

2 ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Цель работы - овладение практическими навыками разработки, программирования вычислительного процесса разветвляющейся структуры; знакомство с задачами, для решения которых используются операторы ветвления.

1.1 Подготовка к лабораторной работе

При подготовке к лабораторной работе необходимо изучить правила вычисления выражений с использованием операций сравнения и логических бинарных операций, освоить особенности использования условной операции, составного оператора, оператора ветвления.

1.2 Теоретические сведения

Условная операция.

В отличие от унарных и бинарных операций в условной операции используется три операнда. Форма представления:

Выражение1 ? Выражение2 : Выражение3;

Первым вычисляется значение *выражения1*. Если оно истинно, то вычисляется значение *выражения2*, которое становится результатом.

Если при вычислении *выражения1* получится 0, то в качестве результата берется значение *выражения3*.

Пример. Условная операция

$x < 0 ? -x : x$; //вычисляется абсолютное значение x.

Составной оператор. К составным операторам относят собственно составные операторы и блоки. В обоих случаях это последовательность операторов, заключенная в фигурные скобки. *Блок отличается от составного оператора наличием определений в теле блока.*

```
{
n++;           // это составной оператор
summa+=n;
}
{
int n=0;
n++;           // это блок
summa+=n;
}
```

Блок представляет собой последовательность инструкций, включая, возможно, объявления, заключенных в фигурные скобки. Общий вид *блока*:

```
{
[ объявления ]
...
инструкция [ инструкция ]
...
}
```

В конце блока точка с запятой не ставится.

Объявление локальной переменной можно разместить в любом месте программы перед первым обращением к этой переменной.

Блоки инструкций могут быть вложены друг в друга на любую глубину. На переменную, объявленную внутри блока, можно ссылаться только внутри этого или внутри вложенных блоков (если эта переменная не скрыта переменной с таким же именем внутри вложенного блока).

Выполнение блока заключается в последовательном выполнении составляющих его инструкций, включая вложенные в него блоки:

Оператор ветвления if. Оператор ветвления **if** иначе называют **условным** оператором. Он имеет две формы: полную и краткую.

Полная форма оператора ветвления выглядит так:

if <логическое выражение> <оператор 1>;
 else <оператор 2>;

Действие оператора. Если <логическое выражение> истинно, то выполняется <оператор 1>, в противном случае – <оператор 2>. Операторы 1 и 2 могут быть простыми или составными.

Алгоритмическая структура, соответствующая этому оператору, показана на рис.2.1.

Пример. Пусть $x = 9$. Тогда в результате выполнения операторов:

if $x > 7$ $y = x * x$; **else** $y = \sin(x)$;

if $x < 5$ $z = \exp(x)$; **else** $z = \sqrt{x}$;

получим $y = 81$, $z = 3$.

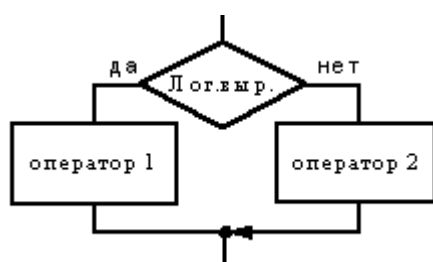


Рис. 2.1. Алгоритмическая структура выполнения полной формы оператора ветвления.

Пример. Пусть $a = 5$. Тогда в результате выполнения операторов:

if $a < 7$

{ $b = a - 2$; $c = 1 + 2 * a$; }

else

{ $b = 2 + 5 * a$; $c = 12 - 4 * (a - 3)$; }

получим $b = 3$, $c = 11$.

Краткая форма оператора ветвления **if** выглядит так:

if <логическое выражение> <оператор 1>;

Действие оператора. Если <логическое выражение> истинно, то выполняется <оператор 1>, иначе выполняется оператор, который находится после данной конструкции.

Алгоритмическая структура, соответствующая этому оператору, показана

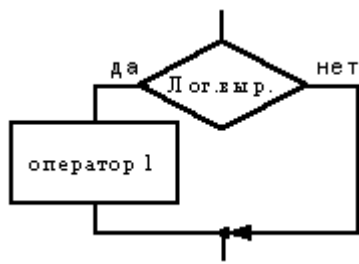


Рис. 2.2. Алгоритмическая структура выполнения краткой формы оператора ветвления.

Пример. Пусть $x = 25$. Тогда в результате выполнения операторов

If $x > 12$ $y = 2 * x$; $z = 10$;

If $x < 5$ $z = x / 2$;

получим $y = 50$, $z = 10$.

Задача 1.1. Определить, попадет ли точка с координатами (x, y) в ограниченную область (рис. 2.3).

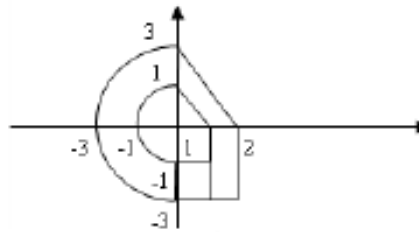


Рис. 2.3. Границы области для задачи 1.1.

Исходные данные: x, y

Результат: да (1) или нет (0)

Математическая модель:

$Ok = I \parallel II \parallel III \parallel VI$, где I, II, III, IV – условия попадания точки в заштрихованную область для каждого квадранта.

Квадрант I: Область формируется прямыми OX и OY , прямой, проходящей через точки $(0,1)$ и $(1,0)$ и прямой, проходящей через точки $(0,3)$ и $(2,0)$.

Необходимо определить уравнения прямых $y = ax + b$. Решаем две системы уравнений:

$$\begin{cases} 1 = a * 0 + b \\ 0 = a * 1 + b \end{cases} \quad \begin{cases} 3 = a * 0 + b \\ 0 = a * 2 + b \end{cases}$$

Из этих систем получаем следующие уравнения прямых:

$$y = -1x + 1;$$

$$y = -\frac{3}{2}x + 3;$$

Тогда условие попадания точки в I квадрант будет выглядеть следующим образом:

$$y \geq -x + 1 \ \&\& \ y \leq -\frac{3}{2}x + 3 \ \&\& \ y \geq 0 \ \&\& \ x \geq 0.$$

Квадранты II и III: Область формируется прямыми OX и OY и двумя окружностями, описываемыми формулами $x^2 + y^2 = 1$, $x^2 + y^2 = 9$.

Тогда условие попадания точки во II и III квадранты будет выглядеть следующим образом:

$$x^2 + y^2 >= 1 \ \&\& \ x^2 + y^2 <= 9 \ \&\& \ x <= 0.$$

Квадрант IV:

Область формируется двумя прямоугольниками. Точка может попадать либо в первый прямоугольник, либо во второй.

Условие попадания точки в IV квадрант будет выглядеть следующим образом:

$$(x >= 0 \ \&\& \ x <= 1 \ \&\& \ y <= -1 \ \&\& \ y >= -3) \ || \ (x >= 1 \ \&\& \ x <= 2 \ \&\& \ y <= 0 \ \&\& \ y >= -3).$$

Вариант 1. Программа имеет вид (без использования разветвляющейся структуры):

```
#include <iostream>
#include <math.h>
#include <windows.h>
#include <conio.h> // файл, где определена функция getch()
using namespace std;
void main()
{
    SetConsoleOutputCP(1251);
    float x,y;
    cout<<"Введите x,y\n";
    cin>>x>>y;
    bool Ok=(y>=-x+1 && y<=-3/2*x+3 && x>=0 && y>=0)//
    (pow(x,2)+pow(y,2)>=1 && pow(x,2)+pow(y,2)<=9 && x<=0)//
    (x>=0 && x<=1 && y<=-1 && y>=-3)//
    (x>=1 && x<=2 && y<=0 && y>=-3);
    cout<<"\n"<<Ok;
    getch(); // ждать нажатия любой клавиши
}
```

Вариант 2. Программа имеет вид (с использованием условной операции):

```
void main()
{
    SetConsoleOutputCP(1251);
    float x,y;
    cout<<"Введите x,y\n";
    cin>>x>>y;
    ((y>=-x+1 && y<=-3/2*x+3 && x>=0 && y>=0)//
    (pow(x,2)+pow(y,2)>=1 && pow(x,2)+pow(y,2)<=9 && x<=0)//
    (x>=0 && x<=1 && y<=-1 && y>=-3)//
    (x>=1 && x<=2 && y<=0 && y>=-3)) ?
    cout<<"точка попадает в область\n":
    cout<<"точка не попадает в область\n";
    getch(); // ждать нажатия любой клавиши
}
```

Вариант 3. Программа имеет вид (с использованием оператора ветвления):

```
void main()
{
    SetConsoleOutputCP(1251);
    float x,y;
    cout<<"Введите x,y\n";
    cin>>x>>y;
    if ((y>=-x+1 && y<=-3/2*x+3 && x>=0 && y>=0)//
        (pow(x,2)+pow(y,2)>=1 && pow(x,2)+pow(y,2)<=9 && x<=0)//
        (x>=0 && x<=1 && y<=-1 && y>=-3)//
        (x>=1 && x<=2 && y<=0 && y>=-3))
        cout<<"точка попадает в область\n";
    else cout<<"точка не попадает в область\n";
    getch();        // ждать нажатия любой клавиши
}
```

Тесты приведены в табл. 2.1.

Таблица 2.1

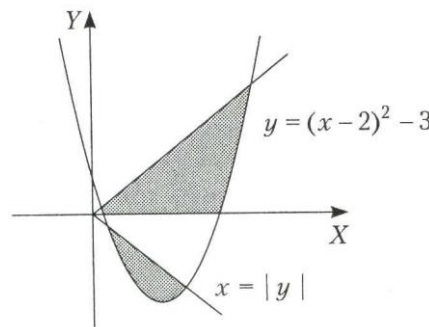
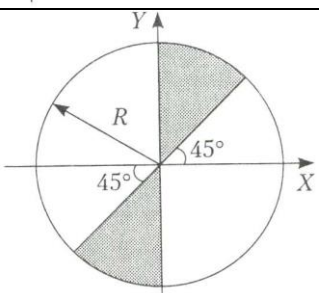
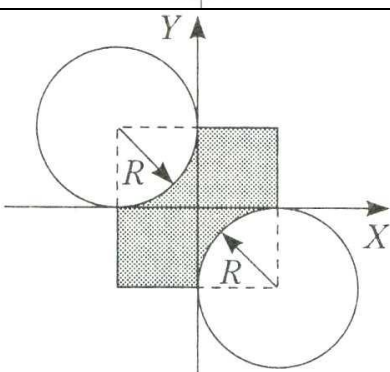
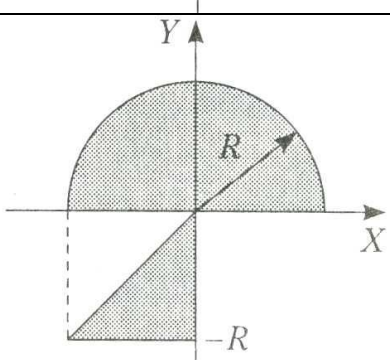
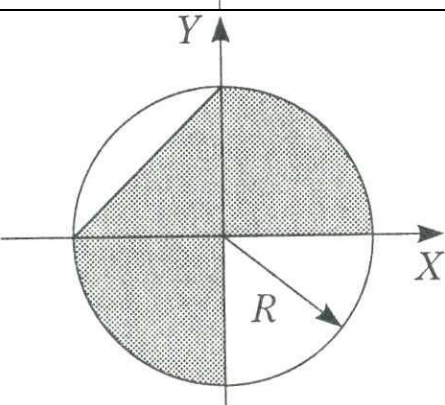
Тесты к задаче 2.1		
Квадрант	Исходные данные (X,Y)	Результат (Ok)
I	0,2, 0,2	0
I	0,7, 0,5	1
II	-0,5, 0,5	0
II	-2,0	1
III	-0,5, -0,5	0
III	-2, -1	1
IV	0,5, -0,5	0
IV	1,5, -1	1
Центр системы координат	0,0	0

1.3 Варианты заданий

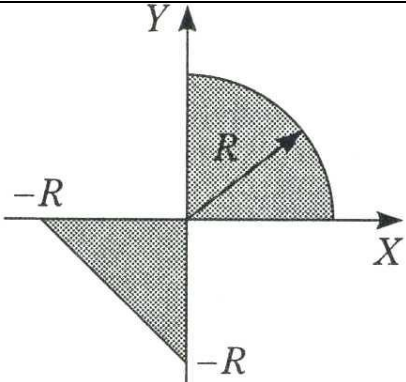
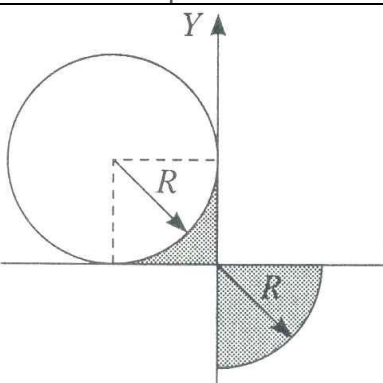
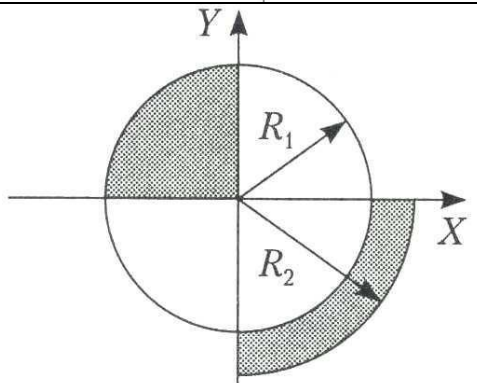
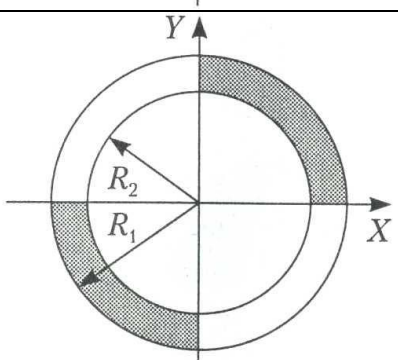
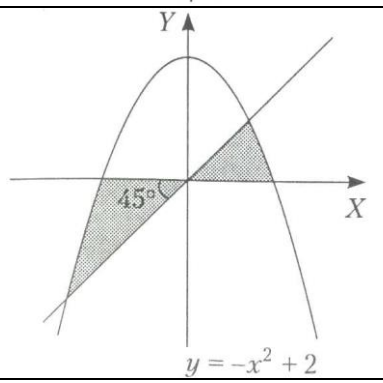
Определить, попадет ли точка с координатами (x, y) в ограниченную область (табл.2.2). Разработать математическую модель: определить исходные данные и результаты решения задачи, а также математические формулы, с помощью которых можно перейти от исходных данных к конечному результату. Составить и отладить программы без использования разветвляющейся структуры, с использованием условной операции и оператора ветвления:

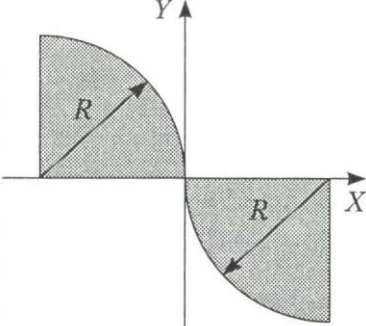
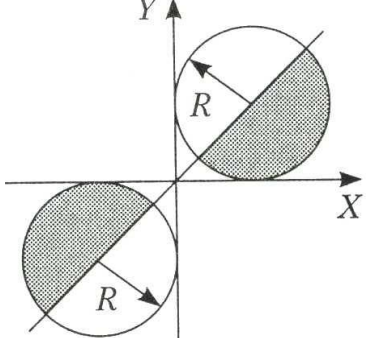
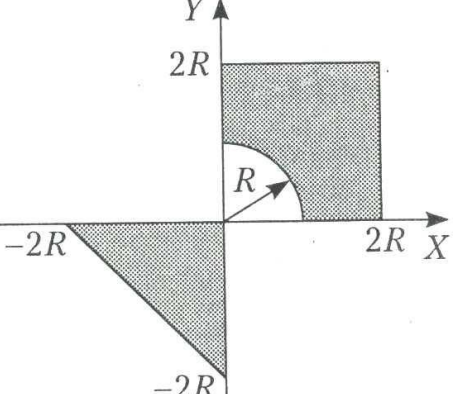
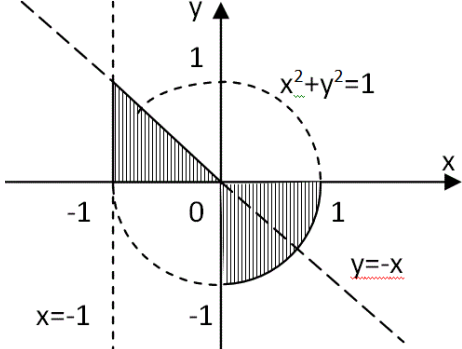
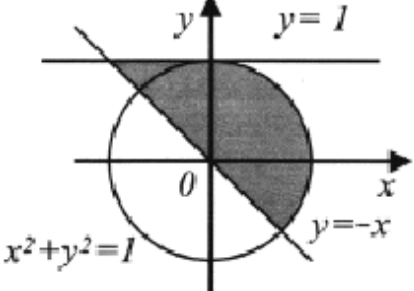
Таблица 2.1

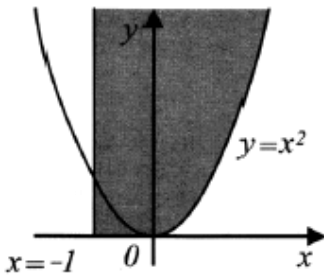
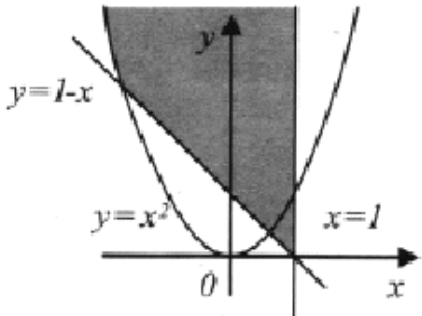
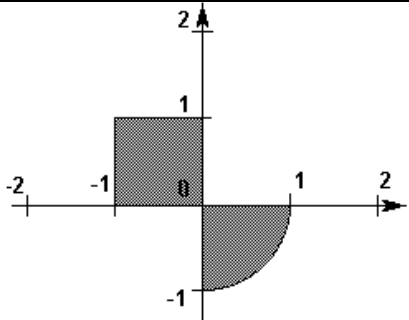
Варианты заданий

№ вар.	Область
1	2
1.	
2.	
3.	
4.	
5.	

1	2
6.	
7.	
8.	
9.	
10.	

1	2
11.	
12.	
13.	
14.	
15.	

1	2
16.	
17.	
18.	
19.	
20.	

1	2
21.	
22.	
23.	

2.4 Контрольные вопросы

1. Что такое логическое выражение? Какие значения оно может принимать?
2. Что такое разветвляющийся вычислительный процесс?
3. Правила выполнения условной операции?
4. Какие операторы называются простыми?
5. Что такое составной оператор?
6. Какие известны формы оператора ветвления?
7. Как происходит выполнение оператора ветвления?
8. Какие условные операторы называются вложенными?
9. Составьте последовательность операторов для вычисления величины $z=0$, если $x < -2$; $z=1$, если $-2 \leq x \leq 2$; $z=-1$, если $x > 2$.
10. Зачем необходимо при отладке программы тестировать все ветви алгоритма?