Организация ввода-вывода данных. Форматирование.

Программа при вводе данных и выводе результатов взаимодействует с внешними устройствами. Совокупность стандартных устройств ввода (клавиатура) и вывода (экран) называется консолью. В языке C# нет операторов ввода и вывода. Вместо них для обмена данными с внешними устройствами используются специальные классы. В частности, для работы с консолью используется стандартный класс *Console*, определенный в пространстве имен *System*.

Вывод данных

В приведенных выше примерах мы уже рассматривали метод *WriteLine*, реализованный в классе *Console*, который позволяет организовывать вывод данных на экран. Однако существует несколько способов применения данного метода:

1. Console.WriteLine(x); *//на экран выводится значение идентификатора x*
2. Console.WriteLine("x=" + x +"y=" + y); */* на экран выводится строка, образованная последовательным слиянием строки "x=", значения x, строки "y=" и значения y */*
3. Console.WriteLine("x={0} y={1}", x, y); */* на экран выводится строка, формат которой задан первым аргументом метода, при этом вместо параметра {0} выводится значение x, а вместо {1} – значение y*/*

Далее мы будем использовать только третий вариант, поэтому рассмотрим его более подробно. Пусть нам дан следующий фрагмент программы:

```
int i=3, j=4;
Console.WriteLine("{0} {1}", i, j);
```

При обращении к методу WriteLine через запятую перечисляются три аргумента: **"{0} {1}", i, j**. Первый аргумент **"{0} {1}"** определяет формат выходной строки. Следующие аргументы нумеруются с нуля, так переменная i имеет номер 0, j – номер 1. Значение переменной i будет помещено в выходную строку на место параметра {0}, а значение переменной j - на место параметра {1}. В результате на экран будет выведена строка: 3 4.

Если мы обратимся к методу WriteLine следующим образом:

```
Console.WriteLine("{0} {1} {0)", j, i);
```

то на экран будет выведена строка: 4 3 4.

Данный вариант использования метода WriteLine является наиболее универсальным, потому что он позволяет не только выводить данные на экран, но и управлять форматом их вывода. Рассмотрим несколько примеров:

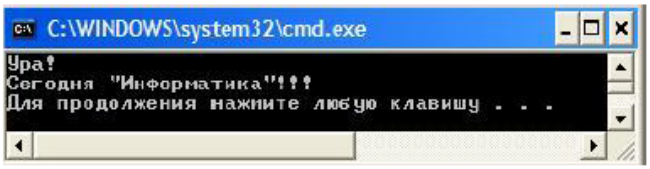
1) *Использование управляющих последовательностей:*

   Управляющей последовательностью называют определенный символ, предваряемый обратной косой чертой. Данная совокупность символов интерпретируется как одиночный символ и используется для представления кодов символов, не имеющих графического обозначения (например, символа перевода курсора на новую строку) или символов, имеющих специальное обозначение в символьных и строковых константах (например, апостроф). Рассмотрим управляющие символы:

Вид	Наименование
\a	Звуковой сигнал
\b	Возврат на шаг назад
\f	Перевод страницы
\n	Перевод строки
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\\	Обратная косая черта
\’	Апостроф
\”	Кавычки

Пример:

```
static void Main()
{
   Console.WriteLine("Ура!\nСегодня \"Информатика\"!!!");
}
```



Замечание. Вместо управляющей последовательности \n можно использовать константу Enviropment.NewLine. Она более универсальна, т.к. ее значение зависит от контекста и операционной системы, в которой запускается программа.

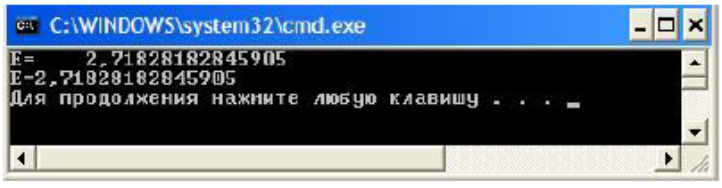
Задание. Измените программу так, чтобы все сообщение выводилось в одну строку, а после вывода сообщения раздавался звуковой сигнал.

2) *Управление размером поля вывода:*

Первым аргументом WriteLine указывается строка вида {n, m} – где n определяет номер идентификатора из списка аргументов метода WriteLine, а m – количество позиций (размер поля вывода), отводимых под значение данного идентификатора. При этом значение идентификатора выравнивается по правому краю. Если выделенных позиций для размещения значения идентификатора окажется недостаточно, то автоматически добавится необходимое количество позиций.

Пример:

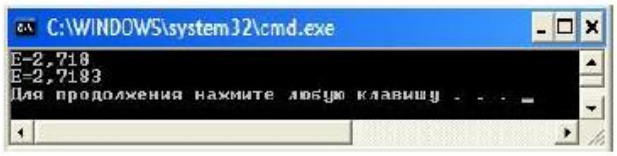
```
static void Main()
{
   double x= Math.E;
   Console.WriteLine("E={0,20}", x);
   Console.WriteLine("E={0,10}", x);
}
```



3) *Управление размещением вещественных данных:*

Первым аргументом WriteLine указывается строка вида {n: ##.###} – где n определяет номер идентификатора из списка аргументов метода WriteLine, а ##.### определяет формат вывода вещественного числа. В данном случае, под целую часть числа отводится две позиции, под дробную – три. Если выделенных позиций для размещения целой части значения идентификатора окажется недостаточно, то автоматически добавится необходимое количество позиций. Пример:

```
static void Main()
{
   double x= Math.E;
   Console.WriteLine("E={0:##.###}", x);
   Console.WriteLine("E={0:.####}", x);
}
```



Задание. Измените программу так, чтобы число e выводилось на экран с точностью до 6 знаков после запятой.

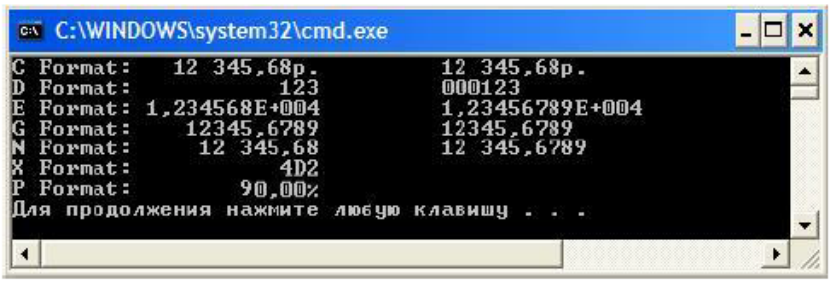
4) *Управление форматом числовых данных:*

Первым аргументом WriteLine указывается строка вида {n:<спецификатор>m} – где n определяет номер идентификатора из списка аргументов метода WriteLine, <спецификатор> - определяет формат данных, а m – количество позиций для дробной части значения идентификатора. В качестве спецификаторов могут использоваться следующие значения:

Параметр	***Формат***	***Значение***
C или c	Денежный. По умолчанию ставит денежный знак, определенный текущими региональными настройками. В русской Windows это р.	Задается количество десятичных разрядов.
D или d	Целочисленный (используется только с целыми числами)	Задается минимальное количество цифр. При необходимости результат дополняется начальными нулями
E или e	Экспоненциальное представление чисел	Задается количество символов после запятой. По умолчанию используется значение 6.
F или f	Представление чисел с фиксированной точкой	Задается количество символов после запятой
G или g	Общий формат (или экспоненциальный, или с фиксированной точкой)	Задается количество символов после запятой. По умолчанию выводится целая часть
N или n	Стандартное форматирование с использованием запятых и пробелов в качестве разделителей между разрядами	Задается количество символов после запятой. По умолчанию – 2, если число целое, то ставятся нули
X или x	Шестнадцатеричный формат	
P или p	Процентный	

Пример:

```
static void Main()
{
   Console.WriteLine("C Format:{0,14:C} \t{0:C2}", 12345.678);
   Console.WriteLine("D Format:{0,14:D} \t{0:D6}", 123);
   Console.WriteLine("E Format:{0,14:E} \t{0:E8}", 12345.6789);
   Console.WriteLine("G Format:{0,14:G} \t{0:G10}", 12345.6789);
   Console.WriteLine("N Format:{0,14:N} \t{0:N4}", 12345.6789);
   Console.WriteLine("X Format:{0,14:X} ", 1234);
   Console.WriteLine("P Format:{0,14:P} ", 0.9);
}
```



Ввод данных

Для ввода данных обычно используется метод ReadLine, реализованный в классе Console. Данный метод в качестве результата возвращает строку, тип которой string. Пример:

```
static void Main()
{
  string s = Console.ReadLine();
  Console.WriteLine(s);
}
```

Для того чтобы получить числовое значение, необходимо воспользоваться преобразованием данных. Пример:

```
static void Main()
{
  string s = Console.ReadLine();
  int x = int.Parse(s);  //преобразование строки в число
  Console.WriteLine(x);
}
```

Или сокращенный вариант:

```
static void Main()
{
  //преобразование введенной строки в число
  int x = int.Parse(Console.ReadLine());
  Console.WriteLine(x);
}
```

Для преобразования строкового представления целого числа в тип int мы используем метод Parse(), который реализован для всех числовых типов данных. Таким образом, если нам потребуется преобразовать строковое представление в вещественное, мы можем воспользоваться методом float.Parse() или double.Parse(). В случае, если соответствующее преобразование выполнить невозможно, то выполнение программы прерывается и генерируется исключение. Например, если входная строка имела неверный формат, то будет сгенерировано исключение System.FormatExeption.

Задания.

1. Подумайте, какие еще исключения могут возникнуть при использовании метода Parse. Проверьте свои предположения на практике.

2. Измените предыдущий фрагмент программы так, чтобы с клавиатуры вводилось вещественное число, а на экран это число выводилось с точностью до 3 знаков после запятой.

Выражения и преобразование типов

Выражение – это синтаксическая единица языка, определяющая способ вычисления некоторого значения. Выражения состоят из операндов, операций и скобок. Каждый операнд является в свою очередь выражением или одним из его частных случаев – константой, переменной или функций.

Замечание. *Список математических функций, реализованных в C# приведен в приложении 2.*

Примеры выражений:

(a + 0.12)/6 x && y || !z (t * Math.Sin(x)-1.05e4)/((2 * k + 2) * (2 * k + 3))

Вычисление значения выражения происходит с учетом приоритета операций (см. Приложение 1), которые в нем участвуют. Если в выражении соседствуют операции одного приоритета, то унарные операции, условная операция и операции присваивания выполняются *справа налево,* остальные — *слева направо*. Например,

a = b = c означает a=(b=c),
a+b+c означает (a + b) + c.

Если необходимо изменить порядок выполнения операций, то в выражении необходимо поставить круглые скобки.

Задания

1. Укажите последовательность выполнения операций в данном выражении:
   (x*x+Math.Sin(x+1))/x-2.
2. Запишите заданное математическое выражение по правилам языка C#:

a) $\sqrt{x^4+\sqrt{|x+1|}}$; b) $\dfrac{a^2+b^2}{1-\dfrac{a^3-b}{3}}$; c) $\ln\left|(y-\sqrt{|x|})(x-\dfrac{y}{x+\dfrac{x^2}{4}})\right|$

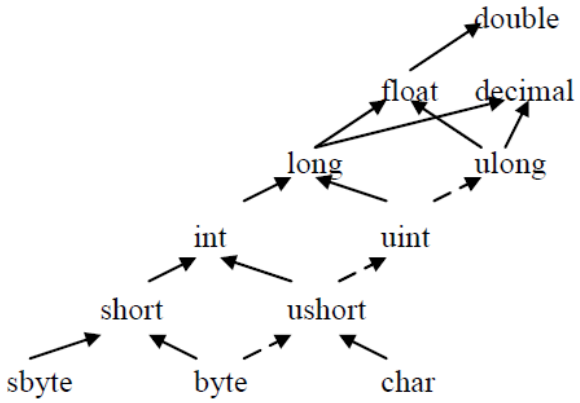
Задание. Напишите консольное приложение для вычисления значений выражений из задания 2 (см. выше). Значения всех необходимых переменных вводите с клавиатуры. Проведите проверку правильности вычислений с помощью калькулятора или MS Excel/

Результат вычисления выражения характеризуется значением и типом. Например, если *a* и *b* — переменные целого типа и описаны так:

int a = 2, b = 5;

то выражение *a + b* имеет значение *7* и тип *int*.

В выражение могут входить операнды различных типов. Если операнды имеют одинаковый тип, то результат операции будет иметь тот же тип. Если операнды разного типа, то перед вычислениями выполняются преобразования более коротких типов в более длинные для сохранения значимости и точности. Иерархия типов данных приведена в следующей схеме:



Преобразование типов в выражениях происходит *неявно* (без участия программистов) следующим образом. Если одни из операндов имеет тип, изображенный на более низком уровне, чем другой, то он приводится к типу второго операнда при наличии пути между ними. Если пути нет, то возникает ошибка компиляции (чтобы ее избежать, необходимо воспользоваться операцией явного преобразования). Если путей преобразования несколько, то выбирается наиболее короткий, не содержащий пунктирных линий.

Вернемся к рассмотрению операции явного преобразования из предыдущего раздела:

```
static void Main()
{
   int i = -4;
   byte j = 4;
   int a = (int)j;  //1
   byte b = (byte)i; //2
   Console.WriteLine("{0} {1}", a, b);
}
```

В строке 1 можно обойтись без явного преобразования типа, т.е. можно записать a=j, т.к. тип int в иерархии типов находится выше типа byte и существует путь для неявного преобразования типа byte в тип int. Но пути для неявного преобразования от типа int к типу byte нет, поэтому в строке 2 нельзя записать b=i – компилятор выдаст сообщение об ошибке.

Замечание. *Рассмотрим один важный пример. В C++ допустим следующий фрагмент кода:*

```
int x=10;
int y= (x)? 1: 2;  // 3
```

В данном примере в строке 3 происходит неявное преобразование типа переменной x, т.е. типа int, к логическому типу. При этом, если x принимает значение 0 (или null для ссылочных типов), то ему ставится в соответствие логическая величина false, всем другим значениям в соответствие ставится логическое значение true. В нашем случае переменная x принимает ненулевое значение, поэтому оно будет преобразовано к значению true, и в переменную y запишется значение 1.

В языке C# подобное неявное преобразование невозможно. Необходимо обязательно выполнять операцию сравнения значения x с 0. Например, следующим образом:

```
int x=10;
int y= (x!=0)? 1: 2;  // 3
```

Примеры решения практических задач

1. Написать программу, подсчитывающую площадь квадрата, периметр которого равен p.

Указания по решению задачи. *Прежде чем составить программу, проведем математические рассуждения. Пусть дан квадрат со стороной a, тогда:*

периметр вычисляется по формуле $p=4a$

площадь вычисляется по формуле $s=a^2$

$$\Rightarrow \quad \begin{matrix} a=\frac{p}{4} \\ a=\sqrt{s} \end{matrix} \quad \Rightarrow \quad \frac{p}{4}=\sqrt{s} \quad \Rightarrow \quad s=\left(\frac{p}{4}\right)^2$$

```
using System;
namespace Example
{
   class Program
   {
      static void Main()
       {
         Console.Write("p= ");
         double p = double.Parse(Console.ReadLine());
         double s = Math.Pow(p/4, 2);
         Console.WriteLine("s{0},="  s);
      }
   }
}
```

Задание. Изменить программу так, чтобы она подсчитывала периметр квадрата, площадь которого равна s

2. Определить, является ли сумма цифр натурального двухзначного числа четной.

Указание по решению задачи. *Напомним, что число является четным, если остаток от деления данного числа на 2 равен нулю. А для того, чтобы разложить двухзначное число на цифры, нужно разделить его нацело на 10 – так мы найдем старшую цифру числа, а затем взять остаток от деления на 10 исходного числа – так мы найдем его младшую цифру.*

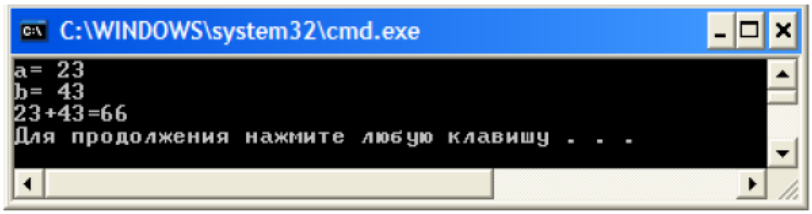
```
using System;
namespace Example
{
    class Program
    {
        static void Main()
        {
            Console.Write("a= ");
            byte a = byte.Parse(Console.ReadLine());
            string result=((a/10+a%10)%2==0)? "четное": "нечетное";
            Console.WriteLine(result);
        }
    }
}
```

Задание. Изменить программу так, чтобы она определяла, является ли сумма цифр трехзначного числа нечетной.

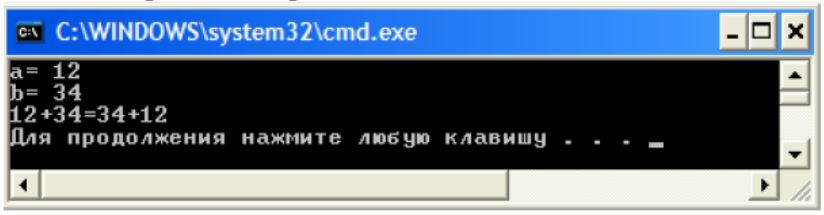
Задания для самостоятельной работы

I. Написать программу, которая, реализует диалог с пользователем:

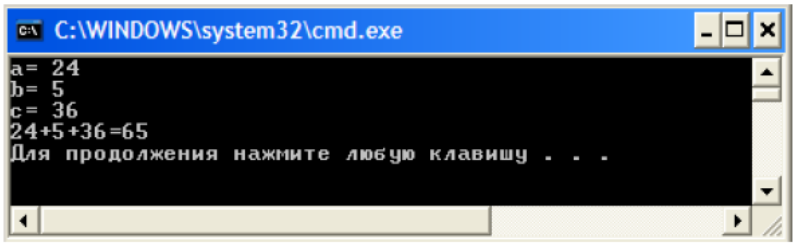
1) запрашивает с клавиатуры два целых числа, и выводит на экран сумму данных чисел:



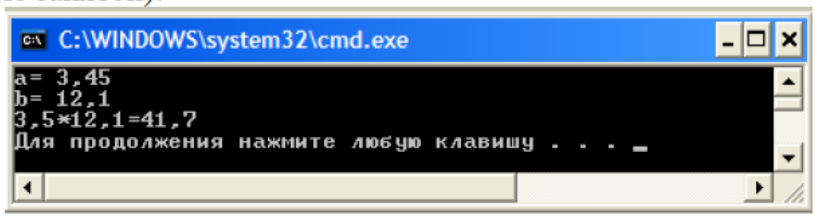
2) запрашивает с клавиатуры два целых числа, и выводит на экран сумму данных чисел в прямом и обратном порядке:



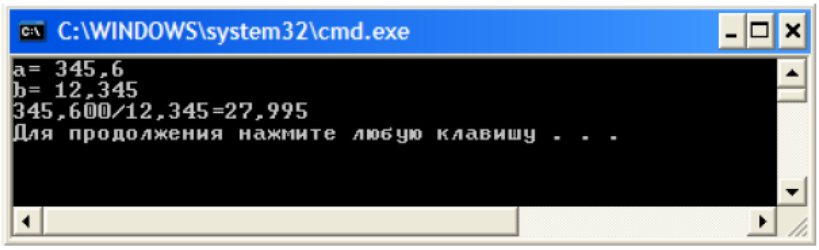
3) запрашивает с клавиатуры три целых числа, и выводит на экран сумму данных чисел:



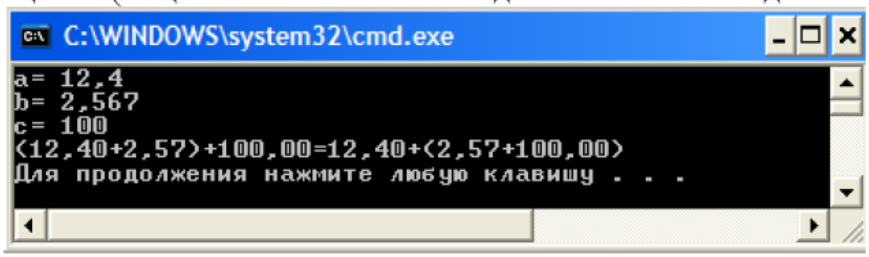
4) запрашивает с клавиатуры два вещественных числа, и выводит на экран произведение данных чисел (вещественные числа выводятся с точностью до 1 знака после запятой):



5) запрашивает с клавиатуры два вещественных числа, и выводит на экран результат деления первого числа на второе (вещественные числа выводятся с точностью до 3 знаков после запятой):

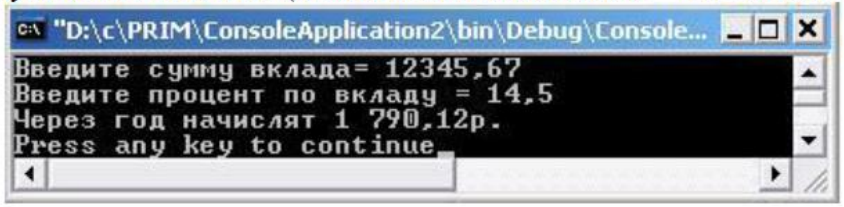


6) запрашивает с клавиатуры три вещественных числа, и выводит на экран следующее сообщение (вещественные числа выводятся с точностью до 2 знаков после запятой):

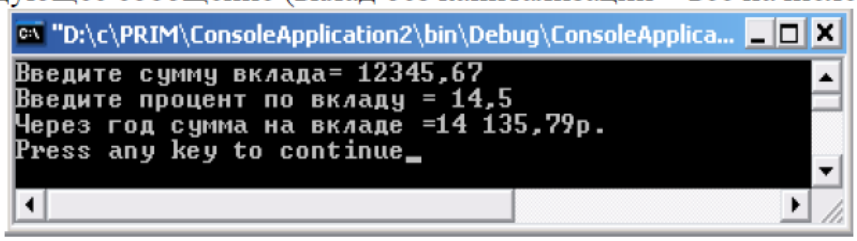


7) запрашивает с клавиатуры номинал купюры и количество купюр, и выводит экран следующее сообщение:

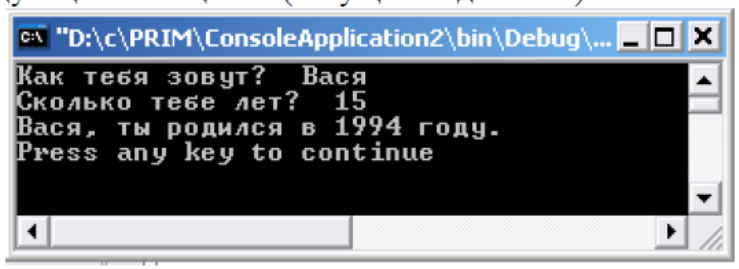
8) запрашивает с клавиатуры сумму вклада и процент по вкладу, и выводит на экран следующее сообщение (вклад без капитализации – все начисления в конце года):



9) запрашивает с клавиатуры сумму вклада и процент по вкладу, и выводит на экран следующее сообщение (вклад без капитализации – все начисления в конце года):



10) запрашивает с клавиатуры имя человека и его возраст, и выводит на экран следующее сообщение (текущий год – 2009):



II. Написать программу, которая подсчитывает:

1) площадь равностороннего треугольника, периметр которого равен p;
2) расстояние между точками с координатами x1, y1 и x2,y2;
3) гипотенузу прямоугольного треугольника по двум данным катетам a, b.
4) площадь прямоугольного треугольника по двум катетам a, b.
5) периметр прямоугольного треугольника по двум катетам a, b.
6) ребро куба, площадь полной поверхности которого равна s;
7) ребро куба, объем которого равен v;
8) периметр треугольника, заданного координатами вершин x1, y1, x2, y2, x3, y3;
9) площадь треугольника, заданного координатами вершин x1, y1, x2, y2, x3, y3;
10) сумму членов арифметической прогрессии, если известен ее первый член, разность и число членов прогрессии

III. Написать программу, которая определяет:

1) наибольшую цифру в натуральном двухзначном числе;
2) наименьшую цифру в натуральном двухзначном числе;
3) одинаковы ли цифры данного двухзначного числа;
4) заканчивается ли сумма цифр двухзначного числа на 0;
5) кратна ли трем сумма цифр двухзначного числа;
6) кратна ли числу A сумма цифр двухзначного числа;
7) какая из цифр трехзначного числа больше: первая или последняя;
8) какая из цифр трехзначного числа больше: первая или вторая;
9) какая из цифр трехзначного числа больше: вторая или последняя;
10) все ли цифры трехзначного числа одинаковые.