2.7. Разветвляющиеся вычислительные процессы

Вычислительный процесс называется разветвляющимся, если он реализуется по одному из нескольких направлений - ветвей. В программе должны быть учтены все возможные ветви вычислений. Выбор той или иной ветви осуществляется **по условию**, включенному в состав условного оператора. Для программной реализации условия используется **логическое выражение**. В сложных структурах с большим числом ветвей применяют оператор выбора.

Логические выражения

Логические выражения строятся из операндов, отношений, логических операций и круглых скобок.

Результатом вычисления логического выражения является одно из двух значений: (истина) (не ноль) или (ложь) (ноль)**.**

В качестве операндов используются константы, переменные и функции.

Отношения

Отношение - это простейший вид логического выражения, состоящего из двух выражений арифметического, символьного или строкового типов, соединенных знаком операции отношения.

Операция отношения - это операция сравнения двух операндов:

< - меньше
<= - меньше либо равно
> - больше
>= - больше либо равно
= = - равно
!= - не равно.

Примеры записи отношений на языке С++

Отношение	Результат
5>3	**(не ноль) т.е 1**
cos(x)>1	**(ноль)**
x*x+y*y<1	**(не ноль)** для всех точек, лежащих внутри круга с единичным радиусом и центром в начале координат
a!=’y’	**(не ноль)**, если значение символьной переменной **a** не равно символу ’y’

Следует помнить, что к операндам вещественного типа не применима операция = = из-за неточного представления чисел в памяти компьютера. Поэтому для вещественных переменных *a* и *b* отношение вида a= =b надо заменить отношением fabs(a-b)<E, где *E* - малая величина, определяющая допустимую погрешность.

Логические операции

Математическая запись	Запись на языке C++	Название операции
¬	!	Отрицание
∧	&&	Операция «И» конъюнкция (логическое умножение)
∨	||	Операция «ИЛИ» дизъюнкция (логическое сложение)

Действия логических операций удобно задать таблицами истинности, в которых приняты следующие обозначения: a, b - логические операнды; 1 - истина, 0 - ложь.

a	b	a && b
1	1	1
1	0	0
0	1	0
0	0	0

a	b	a || b
1	1	1
1	0	1
0	1	1
0	0	0

a	! a
1	0
0	1

Порядок выполнения операций в логических выражениях

В бесскобочных логических выражениях операции выполняются слева направо в соответствии с их **приоритетом**:

1. !
   Отношения
2. < <= > >=
3. == !=
4. &&
5. ||

x>0 && x<10
1 3 2

Пример. *В*ычислить логическое выражение:

$$-3 \geq 5 \vee \neg (7<9) \wedge 0 \leq 3$$

Запись на языке С++ имеет вид:

```
-3>= 5 || ! (7 <9) && 0 <= 3
   1   6 3    2      5       4
```

Внизу под операциями указан порядок их выполнения.

Результаты:

1) -3>= 5=>0; 2) (7 <9) => 1; 3) !(1) =>0;

4) 0 <= 3 => 1; 5) 0 && 1=>1; 6) 0 || 0 => 0.

Ответ: FALSE (ноль).

Примеры записи логических выражений

Записать на языке С++ логические выражения, реализующие следующие условия:

а) переменная *x* принадлежит отрезку *[a, b]*.

Ответ: `(x>= a) && (x <= b)`

б)переменная *x* не принадлежит отрезку *[a, b]*.

Ответ: Данное условие можно записать в одном из двух вариантов:

1) `x <a || x> b;`

2) или воспользоваться операцией отрицания:

```
! (x>= a && x <= b)
```

Условные операторы

На языке С++ различают два вида условных операторов: короткий и полный.

Короткий условный оператор

Общий вид записи

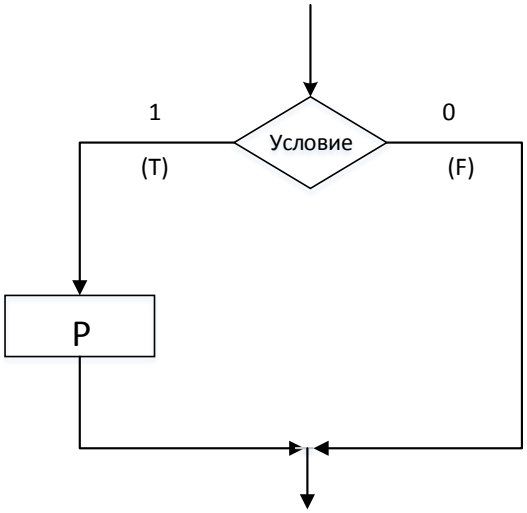
if (логическое выражение) P;

где Р - любой оператор.

Работа оператора

Вычисляем (выполняем) условие. Если условие выполняется (TRUE), то выполняем оператор, который следует за круглой скобкой (Р). Если нет, то (Р)-оператор пропускается.

Графическая интерпретация оператора



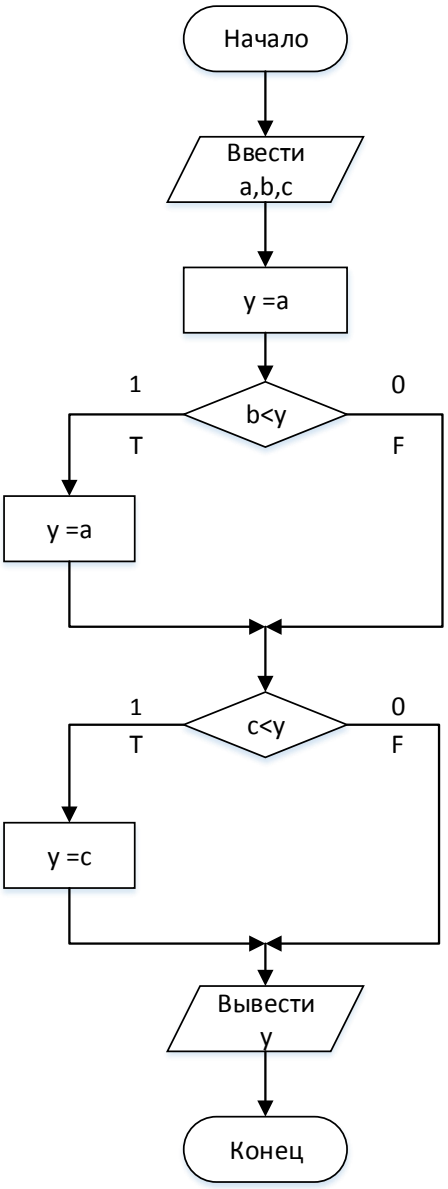
Замечание. По определению, конструкция короткого условного оператора включает единственный оператор P. Если в задаче по заданному условию требуется выполнить несколько операторов, то их необходимо заключить в операторные скобки { }**,** образуя тем самым составной оператор. Тогда запись условного оператора с использованием скобок имеет следующий вид:

```
if (логическое выражение)
  {
     оператор  1;
     ..................
     оператор  N;
  }
```

Составной оператор

Пример. Переменной *y* присвоить минимальное значение из трех различных чисел, т.е. *y* = min(a, *b, c).*

Схема алгоритма



Программа

```
#include "stdio.h"
#include<cmath.h>
int main()
{
float a, b, c,y;
printf("Введите числа a, b, c");
scanf("%f%f%f", &a,&b,&c);
y=a;
  if (b<y)  y = b;
  if (c<y)  y = c;
 printf("y =%6.2f", y);
 return 0;
}
```

```
#include <iostream>
#include <cmath>
using namespace std;
int main()

{

float a,b,c,y;
cout<<"a,b,c";
cin>>a>>b>>c;
y=a;
if (b<y) y=b;
if (c<y) y=c;
 cout <<"y="<<y<<endl;
    return 0;
}
```

Полный условный оператор

Общий вид записи

if (логическое выражение) P1; else P2;

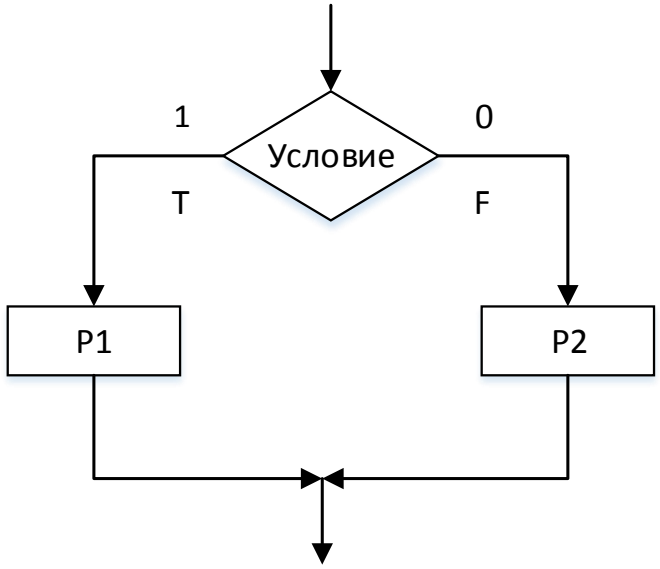
где P1, P2 - любые операторы или даже группы операторов.

Работа оператора

Вычисляется логическое выражение, и если оно имеет значение TRUE(не ноль), то выполняется оператор P1, стоящий после логического выражения. В противном случае (FALSE (ноль)) оператор P1 пропускается, а выполняется оператор P2, стоящий после служебного слова else.

Графическая интерпретация оператора

В схемах алгоритма полному условному оператору соответствует структура ЕСЛИ-ТО-ИНАЧЕ.



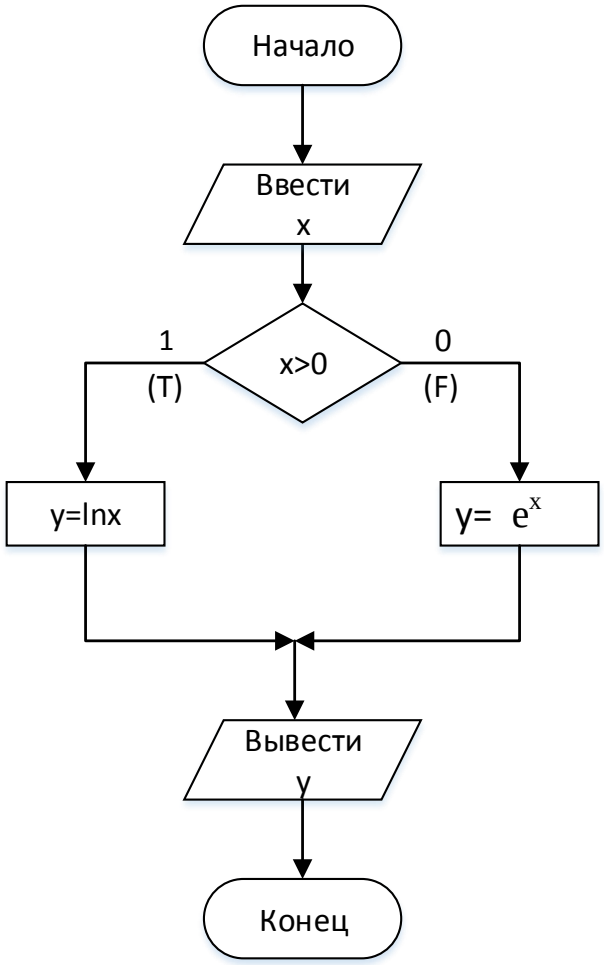
Замечание. Операторы P1 и P2 входят в стандартную конструкцию полного условного оператора как единственные. Если возникает необходимость выполнить в ветвях несколько операторов, то их заключают в операторные скобки { }. Вид записи условного оператора в этом случае следующий:

```
if (логическое выражение)
  {
     оператор 1;
     ……
     оператор n;
      }
 else
      {
     оператор 1;
     ……
     оператор m;
       }
```

Пример 1. Вычислить значение переменной *y* по одной из двух ветвей:

$$y = \begin{cases} \ln x, если \quad x > 0, \\ e^x, если \quad x \le 0. \end{cases}$$

Схема алгоритма



Программа

```
#include "stdio.h"
#include<cmath>
    int main()
{
float x,y;
 printf("Введите число x=  ");
 scanf("%f",&x);
if( x>0)  y = log(x);
else    y= exp(x);
printf("y =%6.2f", y);
return 0;
}
```

```
#include <iostream>
#include <cmath>
#include <windows.h>
using namespace std;
int main()

{
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
float x,y;
 cout <<"Введите число x=  ";
 cin >>x;
if( x>0)  y=log(x);
else  y=exp(x);
cout <<"y= "<<y<<endl;
return 0;
}
```

Пример 2. Вычислить корни полного квадратного уравнения

$$ax^2 + bx + c = 0.$$

В программе предусмотреть проверку дискриминанта на знак. Если дискриминант окажется отрицательным, то вывести сообщение «Корни мнимые».

```
#include "stdio.h"
#include<cmath>
int main()
{
 float a, b, c, d, x1, x2;      // описание переменных
 printf("Введите коэффициенты уравнения    ");
 scanf("%f%f%f",&a, &b, &c); // ввод исходных данных
 d = pow(b,2)-4*a*c;           //вычисление дискриминанта
 if ( d<0 )                    // сравнение дискриминанта с нулем
```

```
 printf ("Корни мнимые");   /*вывод «корни мнимые»,
                             если  d  окажется < 0  */
  else                       // иначе
        {
         x1=(-b+sqrt(d))/(2*a);  // вычисление первого корня
          x2=(-b-sqrt(d))/(2*a);  // вычисление второго корня
     printf("x1=%f    x2=%f",x1,x2);    // вывод корней на экран
        }
 }
```

Пример. Проверка трех условий (x<5, x>5, x==10)

```
#include <windows.h>
#include <cmath>
using namespace std;//пространство имен
int main()

{
   SetConsoleCP(1251);
SetConsoleOutputCP(1251);
int x;
cout<<"Введите свою оценку"<<endl;
cin>>x;
if (x<5) {
      cout<<"Вы полный чайник!!!"<<endl;
}
else if (x==10){
   cout <<"Мне просто повезло!!!!"<<endl;
}
else {
  cout <<"Я не хакер, только учусь!!!!"<<endl;
}
return 0;
}
```

Вложенные структуры условных операторов

Структура называется вложенной, если после служебного слова else или при истинности логического выражения вновь используются условные операторы. Число вложений может быть произвольным. При этом справедливо следующее правило: служебное слово else всегда относится к ближайшему выше слову if.

Пример. Вычислить значение y по одной из трех ветвей:

$$y=\begin{cases}\frac{1}{2}\sqrt{x}, & \text{если} \quad x\geq 1,\\ \frac{1}{3}\sqrt[3]{x}, & \text{если} \quad 0<x<1\\ \frac{1}{4}\sqrt[4]{|x|}, & \text{если} \quad x\leq 0\end{cases}$$

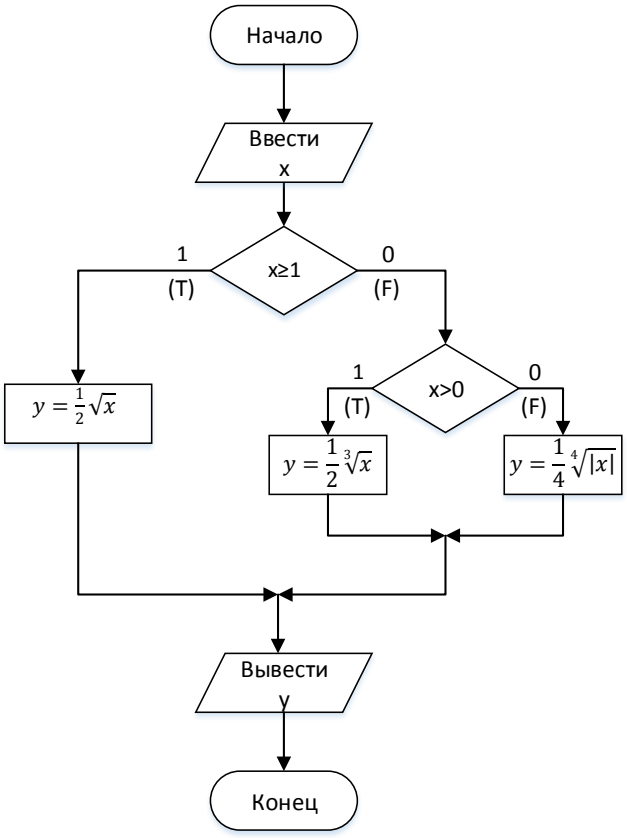
При решении данной задачи возможны два варианта прораммирования:

1) без вложенной структуры;
2) с вложенной структурой.

Ниже рассмотрены оба варианта решения задачи.

Вариант 1 (с использованием вложенной структуры)

Схема алгоритма

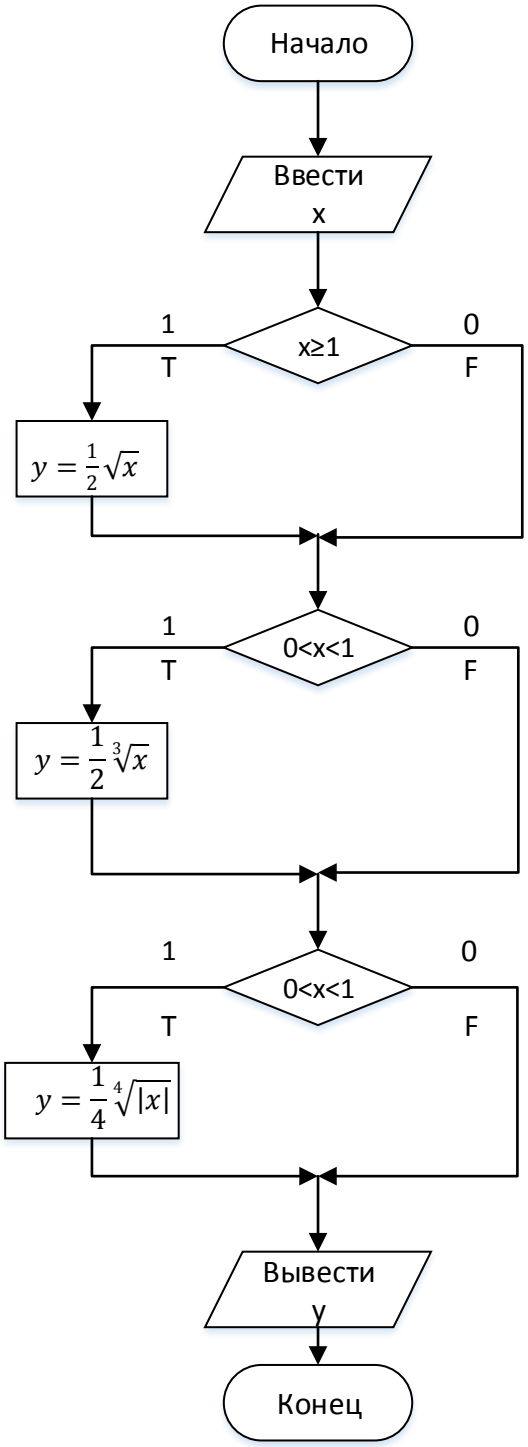


Программа

```
#include "stdio.h"
#include<cmath>
  int main()
   {
 float a,x, y;
 printf("Введите число x= ");
   scanf("%f",&x);
   if (x>=1)
     y = sqrt(x)/2;
   else
if(x>0){a=1.0/3.0;
y=pow(x,a)/2;}
else
{a=1.0/4.0; y=pow(x,a)/4;}
printf("y=%6.2f",y);
return 0;
}
```

Вариант 2 (без использования вложенной структуры)

Схема алгоритма



Программа

```
include "stdio.h"
#include<cmath.h>
int  main()
{
float a,x,y;
printf("Введите число x=" );
scanf("%f",&x);
if (x>=1)
y = sqrt(x)/2;
if (x>0 && x<1)
{ a= 1.0/3.0;
y =pow(x,a)/2;
}
if(x<=0)
{a= 1.0/4;
y=pow(x,a)/4;
}
printf("y=%6.2f",y);
return 0;
}
```

Оператор выбора switch-case

При многократном вложении условных операторов программная конструкция становится громоздкой и ее трудно понять. Считается, что число вложений не должно превышать двух-трех. При большем числе вложений рекомендуется использовать оператор выбора switch-case.

Оператор-переключатель предназначен для выбора одного из нескольких альтернативных путей выполнения программы. Вычисление оператора-переключателя начинается с вычисления *выражения*, после чего управление передается *оператору*, помеченному *константным выражением*, равным вычисленному значению *выражения*. Выход из оператора-переключателя осуществляется оператором **break**. Если значение *выражения* не равно ни одному *константному выражению*, то управление передается *оператору*, помеченному ключевым словом *default*, если он есть.

Общий вид записи оператора

```
switch (селектор)
   {
   case константное_целое_выражение1:   оператор 1;  break;
   ……
   case константное_целое_выражение n:   оператор n;   break;
   default: оператор n+1;
    }
```

Пример:

```
int n;
switch (n )
{
case 1:   cout<<”1”;  break;
case 2:   cout<<”2”;  break;
……
case 9:   :   cout<<”9”;      break;
default:  cout<<”остальное”;  ;
    }
```

Работа оператора

Сначала вычисляется выражение в круглых скобках (назовем его селектором). Затем вычисленное значение селектора последовательно сравнивается с константным выражением, следующим за **case**.

Если селектор равен какому-либо константному выражению, стоящему за **case**, то управление передается оператору, помеченному соответствующим оператором **case**. Если селектор не совпадает ни с одной меткой варианта, то управление передается на оператор, помеченным словом **default**. Если **default** отсутствует, то управление передается следующему за **switch** оператору. Отметим, что после передачи управления по какой-либо одной из меток дальнейшие операторы выполняются подряд. Поэтому, если необходимо выполнить только часть из них, нужно позаботиться о выходе из **switch**. Это обычно делается с помощью оператора **break**, который осуществляет немедленный выход из тела оператора **switch**.

Селектор - это выражение целого или символьного типа.

case -константы выбора - возможные значения селектора.

default – осуществляет обработку непредусмотренного значения селектора. Наличие этой метки в операторе switch необязательно.

Пример 1:

```
#include <iostream>
#include <cmath>
#include <windows.h>
using namespace std;
int main()

{
   SetConsoleCP(1251);
SetConsoleOutputCP(1251);
 int i, d;
    cout<<"Задайте целое значение i"<<endl;
    cin>>i;
switch ( i ){
    case 1: case 2: case 3: cout<<" i="<< i <<"\n";
    case 4: cout<<" i="<< i <<" i^2= "<< i*i<<"\n";
    d=3*i - 4; cout<<" d=" << d <<"\n";
    break;
    case 5: cout<<"i=5.\n"; break;
    default: cout<<" Значение i меньше 1 или больше 5.\n";
    }
  return 0;
}
```

Если будет введено число 2, то будет напечатано

```
i=2
i=2 i^2=4
d=2.
```

Если **i** равно 4 , то будет выведено

```
i=4 i^2=16
d=8.
```

При **i**=5 будет выведено

```
i=5.
```

При всех остальных значениях **i** будет меньше 1 или больше 5.

Замечание. Если в строке выбора необходимо записать несколько операторов, то их заключают в операторные скобки {...}.

Пример 2. Вычислить значение *y*.

$$y = \begin{cases} \sin x, & \text{если} \quad 1 \le x < 2 \\ e^{-x}, & \text{если} \quad 2 \le x < 3 \\ \ln x, & \text{если} \quad 3 \le x < 4 \\ tgx, & \text{если} \quad 4 \le x < 5 \end{cases}$$

Если значение x не принадлежит рассматриваемым промежуткам, то вывести на экран соответствующее сообщение.

В задаче переменная *x* является вещественной и не может использоваться в качестве селектора оператора case. Введем новую переменную целого типа n, которой присваивается целая часть значения *x*. Тогда программа решения данной задачи с использованием оператора выбора может быть составлена следующим образом.

```
#include "stdafx.h"
#include<math.h>
  int main()
  {
     float x, y;
     int n;
    printf("Введите число x=   ");
    scanf("%f",&x);
    if( (x<1) || (x>=5))
printf("x не принадлежит рассматриваемой области\n");
     else
         { n = x;
     switch (n)
    {
     case 1:  y =sin(x);   break;
     case 2:  y = exp(-x); break;
     case 3:  y =log(x);   break;
     case 4:  y = tan(x);  break;
```

```
    default:  printf(" такого решения нет \n");
              break;
    }
    if ((n==1)||(n==2)||(n==3)||(n==4))
      printf("y=%6.2f", y);
            }
  return 0;
  }
```

Контрольные задания

1)Записать на языке С++ логические выражения

а) $\neg a \vee b$;

б) $-1 \le x \le 1$ *или* $2 \le x \le 4$;

в) переменная x находится вне интервала *[a, b];*

г) все точки на плоскости находятся выше оси абсцисс;

д) все точки на плоскости находятся либо в первом, либо в третьем квадрантах;

е) все точки на плоскости лежат выше прямой y = 1 +x.

2) Начертить на плоскости область, в которой логическое выражение имеет значение TRUE

а) (x>= 0) && (y>= 0) && (y <= x) || (y <= -1);

б) (x*x +y*y <= 1) && (y>= 0) && !(y <x);

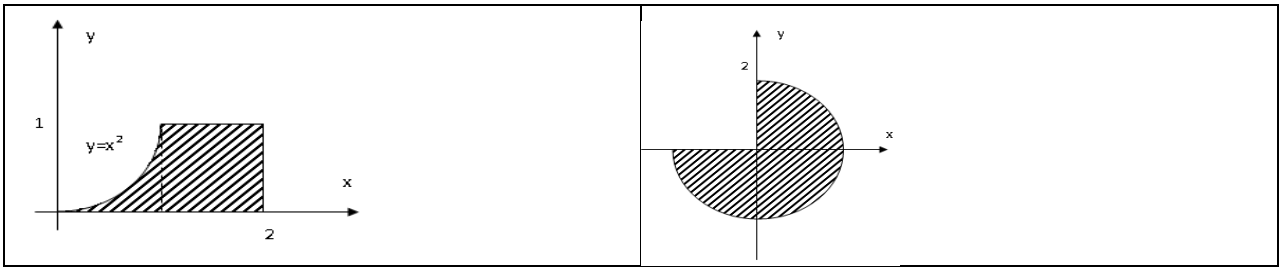
в) (y>= x) && (y>= -x);

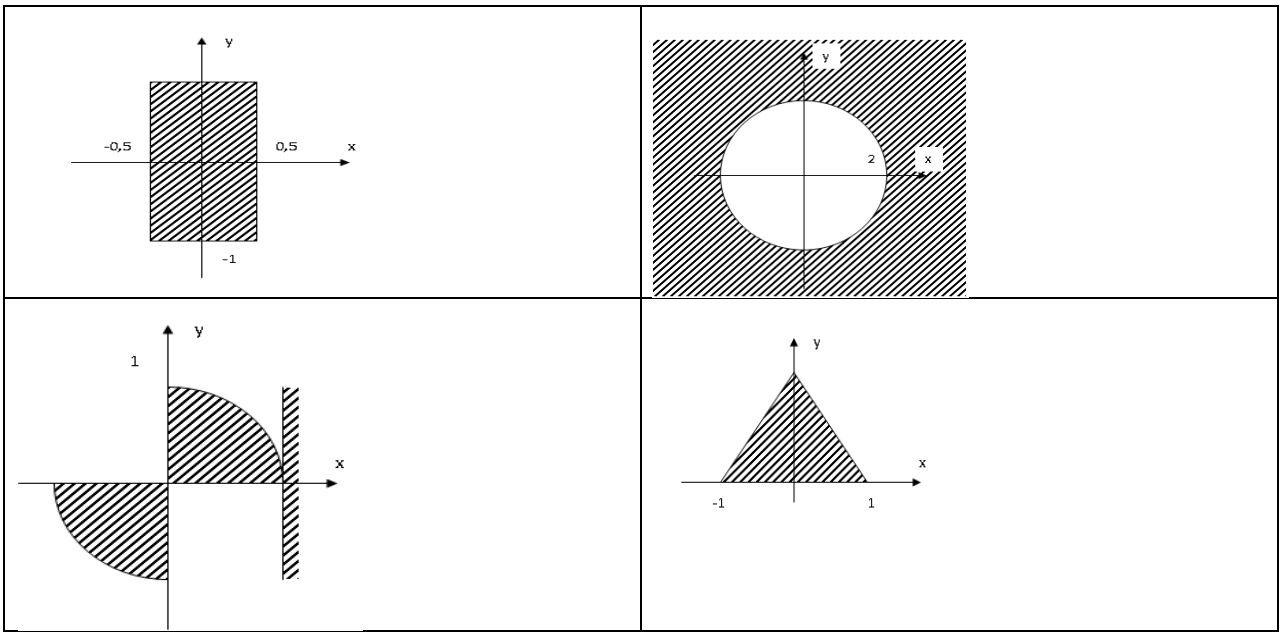
г) (y <= 2) && (x <- 2) || (x*y <0);

д) (x*y>= 0) && (y <= x) && (x <1) && (y>-1);

е) (x*x + y*y <= 4) &&(y>= x*x).

3) Записать на языке С++ логические выражения, принимающие значение TRUE для точек, принадлежащих заштрихованной области (рисунок 1).





4) Записать на языке С++ логические выражения, расставить действия и вычислить при: i = 5, j = 2, *к=2, a* = TRUE, b = FALSE

а) $i \neq 1 \vee a \wedge \neg(b \wedge j > k)$;

б) $(i \leq 1 \vee a) \wedge (b \vee j = k)$;

в) $\neg(i = j^2 + 1) \vee a \wedge b$;

г) $i > j + k \wedge a \wedge \neg b$;

д) $a \vee b \wedge (i \cdot j > k^2)$;

е) $(a \vee \neg b \wedge j^2 = k^2) \wedge \neg b$.