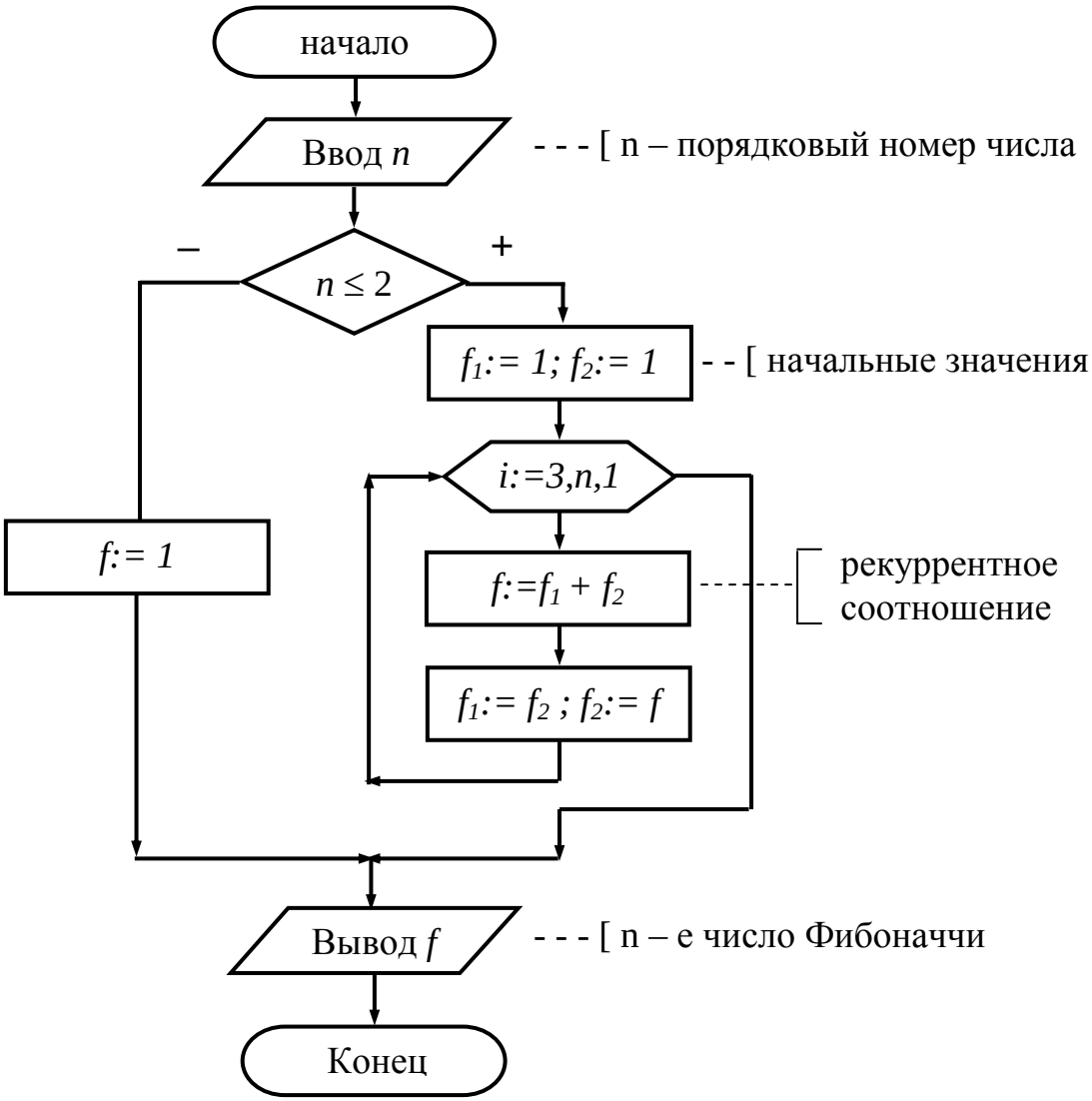
В следующей задаче используется вложенная структура цикл с параметром в развилке.

Задача.

Найти n-й член ряда Фибоначчи. (Ряд Фибоначчи задается начальными значениями $f_1=1$, $f_2=1$ и рекуррентным соотношением $f_n=f_{n-1}+f_{n-2}$ : 1, 1, 2, 3, 5, 8…).

Решение.

***Блок-схема***




***Алгоритмический язык***

```
алг  фибоначчи
нач   цел n,f1,f2,f,i
| вывод "Введите n:"
| ввод n
| если n<=2
| | то f:=1
| | иначе f1:=1;f2:=1
| |  нц для i от 3 до n
| | |  f:=f1+f2;
| | |   f1:=f2;
| | |  f2:=f;
| |  кц
| все
| вывод нс, n,"-е число Фибоначчи=",f
кон
```

***Бейсик***

```
'Числа Фибоначчи
INPUT " Введите n:"; n
IF n <= 2 THEN
f = 1
ELSE
  f1 = 1
  f2 = 1
  FOR i = 3 TO n
    f = f1 + f2
    f1 = f2
    f2 = f
  NEXT
END IF
PRINT n; "-е число Фибоначчи="; f
END
```

***Паскаль***

```
{Числа Фибоначчи}
program fib;
var i,n,f,f1,f2:integer;
begin
  write('Введите n:');readln(n);
  if n<=2
    then f:=1
    else
      begin
        f1:=1;
        f2:=1;
        for i:=3 to n do
          begin
            f:=f1+f2;
            f1:=f2;
            f2:=f;
          end;
      end;
  writeln(n,'-е число Фибоначчи=',f)
end.
```

***Си***

```
/* Числа Фибоначчи*/
```

```
#include<stdio.h>
void main()
{
 int f1,f2,n,f,i;
 printf("\nВведите n: ");
 scanf("%d",&n);
 if(n<=2)
   f=1;else
         {f1=1; f2=1;
          for(i=3;i<=n;i=i+1)
          {f=f1+f2;
          f1=f2;
          f2=f; }
          }
 printf("%d–е число Фибоначчи=%d",n,f);
 }
```