Начало работы с C#

Организация ввода-вывода данных

- 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 объекты.
- В частности, для работы с консолью используется стандартный класс **Console**, определенный в пространстве имен **System**.

Ввод данных

- Для ввода данных используется метод **ReadLine**, реализованный в классе **Console**.
- Особенностью метода является то, что в качестве результата он возвращает строку (**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** используется метод **int.Parse**(), который реализован для всех числовых типов данных.

Например,

- **float.Parse**() или **double. Parse**()

Вывод данных

- В примерах мы уже рассматривали метод **WriteLine**, реализованный в классе **Console**, который позволяет организовывать вывод данных на экран.
- Если использовать при выводе вместо метода WriteLine метод Write, вывод будет выполняться без перевода строки.
- Существует несколько способов применения данного метода.

Console.WriteLine(x);	на экран выводится значение идентификатора x
Console.WriteLine("x=" + x +"y=" + y);	на экран выводится строка, образованная последовательным слиянием строки "x=", значения x, строки "y=" и значения y
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

Управляющие символы

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

Пример. Вывести сообщение о версии установленной операционной системы, текущую дату и время.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
class Program {
static void Main(string[] args)
{
// вывести версию операционной системы
OperatingSystem os = System.Environment.OSVersion;
Console.WriteLine("Platform: {0}",os.Platform);
System.Console.WriteLine("The current date and time is " +
System.DateTime.Now); // дата и время
System.Console.ReadLine(); } } }
```

Форматированный ввод-вывод

- Вывод, производимый методами **System.Console.Write()** и **System.Console.WriteLine()**, можно форматировать.
- Форматирование позволяет указывать формат целых чисел, чисел с плавающей точкой и других типов данных.

Ранее переменные выводились таким образом:

System.Console.WriteLine ("value =" + value);

Можно вывести значение value, используя *требуемое число позиций* (например 5):

System.Console.WriteLine("value = {0, 5}", value);

Первое число в скобках – 0 - номер переменной, что соответствует первой переменной value в списке метода System.Console.WriteLine().

Второе число в скобках - количество позиций, отведенное для отображения переменной.

При выводе переменной длина ее представления будет дополнена пробелами слева. Если количество позиций меньше чем число знаков переменной, то оно будет выведено без форматирования.

Можно задать форматирование для вывода каждой переменной:

- int a = -12; int b = 20;
- System.Console.WriteLine("a = {0, 4}, b = {1, 3}", a, b);

- Результат вывода: a = -12, b = 20

Форматированный вывод чисел с плавающей точкой

Пусть определена переменная типа *double* с именем **myDouble**:

- double myDouble = 1234.56789;

Следующий пример выводит значения myDouble, отведя под него 10 знаков, и округлив его до 3 цифр после запятой:

- **System.Console.WriteLine("myDouble = {0, 10:f3}", myDouble);**
- Символы f3 означают, что значение выводится как число с плавающей точкой (символ f), в дробной части будет выведено 3 цифры.

Такое же форматирование можно применять для типов float и decimal. Напр.:

```
float myFloat = 1234.56789f;
System.Console.Write("myFloat =  {0, 10:f3}",
myFloat);

decimal myDecimal = 1234.56789m;
System.Console.Write("myDecimal = {0, 10:f3}",
myDecimal);
```

- В списке аргументов методов **WriteLine** или **Write** задается строка вида
- **{n, w: *спецификатор* k}**
- где **n** определяет номер идентификатора из списка аргументов метода, *спецификатор - определяет формат данных,* **w** - целая константа без знака, задает количество символов (длину поля), а k - количество позиций для дробной части значения идентификатора.
- Для каждого типа данных существует своя форма представления.

Тип данных		Форма
Числовые	Целые	**W**
	Вещественные с фиксированной точностью	**W:Fk**
	Вещественные в экспоненциальном формате	**W:Ek**
Логические		**W**
Символьные		**W**

Пример. Форматированный вывод данных различного типа

```
public static void Main()
{
int a = -14;
float c = -0.00151f;
double i = 1234.56789;
bool l=false;
string name="Petrov";
System.Console.WriteLine("name = {0, 6}, l ={1, 4}" , name, l);
System.Console.WriteLine("a ={0, 4}, c = {1,10:f5}, i = {1,20:e8}", a,c,i);
System.Console.WriteLine(" ");
System.Console.WriteLine("Для выхода нажмите на Enter");
System.Console.ReadLine();
}
```

Оператор условного перехода if

If *(условия выбора)*

{

// Если условие выбора истинно, то будут выполняться оператор или операторы блока 1.

}

else

{

// В противном случае (иначе): если условие выбора ложно, то будут выполняться оператор или операторы блока 2.

}

// Записанные далее операторы будут выполняться в любом случае, независимо от условия выбора.

Пример

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.

Управляющие операторы для организации циклов.

В C# есть три основных вида циклов:

- цикл **for** с параметром (счётчиком)
- цикл **while** с предусловием
- цикл **dowhile** с постусловием.

Оператор цикла for

- Оператор цикла for (для) служит для организации циклов с параметром (счётчиком).

```
for (выражение 1; условие; выражение 2)
{
// тело цикла
}
```

Пример.

```
int s, p, n; s = 0; p = 1;
for (n = 1; n <= 10; n = n + 1)
{
s = s + n; p = p * n;
}
Console.WriteLine(" s=" + s + " p=" + p);
```

Оператор цикла while

Форма записи оператора:

while *(условие)*

{

// тело цикла

}

Операторы в теле цикла исполняются до тех пор, пока условие цикла выполняется, т.е. имеет значение **true**.

Если вместо условия указано служебное слово true, т.е. while (true), то цикл будет бесконечным.

Пример.

```
int a,b,s,k;
Console.WriteLine(" Введите a" +"- левую границу отрезка.");
a = int.Parse((Console.ReadLine()));
Console.WriteLine(" Введите b" +"- правую границу отрезка.");
b = int.Parse((Console.ReadLine()));
s = 0; k = 0;
while ( a <= b)
{
S += a; // означает s = s + a
k++;     // означает k = k+1
a++;     // означает a = a + 1
}
Console.WriteLine('\t' + " РЕЗУЛЬТАТ:" + " s =" + s + " k=" + k + '\n');
```

Оператор цикла do-while

Оператор цикла do-while является версией цикла while с постусловием.

Форма записи оператора:

do {

//тело цикла

}

while *(условие)*

```
int a, b, s, k;
Console.WriteLine(" Введите a"+" - левую границу отрезка");
a =int.Parse((Console.ReadLine()));
Console.WriteLine(" Введите b"+" - правую границу отрезка");
b = int.Parse((Console.ReadLine()));
s = 0; k = 0;
do
{s += a; k++; a++;}
while (a <= b);
Console.WriteLine('\t' + " РЕЗУЛЬТАТ:" +" s =" + s + " k=" + k +
'\n');
```