МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

Учебно-методическое пособие для вузов

Составители:
Г.Э. Вощинская,
Е.Е. Михайлова

Издательско-полиграфический центр
Воронежского государственного университета
2012

Содержание

14. $\ln(1+x) = \frac{x}{1} - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$ (R = 1).

15. $\mathrm{ch}(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots$ (R = ∞).

16. $\mathrm{sh}(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots$ (R = ∞).

17. $e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \ldots + (-1)^N * \frac{x^N}{N!}$ (R = ∞).

18. $e^{x} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^N}{N!}$ (R = ∞).

19. $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$ (R = ∞).

20. $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$ (R = ∞).

Консольный ввод-вывод

Главные методы класса Console – это методы **`Read(), ReadLine(), Write(), WriteLine().`**

Метод **`static int Read()`** используется для считывания одного символа. Результат возвращает как значение типа int, которое должно быть приведено к типу char.

Метод **`static string ReadLine()`** считывает строку символов. При вводе числовых значений они должны быть переведены с помощью следующих функций (табл.1).

Таблица 1

Методы преобразования строки

Структура	***Метод преобразования***
`Decimal`	**`static decimal Parse(string str)`**
`Double`	**`static double Parse(string str)`**
`Single`	**`static float Parse(string str)`**
`Int64`	**`static long Parse(string str)`**
`Int32`	**`static int Parse(string str)`**
`Int16`	**`static short Parse(string str)`**
`UInt64`	**`static ulong Parse(string str)`**
`UInt32`	**`static uint Parse(string str)`**
`UInt16`	**`static ushort Parse(string str)`**
`Byte`	**`static byte Parse(string str)`**

Вывод информации в окно приложения обеспечивают методы **`Console.Write()`** и **`Console.WriteLine()`**.

WriteLine() отличается тем, что завершает свою работу обязательным выводом Escape-последовательности line feed/carriage return.

```
Console.WriteLine("The Captain is on the board!"); // Вывод строки.
Console.WriteLine(314); // Символьное представление целочисленного значения.
Console.WriteLine(3.14); // Символьное представление значения типа float.
```

Escape-последовательность **\n** внутри строки означает переход на новую строку.

Escape-последовательность **\t** внутри строки означает табуляцию.

Методы с одним параметром достаточно просты. На стадии компиляции при выяснении типа выводимого значения подбирается соответствующий вариант перегруженной функции вывода.

При выводе значения определяется его тип, производится соответствующее стандартное преобразование к символьному виду, которое в виде последовательности символов, соответствующей существующей в операционной системе настройке языковых параметров, представления чисел, времени и дат, и выводится в окно представления.

При использовании метода **WriteLine()** с несколькими параметрами первый *обязательно строковый* параметр используется как управляющий шаблон для представления выводимой информации. Значения следующих за строковым параметром выражений будут выводиться в окно представления лишь в том случае, если первый параметр-строка будет явно указывать места расположения выводимых значений, соответствующих этим параметрам. Явное указание обеспечивается маркерами выводимых значений, которые в самом простом случае представляют собой заключенные в фигурные скобки целочисленные литералы (например, {3}). Таким образом, оператор

```
Console.WriteLine("The sum of {0} and {1} is {2}", 314,3.14,314+3.14);
```

обеспечивает вывод следующей строки:

The sum of 314 and 3.14 is 317.14

Помимо индекса параметра маркер выводимого значения может содержать дополнительные сведения относительно формата представления выводимой информации. Выводимые значения преобразуются к символьному представлению, которое, в свою очередь, при выводе в окно приложения может быть дополнительно преобразовано в соответствии с предопределенным «сценарием преобразования». Вся необходимая для дополнительного форматирования информация размещается непосредственно в маркерах и отделяется запятой от индекса маркера.

```
WriteLine(“строка_форматирования”, арг0, арг1,…,аргN)
строка_форматирования::={номер_аргумента,
ширина:формат}
```

При наличии элемента *ширина* выводимые данные дополняются пробелами, которые гарантируют, что поле, занимаемое выводимым значением, будет иметь минимальную ширину. Если значение *ширина* положительно, выводимые данные выравниваются по правому краю, если отрицательно – то по левому.

Кроме того, в маркерах вывода могут также размещаться дополнительные строки форматирования (FormatString). При этом маркер приобретает достаточно сложную структуру, внешний вид которой в общем случае можно представить следующим образом (M – значение индекса, N – область позиционирования):

{M, N:FormatString},

либо

{M:FormatString},

если не указывается значение области позиционирования.

Непосредственно за символом форматирования может быть расположена целочисленная ограничительная константа, которая, в зависимости от типа выводимого значения, может определять количество выводимых знаков после точки либо общее количество выводимых символов. При этом дробная часть действительных значений округляется или дополняется нулями справа. При выводе целочисленных значений ограничительная константа игнорируется, если количество выводимых символов превышает ее значение. В противном случае выводимое значение слева дополняется нулями.

Следующие примеры иллюстрируют варианты применения маркеров со строками форматирования:

```
Console.WriteLine("Integer fotmating - {0:D3},{1:D5}",
12345, 12);
Console.WriteLine("Currency formatting - {0:C},{1:C5}",
99.9, 999.9);
Console.WriteLine("Exponential formatting - {0:E}",
1234.5);
Console.WriteLine("Fixed Point formatting - {0:F3}",
1234.56789);
Console.WriteLine("General formatting - {0:G}",
1234.56789);
Console.WriteLine("Number formatting - {0:N}",
1234567.89);
Console.WriteLine("Hexadecimal formatting -
{0:X7}",12345);//Integers only!
```