Лекция 12

Система ввода/вывода языка программирования

Методы класса **Console**

Beep: подача звукового сигнала

Clear: очистка консоли

WriteLine: вывод (возврат) строки текста, включая символ возврата каретки (то есть с переводом на новую строку)

Write: вывод строки текста, но без символа возврата каретки

ReadLine: считывание строки текста из входного потока

Read: считывание введенного символа в виде числового кода данного символа. С помощью преобразования к типу **char** можно получить введенный символ

ReadKey: считывание нажатой клавиши клавиатуры

Пример 1

```
using System;namespace ConsoleApplication1{
   class Class1 {
      static void Main() {
         int     i = 3;
        double  y = 4.12;
        decimal d = 600m;
        string  s = "Вася";
       Console.WriteLine("i = " + i);      // 1
       Console.WriteLine("s = " + s);      // 2
       Console.WriteLine("y = {0} \nd = {1}", y-1, d+1);    // 3
       Console.WriteLine($"y = {y} \nd = {d}");             // 4
       Console.ReadKey();  // остановка экрана
     }
   }
}
```

Пример 1

Результат работы программы:

i = 3

s = Вася

y = 3,12

d = 601

y = 4,12

d = 600

В классе **Console** определены **методы ввода** строки и отдельного символа, но нет методов, которые позволяют непосредственно считывать с клавиатуры числа.

Ввод числовых данных выполняется в два этапа:

1. Символы, представляющие собой число, вводятся с клавиатуры в строковую переменную.
2. Выполняется преобразование из строки в переменную соответствующего типа.

Преобразование можно выполнить либо с помощью специального класса **Convert**, определенного в пространстве имен **System**, либо с помощью метода **Parse**, имеющегося в каждом стандартном арифметическом классе.

Пример 2

```
using System;namespace ConsoleApplication1{
  class Class1{
    static void Main(){
        Console.WriteLine("Введите строку");
        string s = Console.ReadLine();                    // 1
        Console.WriteLine("s = " + s);
        Console.WriteLine("Введите символ");
        char c = (char)Console.Read();                    // 2
        Console.ReadLine();                               // 3
        Console.WriteLine("c = " + c);
        string buf;       // строка – буфер для ввода чисел
        Console.WriteLine("Введите целое число");
        buf = Console.ReadLine();
        int i = Convert.ToInt32(buf);                     // 4
        Console.WriteLine(i);
```

Пример 2

```
            Console.WriteLine("Введите вещественное число");
            buf = Console.ReadLine();
            double x = Convert.ToDouble(buf);          // 5
            Console.WriteLine(x);
            Console.WriteLine("Введите вещественное число");
            buf = Console.ReadLine();
            double y = double.Parse(buf);              // 6
            Console.WriteLine(y);
            Console.WriteLine("Введите вещественное число");
            buf = Console.ReadLine();
            decimal z = decimal.Parse(buf);            // 7
            Console.WriteLine(z);
        }
    }
}
```

Свойства класса **Console**

- **BackgroundColor**: цвет фона консоли
- **ForegroundColor**: цвет шрифта консоли
- **BufferHeight**: высота буфера консоли
- **BufferWidth**: ширина буфера консоли
- **Title**: заголовок консоли
- **WindowHeight** и **WindowWidth**: высота и ширина консоли

Пример 3

```
class Program{
  static void Main(string[] args) {
    // установка зеленого цвета шрифта
    Console.ForegroundColor=ConsoleColor.DarkGreen;
    try{
      do{
          Console.WriteLine("Введите первое число");
          int num1 = Int32.Parse(Console.ReadLine());
          Console.WriteLine("Введите второе число");
          int num2 = Int32.Parse(Console.ReadLine());
```

Пример 3

```
            // сиреневый цвет
            Console.ForegroundColor =
ConsoleColor.DarkMagenta;
            Console.WriteLine("Сумма чисел {0} и {1} равна
{2}", num1, num2, num1 + num2);
            Console.WriteLine("Для выхода - Escape; для
продолжения - любая другая клавиша");
        } while (Console.ReadKey().Key != ConsoleKey.Escape);
      } catch (Exception ex) {
            Console.WriteLine(ex.Message);
            Console.ReadLine();
    }
  }
}
```

Метод **ReadKey**()

В классе **Console** включен метод **ReadKey**(), позволяющий непосредственно считывать отдельно введенные с клавиатуры символы без построчной буферизации. При нажатии клавиши метод **ReadKey**() немедленно возвращает введенный с клавиатуры символ. И в этом случае пользователю уже не нужно нажимать дополнительно клавишу <**Enter**>.

Две формы объявления метода **ReadKey** ().

static ConsoleKeyInfo ReadKey()

static ConsoleKeyInfo ReadKey(bool intercept)

Если значение параметра **intercept** равно **true**, то введенный символ не отображается.

ReadKey() = ReadKey(false) // по умолчанию

Метод **ReadKey**()

Метод **ReadKey**() возвращает информацию о нажатии клавиши в объекте типа **ConsoleKeyInfo**, который представляет собой структуру, состоящую из приведенных ниже свойств, доступных только для чтения.

char **KeyChar** - содержит эквивалент **char** введенного с клавиатуры символа

ConsoleKey **Key** - содержит значение из перечисления **ConsoleKey** всех клавиш на клавиатуре

ConsoleModifiers **Modifiers** - содержит описание одной из модифицирующих клавиш (<**Alt**>, <**Ctrl**> или <**Shift**>), которые были нажаты, если это действительно имело место, при формировании ввода с клавиатуры

Пример 4

```
using System;
class ReadKeys {
    static void Main() {
        ConsoleKeyInfo keypress;
        Console.WriteLine("Введите несколько символов,
   а по окончании - <0>.");
        do {
            // считать данные о нажатых клавишах
             keypress = Console.ReadKey(true);
             Console.WriteLine(" Вы нажали клавишу: " +
   keypress.KeyChar);
```

Пример 4

```
// Проверить нажатие модифицирующих клавиш
      if ((ConsoleModifiers.Alt & keypress.Modifiers) != 0)
            Console.WriteLine("Нажата клавиша <Alt>.");
      if ((ConsoleModifiers.Control & keypress.Modifiers) != 0)
            Console.WriteLine("Нажата клавиша <Control>.");
      if ((ConsoleModifiers.Shift & keypress.Modifiers) != 0)
            Console.WriteLine("Нажата клавиша <Shift>.");
    } while(keypress.KeyChar != '0');
  }
}
```

Средства форматирования строк в C#

Методы *WriteLine* и *Write* используются для вывода информации в консоль, и при этом дают возможность отформатировать вывод.

Метод ***Format*** класса ***String*** предназначен конкретно для форматирования. Форматирование в методе ***ToString*** можно задать только для чисел и дат.

Общая структура форматирования выходной информации (строк) имеет следующий вид:

Console.**WriteLine**("строка формата", arg0, arg1, …, argn);

String.**Format**("строка формата", arg0, arg1, …, argn);

arg0 и arg1 здесь – аргументы форматирования (числа, строки, даты, и т. д.), из которых в результате будет создана новая отформатированная строка.

Средства форматирования строк в C#

Строка формата может содержать *обычные символы*, которые будут отображены в том виде, в котором они заданы, *и команды форматирования*. Команда форматирования заключается в фигурные скобки и имеет следующую структуру:

{[номер аргумента], [ширина]:[формат]}

По **[номеру аргумента]** указывается к какому аргументу будет применена данная команда (отсчет аргументов начинается с нуля).

[ширина] задает минимальный размер поля.

[формат] – спецификатор формата.

Параметры [ширина] и [формат] не являются обязательными.

Средства форматирования строк в C#

```
// выравнивание по правому краю
Console.WriteLine("Result is {0, 6}", 1.2789);
Console.WriteLine("Result is {0, 6}", 7.54);
// выравнивание по левому краю
Console.WriteLine("Result is {0, -6}", 1.2789);
Console.WriteLine("Result is {0, -6}", 7.54);
```

В результате получится:

```
Result is 1,2789
Result is     7,54
Result is 1,2789
Result is 7,54
```

Параметры форматирования

Параметр	Значение
C, c	Используется для вывода значений в денежном (**Currency**) формате. По умолчанию перед выводимым значением подставляется символ доллара ($), хотя можно отменить подстановку этого символа при помощи объекта **NumberFormatInfo**.
D, d	Используется для вывода целых десятичных значений (**Decimal**). После этого символа можно указать количество значащих цифр.
E, e	Для вывода значений в экспоненциальном формате (**Scientific**).
F, f	Вывод значений с фиксированной точкой (**Fixed-point**).
G, g	Общий (**General**) формат. Применяется для вывода значений с фиксированной точностью или в экспоненциальном формате.

Параметры форматирования

Параметр	Значение
N, n	Стандартное числовое форматирование (**Number**) с использованием разделителей (пробелов или запятых) между разрядами.
X, x	Вывод значений в шестнадцатеричном формате (**Hexadecimal**). Указывает минимальное количество цифр. При необходимости добавляются нули. Если использовать заглавную X, то символы в шестнадцатеричном формате также будут заглавными.
R, r	Округление (**Round-trip**). Форматирует число в строку таким образом, что его можно обратно преобразовать без потерь точности.
P, p	Проценты. Умножает число на 100 и выводит со знаком процентов. Указывает количество десятичных знаков.

Пример 5

```
using System;
namespace Test{
  class BasicIO  {
    static void Main()  {   // Форматирование без параметров
      string theString = "Привет всем!";
      int theInt = 15;
      float theFloat = 9.99F;
      double theDouble = 123.456789;
      BasicIO theClass = new BasicIO();
     Console.WriteLine("Без параметров форматирования:");
     Console.WriteLine("string: {0}\nint: {1}\nfloat:
    {2}\ndouble: {3}\nobject: {4}", theString, theInt, theFloat,
    theDouble, theClass);
     Console.Write("\n\n"); // две пустые строки
```

Пример 5

```
object[] array = {"Привет!", 20.9, 1, "55", Math.PI};
Console.WriteLine("Элементы массива:\n{0}; {1}; {2}; {3}; {4}", array);
Console.WriteLine("С параметрами форматирования:");
Console.WriteLine("C format: {0:C}", 99989.987);
Console.WriteLine("c format: {0:c1}", 99989.987);
Console.WriteLine("D9 format: {0:D9}", 99999);
Console.WriteLine("E format: {0:E}", .31415926538 * 10);
Console.WriteLine("F format: {0:F3}", 55555.6666);
Console.WriteLine("N format: {0:N}", 99999);
Console.WriteLine("X format: {0:X}", 99999);
Console.WriteLine("x format: {0:x}", 99999);
Console.WriteLine("p format: {0:p}", 0.55);
```

Пример 5

```
        string str;
         str = String.Format("C format: {0:C}", 99989.987);
         Console.WriteLine("Предварительное форматирование
    в символьную строку:");
         Console.WriteLine(str);
         str = "ФИО: {{ {0} }}, Возраст: {{ {1} }}";
         str = String.Format(str, "Огоньков В.М.", 55);
         Console.WriteLine("\n\n Предварительное
    форматирование в символьную строку:");
        Console.WriteLine(str);
       while(true);
     }
  }
}
```

Пользовательский формат числовых данных

Специальный символ	Значение
0	Цифра или ноль
#	Цифра
.	Разделитель дроби
,	Разделитель тысяч
%	Процент
e	Экспонента
;	Определяет разделы, которые описывают форматы для положительных, отрицательных чисел и нуля
\	Экранирование специальных символов. Позволяет вставлять спецсимволы как текст

Примеры пользовательских форматов

```
Console.WriteLine("{0:0000.00}", 1024.32);    // 1024,32
Console.WriteLine("{0:00000.000}", 1024.32);  // 01024,320
Console.WriteLine("{0:####.###}", 1024.32);    // 1024,32
Console.WriteLine("{0:####.#}", 1024.32);        // 1024,3
Console.WriteLine("{0:#,###.##}", 1024.32);     // 1 024,32
Console.WriteLine("{0:##%}", 0.32);                // 32%
Console.WriteLine("{0:<####.###>;[####.###];ноль}", 1024.32);
                                                   // <1024,32>
Console.WriteLine("{0:<####.###>;[####.###];ноль}", -1024.32);
                                                   // [1024,32]
Console.WriteLine("{0:<####.###>;[####.###];ноль}", 0);
                                                    // ноль
```

Встроенные форматы даты и времени

Специальный символ	Формат	Пример
d	Короткая дата	30.06.2014
D	Длинная дата	30 июня 2014 г.
t	Короткое время	22:30
T	Длинное время	22:30:10
f	Длинная дата/короткое время	30 июня 2014 г. 22:30
F	Длинная дата/длинное время	30 июня 2014 г. 22:30:10
g	Короткая дата/короткое время	30.06.2014 22:30
G	Короткая дата/длинное время	30.06.2014 22:30:10
M/m	Месяц и день	июня 30
O/o	Обратный	2014-06-30 22:30:10.0000000
R/r	RFC1123	Mon, 30 Jun 2014 22:30:10 GMT
s	Для сортировки	2014-06-30 22:30:10
u	Локальное, в универсальном формате	2014-06-30 22:30:10Z
U	GMT	30 июня 2014 г. 19:30:10
Y	Год и месяц	Июнь 2014

Примеры стандартных форматов даты и времени

```
Console.WriteLine("{0:d}", DateTime.Now);
                                    // 30.06.2014
Console.WriteLine("{0:D}", DateTime.Now);
                                    // 30 июня 2014 р.
Console.WriteLine("{0:t}", DateTime.Now);
                                    // 2:57
Console.WriteLine("{0:T}", DateTime.Now);
                                    // 2:57:53
Console.WriteLine("{0:U}", DateTime.Now);
                                    // 29 июня 2014 р. 23:57:53
Console.WriteLine("{0:Y}", DateTime.Now);
                                    // Июнь 2014 р.
```

Пользовательский формат даты и времени

Специальный символ	Значение	Результат
y	Год 0-9	4
yy	Год 00-99	14
yyyy	Год, 4 цифры	2014
M	Месяц 1-12	6
MM	Месяц 01-12	06
d	День 1-31	0
dd	День 01-31	30
h	Час 1-12	2
hh	Час 01-12	10
H	Час 1-24	22
H	Час 01-24	22
m	Минута 0-59	30
mm	Минута 00-59	30
s	Секунда 0-59	10
ss	Секунда 00-59	10

Пользовательский формат даты и времени

Специальный символ	Значение	Результат
f – ffffff	Доли секунды	12 (для ff)
F-FFFFFF	Доли секунды, если они не равны 0	12 (для ff)
MMM	Сокращенное имя месяца	Июн
MMMM	Имя месяца	Июнь
ddd	Сокращенное имя дня недели	Пн
dddd	Имя дня недели	понедельник
tt	Маркер для "AM" и "PM" 12-часового формата	PM
zz	Смещение временной зоны, короткое	+03
zzz	Смещение временной зоны, полное	+03:00
gg	Эра	A.D.
:	Разделитель времени	22:30:10
/	Разделитель даты	30.06.2014
****	Экранирование	

Примеры пользовательских форматов даты и времени

```
Console.WriteLine("{0:y yy yyy yyyy}",
    DateTime.Now);              // 14 14 2014 2014
Console.WriteLine("{0:d dd ddd dddd}",
    DateTime.Now);              // 30 30 Пн понедельник
Console.WriteLine("{0:M MM MMM}",
    DateTime.Now);              // 6 06 Июн
Console.WriteLine("{0:HH.mm.ss dd-MMM-yyyy}",
    DateTime.Now);              // 03.21.22 30-Июн-2014
Console.WriteLine("{0:z zz zzz}", DateTime.Now);
                                // +3 +03 +03:00
Console.WriteLine(DateTime.Now.ToString("dd
    MMM yyyy"));                // 30 Июн 2014
```

Региональные параметры CultureInfo

```
string formattedString = string.Format(new
    System.Globalization.CultureInfo("en-US"), "{0:dddd}
    Money - {1:c}", DateTime.Now, 15);
Console.WriteLine("{0:dddd} Money - {1:c}",
    DateTime.Now, 15); // понедельник Money - 15,00 руб.
Console.WriteLine(formattedString);
                                // Monday Money - $15.00

formattedString = string.Format(new
    System.Globalization.CultureInfo("uk-UA"), "{0:dddd}
    Money - {1:c}", DateTime.Now, 15);
Console.WriteLine(formattedString);
                                // понеділок Money - 15,00 грн.
```

Региональные параметры CultureInfo

```
 decimal[] amounts = {16305.32m, 18794.16m};
Console.WriteLine("   Beginning Balance
   Ending Balance");
Console.WriteLine("   {0,-28:C2}{1,14:C2}",
   amounts[0], amounts[1]);
 //        Beginning Balance          Ending Balance
 //        $16,305.32                 $18,794.16
```

Встроенные потоки

Для всех программ, в которых используется пространство имен **System**, доступны встроенные потоки, открывающиеся с помощью свойств **Console.In** - связано с потоком ввода, **Console.Out** - связано с потоком вывода и **Console.Error** - связано со стандартным потоком сообщений об ошибках, которые по умолчанию также выводятся на консоль.

Поток **Console.In** является экземпляром объекта класса **TextReader**, и поэтому для доступа к нему могут быть использованы методы и свойства, определенные в классе **TextReader**.

Пример 6

```
using System;
class ReadChars2 {
    static void Main() {
        string str;
        Console.WriteLine("Введите несколько
   символов.");
    // вызвать метод ReadLine() класса TextReader
        str = Console.In.ReadLine();
        Console.WriteLine("Вы ввели: " + str);
    }
}
```

Пример 7

```
using System;
    class ErrOut {
        static void Main() {
            int a=10, b=0, results;
            Console.Out.WriteLine ("Деление на нуль приведет
  к исключительной ситуации.");
            try {
// сгенерировать исключение при попытке деления на нуль
                result = a / b;
            } catch(DivideByZeroException exc) {
                Console.Error.WriteLine(exc.Message);
            }
        }
    }
```

Контрольные вопросы

1 Какой класс используется для консольного ввода-вывода? Приведите примеры ввода и вывода переменных различных типов.
2 Какие средства языка используются для преобразования величин из символьной формы представления во внутреннюю?
3 Какие функции позволяют вводить и выводить информацию различного типа?
4 Какие параметры форматирования используются для вывода информации?