Программирование на C#

Часть 2

Вывод на консоль нескольких значений

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "Tom";
            int age = 34;
            double height = 1.7;
            Console.WriteLine("Имя: {0}  Возраст: {2}  Рост: {1}м", name, height, age);

            Console.ReadKey();
        }
    }
}
```

Вывод на консоль нескольких значений

```
using System;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "Tom";
            int age = 34;
            double height = 1.7;
            Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м");

            Console.ReadKey();
        }
    }
}
```

Консольный ввод

```
1  using System;
2
3  namespace HelloApp
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.Write("Введите свое имя: ");
10             string name = Console.ReadLine();
11             Console.WriteLine($"Привет {name}");
12
13             Console.ReadKey();
14         }
15     }
16 }
```

```
Введите свое имя: Том
Привет Том
```

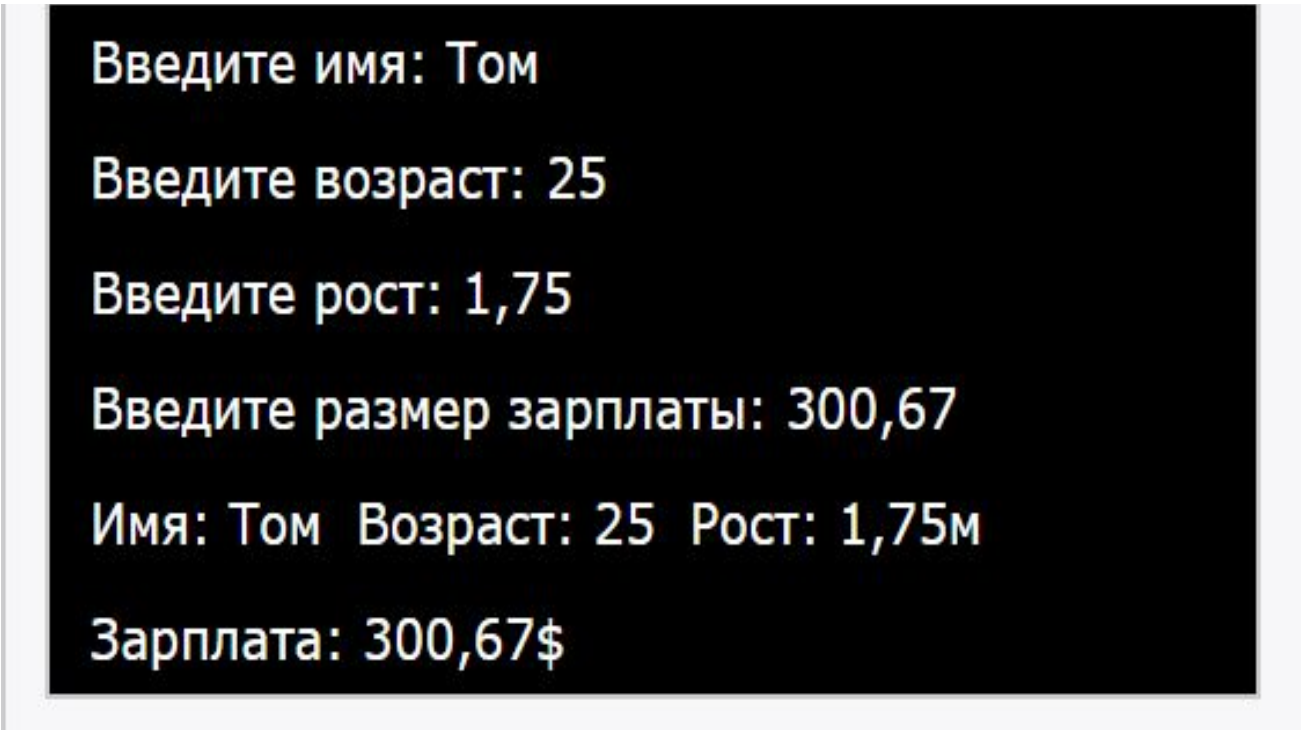
Некоторые методы для преобразования к типам

- **Convert.ToInt32()** (преобразует к типу int)
- **Convert.ToDouble()** (преобразует к типу double)
- **Convert.ToDecimal()** (преобразует к типу decimal)

Пример ввода значений

```
    {
        static void Main(string[] args)
        {
            Console.Write("Введите имя: ");
            string name = Console.ReadLine();

            Console.Write("Введите возраст: ");
            int age = Convert.ToInt32(Console.ReadLine());

            Console.Write("Введите рост: ");
            double height = Convert.ToDouble(Console.ReadLine());

            Console.Write("Введите размер зарплаты: ");
            decimal salary = Convert.ToDecimal(Console.ReadLine());

            Console.WriteLine($"Имя: {name}  Возраст: {age}  Рост: {height}м  Зарплата: {salary}$");

            Console.ReadKey();
        }
    }
}
```

Пример работы программы



Работа с файлами

- **Файл** – это набор данных, который хранится на внешнем запоминающем устройстве (например на жестком диске).
- Файл имеет имя и расширение. Расширение позволяет идентифицировать, какие данные и в каком формате хранятся в файле.

  Под работой с файлами подразумевается:

  - создание файлов;
  - удаление файлов;
  - чтение данных;
  - запись данных;
  - изменение параметров файла (имя, расширение…);
  - другое.

Работа с файлами

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
```

Создание файла

```
static void Main(string[] args)
{
   File.Create("D:\\new_file.txt");
}
```

```
static void Main(string[] args)
{
   File. WriteAllText("D:\\new_file.txt", "текст");
}
```

```
static void Main(string[] args)
{
   File.AppendAllText("D:\\new_file.txt", "текст метода AppendAllText ()"); //
допишет текст в конец файла
}
```

Удаление файла

```
static void Main(string[] args)
{
  File.Delete("d:\\test.txt"); //удаление файла
}
```

Чтение/запись в файл. Потоки

- **Поток** – это абстрактное представление данных (в байтах), которое облегчает работу с ними. В качестве источника данных может быть файл, устройство ввода-вывода, принтер.

  Класс **Stream** является абстрактным базовым классом для всех потоковых классов в Си-шарп. Для работы с файлами нам понадобится класс **FileStream**(файловый поток).

  FileStream - представляет поток, который позволяет выполнять операции чтения/записи в файл.

```
static void Main(string[] args)
{
  FileStream file = new FileStream("d:\\test.txt", FileMode.Open
, FileAccess.Read); //открывает файл только на чтение
}
```

- Режимы открытия **FileMode**:

  - *Append* – открывает файл (если существует) и переводит указатель в конец файла (данные будут дописываться в конец), или создает новый файл. Данный режим возможен только при режиме доступа FileAccess.Write.
  - *Create* - создает новый файл(если существует – заменяет)
  - *CreateNew* – создает новый файл (если существует – генерируется исключение)
  - *Open* - открывает файл (если не существует – генерируется исключение)
  - *OpenOrCreate* – открывает файл, либо создает новый, если его не существует
  - *Truncate* – открывает файл, но все данные внутри файла затирает (если файла не существует – генерируется исключение)

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\file1.txt", FileMode.CreateNew); //
создание нового файла
  FileStream file2 = new FileStream("d:\\file2.txt", FileMode.Open); //открытие
существующего файла
  FileStream file3 = new FileStream("d:\\file3.txt", FileMode.Append); //
открытие файла на дозапись в конец файла
}
```

- Режим доступа **FileAccess**:

  - *Read* – открытие файла только на чтение. При попытке записи генерируется исключение
  - *Write* - открытие файла только на запись. При попытке чтения генерируется исключение
  - *ReadWrite* - открытие файла на чтение и запись.

Чтение из файла

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Open); //создаем файловый поток
  StreamReader reader = new StreamReader(file1); // создаем «потоковый читатель» и связываем его с файловым потоком
  Console.WriteLine(reader.ReadToEnd()); //считываем все данные с потока и выводим на экран
  reader.Close(); //закрываем поток
  Console.ReadLine();
}
```

Запись в файл

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Create); //создаем
файловый поток
  StreamWriter writer = new StreamWriter(file1); //создаем «потоковый
писатель» и связываем его с файловым потоком
  writer.Write("текст"); //записываем в файл
  writer.Close(); //закрываем поток. Не закрыв поток, в файл ничего не
запишется
}
```

Кодировка

- **Кодировка**, в которой будут считываться/записываться данные указывается при создании StreamReader/StreamWriter:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Open);
  StreamReader reader = new StreamReader(file1, Encoding.Unicode);
  StreamWriter writer = new StreamWriter(file1, Encoding.UTF8);
}
```

Работа со строками. Класс String

```
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
}
```

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

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

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

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

  Свойство **Length** возвращает длину строки.

Методы (функции) класса String для работы со строками

- **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

```
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
}
```

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

  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.

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

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

- **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".

```
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.

```
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.

```
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.

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