Платформа .NET и язык программирования C#

Символьный и строковый типы данных

Символьный тип данных

Символы в C# хранятся в кодировке UTF-16.

Для объявления переменной символьного типа используется зарезервированное слово char (от англ. *character* — символ).

Символьная константа — это символ в одинарных кавычках.

Например,
```
char letter = 'z';
```

Значение символьной переменной можно задать про помощи явного преобразования номера символа в кодовом пространстве Юникод в символьный тип.

Например,
```
letter = (char)0x42F;     //задает символ 'Я'
```

Структура char, описывающая символьный тип содержит полезные методы, позволяющие изменить регистр символа, преобразовать его в строку, узнать, является ли символ буквой, цифрой и т. д.

Например,
```
char lowerCaseLetter = char.ToLower(letter);   // задает символ 'я'
```

Строковый тип данных

Строка — это упорядоченная последовательность символов.

Для задания переменной строкового типа в C# используется ключевое слово string.

Строковая константа задается в двойных кавычках.

Например,
```
string str = "Hello, World!";
```

К символу строки можно обратиться по его индексу (его номеру в строке, нумерация начинается с нуля). Для этого после имени строковой переменной в квадратных скобках указывается индекс символа.

Например,
```
char letter = str[7];  // задает символ 'W'
```

Можно задавать пустую строку, не содержащую символов:
```
str = "";
```

Строковые операции и методы

Операция	**Знак**
Конкатенация (слияние строк)	+

Свойство	**Описание**
str.Length	Длина строки str

Метод	**Описание**
str.IndexOf(strN)	Возвращает позицию первого вхождения строки strN в строку str
str.LastIndexOf(strN)	Возвращает позицию последнего вхождения строки strN в строку str

Строковые операции и методы

Метод	Описание
str.ToLower()	Переводит все буквы строки str в нижний регистр
str.ToUpper()	Переводит все буквы строки str в верхний регистр
str.Substring(intStart, intCount)	возвращает intCount символов строки str с позиции intStart

Методы изменения регистра и поиска подстроки не изменяют исходную строку, а создают новую строку.

str.Split(myChars)	Разбивает строку str на массив строк по разделителям из myChars (если myChars — пустой или null, то разделитель — пробел).

Перевод строки в число

Неявное или явное преобразование типа string в числовые типы не работает.
Для преобразования следует использовать один из двух способов:

тип.Parse(выражение)

Например:
int.Parse(myString)

или (считается устаревшим)
Convert.ToСистемныйТип(выражение)

Например:
Convert.ToInt32(myString)

Esc-последовательности

Сочетания символов, состоящих из косой черты (\), за которой следует буква или набор цифр, называются **escape-последовательностями**.

Для представления знака новой строки, кавычки или некоторых других символов в символьной константе, необходимо использовать escape-последовательности.

Escape-последовательность рассматривается как один символ.

\n — новая строка

\t — горизонтальная табуляция

\' — одиночная кавычка

\" — двойная кавычка

\\ — обратная косая черта

\r — возврат каретки

\xHHHH — символ юникода в шестнадцатеричной записи

\b — backspace

Буквальные строки

Буквальные строки используются для удобства использования и чтения, если текст строки содержит символы обратной косой черты, например в путях к файлу.

Чтобы получить буквальную строку, перед ней ставится знак @

Например:

```
string address = @"C:\Users\Student\Documents\";
```

Для встраивания двойных кавычек в буквальную строку используются удвоенные двойные кавычки.

Например:

```
var title = @"Курсовая работа ""Философские основы информатики""";
```

Метод string.Format

Метод возвращает отформатированную по заданному формату строку. В качестве исходного материала для форматирования берется список данных.

Использование:
string.Format("строка с элементами форматирования {}", аргумент1, …)

Каждый элемент форматирования имеет вид:
{индекс , выравнивание : строкаФормата}

индекс определяет номер элемента последующего списка

выравнивание — целое со знаком, указывает желательную ширину поля форматирования

строкаФормата должна соответствовать типу форматируемого объекта

Метод string.Format

Пример:
string.Format("x = {0, 6:F4} y = {1:G}", x, y)

В методе Console.WriteLine(…); вызов string.Format можно опускать, оставляя только строку с элементами форматирования и список аргументов, например:

Console.WriteLine("x = {0, 6:F4} y = {1:G}", x, y);

Подробно о форматах можно прочитать в справке.

Популярные форматы

- *Числовые:*
  G (общий), **F** (с фиксированной точкой),
  E (экспоненциальный), **X** (шестнадцатеричный),
  P (проценты), **C** (денежный)

- *Дата-время*:
  G или **g** (длинная/короткая запись общего вида),
  D или **d** (длинная/короткая запись даты),
  T или **t** (длинная/короткая запись времени),
  F или **f** (длинная/короткая запись даты и времени),
  M или **m** (запись месяца и числа),
  Y или **y** (запись месяца и года),
  составной формат даты-времени, в котором
  y — год, **M** — месяц, **d** — день, **H** или **h** — час (в 24- или 12-часовой системе) , **m** — минуты, **s** — секунды,
  f — доли секунд.

  Например {0: HH:mm dd.MM.yyyy} даст 12:30 31.10.2022

Интерполированная строка

Специальный знак **$** идентифицирует строку как интерполированную.

Интерполированная строка — это строка, которая может содержать выражения интерполяции.

Структура элемента интерполяции в C#:

{выражение,выравнивание:формат}

выражение — определяет значение, которое нужно отформатировать

выравнивание — константа, которая определяет минимальное количество символов, которое отводится для строкового представления значения выражения

формат — строка форматирования (как в методе String.Format())

CultureInfo

Класс CultureInfo предоставляет сведения об определенном языке и региональных параметрах, которые включают имена языков и региональных параметров, систему письма, используемый календарь, порядок сортировки строк и форматы дат и чисел.

Входит в пространство имен System.Globalization

Свойства:

CurrentCulture — возвращает или задает объект CultureInfo, используемые текущим вычислительным потоком.

DefaultThreadCurrentCulture — возвращает или задает объект CultureInfo, используемые по умолчанию вычислительными потоками приложения.

InvariantCulture — возвращает объект CultureInfo, не зависящий от языка и региональных параметров.

Список имен региональных параметров (Table of Language Culture Names) можно посмотреть в справке.