Строки

11 КЛАСС. ЧАСТЬ 2

План занятия

1. Строковые массивы (массивы символов)
2. Строка типа String
3. Функции работы со строками
4. Задания

СТРОКОВЫЕ КОНСТАНТЫ

Для объявления в программе константной строки вам необходимо заключить содержимое строки в двойные кавычки ("My string").

Вы можете делать это практически в любом месте программы: в передаче параметров функции, в инициализации переменных. Мы уже неоднократно применяли строковые константы при выводе данных на экран.

System.Console.WriteLine("Самые большие мониторы имеют размер: {0}", (int) Screens.SuperLarge);

Здесь в качестве одного из параметров функции используется строка "Самые большие мониторы имеют размер: {0}".

string strMessage = "Здравствуй Мир!";

В данном случае константная строка «Здравствуй Мир!» инициализирует переменную strMessage.

СТРОКИ C#

- **Строки в Си** # - это объекты класса String, значением которых является текст. Для работы со строками в этом классе определено множество методов (функций) и в этом уроке мы рассмотрим некоторые из них.

  Чтобы использовать строку, ее нужно сначала создать – присвоить какое-либо значение, иначе мы получим ошибку: "Использование локальной переменной "[имя переменной]", которой не присвоено значение".

ОБЪЯВЛЕНИЕ СТРОК

- Объявим простую строку и выведем ее на экран:

```
static void Main(string[] args)
{
  string s = "Hello, World!";
  Console.WriteLine(s);
}
```

```
static void Main(string[] args)
{
  string s;
  Console.WriteLine(s); // ошибка, строка не создана
}
```

ОБЪЕДИНЕНИЕ (КОНКАТЕНАЦИЯ) СТРОК

- Для **объединения** (конкатенации) строк используется оператор "+".

```
string s = "Hello," + " World!";
```

ДОСТУП К СИМВОЛУ ИЗ СТРОКИ

- Оператор "[]" используется для доступа (только чтение) к символу строки по индексу:

```
string s = "Hello, World!";
char c = s[1]; // 'e'
```

ДЛИНА СТРОКИ

- Свойство **Length** возвращает длину строки.
- string s = "Hello, World!";
- Console.WriteLine(s.Length);

СПЕЦСИМВОЛЫ

- Символ "\" является служебным, поэтому, чтобы использовать символ обратного слэша необходимо указывать его дважды "\\".

Символ табуляции – "\t"
Символ перевода строки – "\r\n"
Двойные кавычки – "\""

МЕТОДЫ (ФУНКЦИИ) КЛАССА STRING ДЛЯ РАБОТЫ СО СТРОКАМИ В C#

- **Как проверить, пуста ли строка?**

  Метод **IsNullOrEmpty()** возвращает True, если значение строки равно null, либо когда она пуста (значение равно ""):

```
static void Main(string[] args)
{
   string s1 = null, s2 = "", s3 = "Hello";
   String.IsNullOrEmpty(s1); // True
   String.IsNullOrEmpty(s2); // True
   String.IsNullOrEmpty(s3); // False
}
```

- Метод **IsNullOrWhiteSpace()** работает как и метод IsNullOrEmpty(), только возвращает True еще и тогда, когда строка представляет собой набор символов пробела и/или табуляции ("\t"):

```
static void Main(string[] args)
{
   string s1 = null, s2 = "\t", s3 = " ", s4 = "Hello";
   String.IsNullOrWhiteSpace(s1); // True
   String.IsNullOrWhiteSpace(s2); // True
   String.IsNullOrWhiteSpace(s3); // True
   String.IsNullOrWhiteSpace(s4); // False
}
```

- **Как проверить, является ли одна строка "больше" другой?**

  Для сравнивания строк используется метод **Compare()**. Суть сравнения строк состоит в том, что проверяется их отношение относительно алфавита. Строка "a" "меньше" строки "b", "bb" "больше" строки "ba". Если обе строки равны - метод возвращает "0", если первая строка меньше второй – "-1", если первая больше второй – "1":

```
static void Main(string[] args)
{
  String.Compare("a", "b"); // возвращает -1
  String.Compare("a", "a"); // возвращает 0
  String.Compare("b", "a"); // возвращает 1
  String.Compare("ab", "abc"); // возвращает -1
  String.Compare("Romania", "Russia"); // возвращает -1
  String.Compare("Rwanda", "Russia"); // возвращает 1
  String.Compare("Rwanda", "Romania"); // возвращает 1
}
```

  Чтобы игнорировать регистр букв, в метод нужно передать, как третий аргумент true.

```
String.Compare("ab", "Ab"); // возвращает -1
String.Compare("ab", "Ab", true); // возвращает 0
```

- **Как перевести всю строку в верхний/нижний регистр?**

  Для этого используются методы **ToUpper()** и **ToLower()**:

```
static void Main(string[] args)
{
   string s = "Hello, World";
   Console.WriteLine(s.ToUpper()); // выводит "HELLO, WORLD"
   Console.WriteLine(s.ToLower()); // выводит "hello, world"
   Console.ReadLine();
}
```

- **Как проверить, содержит ли строка подстроку?**

  Для проверки содержания подстроки строкой используется метод **Contains()**. Данный метод принимает один аргумент – подстроку. Возвращает True, если строка содержит подстроку, в противном случае – False. Пример:

```
static void Main(string[] args)
{
  string s = "Hello, World";

  if (s.Contains("Hello"))
    Console.WriteLine("Содержит");
  Console.ReadLine();
}
```

- Данная программа выводит слово "Содержит", так как "Hello, World" содержит подстроку "Hello".

- **Как найти индекс первого символа подстроки, которую содержит строка?**

  Метод **IndexOf()** возвращает индекс первого символа подстроки, которую содержит строка. Данный метод принимает один аргумент – подстроку. Если строка не содержит подстроки, метод возвращает "-1". Пример:

```
static void Main(string[] args)
{
   string s = "Hello, World";
   Console.WriteLine(s.IndexOf("H")); // 0
   Console.WriteLine(s.IndexOf("World")); // 7
   Console.WriteLine(s.IndexOf("Zoo")); // -1
   Console.ReadLine();
}
```

- **Как узнать, начинается/заканчивается ли строка указанной подстрокой?**

  Для этого используются соответственно методы **StartsWith()** и **EndsWith()**, которые возвращают логическое значение. Пример:

```
static void Main(string[] args)
{
   string s = "Hello, World";
   Console.WriteLine(s.StartsWith("Hello")); // True
   Console.WriteLine(s.StartsWith("World")); // False
   Console.WriteLine(s.EndsWith("World")); // True
   Console.ReadLine();
}
```

- **Как вставить подстроку в строку, начиная с указанной позиции?**

  Метод **Insert()** используется для вставки подстроки в строку, начиная с указанной позиции. Данный метод принимает два аргумента – позиция и подстрока. Пример:

```
static void Main(string[] args)
{
   string s = "Hello World";
   Console.WriteLine(s.Insert(5,",")); // вставляет запятую на 5 позицию
   Console.ReadLine();
}
```

- **Как обрезать строку, начиная с указанной позиции?**

  Метод **Remove()** принимает один аргумент – позиция, начиная с которой обрезается строка:

```
static void Main(string[] args)
{
   string s = "Hello, World";
   Console.WriteLine(s.Remove(5)); // удаляем все символы, начиная с 5 позиции, на экран выведется "Hello"
   Console.ReadLine();
}
```

- В метод Remove() можно передать и второй аргумент – количество обрезаемых символов. Remove(3, 5) – удалит из строки пять символов начиная с 3-го.

- **Как получить подстроку из строки, начиная с указанной позиции?**

  Для этого используется метод **Substring()**. Он принимает один аргумент – позиция, с которой будет начинаться новая подстрока:

```
static void Main(string[] args)
{
   string s = "Hello, World";
   Console.WriteLine(s.Substring(7)); // получаем строку начиная с 7 позиции, выведет "World"
   Console.ReadLine();
}
```

- В метод Substring(), как в метод Remove() можно передать и второй аргумент – длина подстроки. Substring (3, 5) – возвратит подстроку длиной в 5 символов начиная с 3-й позиции строки.

- **Как заменить в строке все подстроки указанной новой подстрокой?**

  Метод **Replace()** принимает два аргумента – подстрока, которую нужно заменить и новая подстрока, на которую будет заменена первая:

```
static void Main(string[] args)
{
   string s = "Hello, World, Hello";
   Console.WriteLine(s.Replace("Hello", "World"));
//выведет "World, World, World"
   Console.ReadLine();
}
```

- **Как преобразовать строку в массив символов?**

  Метод **ToCharArray()** возвращает массив символов указанной строки:

```
static void Main(string[] args)
{
  string s = "Hello, World";
  char[] array = s.ToCharArray(); // элементы массива – 'H', 'e', 'l', 'l'...
}
```

- **Как разбить строку по указанному символу на массив подстрок?**

  Метод **Split()** принимает один аргумент - символ, по которому будет разбита строка. Возвращает массив строк. Пример:

```
static void Main(string[] args)
{
  string s = "Arsenal,Milan,Real Madrid,Barcelona";
  string[] array = s.Split(','); // элементы массива – "Arsenal", "Milan", "Real Madrid",
"Barcelona"
}
```

НЕИЗМЕНЯЕМЫЕ СТРОКИ

- Стоит знать, что объекты класса String представляют собой неизменяемые (Immutable) последовательности символов Unicode. Когда вы используете любой метод по изменению строки (например Replace()), он возвращает новую измененную копию строки, исходные же строки остаются неизменными. Так сделано потому, что операция создания новой строки гораздо менее затратна, чем операции копирования и сравнения, что повышает скорость работы программы. В C# также есть класс **StringBuilder**, который позволяет изменять строки.

ЗАДАНИЕ 1

- Есть некий текст. Необходимо заменить в этом тексте все слова "Nikolay" на "Oleg".

ЗАДАНИЕ 2

Дан текст – «Сегодня мы с вами рассмотрели, как работать со строками в C#. Были описаны основные операторы и методы, которые используются для работы со строками». Обрежьте этот текст так, чтобы осталась только часть «Были описаны основные операторы и методы».

ЗАДАНИЕ 3

Дана строка, которая содержит имена пользователей, разделенные запятой – "Login1,LOgin2,login3,loGin4". Необходимо разбить эту строку на массив строк (чтобы отдельно были логины), и перевести их все в нижний регистр.

ЗАДАНИЕ И.1

- Дано слово, состоящее из четного числа букв. Вывести на экран его первую половину, не используя циклов.

ЗАДАНИЕ И.2

- Дано слово из 12 букв. Поменять местами его трети следующим образом:
- а) первую треть слова разместить на месте третьей, вторую треть — на месте
- первой, третью треть — на месте второй;
- б) первую треть слова разместить на месте второй, вторую треть — на месте
- третьей, третью треть — на месте первой.

ЗАДАНИЕ И.3

- Дано слово. Перенести первые k его букв в конец.
- Задачу решить двумя способами:
- 1) без использования оператора цикла;
- 2) с использованием оператора цикла.

ЗАДАНИЕ И.4

- Дано предложение. Определить:
- а) число вхождений в него буквосочетания ро;
- б) число вхождений в него некоторого буквосочетания из двух букв;
- в) число вхождений в него некоторого буквосочетания.

ЗАДАНИЕ И.5

- Дано предложение. В нем слова разделены одним пробелом (начальные и ко-нечные пробелы и символ "-" в предложении отсутствуют). Определить коли-чество слов в предложении.

ЗАДАНИЕ И.6

- Дано предложение. В нем слова разделены одним или несколькими пробела-ми (символ "-" в предложении отсутствует). Определить количество слов
- в предложении. Рассмотреть два случая:
- 1) начальные и конечные пробелы в предложении отсутствуют;
- 2) начальные и конечные пробелы в предложении имеются.

ЗАДАНИЕ И.7

- Дано предложение. Определить, каких букв в нем больше: м или н.

ЗАДАНИЕ И.8

- Дано предложение, в котором имеются буквы с и Т. Определить, какая из них
- встречается позже (при просмотре слова слева направо). Если таких букв не-сколько, то должны учитываться последние из них. Оператор цикла с услови-ем не использовать.

ЗАДАНИЕ И.9

- Дано предложение, в котором имеется несколько букв е. Найти:
- а) порядковый номер первой из них;
- б) порядковый номер последней из них.

ЗАДАНИЕ И.10

- Дано предложение. Заменить в нем все вхождения буквосочетания бит на рог.

ЗАДАНИЕ И.11

- Дан текст, в котором имеются цифры.
- а) Найти их сумму.
- б) Найти максимальную цифру.

РАСПРЕДЕЛЕНИЕ ЗАДАНИЙ ПО ВАРИАНТАМ

№ варианта	1	2	3	4
1	+	+	+	И.1
2	+	+	+	И.2
3	+	+	+	И.3
4	+	+	+	И.4
5	+	+	+	И.5
6	+	+	+	И.6
7	+	+	+	И.7
8	+	+	+	И.8
9	+	+	+	И.9
10	+	+	+	И.10
11	+	+	+	И.11