ОПЕРАТОРЫ ЯЗЫКА C#

Программа на языке C# состоит из последовательности операторов, каждый из которых определяет законченное описание некоторого действия и заканчивается точкой с запятой. Все операторы можно разделить на 4 группы: операторы следования, операторы ветвления, операторы цикла и операторы передачи управления.

Операторы следования

Операторы следования выполняются компилятором в естественном порядке: начиная с первого до последнего. К операторам следования относятся: выражение и составной оператор.

Любое *выражение*, завершающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается вычислением значения выражения или выполнением законченного действия, например, вызовом метода. Например:

```
++i;          //оператор инкремента
x += y;       //оператор сложения с присваиванием
Console.WriteLine(x);    //вызов метода
x = Math.Pow(a,b)+a*b;   //вычисление сложного выражения
```

Составной оператор, или *блок* представляет собой последовательность операторов, заключенных в фигурные скобки {}. Блок обладает собственной *областью видимости*: объявленные внутри блока имена доступны только внутри данного блока или блоков, вложенных в него. Составные операторы применяются в случае, когда правила языка предусматривают наличие только одного оператора, а логика программы требует нескольких. Например, тело цикла *while* должно состоять только из одного оператора. Если заключить несколько операторов в фигурные скобки, то получится блок, который будет рассматриваться компилятором как единый оператор.

Операторы ветвления

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор *if* и оператор выбора *switch.*

Условный оператор if

Условный оператор *if* используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: *сокращенную* или *полную*.

Форма сокращенного оператора if:

```
if (B) S;
```

где *B* – логическое выражение, истинность которого проверяется; *S* – простой, или составной оператор.

При выполнении сокращенной формы оператора *if* сначала вычисляется выражение *B*, а затем проводится анализ его результата: если *B* истинно, то выполняется оператор *S*; если *B* ложно, то оператор *S* пропускается. Таким образом, с помощью сокращенной формы оператора *if* можно либо выполнить оператор S, либо пропустить его.

Форма полного оператора if:

```
if (B) S1; else S2;
```

где *B* – логическое выражение, истинность которого проверяется; *S1*, *S2* – простые, или составные операторы.

При выполнении полной формы оператора *if* сначала вычисляется значение выражения *B*, затем анализируется его результат: если *B* истинно, то выполняется оператор *S1*, а оператор *S2* пропускается; если *B* ложно, то выполняется оператор *S2*, а *S1* – пропускается. Таким образом, с помощью полной формы оператора *if* можно выбрать одно из двух альтернативных действий процесса обработки данных.

Рассмотрим несколько примеров записи условного оператора *if*

Сокращенная форма оператора if

```
// с простым оператором
if (a > 0)
   x = y;
```

```
// с составным оператором
if (i > 0)
{
   x = y;
   y = 2*z;
}
```

Полная форма оператора if

```
// с простым оператором
if (a > 0 || b < 0)
   x = y;
else
   x = z;
```

```
// с составными операторами
if (i != j-1)
{
   x = 0;
   y = 1;
}
else
{
   x = 1;
   y = 0;
}
```

Операторы S1 и S2 могут также являться операторами *if*. Такие операторы называют вложенными. При этом ключевое слово *else* связывается с ближайшим предыдущим словом *if*, которое еще не связано ни с одним *else*. Рассмотрим несколько примеров алгоритмов с использованием вложенных условных операторов:

Пример 1

```
if (A>B)
if (C>D) X=Y;
else X=Z;
else X=R;
```

2 1

Пример 2

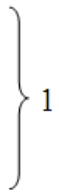
```
if (A>B) X=Y;
else if (C>D) X=Z;
     else X=R;
```

2 1

ERAM Systems

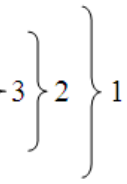
Пример 3

```
if (A < B)
   if (C < D) X =Y;
   else  X = Z;
else
   if (E < F) X= R;
   else  X = Q;
```



Пример 4

```
if (A< B)
   if (C < D)
      if (E < F) X= Q;
      else X = R;
   else X= Z;
else X = Y;
```



Замечание

Т.к. оператор if допускает наличие одного оператора действия, то можно записывать его без использования блока {}, например, так:

```
if (x < 0) y = 0; else y=1;
```

Однако предпочтительнее и в этом случае ставить блок, начиная его с новой строки. Это улучшает читабельность программы и значительно сокращает вероятность ошибки при внесении последующих изменений. В нашем случае оператор if следует записать так:

```
if (x < 0)
{
   y = 0;
}
else
{
   y = 1;
}
```

Далее будем придерживаться данного правила не только для оператора if, но и для других операторов.

Рассмотрим несколько примеров использования оператора if.

Пример 1

Найдем наибольшее значение из двух вещественных чисел.

```
static void Main()
{
   Console.Write("x= ");
   double x = double.Parse(Console.ReadLine());
   Console.Write("y=");
   double y = double.Parse(Console.ReadLine());
   double max;
   if (x > y )
   {
      max = x;
   }
```

EPAM Systems

```
      else
      {
         max = y;
      }
      Console.WriteLine("max= {0}", max);
   }
```

Результат работы программы:

```
x  y  max
0  0   0
1  -1  1
-2 2   2
```

Задание 1

Объясните, почему в данном примере не требуется инициализация переменной max.

Задание 2

Измените программу так, чтобы вычислялось наименьшее значение из двух вещественных чисел.

Замечание

В реальных программах разработчику нет необходимости писать подобные конструкции самому. Вычислить максимум из двух чисел можно с помощью метода Math.Max(x,y).

Пример 2

Найдем наибольшее значение из трех вещественных чисел.

```
   static void Main()
   {
      Console.Write("x= ");
      double x = double.Parse(Console.ReadLine());
      Console.Write("y=");
      double y = double.Parse(Console.ReadLine());
      Console.Write("z=");
      double z = double.Parse(Console.ReadLine());
      double max;
      if (x > y && x > z)
      {
         max = x;
      }
      else
      {
         if (y > z)
         {
            max = y;
         }
         else
         {
            max = z;
         }
      }
      Console.WriteLine("max= {0}", max);
   }
```

ERA Systems

Результат работы программы:

```
x   y   z   max
0   0   0   0
1   -1  2   2
-2  12  2   12
4   -1  -3  4
```

Задание

Измените программу так, чтобы вычислялось наименьшее значение из трех вещественных чисел.

Оператор выбора switch

Оператор выбора *switch* предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```
switch (<выражение>)
{
   case <константное_выражение_1>:
      [<оператор 1>]; <оператор перехода>;
   case <константное_выражение_2>:
      [<оператор 2>]; <оператор перехода>;
   ...
   case <константное_выражение_n>:
      [<оператор n>]; <оператор перехода>;
   [default: <оператор>; <оператор перехода>;]
}
```

Замечание

Операторы, записанные в квадратных скобках, являются необязательными элементами в операторе switch. Если они отсутствуют, то могут отсутствовать и соответствующие им операторы перехода.

Выражение, стоящее за ключевым словом *switch*, должно иметь арифметический, символьный, строковый тип, или тип указатель. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящего внутри скобок *switch*, или приводиться к нему. Ключевое слово *case* и расположенное после него константное выражение называют также меткой *case*.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом *switch*. Полученный результат сравнивается с меткой *case*. Если результат выражения соответствует метке *case*, то выполняется оператор, стоящий после этой метки, за которым *обязательно* должен следовать оператор перехода: *break*, *goto* и т.д. В случае отсутствия оператора перехода компилятор выдаст сообщении об ошибке. При использовании оператора *break* происходит выход из *switch* и управление передается оператору, следующему за *switch*. Если же используется оператор *goto*, то управление передается оператору, помеченному меткой, стоящей после *goto*.

Замечание 1

Оператор перехода goto лучше использовать для перехода по меткам внутри switch, и не использовать его для выхода из оператора switch.

Замечание 2

Для повышения производительности рекомендуется размещать ветви, вероятность выбора которых является наибольшей, ближе к началу. В том случае будет тратиться меньше времени на выбор требуемого варианта.

Пример

По заданному виду арифметической операции (сложение, вычитание, умножение и деление) и двум операндам, вывести на экран результат применения данной операции к операндам.

```
static void Main()
{
   Console.Write("OPER= ");
   char oper = char.Parse(Console.ReadLine());
   bool ok=true;
   Console.Write("A= ");
   double a = double.Parse(Console.ReadLine());
   Console.Write("B= ");
   double b = double.Parse(Console.ReadLine());
   double res = 0;
   switch (oper)
   {
      case '+' : res = a + b; break;    //1
      case '-' : res = a - b; break;
      case '*' : res = a * b; break;
      case ':' : if (b != 0)            //2
                 {
                    res = a / b;
                    break;
                 }
                 else
                 {
                    goto default;
                 }
      default: ok = false; break;
   }
   if (ok)
   {
      Console.WriteLine("{0} {1} {2} = {3}", a, oper, b, res);
   }
   else
   {
      Console.WriteLine("error");
   }
}
```

Результат выполнения программы:

```
oper   x   y   rez
  +    4   5    9
  :    4   0   error
  %    4   3   error
```

EPAM Systems

Задание 1

Замените в строке 1 оператор break, *на оператор* goto case '-' *и посмотрите, что произойдет, если в качестве операции ввести* +.

Задание 2

В условном операторе if *(см. строку 2) уберите ветку* else *и посмотрите, что произойдет в случае ввода* b = 0. *Дайте этому объяснение*.

Если необходимо, чтобы для разных меток выполнялось одно и то же действие, то метки перечисляются через двоеточие. Например:

```
switch (oper)
{
   case '+': res = a + b; break;
   case '-': res = a - b; break;
   case '*': res = a * b; break;
   case ':':
   case '/':  //перечисление меток
              if (b != 0)
              {
                 res = a / b;
                 break;
              }
              else
              {
                 goto default;
              }
   default: ok = false; break;
}
```

Операторы цикла

Операторы цикла используются для организации многократно повторяющихся вычислений. К операторам цикла относятся: цикл с предусловием *while*, цикл с постусловием *do while*, цикл с параметром *for* и цикл перебора *foreach.*

Цикл с предусловием while

Оператор цикла *while* организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Формат цикла *while*:

```
while (B) S;
```

где *B* – выражение, истинность которого проверяется (условие завершения цикла); *S* – тело цикла (простой или составной оператор).

Перед каждым выполнением тела цикла анализируется значение выражения *B*: если оно истинно, то выполняется тело цикла, и управление передается на повторную проверку условия *B*; если значение *B* ложно, цикл завершается и управление передается на оператор, следующий за оператором *S*.

Если результат выражения *B* окажется ложным при первой проверке, то тело цикла не выполнится ни разу. Отметим, что если внутри цикла не будет оператора (или операторов), в результате выполнения которых условие *B* на какой-то итерации станет ложным, то произойдет *зацикливание*, то есть невозможность выхода из цикла. Поэтому внутри тела

ЕРАМ Systems

должны находиться операторы, приводящие к изменению значения выражения *B* так, чтобы цикл мог корректно завершиться.

В качестве иллюстрации выполнения цикла *while* рассмотрим программу вывода на экран целых чисел из интервала от 1 до *n*.

```
static void Main()
{
   Console.Write("N= ");
   int n = int.Parse(Console.ReadLine());
   int i = 1;
   while (i <= n)           //пока i меньше или равно n
   {
      //выводим i на экран, затем увеличиваем его на 1
      Console.Write(" {0}", i++ );
   }
}
```

Результаты работы программы:

```
n                  ответ
10      1  2  3  4  5  6  7  8  9  10
```

Задание 1

Измените программу так, чтобы числа выводились в обратном порядке.

Задание 2

Измените программу так, чтобы выводились только нечетные числа.

Цикл с постусловием do while

Оператор цикла *do while* также организует выполнение одного оператора (простого или составного) неизвестное заранее число раз. Однако, в отличие от цикла *while*, условие завершения цикла проверяется после выполнения тела цикла. Формат цикла *do while*:

```
do S while (B);
```

где *B* – выражение, истинность которого проверяется (условие завершения цикла); *S* – тело цикла (простой или составной оператор).

Сначала выполняется оператор *S*, а затем анализируется значение выражения *B*: если оно истинно, то управление передается оператору *S*, если ложно, цикл завершается, и управление передается на оператор, следующий за условием *B*. Так как условие *B* проверяется после выполнения тела цикла, то в любом случае тело цикла выполнится хотя бы один раз.

В операторе *do while*, так же как и в операторе *while*, возможна ситуация *зацикливания* в случае, если условие *B* всегда будет оставаться истинным.

В качестве иллюстрации выполнения цикла *do while* рассмотрим программу вывода на экран целых чисел из интервала от 1 до *n*.

```
static void Main()
{
   Console.Write("N= ");
   int n = int.Parse(Console.ReadLine());
   int i = 1;
```

```
      do
         //выводим i на экран, затем увеличиваем его на 1
         Console.Write(" {0}", i++);
      while (i <= n);           //пока i меньше или равно n
   }
```

Задание 1

Измените программу так, чтобы числа выводились в обратном порядке.

Задание 2

Измените программу так, чтобы выводились только четные числа.

Цикл с параметром for

Цикл с параметром имеет следующую структуру:

```
for ( <инициализация>; <выражение>; <модификация>) <оператор>;
```

Инициализация используется для объявления и/или присвоения начальных значений величинам, используемым в цикле в качестве параметров (счетчиков). В этой части можно записать несколько операторов, разделенных запятыми. Областью действия переменных, объявленных в части инициализации цикла, является цикл и вложенные блоки. Инициализация выполняется один раз в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результатом является истина, цикл выполняется. Истинность выражения проверяется перед каждым выполнением тела цикла. Таким образом, цикл с параметром реализован как цикл с предусловием.

Модификация выполняется после каждой итерации цикла и служит обычно для изменения параметров цикла. В части <модификация> можно записать несколько операторов через запятую.

Оператор (простой или составной) представляет собой тело цикла.

Любая из частей оператора *for* (инициализация, выражение, модификация, оператор) может отсутствовать, но точку с запятой, определяющую позицию пропускаемой части, надо оставить.

```
static void Main()
{
   Console.Write("N= ");
   int n = int.Parse(Console.ReadLine());
   for (int i=1; i<=n; i++)
   {
      Console.Write(" {0} ,", i);
   }
}
```

Задание 1

Измените программу так, чтобы решалась поставленная задача, а блок модификации оказался пустым.

Задание 2

Измените программу так, чтобы числа выводились в обратном порядке.

Задание 3

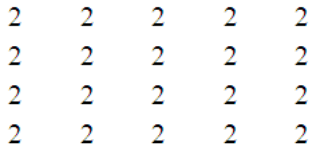
Измените программу так, чтобы выводились квадраты чисел.

Цикл перебора foreach предназначен для работы с наборами данных и будет рассмотрен позже.

Вложенные циклы

Циклы могут быть простые или вложенные (кратные, циклы в цикле). Вложенными могут быть циклы любых типов: *while, do while, for*. Каждый внутренний цикл должен быть полностью вложен во все внешние циклы. «Пересечения» циклов не допускаются.

Рассмотрим пример использования вложенных циклов, который позволит вывести на экран числа следующим образом:




```
static void Main()
{
   for (int i = 1; i <= 4; i++)
   {
      for (int j = 1; j <= 5; j++)
      {
         Console.Write(”2 ”);
      }
      Console.WreLine();
   }
}
```

Задание

Измените программу так, чтобы таблица содержала n строк и m столбцов (значения n и m вводятся с клавиатуры).

Операторы безусловного перехода

В C# есть несколько операторов, изменяющих естественный порядок выполнения команд: оператор безусловного перехода *goto*, оператор выхода *break*, оператор перехода к следующей итерации цикла *continue*, оператор возврата из метода *return* и оператор генерации исключения *throw*.

Оператор безусловного перехода goto

Оператор безусловного перехода *goto* имеет формат:

```
goto <метка>;
```

В теле того же метода должна присутствовать ровно одна конструкция вида:

```
<метка>: <оператор>;
```

Оператор *goto* передает управление оператору с меткой. Рассмотрим пример использования оператора goto:

EPAM Systems

```
static void Main()
{
   float x;
label: Console.WriteLine("x="); //оператор, помеченный меткой
   x = float.Parse(Console.ReadLine());
   if (x!=0)
   {
      Console.WriteLine("y({0})={1}", x, 1 / x );
   }
   else
   {
      Console.WriteLine("функция не определена");
      goto label;    // передача управление метке
   }
}
```

Следует учитывать, что использование оператора *goto* затрудняет чтение больших по объему программ, поэтому использовать метки нужно только в крайних случаях.

Оператор выхода break

Оператор *break* используется внутри операторов цикла и оператора выбора для обеспечения перехода в точку программы, находящуюся непосредственно за оператором, внутри которого находится *break*.

Мы уже применяли оператор *break* для выхода из оператора *switch*. Аналогичным образом он может применяться для выхода из операторов цикла.

Оператор перехода к следующей итерации цикла continue

Оператор перехода к следующей итерации цикла *continue* пропускает все операторы, оставшиеся до конца тела цикла, и передает управление на начало следующей итерации (повторение тела цикла). Рассмотрим оператор *continue* на примере.

```
static void Main()
{
   Console.WriteLine("n=");
   int n = int.Parse(Console.ReadLine());
   for (int i = 1; i <= n; i++)
   {
      if (i % 2 == 0)
      {
         continue;
      }
      Console.Write(" {0} ", i);
   }
}
```

Операторы return и throw будут рассмотрены позже.

Примеры решения задач

Пример 1

Для произвольных значений аргументов вычислить значение функции, заданной следующим образом:

$$y(x)=\frac{1}{x}+\sqrt{x+1}$$

Если в некоторой точке вычислить значение функции окажется невозможно, то вывести на экран сообщение «функция не определена».

Указание по решению задачи. Данную задачу можно решить двумя способами.

I способ. Заданная функция не определена в том случае, когда знаменатель первого слагаемого равен нулю, или подкоренное выражение второго слагаемого отрицательное, т.е., когда x=0, или x<-1. В остальных случаях функция определена. В данном случае программа выглядит следующим образом:

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("x= ");
         double x = double.Parse(Console.ReadLine());
         if (x==0 || x<-1)
         {
            Console.WriteLine("Функция не определена");
         }
         else
         {
            double y = 1/x + Math.Sqrt(x+1);
            Console.WriteLine("y({0:f2})={1:f2}",x,y);
         }
      }
   }
}
```

II способ. Заданная функция определена в том случае, когда знаменатель первого слагаемого не равен нулю и подкоренное выражение второго слагаемого неотрицательно, т.е. x≠0 и x≥-1. В остальных случаях функция не определена. В данном случае программа выглядит следующим образом:

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("x= ");
         double x = double.Parse(Console.ReadLine());
         if (x!=0 && x>=-1)
         {
            double y = 1/x + Math.Sqrt(x+1);
            Console.WriteLine("y({0:f2})={1:f2}",x,y);
         }
```

EPAM Systems

```
            else
            {
               Console.WriteLine("Функция не определена");
            }
         }
      }
   }
```

Обе программы дадут нам следующий результат:

```
x           Сообщение на экране
0           функция не определена
2,5         y(2,50)=2,27
-2          функция не определена
-1          y(-1,00)=-1,00
```

Пример 2

Для произвольных значений аргументов вычислить значение функции, заданной следующим образом:

$$y(x)=\begin{cases}\left(x^3+1\right)^2, \text{ при } x<0;\\ 0, \text{ при } 0\le x<1;\\ \left|x^2-5x+1\right|, \text{ при } x\ge 1.\end{cases}$$

Указания по решению задачи. Вся числовая прямая Ox разбивается на три непересекающихся интервала: (-∞; 0), [0; 1) и [1; +∞). На каждом интервале функция задается своею ветвью. Заданная точка x может попасть только в один из указанных интервалов. Чтобы определить, в какой из интервалов попала точка, воспользуемся следующим алгоритмом. Если *x<0*, то *x* попадает в первый интервал, и функцию высчитываем по первой ветви, после чего проверка заканчивается. Если это условие ложно, то истинно условие *x≥0*, и для того чтобы точка попала во второй интервал достаточно, чтобы выполнялось условие *x<1*. Если выполняется это условие, то точка x попадает во второй интервал, и мы определяем функцию по второй ветви, после чего заканчиваем вычисления. В противном случае, точка может принадлежать только третьему интервалу, поэтому дополнительная проверка не проводится, а сразу вычисляем функцию по третьей ветви. Приведенный алгоритм можно реализовать с помощью вложенных операторов *if*

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("x= ");
         double x = double.Parse(Console.ReadLine());
         double y;
         if (x<0)    //проверяем условие первой ветви
         {
            y = Math.Pow(Math.Pow(x, 3)+1, 2);
         }
```

```
            else
            {
               if (x<1)    //проверяем условие второй ветви
               {
                  y=0;
               }
               else
               {
                  y = Math.Abs(x*x-5*x+1);
               }
            }
            Console.WriteLine("y({0:f2})={1:f2}", x, y);
         }
      }
   }
```

Результат работы программы:

```
x     Сообщение на экране
-2      y(-2,00)=49,00
0       y(0,00)=0,00
1,5     y(1,50)=4,25
```

Пример 3

Дана точка на плоскости с координатами (x, y). Составить программу, которая выдает одно из сообщений «Да», «Нет», «На границе» в зависимости от того, лежит ли точка внутри заштрихованной области, вне заштрихованной области, или на ее границе.

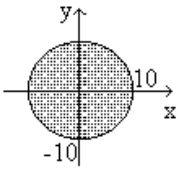


Рис. 9.

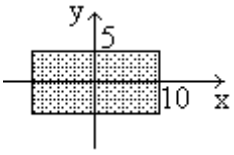


Рис. 10.

Указания по решению задачи. Всю плоскость можно разбить на три непересекающихся множества точек: I_1 – множество точек, лежащих внутри области; I_2 – множество точек, лежащих вне области; I_3 – множество точек, образующих границу области. Точка с координатами (x, y) может принадлежать только одному из них. Поэтому проверку принадлежности точки одному из указанных множеств можно проводить по аналогии с алгоритмом, приведенном в примере 2. Однако множества I_1, I_2, I_3 значительно труднее описать математически, чем интервалы в примере 2. Поэтому для непосредственной проверки выбираются те два множества, которые наиболее просто описать математически. Обычно труднее всего описать точки границы области. Например, для Рис. 9 множества задаются следующим образом:

I_1: $x^2 + y^2 < 10^2$; I_2: $x^2 + y^2 > 10^2$; I_3: $x^2 + y^2 = 10^2$

Для Рис. 10 множества задаются следующим образом:

I_1: |x|< 10 и |y|< 5 ;
I_2: |x| > 10 или |y|> 5;
I_3: (|x|<=10 и y = 5) или (|x|<=10 и y = -5) или (|y|<5 и x = 10) или (|y|<5 и x = -10).

Таким образом, для Рис. 9 описание всех множеств равносильно по сложности, а для Рис. 10 описать множество I_3 значительно сложнее. Следует отметить, что для множества I_3 необходимо выполнять проверку на равенство, что очень тяжело сделать для вещественных чисел из-за ошибок, связанных с округлением.

Для Рис. 9:

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("x= ");
         double x = double.Parse(Console.ReadLine());
         Console.Write("y= ");
         double y = double.Parse(Console.ReadLine());
         const int r = 10;
         if (x*x+y*y<r*r)
         {
            Console.WriteLine("да");
         }
         else
         {
            if (x*x+y*y>r*r)
            {
               Console.WriteLine("нет");
            }
            else
            {
               Console.WriteLine("на границе");
            }
         }
      }
   }
}
```

Результаты работы программы:

```
x       y         сообщение на экране
0       0             да
10      0             на границе
-12     13            нет
```

Замечание

Чтобы не вычислять два раза расстояние от заданной точки до начала координат можно ввести промежуточную переменную, в которую записать значение этого расстояния, а затем сравнить его с радиусом. Сделать это можно, например, следующим образом:

```
const int r = 10;
double n = Math.Sqrt(x*x+y*y);
```

EPAM Systems

```
if (n<r)
{
   Console.WriteLine("да");
}
else
{
   if (n>r)
   {
      Console.WriteLine("нет");
   }
   else
   {
      Console.WriteLine("на границе");
   }
}
```

Для Рис. 10:

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("x= ");
         double x = double.Parse(Console.ReadLine());
         Console.Write("y= ");
         double y = double.Parse(Console.ReadLine());
         const int sizeX = 10;
         const int sizeY = 5;
         if (Math.Abs(x)<sizeX && Math.Abs(y)<sizeY)
         {
            Console.WriteLine("да");
         }
         else
         {
            if (Math.Abs(x)>sizeX || Math.Abs(y)>sizeY)
            {
               Console.WriteLine("нет");
            }
            else
            {
               Console.WriteLine("на границе");
            }
         }
      }
   }
}
```

EPAM Systems

Результаты работы программы:

```
x     y       сообщение на экране
0     0             да
10    5             на границе
-12   13            нет
```

Пример 4

Написать программу, которая выводит на экран квадраты всех четных чисел из диапазона от A до B (A и B целые числа, при этом A≤B).

Указания по решению задачи. Из диапазона целых чисел от A до B необходимо выбрать только четные числа. Напомним, что четными называются числа, которые делятся на два без остатка. Кроме того, четные числа представляют собой упорядоченную последовательность, в которой каждое число отличается от предыдущего на 2. Решить эту задачу можно с помощью любого из рассмотренных операторов цикла.

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("a= ");
         int a = int.Parse(Console.ReadLine());
         Console.Write("b= ");
         int b = int.Parse(Console.ReadLine());
         int i;
         Console.Write("FOR: ");
         a = (a%2==0) ? a: a+1;
         for (i=a; i<=b; i+=2)
         {
            Console.Write(" {0}", i*i);
         }
         Console.Write("\nWHILE: ");
         i = a;
         while (i<=b)
         {
            Console.Write(" {0}", i*i);
            i += 2;
         }
         Console.Write("\nDO: ");
         i = a;
         do
         {
            Console.Write(" {0}", i*i);
            i += 2;
         }
         while (i<=b);
      }
   }
}
```

EPAM Systems

Результаты работы программы:

```
a    b    сообщение на экране
-11  11   FOR: 100  64  36  16  4  0  4  16  36  64  100
          WHILE: 100  64  36  16  4  0  4  16  36  64  100
          DO: 100  64  36  16  4  0  4  16  36  64  100
```

Пример 5

Постройте таблицу значений функции

$$y(x)=\begin{cases}\left(x^3+1\right)^2, \text{ при } x<0;\\ 0, \text{при } 0\le x<1;\\ \left|x^2-5x+1\right|, \text{при } x\ge 1.\end{cases}$$

для x∈[a, b] с шагом h.

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("a= ");
         double a = double.Parse(Console.ReadLine());
         Console.Write("b= ");
         double b = double.Parse(Console.ReadLine());
         Console.Write("h= ");
         double h = double.Parse(Console.ReadLine());
         double y;
         int i = 1;
         Console.WriteLine("{0,3} {1,5} {1,5}","#","x", "f(x)");
         for (double x = a; x <= b; x+=h, i++)
         {
            if (x<0)
            {
               y = Math.Pow(Math.Pow(x,3)+1,2);
            }
            else
            {
               if (x<1)
               {
                  y = 0;
               }
               else
               {
                  y = Math.Abs(x*x-5*x+1);
               }
            }
            Console.WriteLine("{0,3} {1,5:f2} {2,5:f2}",i,x,y);
         }
      }
```

EPAM Systems

```
      }
   }
```

Результат работы программы:

```
#   x      f(x)
1   -1,00  0,00
2   -0,50  0,77
3   0,00   0,00
4   0,50   0,00
5   1,00   3,00
6   1,50   4,25
7   2,00   5,00
```

Пример 6

Написать программу, которая по признаку геометрической фигуры (‘п’ – прямоугольник, ‘т’ – треугольник), запрашивает необходимые данные для расчетов и выводит на экран периметр и площадь заданной фигуры.

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.WriteLine("Введите признак фигуры: п -
                           прямоугольник, т - треугольник ");
         char n = char.Parse(Console.ReadLine());
         switch (n)
         {
            case 'п':
               Console.WriteLine("Введите стороны прямоугольника: ");
               Console.Write("a= ");
               double a = double.Parse(Console.ReadLine());
               Console.Write("b= ");
               double b = double.Parse(Console.ReadLine());
               if (a>0 && b>0)
               {
                  Console.WriteLine("p={0:f2} s={1:f2}",
                                          2*(a+b),a*b);
               }
               else
               {
                  Console.WriteLine("Прямоугольник с
                          заданными длинами сторон не существует");
               }
               break;
            case 'т':
               Console.WriteLine("Введите стороны треугольника: ");
               Console.Write("a= ");
               double a = double.Parse(Console.ReadLine());
               Console.Write("b= ");
               double b = double.Parse(Console.ReadLine());
```

EPAM Systems

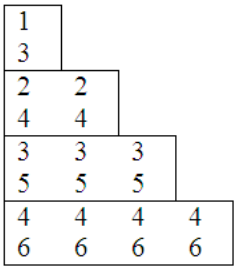
```
                    Console.Write("c= ");
                    double c = double.Parse(Console.ReadLine());
                    if (a+b>c && a+c>b && b+c>a)
                    {
                       double p = a + b + c;
                       double p2 = p / 2;
                       double s = Math.Sqrt(p2*(p2-a)*(p2-b)*(p2-c));
                       Console.WriteLine("p={0:f2}  s={1:f2}",p,s);
                    }
                    else
                    {
                       Console.WriteLine("Треугольник с заданными длинами
                                          сторон не существует");
                    }
                    break;
                 default:
                    Console.WriteLine("Вы неверно указали признак
                                       фигуры!");
                    break;
              }
           }
        }
```

Пример 7

Напечатать числа в виде следующей таблицы:

```
1
3
2   2
4   4
3   3   3
5   5   5
4   4   4   4
6   6   6   6
…
```

Указания по решению задачи. Исходную таблицу можно схематично разбить на четыре группы строк:



В каждой группе две строки, для элементов которых выполняется следующие правило: в первой строчке *i*-группы напечатано число *i*, причем *i* раз, а во второй строчке *i*-группы напечатано число *i+2* *i* раз. Для решения поставленной задачи можно использовать два последовательных оператора *for*, вложенных во внешний по отношению к ним оператор *for*. Внешний оператор следит за номером текущей группы *i*, первый вложенный цикл *i* раз печатает число *i*, второй вложенный цикл *i* раз печатает число *i+2*.

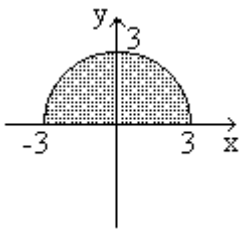
```
using System;
namespace Example
{
   class Program
   {
      static void Main()
      {
         Console.Write("n= ");
         byte n = byte.Parse(Console.ReadLine());
         for (byte i = 1; i <= n; i++)
         {
            for (byte j = 1; j <= i; j++)
            {
               Console.Write("{0,4}",i);
            }
            Console.WriteLine();
            for (byte j = 1; j <= i; j++)
            {
               Console.Write("{0,4}",i+2);
            }
            Console.WriteLine();
         }
      }
   }
}
```

Практикум №3

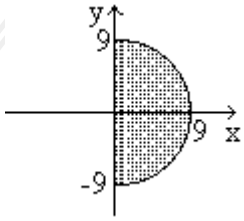
Задание 1

Дана точка на плоскости с координатами (x, y). Составить программу, которая выдает одно из сообщений «Да», «Нет», «На границе» в зависимости от того, лежит ли точка внутри заштрихованной области, вне заштрихованной области, или на ее границе. Области задаются графически следующим образом:

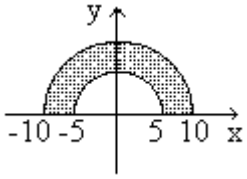
1.



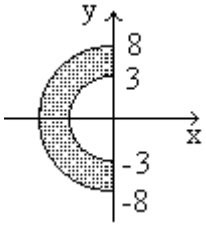
2.



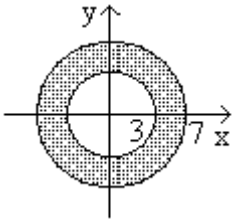
3.



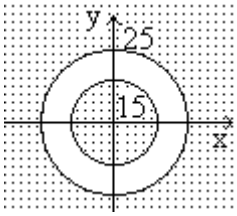
4.



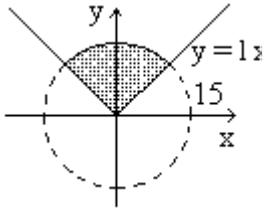
5.



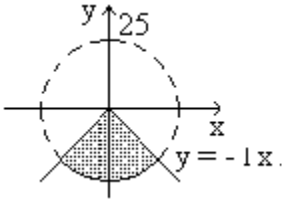
6.



7.

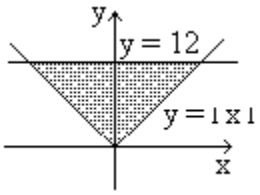


8.

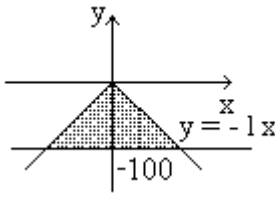


EPAM Systems

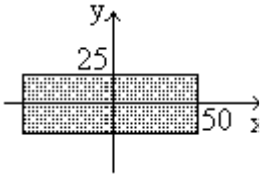
9.



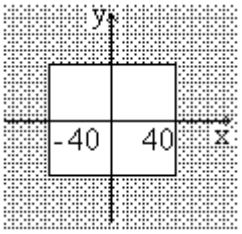
10.



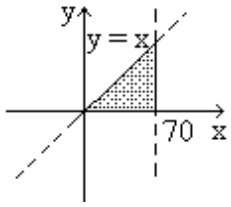
11.



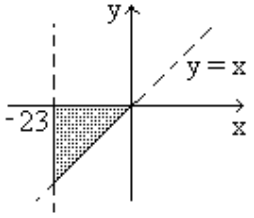
12.



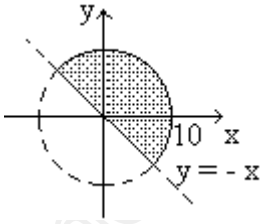
13.



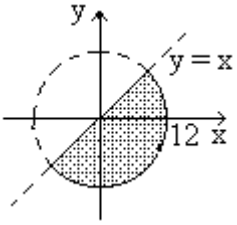
14.



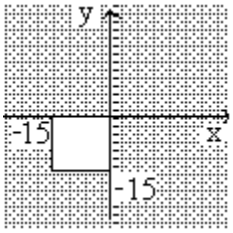
15.



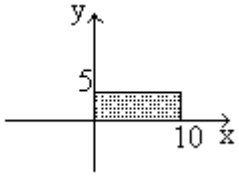
16.



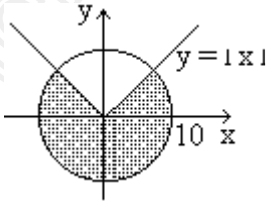
17.



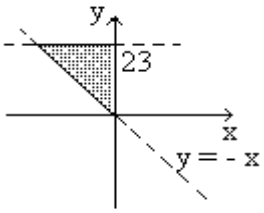
18.



19.



20.



Задание 2

Составить программу.

1. Дан порядковый номер месяца, вывести на экран его название.
2. Дан порядковый номер дня месяца, вывести на экран количество дней оставшихся до конца месяца.
3. Дан номер масти m ($1 \le m \le 4$), определить название масти. Масти нумеруются: «пики» – 1, «трефы» – 2, «бубны» – 3, «червы» – 4.
4. Дан номер карты k ($6 \le k \le 14$), определить достоинство карты. Достоинства определяются по следующему правилу: «туз» – 14, «король» – 13, «дама» – 12, «валет» – 11, «десятка» – 10, …, «шестерка» – 6.
5. С 1 января 1990 года по некоторый день прошло m месяцев. Определить название текущего месяца.
6. Дано расписание приемных часов врача на неделю. Вывести на экран приемные часы врача в заданный день недели (расписание придумать самостоятельно).
7. Проведен тест, оцениваемый в целочисленный баллах от нуля до ста. Вывести на экран оценку тестируемого в зависимости от набранного количества баллов: от 90 до 100 – «отлично», от 70 до 89 – «хорошо», от 50 до 69 – «удовлетворительно», менее 50 – «неудовлетворительно».
8. Дан год. Вывести на экран название животного, символизирующего этот год по восточному календарю.

ERAM Systems

9. Дан возраст человека мужского пола в годах. Вывести на экран возрастную категорию: до года – «младенец», от года до 11 лет – «ребенок», от 12 до 15 лет – «подросток», от 16 до 25 лет – «юноша», от 26 до 70 лет – «мужчина», более 70 лет – «пожилой человек».
10. Дан признак транспортного средства: а – автомобиль, в – велосипед, м – мотоцикл, с – самолет, п – поезд. Вывести на экран максимальную скорость транспортного средства в зависимости от введенного признака (максимальные значения скорости задать самостоятельно в теле программы).

Замечание

При решении данных задач можно использовать как оператор switch, *так и вложенные операторы* if. *Свой выбор обоснуйте*.

Задание 3

Вывести на экран:

1. целые числа 1, 3, 5, …, 21 в строчку через пробел:
2. целые числа 10, 12, 14, …, 60 в обратном порядке в столбик;
3. числа следующим образом:

```
10  10.4
11  11.4
…
25  25.4
```

4. числа следующим образом:

```
25  25.5  24.8
26  26.5  25.8
…
35  35.5  34.8
```

5. таблицу соответствия между весом в фунтах и весом в килограммах для значений 1, 2, 3, …, 10 фунтов (1 фунт = 453 г);
6. таблицу перевода 5, 10, 15, …, 120 долларов США в рубли по текущему курсу (значение курса вводится с клавиатуры);
7. таблицу стоимости для 10, 20, 30,…, 100 штук товара, при условии, что одна штука товара стоит *x* руб (значение *x* водится с клавиатуры);
8. таблицу перевода расстояний в дюймах в сантиметры для значений 2, 4, 6, …, 12 дюймов (1 дюйм = 25.4 мм);
9. кубы всех целых чисел из диапазона от A до B (A≤B) в обратном порядке;
10. все целые числа из диапазона от A до B (A≤B), оканчивающиеся на цифру X;
11. все целые числа из диапазона от A до B (A≤B), оканчивающиеся на цифры X или У;
12. все целые числа из диапазона от A до B (A≤B), оканчивающиеся на любую четную цифру;
13. только положительные целые числа из диапазона от A до B (A≤B);
14. все целые числа из диапазона от A до B, кратные трем (A≤B);
15. все четные числа из диапазона от A до B, кратные трем (A≤B);
16. только отрицательные четные числа из диапазона от A до B (A≤B);
17. все двухзначные числа, в записи которых все цифры разные;
18. все двухзначные числа, в которых старшая цифра отличается от младшей не больше чем на 1;
19. все трехзначные числа, которые начинаются и заканчиваются на одну и ту же цифру;
20. все трехзначные числа, в которых хотя бы две цифры повторяются.

Замечание

Решите каждую задачу тремя способами – используя операторы цикла while, do while *и* for.

EPAM Systems

Задание 4

Вывести на экран числа следующим образом:

1.
```
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
```

2.
```
1 2 3 … 10
1 2 3 … 10
1 2 3 … 10
1 2 3 … 10
```

3.
```
-10 -9 -8 … 12
-10 -9 -8 … 12
-10 -9 -8 … 12
-10 -9 -8 … 12
-10 -9 -8 … 12
```

4.
```
41 42 43 … 50
51 52 53 … 60
61 62 63 … 70
…
71 72 73 … 80
```

5.
```
5
5 5
5 5 5
5 5 5 5
5 5 5 5 5
```

6.
```
1 1 1 1 1
1 1 1 1
1 1 1
1 1
1
```

7.
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

8.
```
6 6 6 6 6
7 7 7 7
8 8 8
9 9
10
```

9.
```
7
6 6
5 5 5
4 4 4 4
3 3 3 3 3
```

10.
```
8 8 8 8 8
7 7 7 7
6 6 6
5 5
4
```

11.
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

12.
```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

13.
```
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

14.
```
4 3 2 1 0
3 2 1 0
2 1 0
1 0
0
```

15.
```
1
0
2 2
0 0
3 3 3
0 0 0
4 4 4 4
0 0 0 0
5 5 5 5 5
0 0 0 0 0
```

16.
```
8
7
7 7
6 6
6 6 6
5 5 5
5 5 5 5
4 4 4 4
```

17.
```
1
6
2 2
7 7
3 3 3
8 8 8
4 4 4 4
9 9 9 9
```

18.
```
9
4
8 8
3 3
7 7 7
2 2 2 2
6 6 6 6 6
1 1 1 1 1
```

19.
```
3
0
2  3
9  0
2  2  3
8  9  0
2  2  2  3
7  8  9  0
2  2  2  2  3
6  7  8  9  0
```

20.
```
2  2  2  2  2
3  4  5  6  7
2  2  2  2
2  3  4  5
2  2  2
1  2  3
2  2
0  1
2
```

Задание 5

Постройте таблицу значений функции y=f(x) для x∈[a, b] с шагом h.

1. $y=\begin{cases}\dfrac{1}{(0.1+x)^2}, & \text{если } x\ge 0.9;\\ 0.2x+0.1, & \text{если } 0\le x<0.9;\\ x^2+0.2, & \text{если } x<0.\end{cases}$

2. $y=\begin{cases}\sin(x), & \text{если } |x|<3;\\ \dfrac{\sqrt{x^2+1}}{\sqrt{x^2+5}}, & \text{если } 3\le|x|<9;\\ \sqrt{x^2+1}-\sqrt{x^2+5}, & \text{если } |x|\ge 9.\end{cases}$

3. $y=\begin{cases}0, & \text{если } x<a;\\ \dfrac{x-a}{x+a}, & \text{если } x>a;\\ 1, & \text{если } x=a.\end{cases}$

4. $y=\begin{cases}x^3-0.1, & \text{если } |x|\le 0.1;\\ 0.2x-0.1, & \text{если } 0.1<|x|\le 0.2;\\ x^3+0.1, & \text{если } |x|>0.2.\end{cases}$

5. $y=\begin{cases}a+b, & \text{если } x^2-5x<0;\\ a-b, & \text{если } 0\le(x^2-5x)<10;\\ ab, & \text{если } x^2-5x\ge 10.\end{cases}$

6. $y=\begin{cases}x^2, & \text{если } (x^2+2x+1)<2;\\ \dfrac{1}{x^2-1}, & \text{если } 2\le(x^2+2x+1)<3;\\ 0, & \text{если } (x^2+2x+1)\ge 3.\end{cases}$

7. $y=\begin{cases}-4, & \text{если } x<0;\\ x^2+3x+4, & \text{если } 0\le x<1;\\ 2, & \text{если } x\ge 1.\end{cases}$

8. $y=\begin{cases}x^2-1, & \text{если } |x|\le 1;\\ 2x-1, & \text{если } 1<|x|\le 2;\\ x^5-1, & \text{если } |x|>2.\end{cases}$

9. $y=\begin{cases}(x^2-1)^2, & \text{если } x<1;\\ \dfrac{1}{(1+x)^2}, & \text{если } x>1;\\ 0, & \text{если } x=1.\end{cases}$

10. $y=\begin{cases}x^2, & \text{если } (x+2)\le 1;\\ \dfrac{1}{x+2}, & \text{если } 1<(x+2)<10;\\ x+2, & \text{если } (x+2)\ge 10;\end{cases}$

EPAM Systems

11. $y = \begin{cases} x^2 + 5, \text{ если } x \le 5; \\ 0, \text{если } 5 < x < 20; \\ 1, \text{ если } x \ge 20. \end{cases}$

12. $y = \begin{cases} 0, \text{ если } x < 0; \\ x^2 + 1, \text{если } x \ge 0 \text{ и } x \ne 1; \\ 1, \text{если } x = 1. \end{cases}$

13. $y = \begin{cases} 1, \text{если } x = 1 \text{ или } x = -1; \\ \dfrac{-1}{1-x}, \text{если } x \ge 0 \text{ и } x \ne 1; \\ \dfrac{1}{1+x}, \text{если } x < 0 \text{ и } x \ne -1. \end{cases}$

14. $y = \begin{cases} 0.2x^2 - x - 0.1, \text{ если } x < 0; \\ \dfrac{x^2}{x - 0.1}, \text{если } x > 0 \text{ и } x \ne 0.1; \\ 0, \text{ если } x = 0.1. \end{cases}$

15. $y = \begin{cases} 1, \text{ если } (x\text{-}1) < 1; \\ 0, \text{ если } (x\text{-}1) = 1; \\ -1, \text{если } (x\text{-}1) > 1. \end{cases}$

16. $y = \begin{cases} x, \text{ если } x > 0; \\ 0, \text{ если } -1 \le x \le 0; \\ x^2, \text{ если } x < -1. \end{cases}$

17. $y = \begin{cases} a + bx, \text{ если } x < 93; \\ \text{b - ac, если } 93 \le x \le 120; \\ \text{abx, если } x > 120. \end{cases}$

18. $y = \begin{cases} x^2 - 0.3, \text{ если } y < 3; \\ 0, \text{ если } 3 \le x \le 5; \\ x^2 + 1, \text{ если } x > 5. \end{cases}$

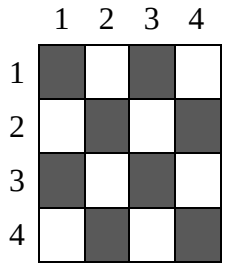
19. $y = \begin{cases} \sqrt{5x^2 + 5}, \text{если} |x| < 2; \\ \dfrac{|x|}{\sqrt{5x^2 + 5}}, \text{ если } 2 \le |x| < 10; \\ 0, \text{ если } |x| \ge 10. \end{cases}$

20. $y = \begin{cases} \sin(x), \text{ если } |x| < \dfrac{\pi}{2}; \\ \cos(x), \text{ если } \dfrac{\pi}{2} \le |x| \le \pi; \\ 0, \text{ если } |x| > \pi. \end{cases}$

Задание 6

Задачи повышенной сложности

1. Дана шахматная доска размером n×n клеток. Верхняя левая клетка доски черная и имеет номер (1, 1). Например, для n=4 шахматная таблица выглядит следующим образом:



   1) для заданного значения n определить количество черных ячеек шахматной доски;
   2) по номеру ячейки (k, m) определить ее цвет;
   3) определить, являются ли ячейки с номерами (k1, m1) и (k2, m2) одного цвета;

EPAM Systems

4) определить, находится ли фигура, стоящая в ячейке с номером (k1, m1), под ударом второй фигуры, стоящей в ячейке с номером (k2, m2), при условии, что ход второй фигуры и ей является:
   a. пешка;
   b. слон;
   c. ладья;
   d. ферзь;
   e. конь.

2. Задана дата в формате <день>.<месяц>.<год>. Определить:
   1) сколько дней прошло с начала года;
   2) сколько дней осталось до конца года;
   3) дату предыдущего дня;
   4) дату следующего дня.
3. Натуральное число из n цифр является числом Армстронга, если сумма его цифр, возведенных в n-ную степень, равна самому числу. Например, $153=1^3+5^3+3^3$. Найти все трехзначные числа Армстронга.
4. Стороны прямоугольника заданы натуральными числами n и m. Найти количество квадратов (стороны которых выражены натуральными числами), на которые можно разрезать данный прямоугольник, если от него каждый раз отрезать квадрат:
   1) наименьшей площади;
   2) наибольшей площади.

EPAM Systems