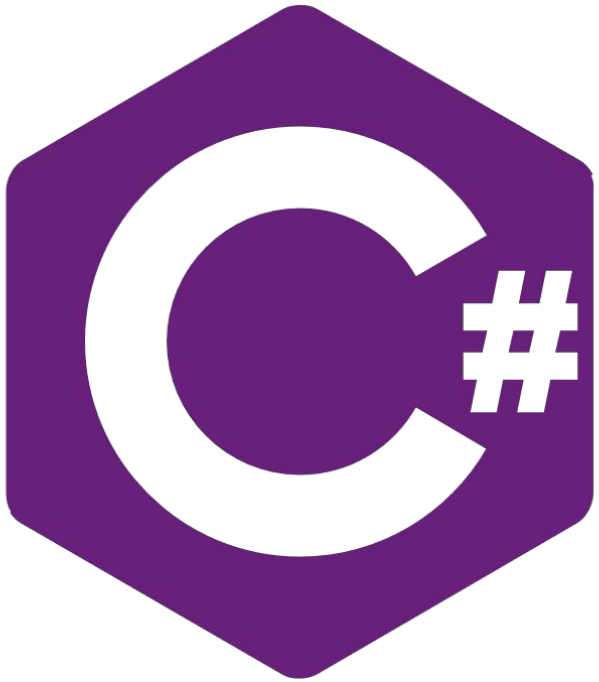


Работа с файлами.
Открытие и чтение из файла.
Форматирование данных.
Работа со списком.

Работа с файлами.
Открытие и чтение из файла.

Подключаем пространство имён для работы с потоками ввода вывода

```
using System.IO;
```

Используем статический класс File

Вызываем метод ReadAllText

```
File.Rea
    OpenRead
    ReadAllBytes
    ReadAllLines
    ReadAllText
    ReadLines
```

Работа с файлами.
Открытие и чтение из файла.

ReadAllText

path – имя файла

encoding – кодировка

```
string File.ReadAllText(string path) (+ 1 overload)
Opens a text file, reads all lines of the file, and then closes the file.
```

```
string File.ReadAllText(string path, Encoding encoding)
Opens a file, reads all lines of the file with the specified encoding, and then closes the file.
encoding: The encoding applied to the contents of the file.
```

```
File.ReadAllText("имя_файла.txt");
```

Кириллица не поддерживается

```
File.ReadAllText("имя_файла.txt", Encoding.Default);
```

`Encoding.Default` - Подключение Кириллицы

Работа с файлами.
Открытие и чтение из файла.

Сохраняем содержимое файла в переменную типа String

```
String content = File.ReadAllText("имя_файла.txt", Encoding.Default);
```

имя_файла.txt

```
bmw     германия
mercedes        _германия
toyota  @       япония
mitsubishi  япония#
ford    usa
chevrolet *usa*
audi германия======
bentley  (='.'=)        великобритания*******
rangeRover      великобритания
citroen $франция$
%daewoo корея
(*-*)kia -> корея
faw китай
fiat италия
honda___ япония (-_-(-_-)-_-)
```

content

```
"bmw \tермания\r\nmercedes\t_германия\r\ntoyota\t@\tяпония\r\nmitsubishi
япония# \r\nford\tusa\r\nchevrolet *usa*\r\naudi гер ...
```

\n – перевод каретки на новую строку
\r – сдвиг каретки в начало строки
\t – табуляция

Форматирование данных

Избавляемся от лишних символов

```
string content = "bmw \trермания\r\nmercedes\t_германия\r\ntoyota\t@\tяпония\r\nmitsubishi
японня# \r\nford\tusa\r\nchevrolet *usa*\r\naudi rep
```

Используем метод Split для разбиения строки на массив данных

```
string[] mas = content.Spl
```

Split

string[] string.Split(params char[] separator) (+ 5 overloads)
Splits a string into substrings that are based on the characters in an array.

Форматирование данных

Создаем массив символов, не связанных с данными, которые будут исключены

```
char[] separators = new char[] { '\r', '\n', '\t', ' ', '@', '_', '#'};

string[] mas = content.Split(separators);
```

Если встретятся несколько подряд идущих символов из separators,
то образуются пустые ячейки при делении, например @@, ###,…

`StringSplitOptions.RemoveEmptyEntries` -> Удаляет пустые ячейки в результирующем массиве mas

```
string[] mas = content.Split(separators, StringSplitOptions.RemoveEmptyEntries);
```

Форматирование данных

В результате имеем все данные сохраненные в массиве

"bmw \trермания\r\nmercedes\t_германия\r\ntoyota\t@\tяпония\r\nmitsubishi

япония# \r\nford\tusa\r\nchevrolet *usa*\r\naudi rep…

content → Split → mas

[0]	"bmw"
[1]	"германия"
[2]	"mercedes"
[3]	"германия"
[4]	"toyota"
[5]	"япония"
[6]	"mitsubishi"
[7]	"япония"
[8]	"ford"
[9]	"usa"
[10]	"chevrolet"
[11]	"usa"
[12]	"audi"
[13]	"германия"
[14]	"bentley"

…

Создание списка данных

Создаем класс в соответствии со структурой данных

[0]	"bmw"
[1]	"германия"
[2]	"mercedes"
[3]	"германия"
[4]	"toyota"
[5]	"япония"
[6]	"mitsubishi"
[7]	"япония"
[8]	"ford"
[9]	"usa"
[10]	"chevrolet"
[11]	"usa"
[12]	"audi"
[13]	"германия"
[14]	"bentley"

```
class Brand
{
    public string name;
    public string country;
}
```

Далее создаём список брендов

```
List<Brand> brands = new List<Brand>();
```

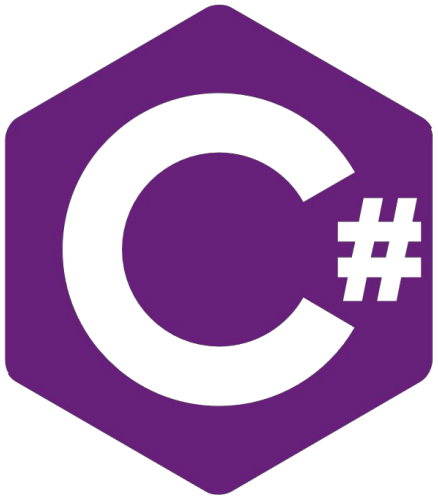
Создание списка данных

Заполняем список brands данными из массива mas

```
string content = File. ReadAllText ("данные.txt", Encoding.Default) ;

char[] separators = new char[] { '\r', '\n', '\t', ' ', '*', '_' } ;

string[] mas = content.Split (separators, StringSplitOptions.RemoveEmptyEntries);

List<Brand> brands = new List<Brand>();

for (int i = 0; i < mas.Length / 2; i++)
{
    brands.Add(new Brand { name = mas[2 * i], country = mas[2 * i + 1] });
}
```

Делим на 2 т.к. объект состоит из двух полей

Имена брендов стоят на чётных позициях а страны производители на нечетных



Готовая программа

```
class Brand
{
    public string name;
    public string country;
}

class Program
{
    static void Main(string[] args)
    {
        string content = File. ReadAllText ("данные.txt", Encoding.Default) ;

        char[] separators = new char[] {'\r', '\n', '\t', ' ', '@', '_', '#', '*', '=', '.', '\'', '(', ')', '%', '*', '-', '>'} ;

        string[] mas = content.Split (separators, StringSplitOptions.RemoveEmptyEntries);

        List<Brand> brands = new List<Brand>();

        for (int i = 0; i < mas.Length / 2; i++)
        {
            brands.Add(new Brand { name = mas[2 * i], country = mas[2 * i + 1] });
        }

        foreach (var item in brands)
        {
            Console.WriteLine(item.name + " " + item.country);
        }
    }
}
```

Заполните пропуски

```
class Department
{
    public string name;
    public int floor;
    public string phone;
}
```

```
class Employee
{
    public string name;
    public string dep_name;
    public string phone;
    public string position;
}
```

```
class Payment
{
    public string postion;
    public double salary;
}
```

```
List<Department> departments = new List<Department>();
List<Employee> employees = new List<Employee>();
List<Payment> payment = new List<Payment>();
var result = from [      ] in [                    ]
             join [      ] in [                    ] on [                    ] equals [                    ]
             join [      ] in [                    ] on [                    ] equals [                    ]
select new {eName = [                ], ePos = [                ], dphone = [                ], eSalary = [                ];
```

Компоненты:

Заполните пропуски

```
class Department
{
    public string name;
    public int floor;
    public string phone;
}
```

```
class Employee
{
    public string name;
    public string dep_name;
    public string phone;
    public string position;
}
```

```
class Payment
{
    public string postion;
    public double salary;
}
```

```
List<Department> departments = new List<Department>();
List<Employee> employees = new List<Employee>();
List<Payment> payment = new List<Payment>();
var result = from [   ] in [   ]

             join [   ] in [   ] on [   ] equals [   ]

             join [   ] in [   ] on [   ] equals [   ]

select new {eName = [   ], ePos = [   ], dphone = [   ], eSalary = [   ] };
```

Компоненты: