

ЛАБОРАТОРНАЯ РАБОТА № 3

Управляющие операторы условного и безусловного переходов.

Разветвляющиеся программы

1. Краткие теоретические сведения

Как известно, из предыдущих лабораторных работ, все программы состоят из последовательности операторов, которые обычно выполняются поочерёдно в том порядке, в каком они записаны в программе. Однако часто возникает необходимость изменить очередность выполнения операторов, т.е. пропустить или наоборот выполнить какую-то группу операторов в зависимости от выполнения или не выполнения некоторых заданных условий. Кроме того иногда необходимо повторить группу операторов несколько раз, то есть организовать цикл. Для выполнения этих задач служат управляющие операторы. Управляющие операторы подразделяются на операторы принятия решения, к ним относятся операторы условного и безусловного переходов, и операторы для организации циклов, которые будут рассмотрены в следующей лабораторной работе.

Одной из важнейших возможностей компьютерного процессора является возможность принятия решения. Под этим выражением подразумевается, что процессор может направить поток выполнения запрограммированных команд по тому или иному пути в зависимости от того истинно или ложно некоторое заданное условие. Любой язык программирования обеспечивает возможность принятия решения. В алгоритмическом языке С#, как и во многих других, основой для такой возможности является оператор услов-

ного перехода $i f$, который действует в C# практически также, как и оператор IF в любом другом языке программирования./

Оператор условного перехода $i f$ и его конструкции

Оператор условного перехода $i f$ (если), как было уже сказано, предназначен для выбора одного из возможных вариантов исполнения операторов программы, поэтому его и называют оператором принятия решения. Существует несколько разновидностей конструкций этого оператора. Рассмотрим их последовательно по мере усложнения.

$i f$ (*условие выбора*)

{

 // Записанные в скобках операторы (оператор)

 // будут выполняться, если *условие выбора* истинно

}

// Записанные далее операторы будут выполняться

// в любом случае, независимо от *условия выбора*.

В качестве *условия выбора* используется значение логического выражения. При выполнении этой конструкции вначале вычисляется значение логического выражения, записанного в скобках. Результат вычисления имеет тип `bool` `ean`. Если вычисленное выражение имеет значение `true` (истина), то выполняются операторы в фигурных скобках и все следующие за ними. Если получено значение `false` (ложь) то операторы в фигурных скобках пропускаются и выполняются только операторы следующие за ними.

Пример 1.

```
// Сохраняем наибольшее значение из двух, a и b,  
// в переменной max  
max = b;  
if (a > b)  
{  
    max = a;  
}
```

В данном фрагменте программы изначально предполагается, что наибольшее значение имеет переменная *b*, и она присваивается переменной *max*. Если это не так, то переменной *max* присваивается значение переменной *a*.

Операторы, записываемые в фигурных скобках можно размещать в одной строке, как в следующем примере.

Пример 2.

```
max = b;  
if (a > b){max = a; }
```

При записи в фигурных скобках нескольких операторов в конце каждого из них ставится точка с запятой. Если в фигурных скобках записывается один оператор, то фигурные скобки можно опустить (см. пример 3).

Пример 3.

```
max = b;  
if (a > b) max = a;
```

Конструкция if-else (если -иначе)

Данную конструкцию целесообразно использовать, когда необходимо задать выполнение одного из двух блоков операторов (или одного из двух операторов), в зависимости от результата проверки *условия выбора*. Конструкция имеет следующий вид записи.

```
if (условия выбора)
{
    // Если условие выбора истинно, то будут выпол-
    // няться оператор или операторы блока 1.
}
else
{
    // В противном случае (иначе)
    // если условие выбора ложно, то будут выполнять-
    // ся оператор или операторы блока 2.
}
// Записанные далее операторы будут выполняться
// в любом случае, независимо от условия выбора.
```

Если результатом проверки *условия выбора* является значение true (истина), то будут выполнены *операторы блока 1*. Далее будет выполняться первый оператор, следующий за последней фигурной скобкой. *Операторы блока 2* выполняться не будут. Если проверка *условия выбора* даст результат false (ложь), то *операторы блока 1* будут пропущены, и будут выполнены *операторы блока 2*. Далее будет выполняться первый

оператор, следующий за последней фигурной скобкой. Каждый из указанных блоков может состоять из одного оператора, тогда фигурные скобки могут быть опущены.

Пример 4.

```
if (a > b)
{
    max = a; // Эти операторы будут выполняться,
    min = b; // если условие выбора  $a > b$  истинно.
}
else
{
    max = b; // Эти операторы будут выполняться,
    min = a; // если условие выбора  $a > b$  ложно.
}
// Записанные далее операторы будут выполняться
// в любом случае, независимо от условия выбора.
```

Если $a > b$, то переменной `max` будет присвоено значение `a`, переменной `min` - значение `b`, в противном случае наоборот переменной `max` присваивается значение `b`, а переменной `min` - значение `a`. Рассмотренную конструкцию допускается записывать в одной строке, как в следующем примере 5.

Пример 5.

```
if (a > b)
{
```

```

    max = a;
    min = b;
}
else
{
    max = b;
    min = a;
}

```

Если в фигурных скобках записано по одному оператору, то скобки можно опустить, как в примере 6.

Пример 6.

```

if (a > b) max = a;
else max = b;

```

Вложенные конструкции оператора if

В том случае, когда определённый *блок операторов* (или один оператор) нужно выполнить после проверки ни одного, а нескольких условий, то используют несколько конструкций оператора if. Операторы if, находящиеся внутри другого оператора if, называются вложенными конструкциями оператора if.

```

if (Условие 1 выбора)
{
    // Если условие 1 выбора истинно будут выполня-
    // ться, записанные в скобках операторы блока 1.
}

```

```

else
// В противном случае будет выполняться
// вложенная конструкция оператора if.
{
    // Начало вложенной конструкции оператора if.
    if (условие 2 выбора)
    {
        // Если условие 2 выбора истинно будут
        // выполняться, записанные здесь в скобках
        // операторы блока 2.
    }
    Else
    {
        // В противном случае,
        // если условие 2 выбора ложно будут вы
        // полняться, записанные здесь в скобках
        // операторы блока 3.
    }
} // Конец вложенной конструкции оператора if

```

Если *условие 1 выбора* истинно, то выполняются *операторы блока 1*, и далее первый оператор, который следует за последней фигурной скобкой, концом вложенной конструкции оператора *if*. В противном случае выполняется вложенная конструкция оператора *if*. Если *условие 2 выбора* вложенного оператора *if* истинно, то выполняются записанные в фигурных скобках *операторы блока 2*, и далее первый оператор, который следует за последней фигурной скобкой, концом вложенной конструкции опе-

ратора `if`. В противном случае, если *условие 2 выбора* ложно выполняются *операторы блока 3*. Рассмотрим пример записи вложенной конструкции оператора `if`.

Пример 7.

```
. . . .
if (x < -1)
{ n = 1; }
else
{
    // Начало вложенной конструкции if.
    if (x > 1)
    { n = 2; }
    else
    { n = 0; }
    // Конец вложенной конструкции if.
}
. . . .
```

Допускаются и другие виды записи вложенной конструкции оператора `if`, например запись в одной строке.

Пример 8.

```
. . . .
if (x < -1)
{ n = 1; }
else
{
    if (x > 1) { n = 2; }
    else { n = 0; }
}
```



```
}
```

```
. . .
```

Поскольку фигурные, операторные скобки являются обязательными только в случае записи в них нескольких операторов, поэтому в данном случае они могут быть опущены.

Пример 9.

```
. . .
```

```
if (x < -1) n = 1;
```

```
else
```

```
if (x > 1) n = 2;
```

```
else n = 0;
```

```
. . .
```

Операторы логического сравнения

Эти операторы называются логическими сравнениями (logical comparisons), поскольку они возвращают результат сравнения в виде значения `true` (истина) или `false` (ложь) имеющие тип `bool`. Для записи и проверки условия равенства двух выражений, в алгоритмическом языке `C#` используется символ `==`. Аналогично: символ `>` используется для проверки условия «больше»; символ `<` для проверки условия «меньше»; `>=` – «больше или равно»; `<=` – «меньше или равно»; `!=` «не равно». Например: `a!=b`, означает, что оператор логического сравнения `!=` возвращает значение `true`, если `a` не равно `b`.

Логические операторы

Для переменных типа `bool` используются специальные составные логические операторы:

$\&$ – конъюнкция (логическое и, and), используется для логического объединения двух выражений;

$|$ – дизъюнкция (логическое или, or), используется, чтобы убедиться в том, что хотя бы одно из выражений true, истинно;

$!$ – отрицание (логическое не, not), возвращает обратное логическое выражение;

\wedge – исключение (логическое исключаящее или), используется для того, чтобы убедиться в том, что одно из двух выражений true, истинно.

Операторы $\&$, $|$ и \wedge обычно используются с целыми типами данных, а также могут применяться к типу данных bool.

Кроме того могут применяться операторы $\&\&$ и $||$, которые отличаются от своих односимвольных версий тем, что выполняют ускоренные вычисления. Например в выражении $a \&\& b$, b вычисляется лишь в том случае, если a равно true, истинно. В выражении $a || b$, b вычисляется в том случае, если a равно false, ложно.

Пример 10.

`if (x > -1 && x < 1)`

В условии оператора `if` записано обычное алгебраическое неравенство $-1 < x < 1$.

Пример 11.

`if (x < -1 || x > 1)`

В условии оператора `if` записаны алгебраические неравенства $x < -1$ либо $x > 1$.

Оператор проверки

Оператор проверки выбирает одно из двух выражений в зависимости от проверки значения логического условия. Его синтаксис:

Имя переменной = (*условие выбора*)?Значение1: значение2

Пример 12.

```
int value = (x < 25) ? 5: 15
```

В этом примере сначала вычисляется выражение $x < 25$ являющееся *условием выбора*. Если оно равно true, то переменной `value` будет присвоено значение равное 5, в противном случае – равное 15.

Оператор безусловного перехода goto

Оператор безусловного перехода `goto` (перейти к) осуществляет переход, без проверки каких-либо условий, к оператору, обозначенному соответствующей меткой. Синтаксис этого оператора выглядит следующим образом:

метка: *оператор*

. . .

`goto` *метка*

. . .

где *метка* - метка. Это любой допустимый идентификатор C#, который помещается слева от *оператора*, которому надо передать управление выполнением программы и отделяется от него двоеточием. Причём *метка* может ставиться у *оператора* расположенного как до оператора `goto`, так и после него. В случае если оператор `goto` используется самостоятельно, без каких либо конструкций, то первый оператор, следующий за оператором `goto`, дол-

жен иметь свою метку, иначе он не будет выполнен в процессе работы программы. Обычно оператор `goto` используется совместно с оператором условного перехода `if`, и используется в программах редко, т. к. есть более эффективные операторы.

Конструкция `switch` (переключатель)

Этот оператор позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программы. Несмотря на то, что это может быть организовано с помощью последовательной записи вложенных операторов `if`, во многих случаях более эффективным оказывается применение оператора `switch`. Ниже приведена общая форма оператора.

`switch` (*выражение*)

```
{  
    case константа 1:  
        последовательность операторов блока 1  
        break;  
    case константа 2:  
        последовательность операторов блока 2  
        break;  
    . . .  
    default  
        последовательность операторов блока n  
        break;  
}
```

Этот оператор работает следующим образом. Значение *выражения* последовательно сравнивается с *константами*. Как только

будет обнаружено совпадение, выполняется оператор или *последовательность операторов*, связанных с этим совпадением, до оператора break. Оператор break передаёт управление оператору, следующему за конструкцией switch. Если совпадений нет, то выполняется последовательность операторов, следующая после оператора default. Эта ветвь не является обязательной.

При использовании конструкции switch действуют следующие правила:

- *выражение* в конструкции switch должно быть целочисленного типа (char, byte, short или int) перечислимого типа или же типа строкового;
- нельзя использовать числа с плавающей точкой;
- *константы* оператора case должны иметь тот же тип, что и *выражение* в конструкции switch;
- в одном операторе switch не допускается наличие двух одинаковых по значению *констант*;
- допускается использовать одну и ту же последовательность операторов, в этом случае оператор *break* не записывается.

Пример 13.

```
int n;  
m1: Console.WriteLine("Введите целое число");  
int a = int.Parse(Console.ReadLine());  
switch (a)  
{  
    case 1:  
        n = 10;  
        break;
```

```

    case 2:
    case 3:
        n = 20;
        break;
    default:
        n = 0;
        break;
}
Console.WriteLine("a = " + a + "    n = " + n);
if (a != 0) goto m1;
Console.Read();

```

В данном примере в программу вводится и присваивается переменной **a**, любое целое число. С помощью конструкции **switch** происходит анализ. Если переменная **a** имеет значение равное 1, переменной **n** присваивается значение 10 и далее следует вывод этих переменных. Если **a** имеет значение равное 2 или 3, то переменной **n** присваивается значение 20 и далее вывод этих переменных. Во всех остальных случаях переменной **n** присваивается значение 0. Программа продолжает работать до тех пор, пока переменной **a**, не будет задано значение 0.

Один оператор **switch** может быть частью последовательности другого внешнего оператора **switch**. Такой оператор называется вложенным. Константы внешнего и внутреннего операторов **switch** могут содержать общие значения, не вызывая каких либо конфликтов.

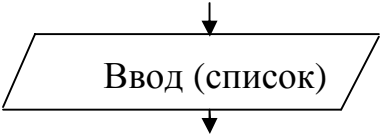
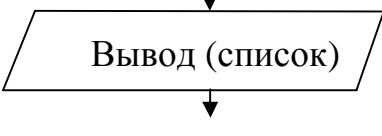
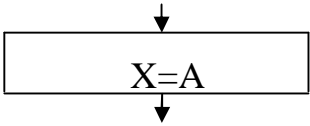
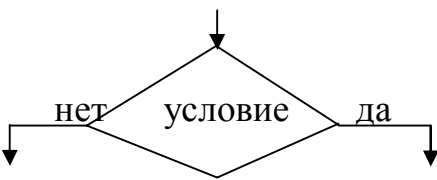
В операторе **switch** отсутствует возможность задания диапазона выбора, что является его недостатком. Например, в языке

программирования Visual Basic в аналогичном операторе задание диапазона выбора допускается.

Разветвляющиеся программы

Разветвляющиеся программы это – такие программы, в которых на определённых этапах происходит анализ значений тех или иных параметров и в зависимости от этого выбирается один из возможных вариантов дальнейшего хода программы. Практически все более или менее сложные программы являются разветвляющимися. Для их написания используются рассмотренные конструкции управляющих операторов принятия решения.

Таблица 3.1

№	Название блока	Графическое изображение блока	Операторы и функции эквивалентные блоку
1	Блок ввода		Операторы ввода, функция InputBox и другие
2	Блок вывода		Операторы вывода, функция MsgBox и другие
3	Блок присваивания		Оператор присваивания
4	Блоки сравнения		Условный оператор i f

При написании разветвляющих программ предварительно составляется блок-схема алгоритма решения задачи. Блок-схема это – графическое изображение алгоритма или последовательности решения задачи программирования.

Для составления блок-схем используются стандартизованные графические изображения (блоки) определённых операторов алгоритмического языка. Некоторые из них представлены в таблице 3.1.

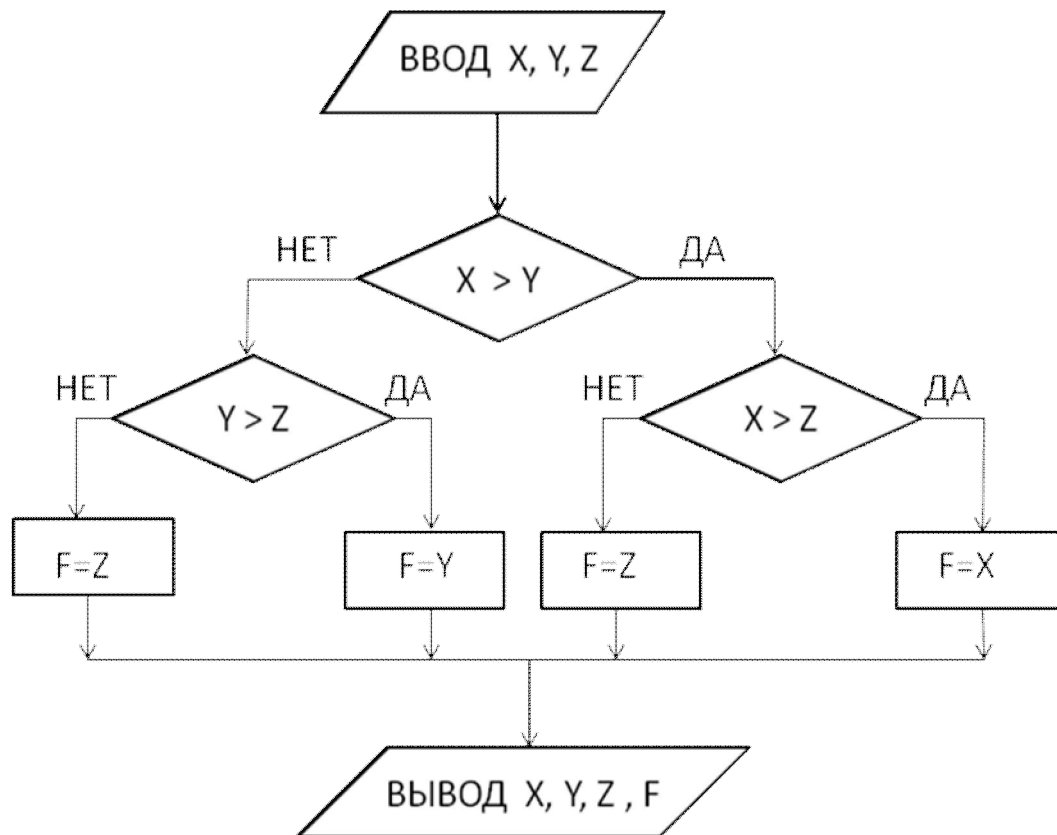
Далее рассмотрены примеры написания разветвляющихся программ, аналогичные тем, которые должен выполнить студент в данной лабораторной работе.

Пример 14.

Составить блок-схему и написать программу для определения наибольшей из трёх заданных величин X , Y и Z . Полученное значение присвоить переменной F , т. е. вычисляет $F = \max(X, Y, Z)$. Замечание: Данный пример является тренировочным, на практике подобные задачи решаются с помощью соответствующих встроенных функций.

Пояснения к блок-схеме. После ввода численных значений для переменных X , Y и Z производится их последовательное сравнение друг с другом на предмет выявления наибольшего из них. Первоначально сравниваются значения переменных X и Y . Если условие $X > Y$ выполняется (истинно), то далее переменная с наибольшим значением, а именно X сравнивается с Z . Если поставленное в блоке сравнения условие $X > Z$ верно, то переменной F будет присвоено значение переменной X в противном случае – значение переменной Z . Аналогично поступаем в случае если условие $X > Y$, не выполняется (ложно).

Блок-схема



После составления блок-схемы по ней пишется программа, при этом каждый блок описывается соответствующим оператором алгоритмического языка.

```
float f;
```

```
m1: Console.WriteLine(" Введите значение X");
```

```
float x = float.Parse(Console.ReadLine());
```

```
Console.WriteLine(" Введите значение Y");
```

```
float y = float.Parse(Console.ReadLine());
```

```
Console.WriteLine(" Введите значение Z");
```

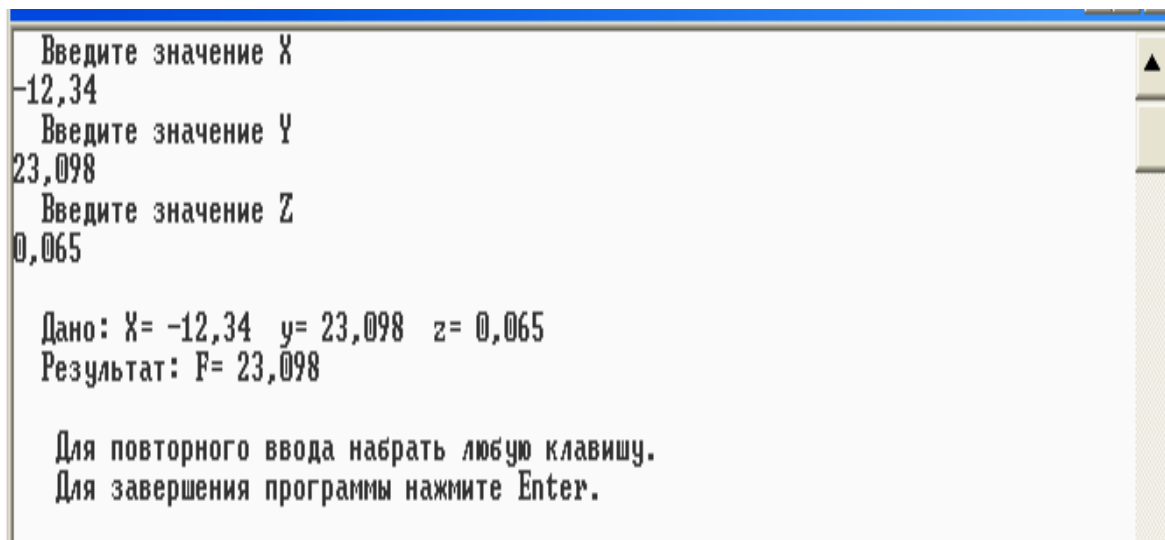
```
float z = float.Parse(Console.ReadLine());
```

```
if (x > y)
```

```

{
    if (x > z) f=x;
    else f=z;
}
else
{ if(y>z) f=y; else f=z; }
Console.WriteLine('\n' + " Дано: X= " +
x + " y= " + y + " z= " + z +
'\n' + " Результат: F= " + f);
Console.WriteLine('\n' + "Для повторного
ввода" + "набрать любую клавишу." +
'\n' + "Для завершения программы нажмите
Enter.");
string p = Console.ReadLine();
if (p != "") goto m1;

```



```

Введите значение X
-12,34
Введите значение Y
23,098
Введите значение Z
0,065

Дано: X= -12,34 y= 23,098 z= 0,065
Результат: F= 23,098

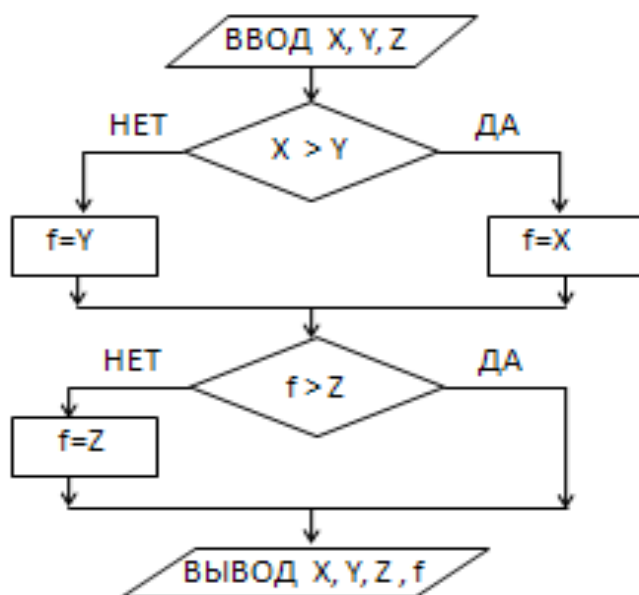
Для повторного ввода набрать любую клавишу.
Для завершения программы нажмите Enter.

```

Рассмотренный алгоритм решения задачи примера 14 не является единственным. Ниже представлена блок-схема другого вари-

анта алгоритма и основной фрагмент программы, с использованием оператора проверки

Блок-схема



```

. . .
f = (x > y) ? x : y;
f = (f > z) ? f : z;
. . .

```

Основной фрагмент программы

Пример 15.

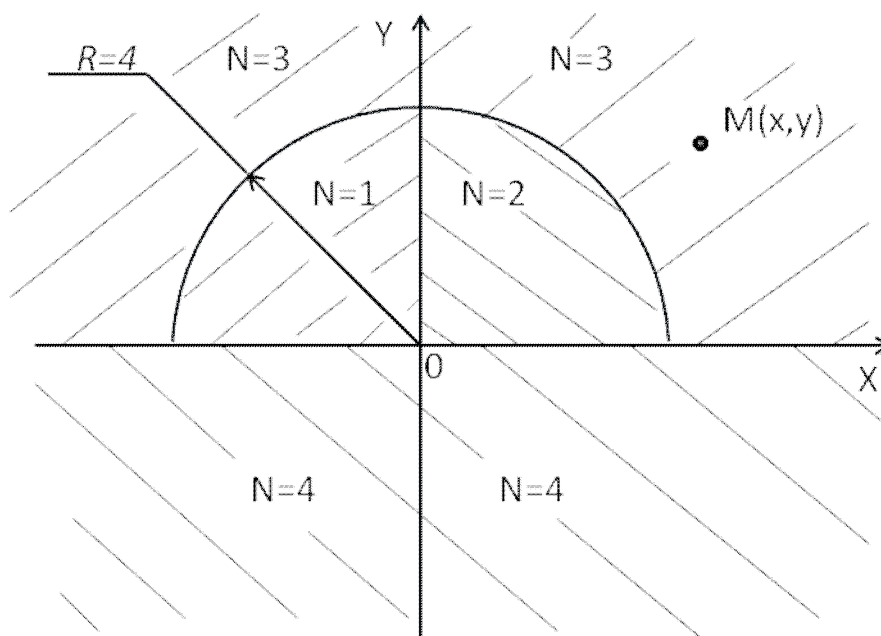
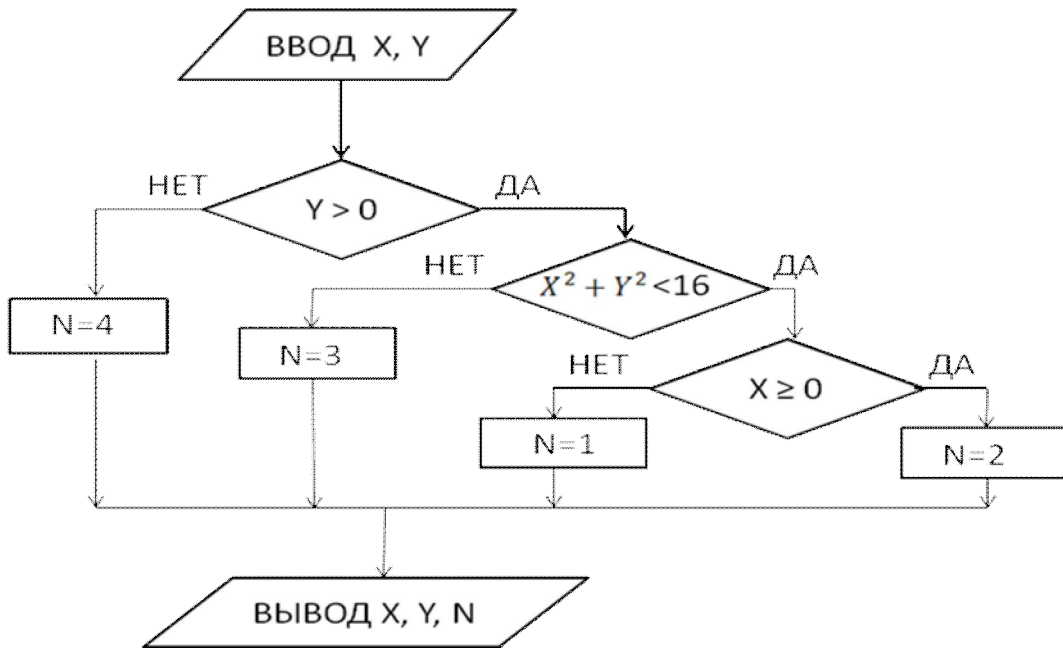


Рис. 3.1

Задание. Составить блок-схему и написать программу, которая определяет номер N области, в которой находится точка $M(x, y)$ с заданными координатами (см. рисунок 3.1). Границы области относить к области с наибольшим номером.

Блок-схема



Пояснения к блок-схеме. В первом блоке производится ввод численных значений для переменных X и Y , которые являются координатами точки M . Далее целесообразно сравнить переменную Y (координата по оси Y) с нулём. В блок-схеме это первый блок сравнения, если его условие $Y > 0$ не выполняется (ложно), то координата по оси Y точки M отрицательна или равна нулю, а это значит, что она расположена ниже оси X или на ней, т. е. в области с номером $N = 4$. Если условие $Y > 0$ первого блока сравнения выполняется (истинно), то точка M расположена выше оси X , а это значит, что она может находиться в одной из областей с номером $N = 1$, $N = 2$ или $N = 3$. Далее для определения номера области

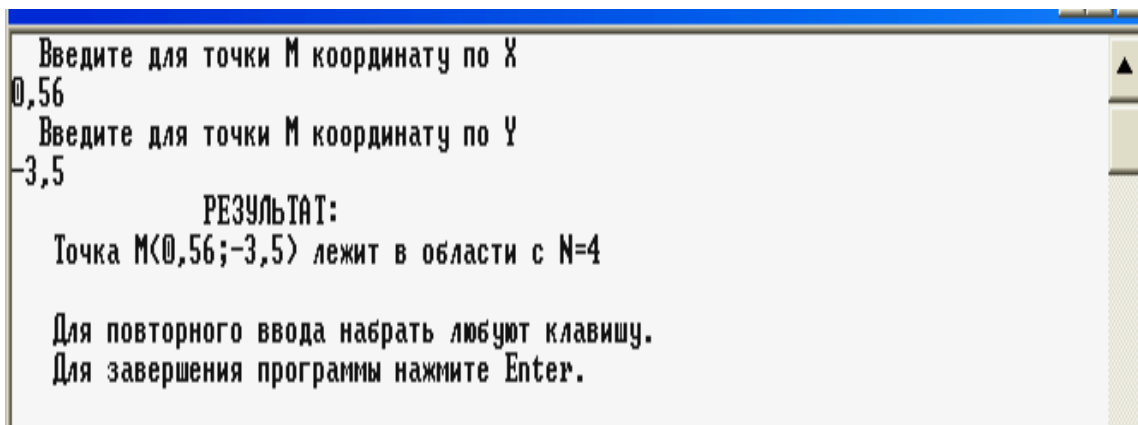
целесообразно задать во втором блоке сравнения условие $X^2 + Y^2 < 16$, которое следует из уравнения окружности $X^2 + Y^2 = R^2$, где радиус окружности. Если заданное условие выполняется, то точка расположена внутри окружности, а так как $Y > 0$, то внутри полуокружности. Согласно условию задачи внутри полуокружности точка может находиться либо в области с номером $N = 1$, либо в области с $N = 2$. Если условие $X \geq 0$ третьего блока сравнения выполняется (истинно), то точка расположена в области с $N = 2$, в противном случае с $N = 1$. После чего идёт печать результата. Если условие $X^2 + Y^2 < 16$ второго блока сравнения не выполняется (ложно), то точка M находится вне полуокружности и над осью X так как $Y > 0$ т. е. в области $N = 3$. Далее представлена программа, составленная по рассмотренной блок-схеме.

```
int N;
m2: Console.WriteLine(" Введите для точки M"+
    " координату по X ");
float x = float.Parse((Console.ReadLine()));
Console.WriteLine(" Введите для точки M" +
    " координату по Y ");
float y = float.Parse((Console.ReadLine()));
if (y > 0)
{
    if (x * x + y * y < 16)
    {
        if (x >= 0) N = 2;
        else N = 1;
    }
}
```

```

else
    { N = 3; }
}
else
    { N = 4; }
Console.WriteLine('\t' + "      РЕЗУЛЬТАТ:");
Console.WriteLine("    Точка М(" + x + "; " + y + " )"
+ " лежит в области с N=" + N);
Console.WriteLine('\n' + "    Для повторного вво
да" + " нажать любую клавишу. " + '\n' +
"    Для завершения программы нажмите Enter.");
string p = Console.ReadLine();
if (p != "") goto m2;

```



Результаты расчёта по программе примера 15

2. Практическая часть

Задания к лабораторной работе

Составить блок-схему и написать программу для выполнения следующих заданий. При этом руководствоваться выше приведёнными примерами выполнения заданий (см. примеры 14 и 15)

Задание 1. Вычислить для своего варианта значение функции F . При получении в знаменателе нуля дать соответствующее сообщение.

Варианты заданий

$$1) \quad F = \frac{\min(x, y) + 0,5}{(\max(x, y))^2 - \sin z}$$

$$2) \quad F = \frac{\min(x, y, z) + x}{(\max(x, z))^2 + y}$$

$$3) \quad F = \frac{\max(x, y) + y}{(\min(x, y, z))^2 + yx}$$

$$4) \quad F = \frac{\min(z, \max(x, y))}{x^2 + z}$$

$$5) \quad F = \frac{\max(x^2, y^2, xz) + x}{(\min(x, y))^2 - y}$$

$$6) \quad F = \frac{\min(x, y + z)}{\max(x^2, y) + z^3}$$

$$7) \quad F = \frac{\max(x^2, y^2, x - y) + x}{(\min(x, y))^2 + y^4}$$

$$8) \quad F = \frac{\min(x, (x + y)^2)}{x^2 + \max(y^3, x)}$$

$$9) \quad F = \frac{\max(x + z, \min(x, y))}{x^2}$$

$$10) \quad F = \frac{\min(x, \max(x + y, z))^2}{x^2 + z^2}$$

$$11) \quad F = \frac{\min(x, y + z)}{\max(x, y) + \sin z}$$

$$12) \quad F = \frac{\min(x, y - x)}{\max(yz, x^2) + \cos 2z^3}$$

$$13) \quad F = \frac{\max(x^2, z^2) + \cos 2x^2}{(\min(x, y))^2 - y}$$

$$14) \quad F = \frac{\min(x^2, y + z)}{x^2 + \max(z^3, xy)}$$

$$15) \quad F = \frac{\max(x + y, \max(x, zy))}{xe^2}$$

$$16) \quad F = \frac{\max(x^3, y^2, xy) + x}{(\min(x, yz))^2 - y}$$

$$17) \quad F = \frac{x(\max(x + z, zy))}{\min(x, y) + x^2} 1$$

$$18) \quad F = \frac{\min(x, \max(x + z, y))^2}{x^3 + z^2}$$

$$19) \quad F = \frac{x^3 + \max(z^2, y)}{(\max(x, z))^2 - y}$$

$$20) \quad F = \frac{\max(x, y + z) + e^{xz}}{\min(x^2, y) + z^3}$$

$$21) F = \frac{\min(x^2, z^4, xy) + x}{(\max(x, y))^2 - y}$$

$$22) F = \frac{\min(x, y+z) + e^x}{\max(x^2, y) + z^3}$$

$$23) F = \frac{\max(x^2, y^2) + \cos 4z^2}{\min(x, y) + x^2}$$

$$24) F = \frac{(\min(x, y))^2 - y}{x^2 + \max(z^3, x)}$$

$$25) F = \frac{\min(x, y+2x)}{\max(y, z) + \sqrt[3]{x}}$$

$$26) F = \frac{\min(x, y-z)}{\max(yz, x^2) + \cos 2x^2}$$

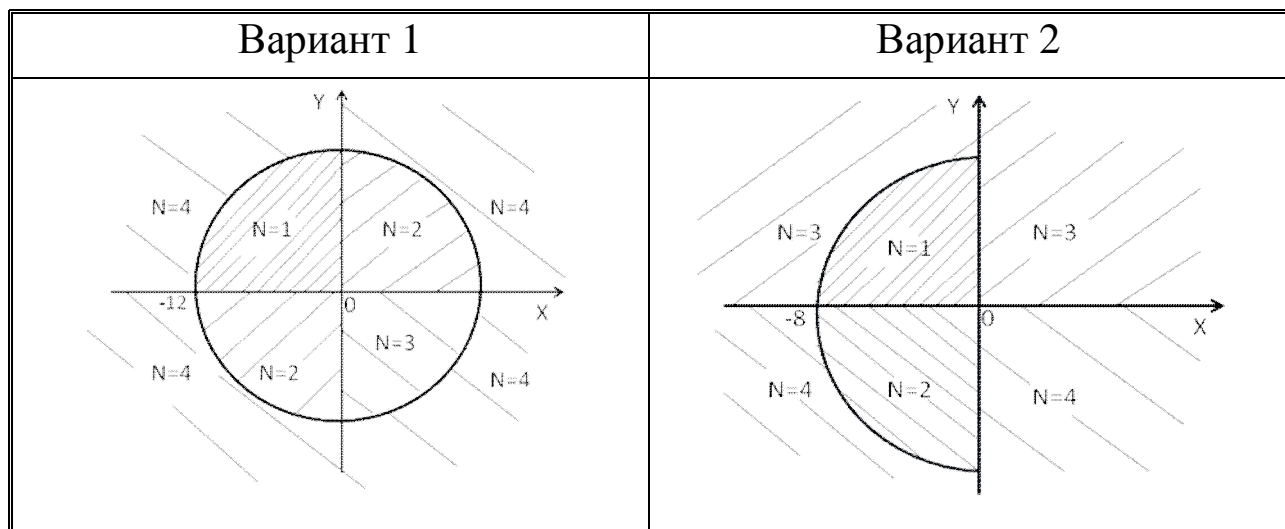
$$27) F = \frac{\max(x^3, z^2) + \cos 4y^2}{\min(y, yz) + \sqrt{x}}$$

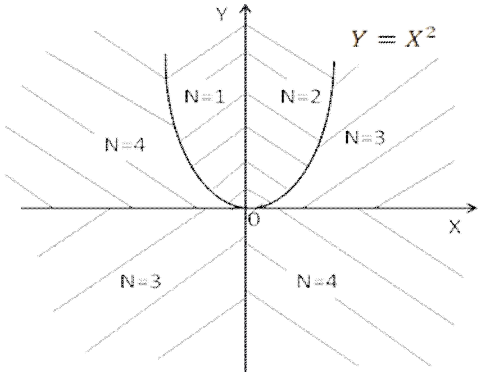
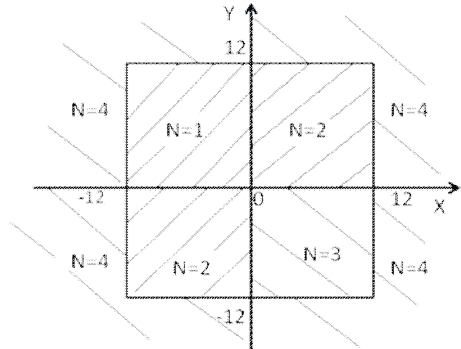
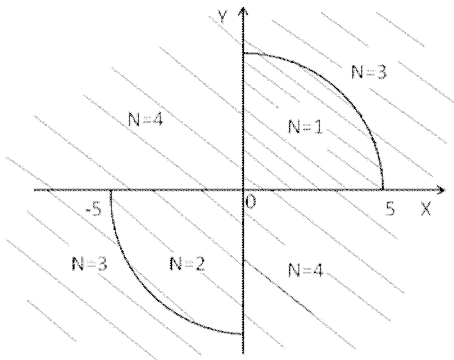
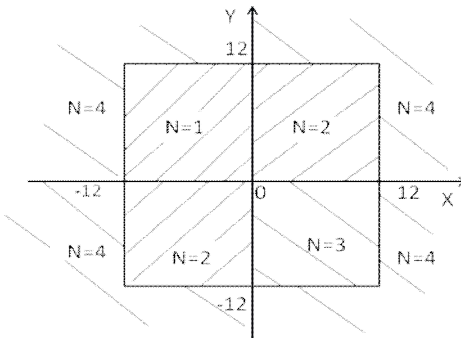
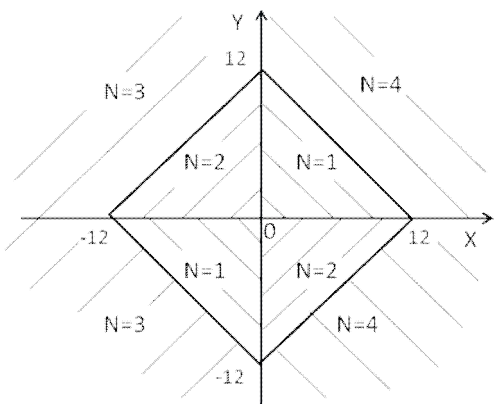
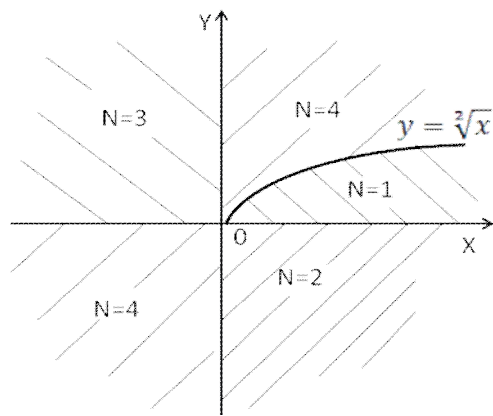
$$28) F = \frac{(\min(x, y))^4 + 2e^x}{x^3 + \max(z^2, x)}$$

$$29) F = \frac{\max(xz, \min(y, z))}{x^2 + \sin zy}$$

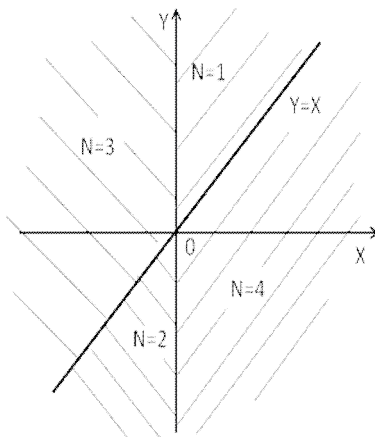
$$30) F = \frac{\max(x, \max(y, z))^4}{\sin 2y + xe^2}$$

Задание 2. Определить для своего варианта номер N области, в которой находится точка M(x,y) с заданными координатами. Границы области относить к области с наибольшим номером.

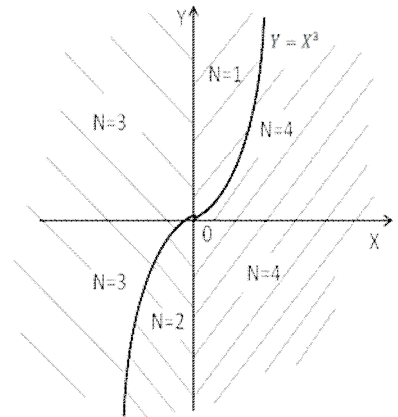


<p style="text-align: center;">Вариант 3</p> 	<p style="text-align: center;">Вариант 4</p> 
<p style="text-align: center;">Вариант 5</p> 	<p style="text-align: center;">Вариант 6</p> 
<p style="text-align: center;">Вариант 7</p> 	<p style="text-align: center;">Вариант 8</p> 

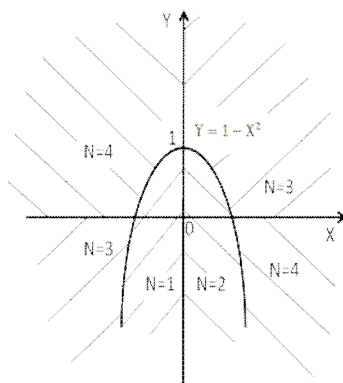
Вариант 9



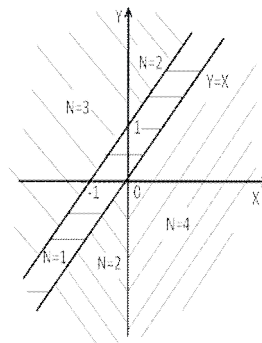
Вариант 10



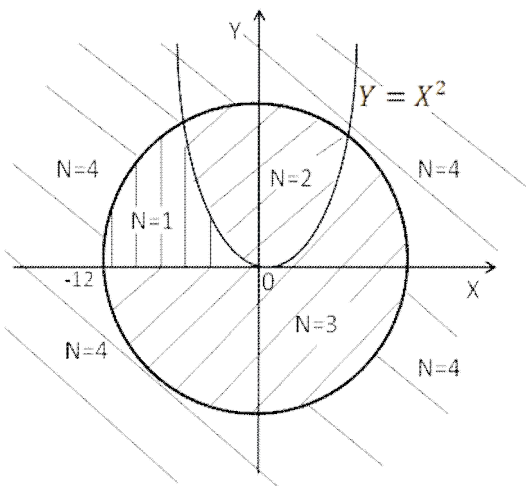
Вариант 11



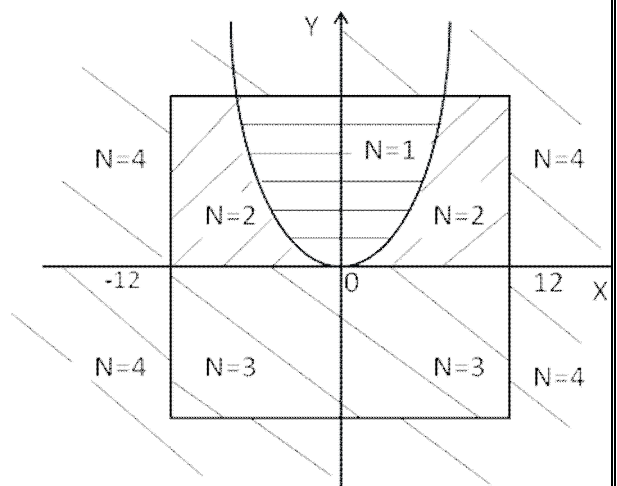
Вариант 12



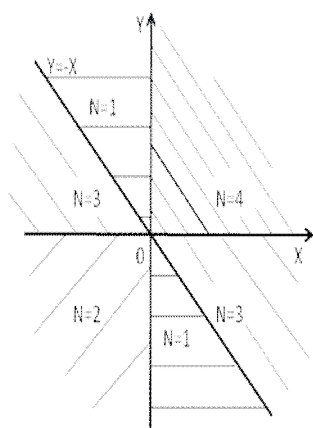
Вариант 13



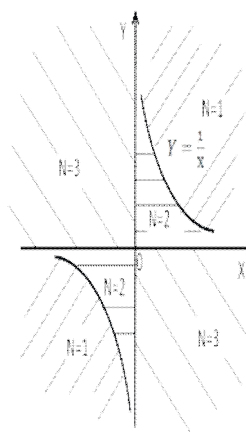
Вариант 14



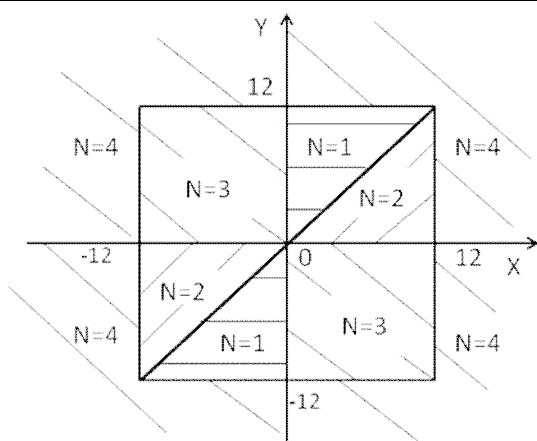
Вариант 15



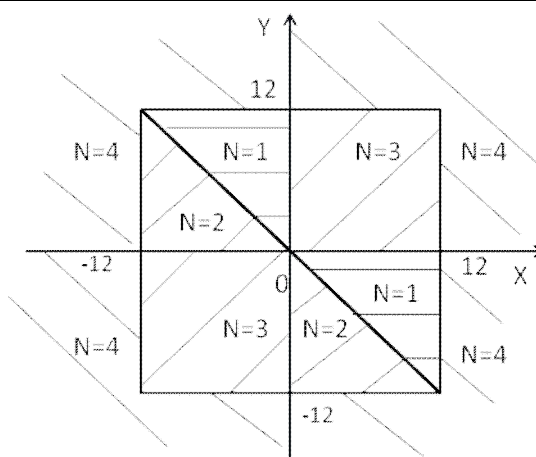
Вариант 16



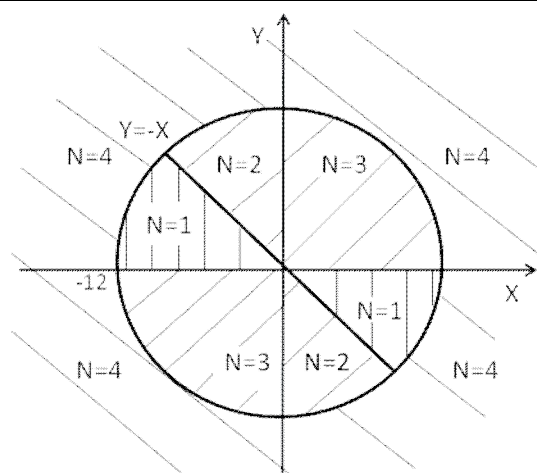
Вариант 17



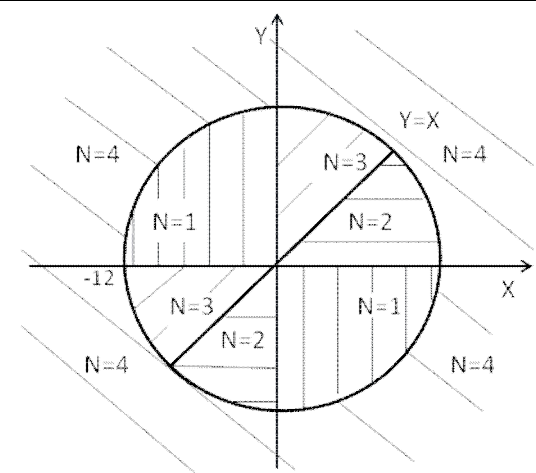
Вариант 18



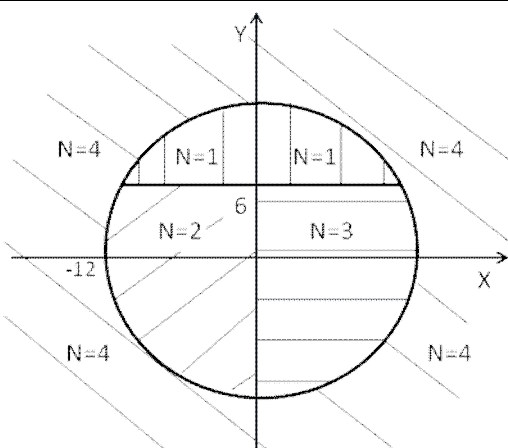
Вариант 19



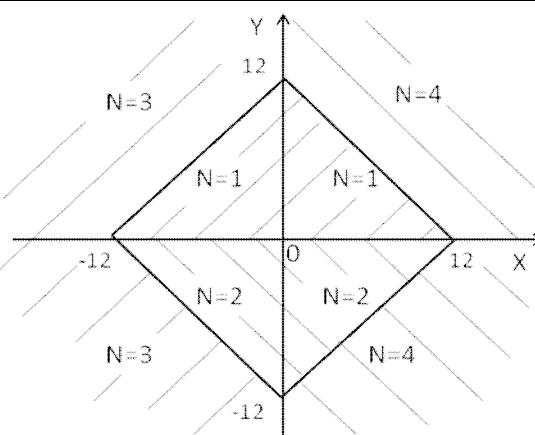
Вариант 20



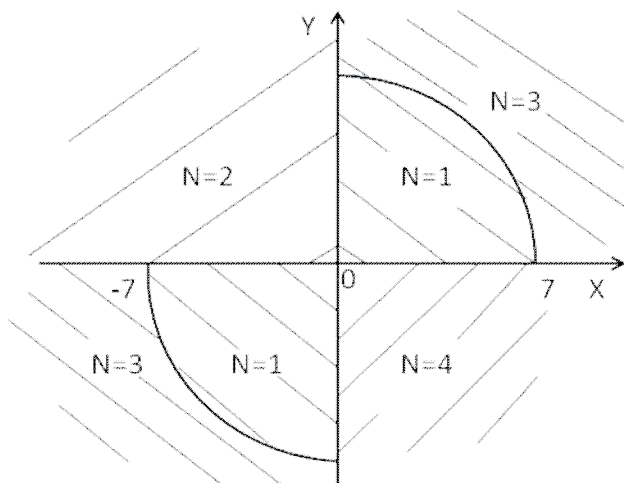
Вариант 21



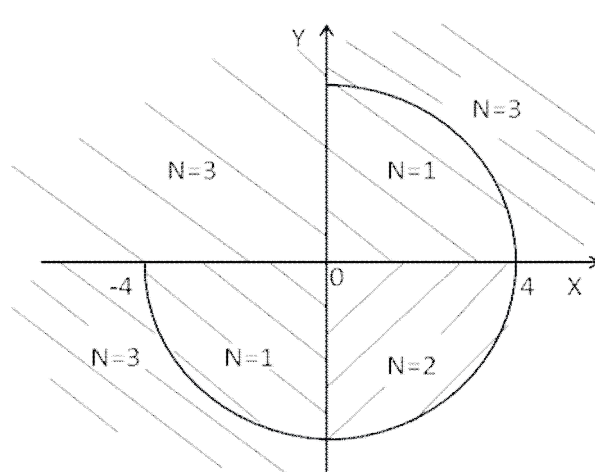
Вариант 22



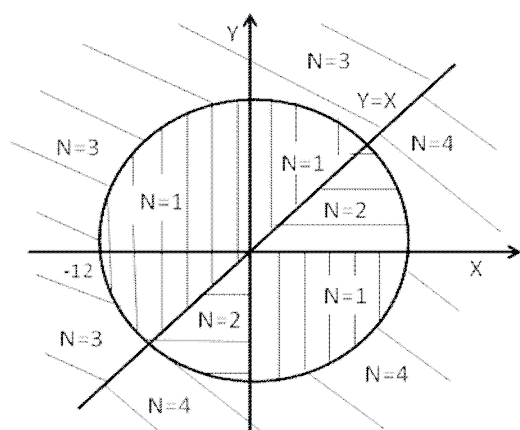
Вариант 23



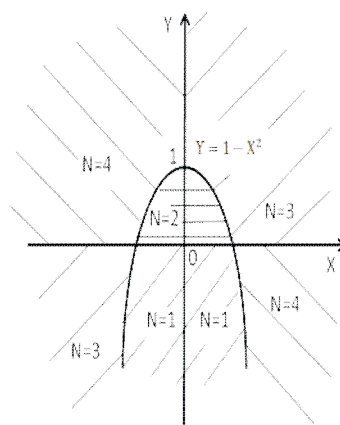
Вариант 24



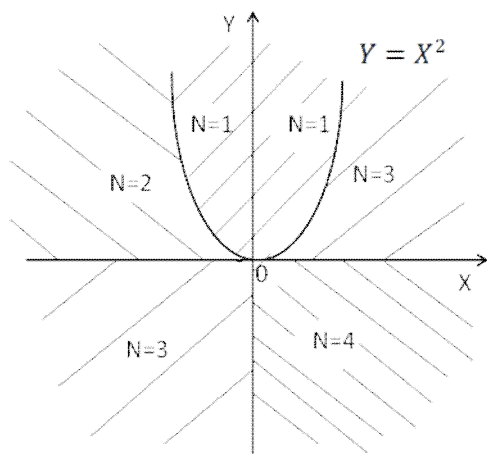
Вариант 25



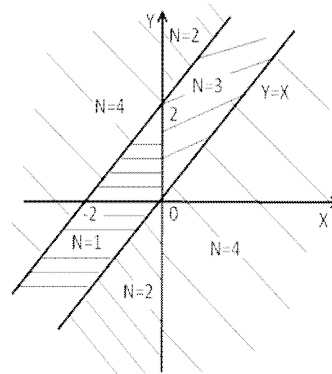
Вариант 26



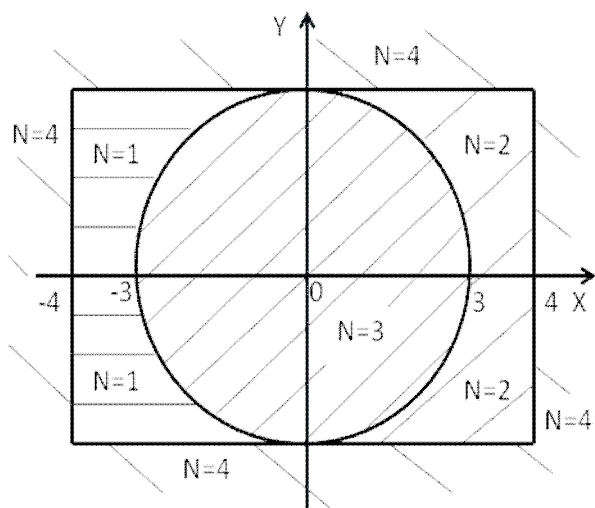
Вариант 27



Вариант 28



Вариант 29



Вариант 30

