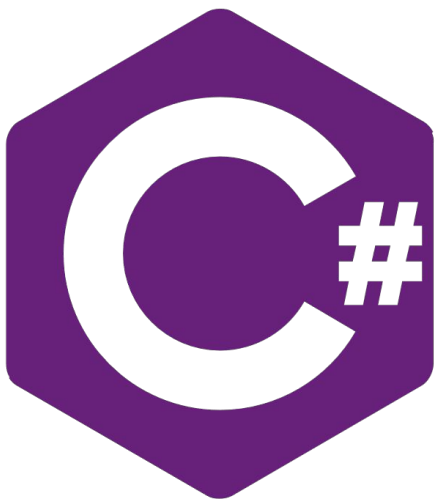


Работа с файлами.
Открытие и чтение из файла.
Форматирование данных.
Работа со списком.



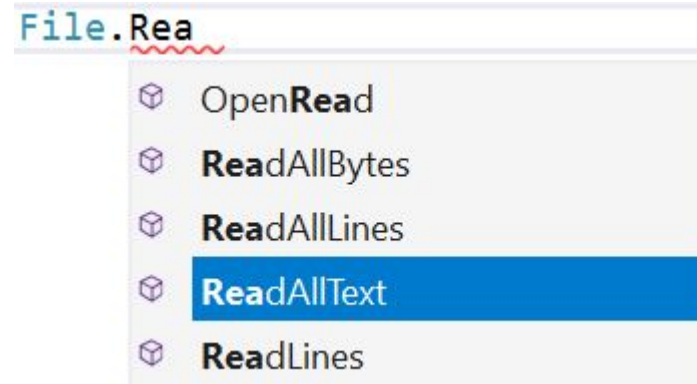
Работа с файлами. Открытие и чтение из файла.

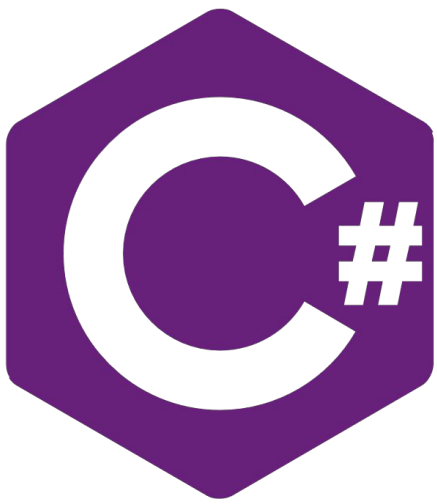
Подключаем пространство имён для работы с потоками ввода вывода

```
using System.IO;
```

Используем статический класс **File**

Вызываем метод `ReadAllText`





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.
```

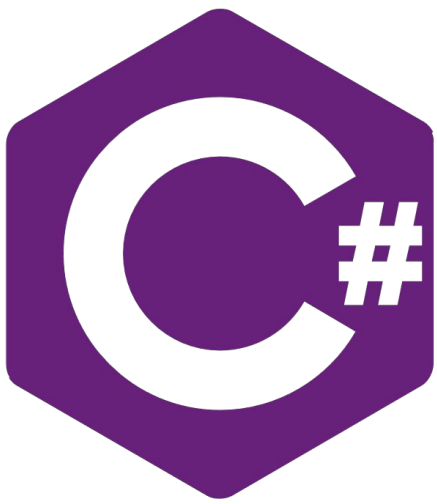
```
File.ReadAllText("имя_файла.txt");
```

Кириллица не
поддерживается

```
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", Encoding.Default);
```

Encoding.Default - Подключение
Кириллицы



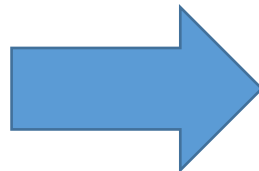
Работа с файлами. Открытие и чтение из файла.

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 корейя
(*-*)kia -> корейя
faw китай
fiat италия
honda__ япония (-_-(-_-)-_-)
```



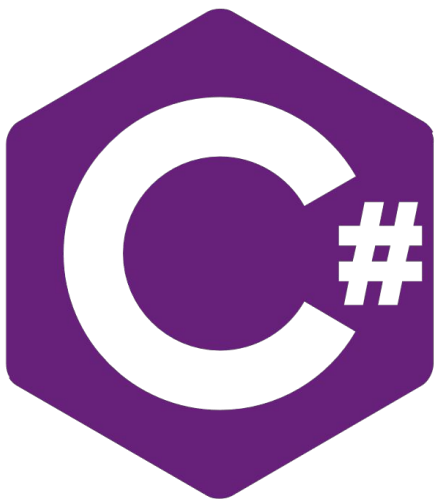
content

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

\n – перевод каретки на новую строку

\r – сдвиг каретки в начало строки

\t – табуляция



Форматирование данных

Избавляемся от лишних СИМВОЛОВ

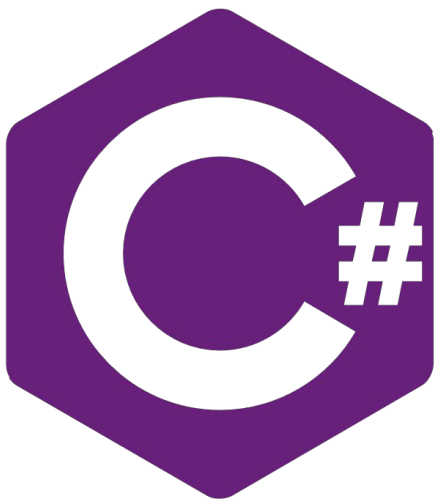
```
string content = "bmw \тремания\r\nmercedes\t_германия\r\ntoyota\t@\tяпония\r\nmitsubishi  
япония# \r\nford\tusa\r\nchevrolet *usa*\r\naudi rep
```

Используем метод Split для разбиения строки на массив данных

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

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

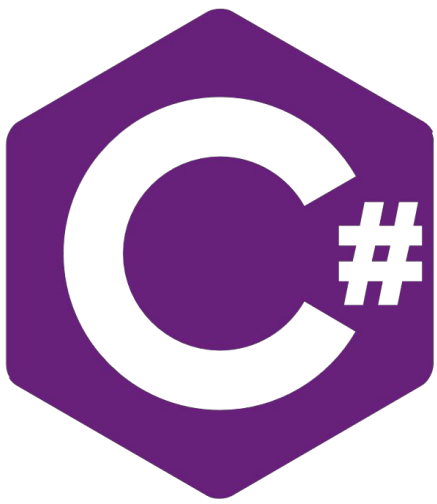
```
char[] separators = new char[] { '\r', '\n', '\t', ' ', '@', '_', '#' };
```

```
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);
```



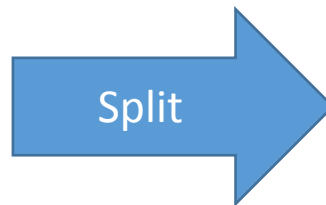
Форматирование данных

В результате имеем все данные сохраненные в массиве

```
"bmw \tгермания\r\nmercedes\t_германия\r\ntoyota\t@\tяпония\r\nmitsubishi
```

```
япония# \r\nford\tusa\r\nchevrolet *usa*\r\naudi rep...
```

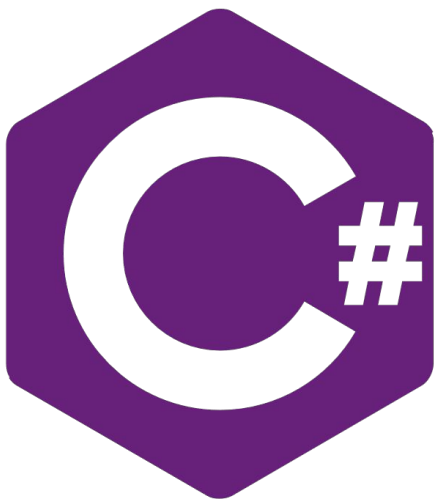
content



mas

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

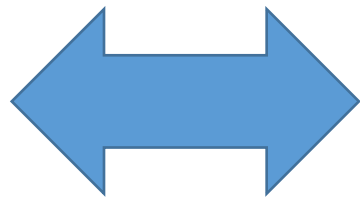
...



Создание списка данных

Создаем класс в соответствии со структурой данных

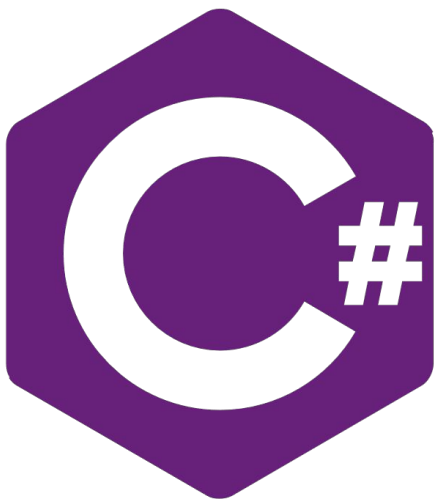
[0]	Q	"bmw"
[1]	Q	"германия"
[2]	Q	"mercedes"
[3]	Q	"германия"
[4]	Q	"toyota"
[5]	Q	"япония"
[6]	Q	"mitsubishi"
[7]	Q	"япония"
[8]	Q	"ford"
[9]	Q	"usa"
[10]	Q	"chevrolet"
[11]	Q	"usa"
[12]	Q	"audi"
[13]	Q	"германия"
[14]	Q	"bentley"



```
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

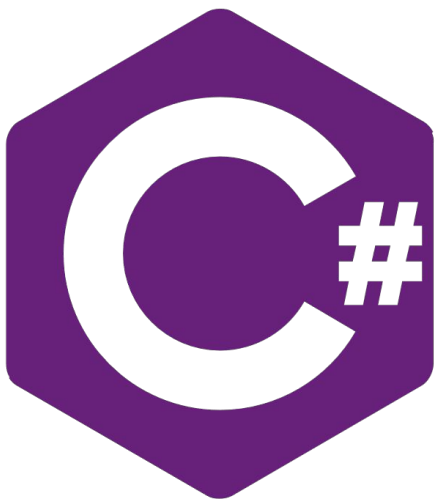
```
string content = File.ReadAllText ("данные.txt", Encoding.Default) ;  
char[] separators = new char[] { '\r', '\n', '\t', ' ', '*', '_' } ;  
string[] mas = content.Split (separators, StringSplitOptions.RemoveEmptyEntries);
```

```
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 нечётных



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: