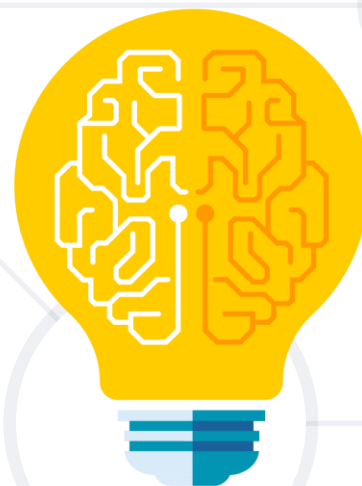


# Въведение в програмирането

Работа с конзола, аритметични операции с числа



СофтУни

Преподавателски екип



SoftUni

Софтуерен университет

<https://softuni.bg>

1. Какво означава да програмираме?
2. Да направим конзолна програма
3. Конзолни програми
4. Променливи и типове данни
5. Работа с конзола – четене и печатане
6. Дебъгване
7. Работа с числа



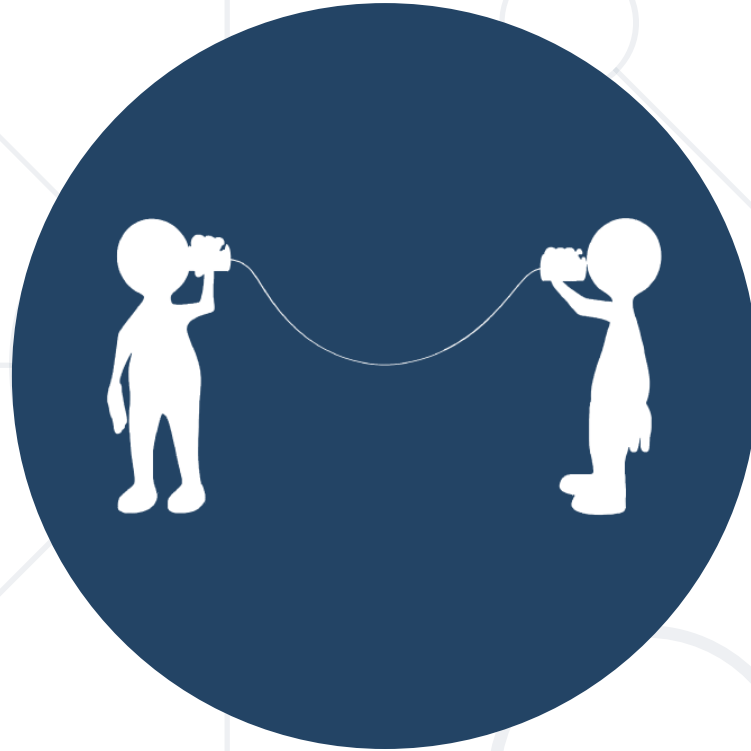


**Какво означава  
"да програмираме"?**

# Какво означава "програмиране"?

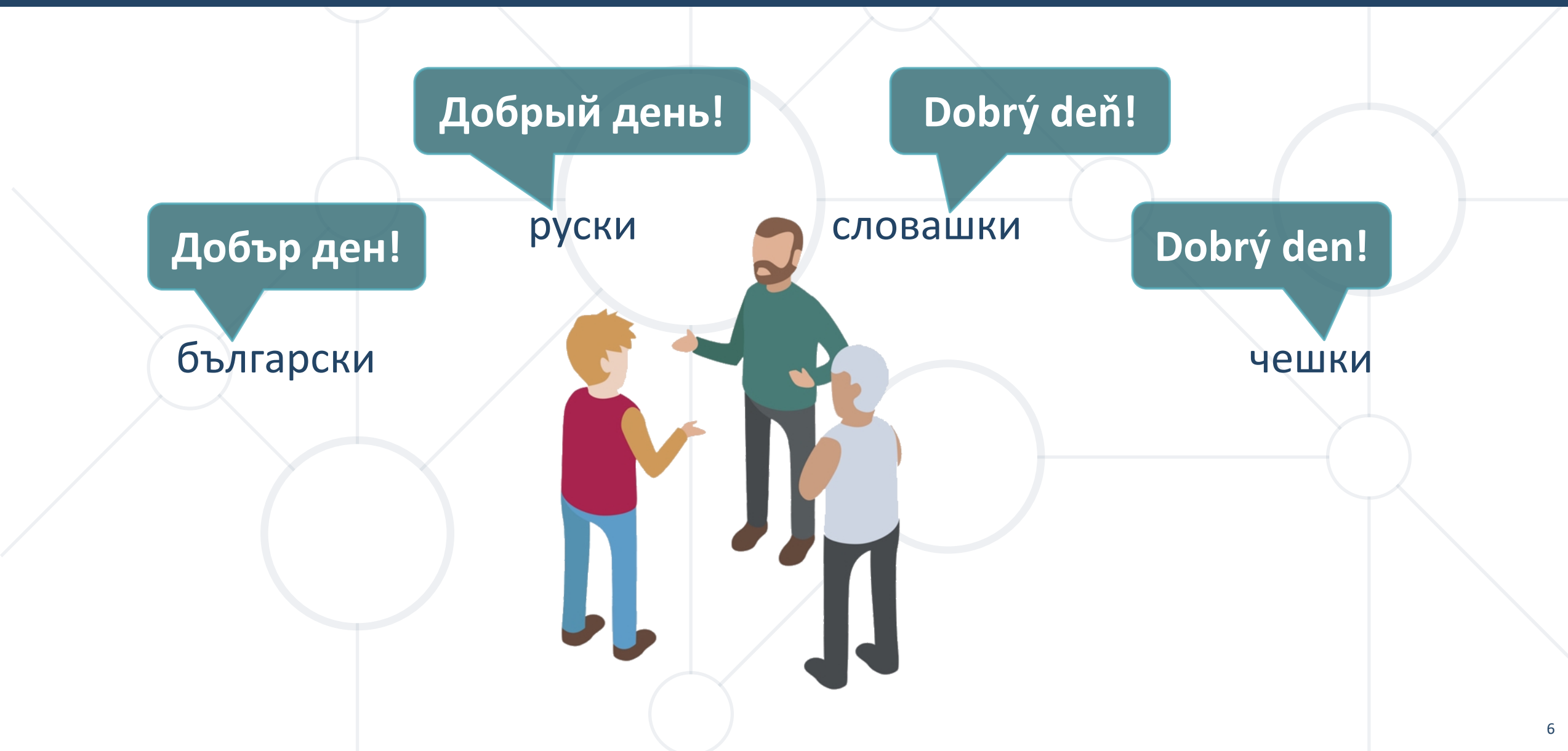
- Да даваме **команди** на компютъра – да "комуникираме"
- Командите се подреждат една след друга
- В поредица, те образуват "**компютърна програма**"





# **Езиците като начин на комуникация**

# Начин на комуникация ()



# Начин на комуникация (2)

```
Console.WriteLine("Hello");
```



```
print("Hello")
```

```
System.out.println("Hello");
```



```
console.log("Hello");
```

# Езици за програмиране

- Програмите се пишат на **език за програмиране**
- Например C#, Java, JavaScript, Python, PHP, C, C++, ...
- Използва се **среда за програмиране** (например Visual Studio)





- Програма == **последователност от команди**
  - Съдържа пресмятания, проверки, повторения, ...
- Програмите се пишат в текстов формат
  - Текстът на програмата се нарича **сорс код**
- Сорс кодът се компилира до изпълним файл
  - Например **Program.cs** се компилира до **Program.exe**



# Интересно за C#

- В **топ 5** на най-популярните езици за програмиране
- Около 31% от всички програмисти го използват редовно
- Третата по големина общност в **StackOverflow** с повече от **1.1 милиона** теми
- Глобално, **всеки месец** се предлагат повече от **17 000** C# позиции



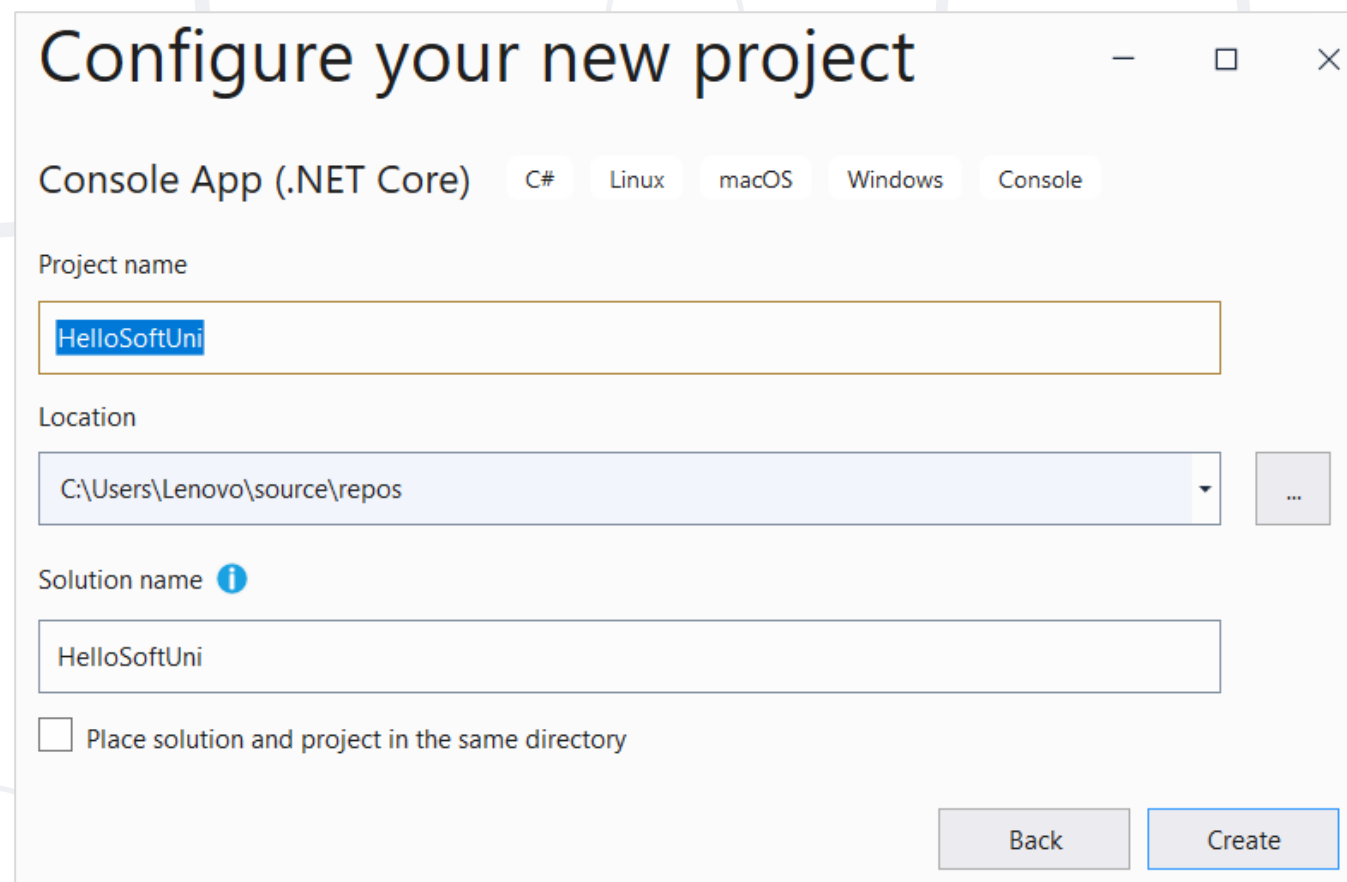
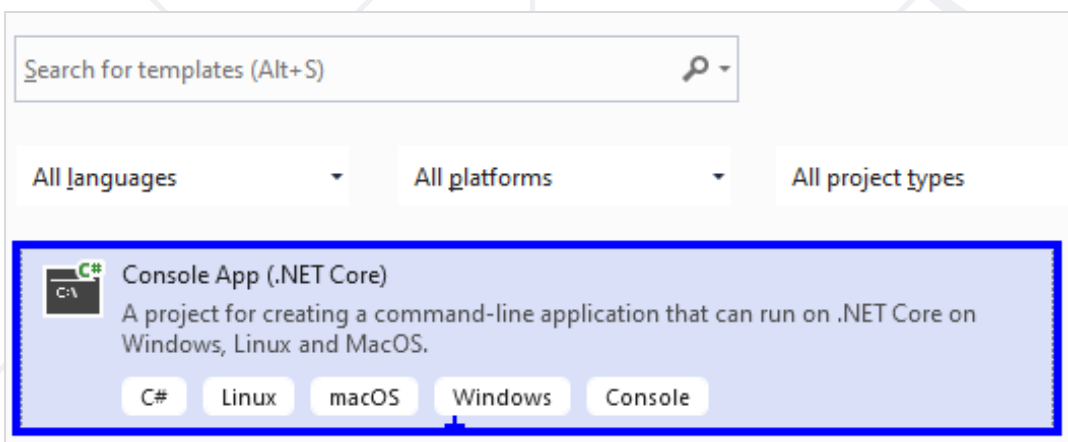


**Демонстрация на живо**

- За да програмирате, ви трябва среда за разработка
  - Integrated Development Environment (**IDE**)
  - **За C# → Visual Studio**; за Java → IntelliJ; за Python → PyCharm
- Инсталирайте си **Microsoft Visual Studio Community 2019**  
<https://visualstudio.com/products/visual-studio-community-vs>
- Visual Studio се предлага за: Windows, Linux, Mac OS

# Създаване на конзолна програма

- Стартирайте Visual Studio
- Нов конзолен проект – [Create a new project] → [Console App (.NET Core)]



# Писане на програмен код (1)

- Сорс кодът на програма се пише в секцията **Main(string[] args)**
  - Между отварящата и затварящата скоба { }
- Натиснете [Enter] след отварящата скоба {
- Кодът на програмата се пише отместен навътре

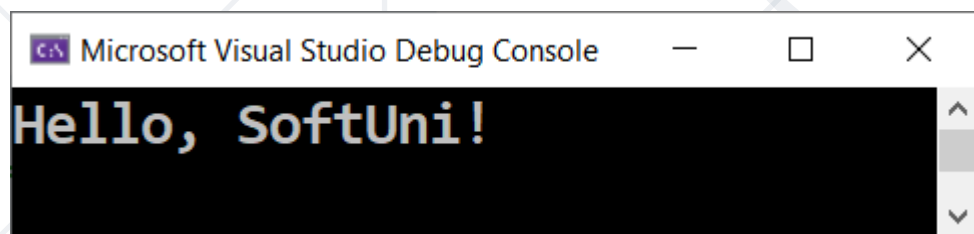
```
namespace HelloSoftUni
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            // TODO: Code here
        }
    }
}
```

- `Console.WriteLine("Hello SoftUni");`

**Напишете следния код:**

```
namespace HelloSoftUni
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello SoftUni!");
        }
    }
}
```

- За стартиране на програмата натиснете [Ctrl + F5]
- Ако няма грешки, програмата ще се изпълни
- Резултатът ще се изпише на конзолата



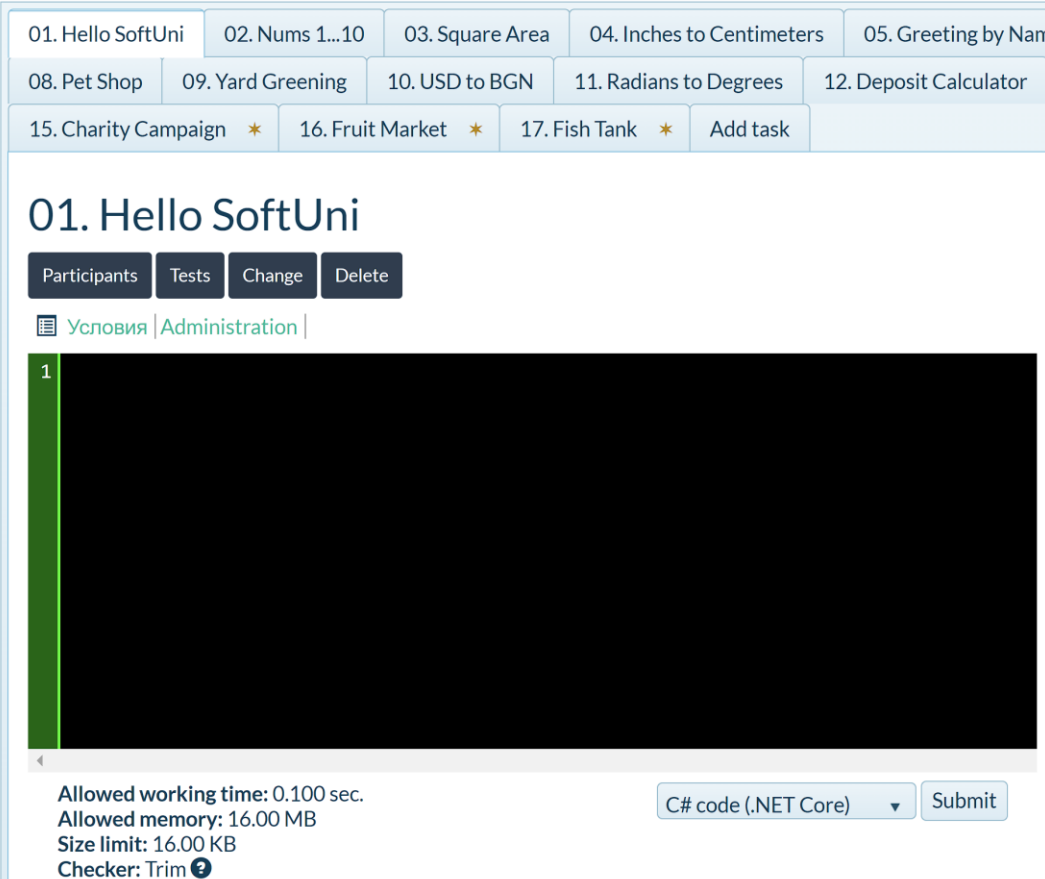


# Тестване на програмата в Judge

- Тествайте кода си в онлайн judge системата:

## Intro to Programming

Submit a solution



The screenshot shows the 'Intro to Programming' contest page. At the top, there's a grid of problem cards: 01. Hello SoftUni, 02. Nums 1...10, 03. Square Area, 04. Inches to Centimeters, 05. Greeting by Name, 08. Pet Shop, 09. Yard Greening, 10. USD to BGN, 11. Radians to Degrees, 12. Deposit Calculator, 15. Charity Campaign, 16. Fruit Market, 17. Fish Tank, and an 'Add task' button. Below this, the '01. Hello SoftUni' problem is selected. It has buttons for 'Participants', 'Tests', 'Change', and 'Delete'. There are links for 'Условия' (Conditions) and 'Administration'. A large black text area for code is shown with a green line number '1' on the left. At the bottom, it specifies 'Allowed working time: 0.100 sec.', 'Allowed memory: 16.00 MB', 'Size limit: 16.00 KB', and 'Checker: Trim'. A dropdown menu is set to 'C# code (.NET Core)' and a 'Submit' button is present.

Тествайте решението в Judge: <https://judge.softuni.bg/Contests/Practice/Index/3153#0>

# Типични грешки в C# програмите (1)

- Писане извън тялото на `Main()` метода:

```
Console.WriteLine("Hello SoftUni!");
```

- Бъркане на малки и главни букви:

```
Console.writeLine("Hello SoftUni!");
```

```
console.WriteLine("Hello SoftUni!");
```



# Типични грешки в C# програмите (2)

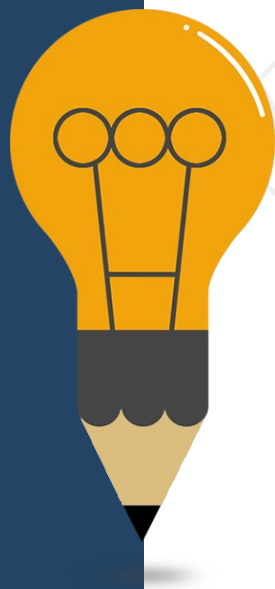
- Липса на ; в края на всяка команда

```
Console.WriteLine("Hello SoftUni!")~
```

- Липсваща кавичка " или липсваща скоба ( или )

```
Console.WriteLine("Hello SoftUni!");~
```

```
Console.WriteLine("Hello SoftUni!");~
```

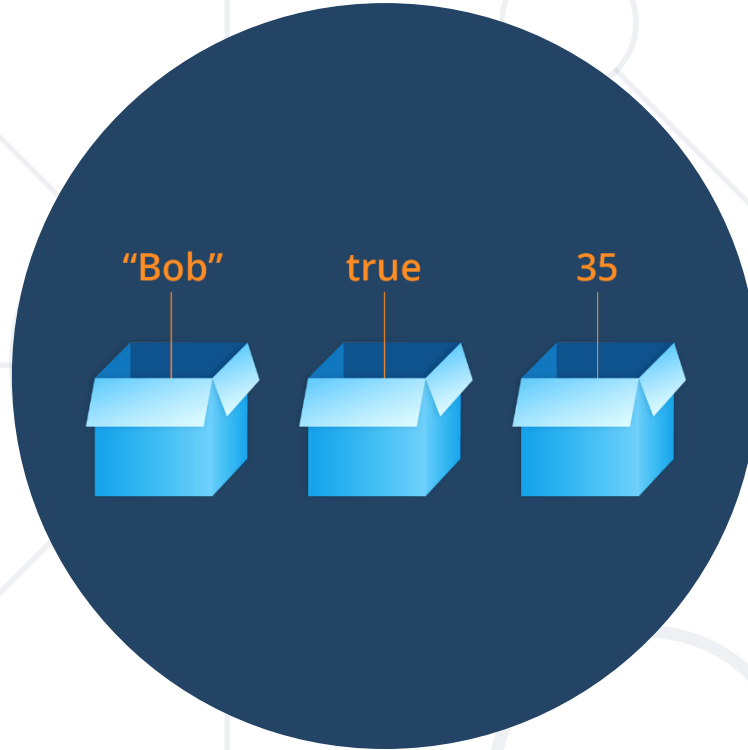




**Решаване на задачи в клас (лаб)**

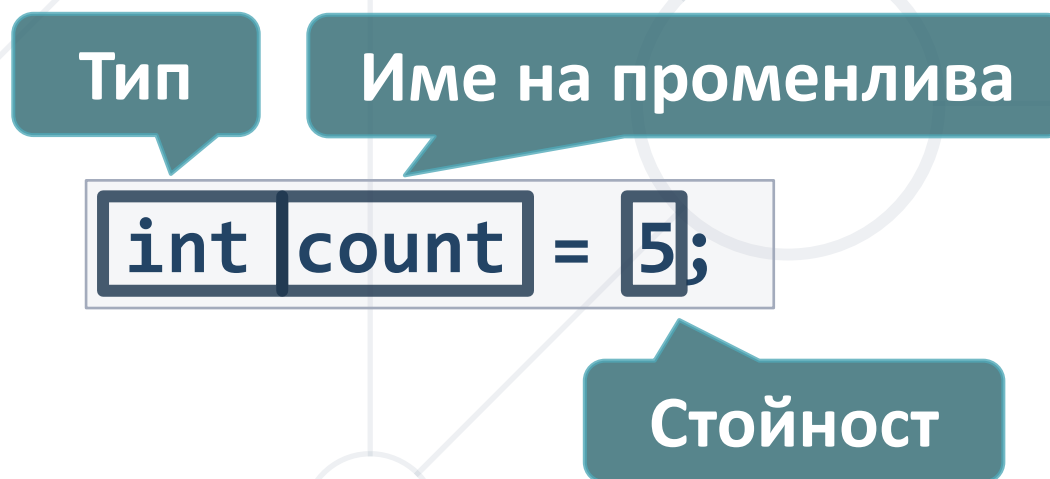
- Напишете програма, която принтира числата от **1** до **10**, всяко на нов ред
- Решение:

```
Console.WriteLine(1);  
Console.WriteLine(2);  
Console.WriteLine(3);  
...  
Console.WriteLine(10);
```



# Променливи и типове данни

- Компютрите са машини, които обработват **данни**
  - Данните се записват в компютърната памет в **променливи**
  - Променливите имат **име**, **тип** и **стойност**
- **Дефиниране** на променлива и **присвояване** на стойност:



# Типове данни (1)

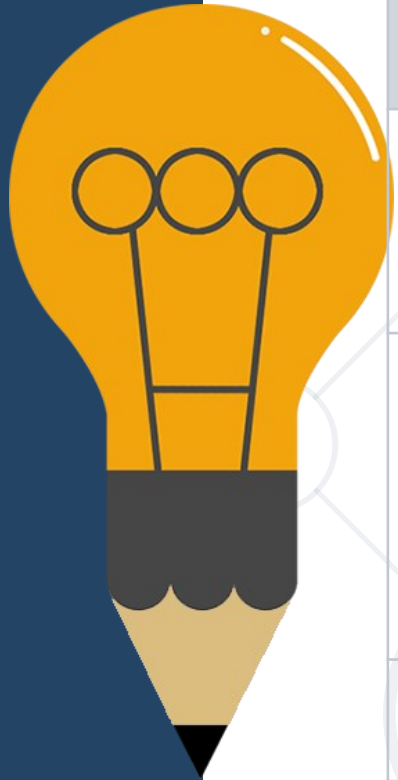
- Променливите съхраняват **стойност от даден тип**
  - Число, буква, текст (низ), дата, цвят, картинка, списък, ...
- Типове данни - примери:
  - **int** - цяло число: 1, 2, 3, 4, 5, ...
  - **double** - дробно число: 0.5, 3.14, -1.5, ...
  - **string** - текст (низ): "Здрасти", "Hi", "Banana", ...





# Типове данни (2)


Тип	Ключова дума	Допустими стойности
цяло число	<code>int</code>	-2,147,483,648 до 2,147,483,647
число с десетична запетая	<code>double</code>	$-1.7 \times 10^{308}$ до $+1.7 \times 10^{308}$
текст(низ)	<code>string</code>	





**Работа с консолью**

# Прочитане на текст

- 
- Всичко, което **получаваме** от конзолата, идва под формата на **текст**
  - Всичко, което **печатаме** на конзолата, се **преобразува в текст**
  - Команда за четене от конзолата:  

```
string name = Console.ReadLine();
```

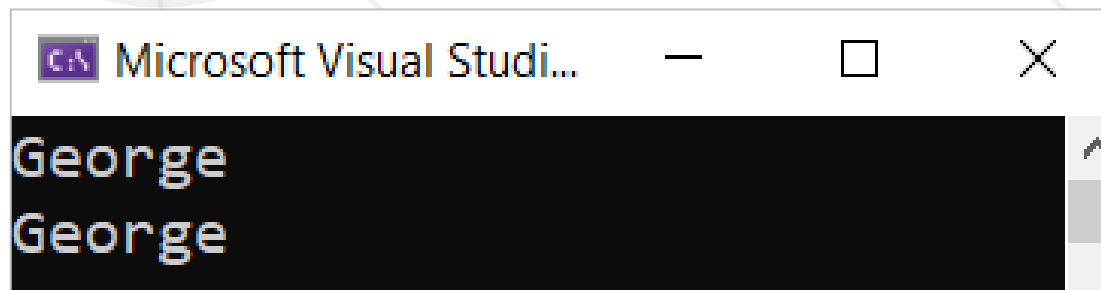
    - Връща ни текстът, въведен от потребителя

- Програма, която **чете** име от конзолата и го **принтира**:

```
string name = Console.ReadLine();  
Console.WriteLine(name);
```

Примерен вход

Изход



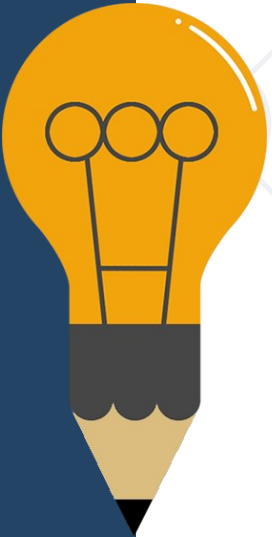
# Четене на числа

- Четене на цяло число:

```
string input = Console.ReadLine();  
int num = int.Parse(input);
```

- Пример: пресмятане на лице на квадрат със страна **a**:

```
int a = int.Parse(Console.ReadLine());  
int area = a * a;  
Console.WriteLine(area);
```



Прочитане на  
цяло число на  
един ред

# Четене на дробно число

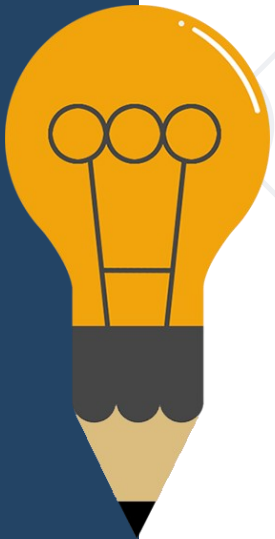
- Четене на дробно число от конзолата:

```
string input = Console.ReadLine();  
double num = double.Parse(input);
```

- Пример: конвертиране от инчове в сантиметри

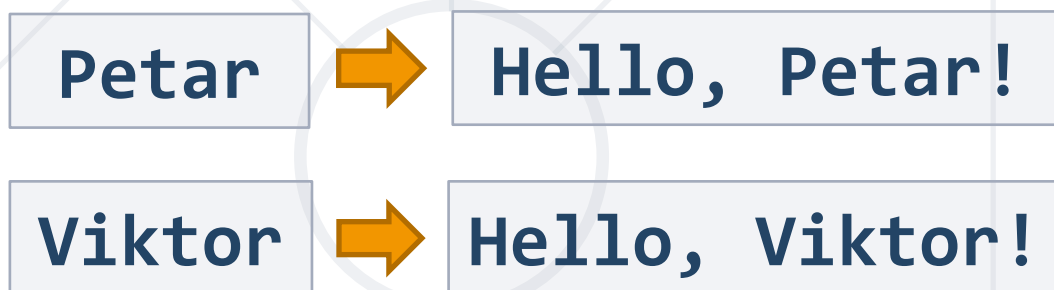
```
double inches = double.Parse(Console.ReadLine());  
double centimeters = inches * 2.54;  
Console.WriteLine(centimeters);
```

Прочитане на  
дробно число на  
един ред



# Поздрав по име – пример

- Да се **напише програма**, която:
  - Чете от конзолата **име** на човек, въведено от **потребителя**
  - Отпечатва "Hello, <name>", където <name> е **въведеното** преди това **име**
- Примерен вход и изход:



# Поздрав по име – решение

```
string name = Console.ReadLine();  
Console.Write("Hello, ");  
Console.Write(name);  
Console.WriteLine("!");
```

Курсорът остава  
на същия ред

```
string name = Console.ReadLine();  
Console.WriteLine("Hello, " + name + "!");
```

Долепяне

```
string name = Console.ReadLine();  
Console.WriteLine($"Hello, {name}!");
```

Интерполация



- Съединяване на текст и число (**оператор "+"**):

```
string firstName = "Maria";  
string lastName = "Ivanova";  
int age = 19;  
string str = firstName + " " + lastName + " @ " + age;  
Console.WriteLine(str);           // Maria Ivanova @ 19
```

```
double a = 1.5;  
double b = 2.5;  
string sum = "The sum is: " + a + b;  
Console.WriteLine(sum);           // The sum is 1.52.5
```

Резултатът е  
долепяне/конкатенация

- Можем да форматираме изхода чрез **интерполация**, която се означава със символа '\$':

```
string firstName = Console.ReadLine();  
string lastName = Console.ReadLine();  
int age = int.Parse(Console.ReadLine());  
string town = Console.ReadLine();  
Console.WriteLine($"You are {firstName} {lastName},  
a {age}-years old person from {town}.");
```

В къдравите скоби  
поставяме името на  
променливата



**Прости операции с дебъгер**

- Процес на проследяване на изпълнението на програмата
  - Това ни позволява да откриваме грешки (бъгове)

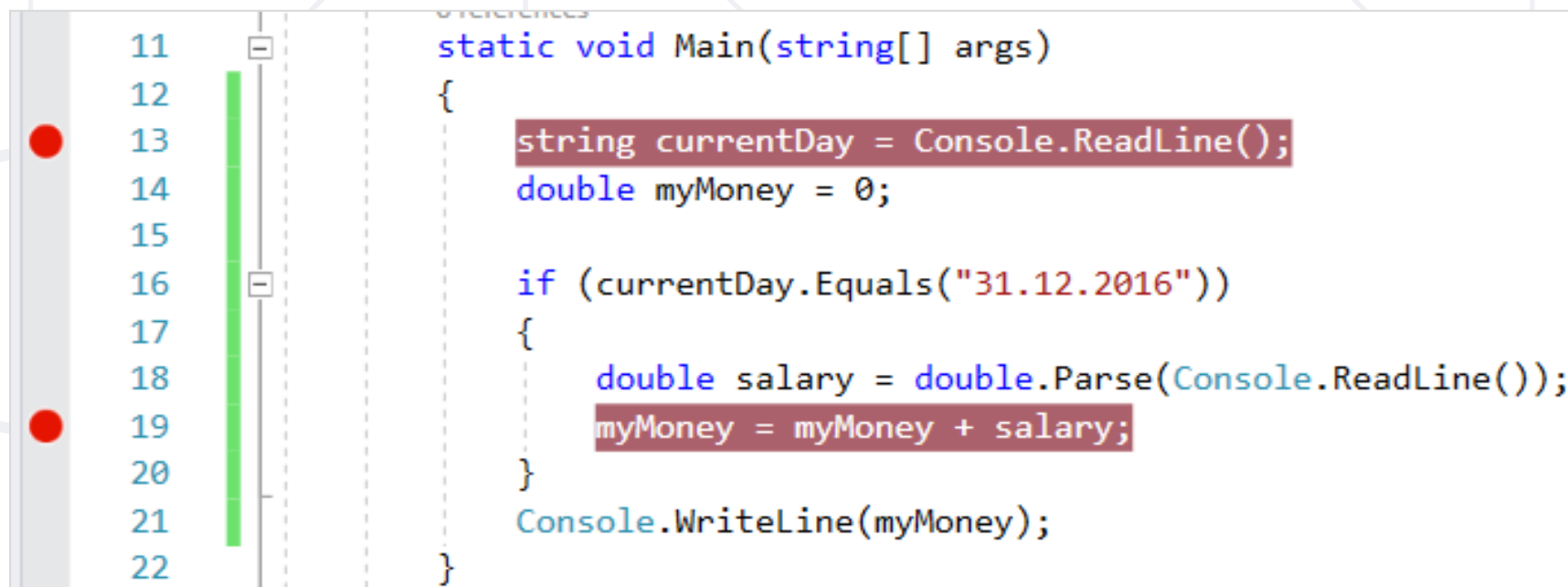
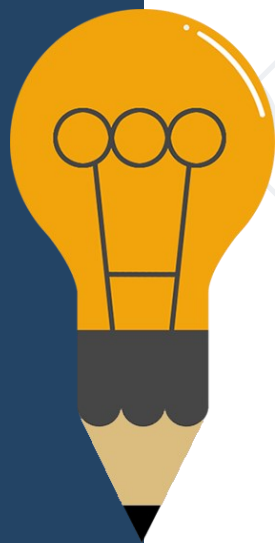


Breakpoint

```
9  class Program
10 {
11     0 references
12     static void Main(string[] args)
13     {
14         string currentDay = Console.ReadLine();
15         double myMoney = 0;
16         if (currentDay.Equals("31.12.2016"))
17         {
18             double salary = double.Parse(Console.ReadLine());
19             myMoney = myMoney + salary;
20         }
21     }
22 }
23 }
```

# Дебъгване във Visual Studio

- Натискане на **[F5]** ще стартира програмата в debug режим
- Можем да преминем към следващата стъпка с **[F10]**
- Можем да създаваме **[F9]** стопери – breakpoints
  - До тях можем директно да стигнем използвайки **[F9]**

A screenshot of the Visual Studio code editor showing a C# program with two breakpoints. The left margin shows line numbers 11 to 22. A green vertical bar indicates the current execution position. Two red dots on lines 13 and 19 represent breakpoints. The code on the right is as follows:

```
static void Main(string[] args)
{
    string currentDay = Console.ReadLine();
    double myMoney = 0;

    if (currentDay.Equals("31.12.2016"))
    {
        double salary = double.Parse(Console.ReadLine());
        myMoney = myMoney + salary;
    }

    Console.WriteLine(myMoney);
}
```



**Работа с числа**

# Аритметични операции: + и -

- Събиране на числа (**оператор +**):

```
int a = 5;  
int b = 7;  
int sum = a + b; // 12
```



- Изваждане на числа (**оператор -**):

```
int a = int.Parse(Console.ReadLine());  
int b = int.Parse(Console.ReadLine());  
int result = a - b;  
Console.WriteLine(result);
```



# Аритметични операции: \* и /

- Умножение на числа (**оператор \***):

```
int a = 5;  
int b = 7;  
int product = a * b; // 35
```

- Деление на числа (**оператор /**):

```
int a = 25;  
int b = a / 4; // 6 - дробната част се отрязва  
double c = a / 4.0; // 6.25 - дробно делене  
int error = a / 0; // Грешка: деление на 0
```





- При деление на цели числа резултатът е **цяло число**:

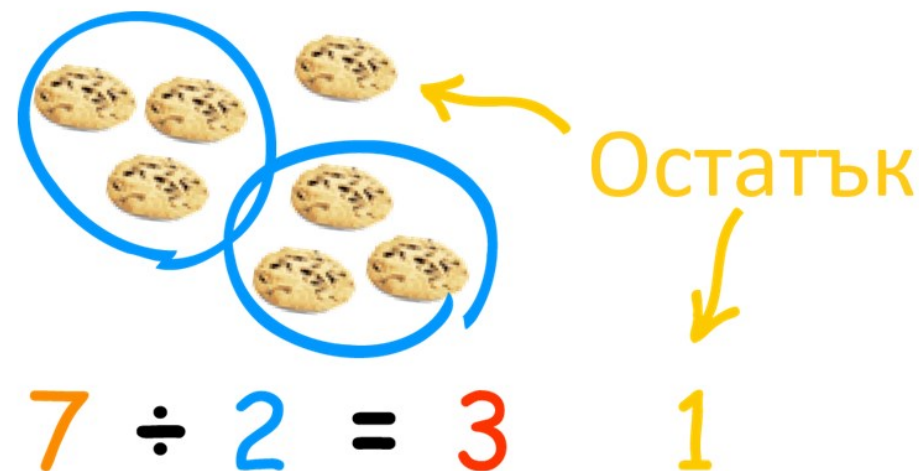
```
int a = 25;  
Console.WriteLine(a / 4);    // Целочислен резултат: 6  
Console.WriteLine(a / 0);    // Грешка: деление на 0
```

- При деление на дробни числа резултатът е **дробно число**:

```
double a = 15;  
Console.WriteLine(a / 2.0);   // Дробен резултат: 7.5  
Console.WriteLine(a / 0.0);   // Резултат: Infinity  
Console.WriteLine(0.0 / 0.0); // Резултат: NaN
```

- Модул/остатък от целочислено деление на числа (**оператор %**):

```
int a = 7;  
int b = 2;  
int product = a % b; // 1
```



```
int odd = 3 % 2; // 1 - числото 3 е нечетно  
int even = 4 % 2; // 0 - числото 4 е четно  
int error = 3 % 0; // Грешка: деление на 0
```

- **Инкрементиране** - увеличаването на стойността на дадена променлива
  - Извършва се чрез оператори за инкрементиране: **префиксни** и **постфиксни**
  - Извършва се само върху променливи, които имат числена стойност

Пример	Име	Резултат
<code>++a</code>	<b>Пре</b> -инкрементация	Увеличава стойността с единица и връща <code>a</code>
<code>a++</code>	<b>Пост</b> -инкрементация	Връща <code>a</code> и увеличава стойността с единица

## ■ Пре-инкрементация

```
int a = 1;  
Console.WriteLine(++a); // 2  
Console.WriteLine(a);   // 2
```

Стойността на променливата а се увеличава с 1 и след това се принтира

## ■ Пост-инкрементация

```
int a = 1;  
Console.WriteLine(a++); // 1  
Console.WriteLine(a);   // 2
```

Първо се принтира променливата а и след това се увеличава с 1

- **Декрементиране** – намаляването на стойността на дадена променлива
  - Извършва се чрез оператори за декрементиране: **префиксни** и **постфиксни**
  - Извършва се само върху променливи, които имат числена стойност

Пример	Име	Резултат
--a	Пре-декрементация	Намалява стойността с единица и връща a
a--	Пост-декрементация	Връща a и намалява стойността с единица

- **Пре-**декрементация

```
int a = 1;  
Console.WriteLine(--a); // 0  
Console.WriteLine(a);  // 0
```

Стойността на променливата a се намалява с 1 и след това се принтира

- **Пост-**декрементация

```
int a = 1;  
Console.WriteLine(a--); // 1  
Console.WriteLine(a);  // 0
```

Първо се принтира променливата a и след това се намалява с 1

- В програмирането можем да закръгляме дробни числа

- Закръгляне до следващо (**по-голямо**) цяло число:

```
double up = Math.Ceiling(23.45); // 24
```

- Закръгляне до предишно (**по-малко**) цяло число:

```
double down = Math.Floor(45.67); // 45
```

- Намиране на **абсолютна** стойност

```
int example1 = Math.Abs(-50); // 50  
int example2 = Math.Abs(50); // 50
```



- Закръгляне до 2 знака след десетичната запетая:

```
double round = Math.Round(45.67852, 2); // 45.68
```

- Форматиране до 2 знака след десетичната запетая:

```
Console.WriteLine("{0:F2}", 123.456); // 123.46
```

Брой символи след десетичната запетая

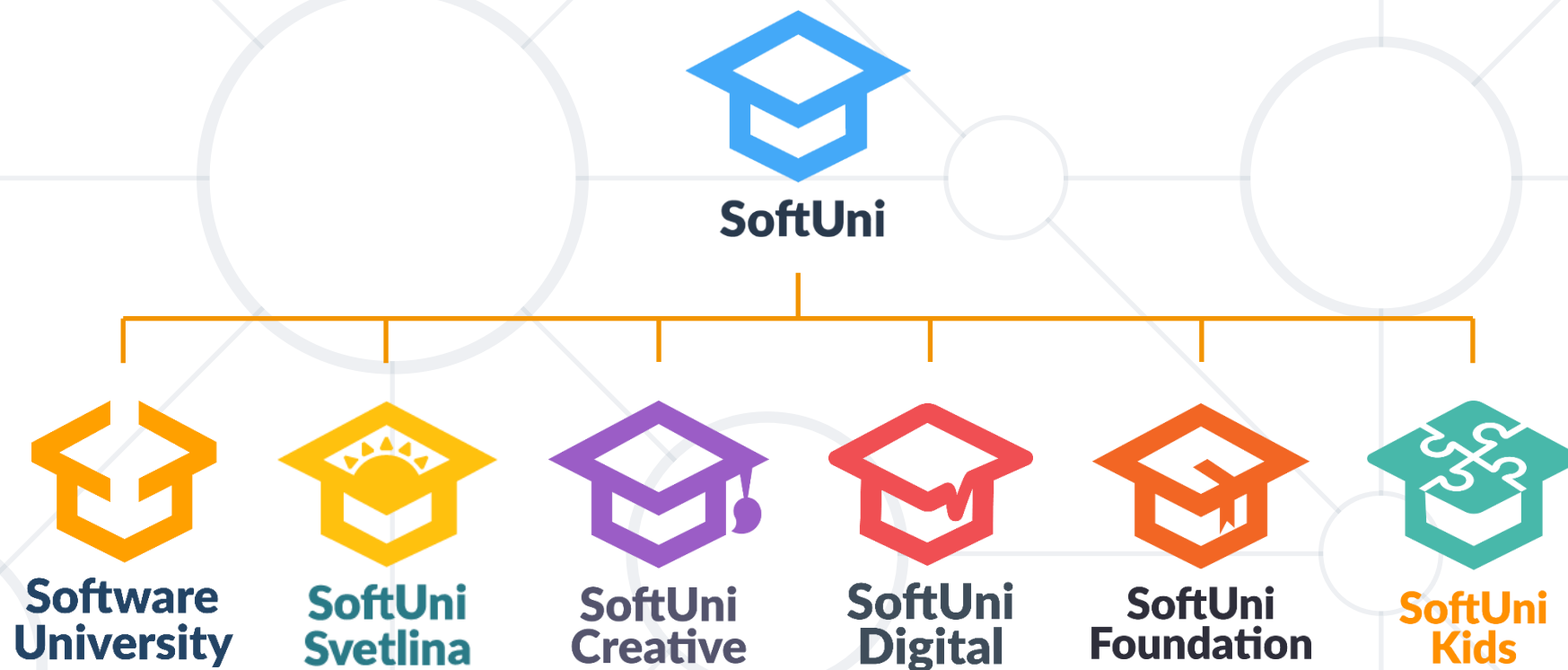
- Разлика между форматиране и закръгляне:

```
Console.WriteLine(Math.Round(45.60000, 4)); // 45.6  
Console.WriteLine("{0:F4}", 45.60000); // 45.6000
```



- **Компютърната програма** е поредица команди
- В C# командите се пишат в частта **Main(...)**
- Въвеждане на текст
- Печатаме с **Console.WriteLine(...)**
- Извеждане на текст по шаблон
- Дебъгване
- Пресмятания с числа (**+**, **-**, **\***, **/**, **%**) и закръгляване

# Въпроси?



- Този курс (презентации, примери, демонстрационен код, упражнения, домашни, видео и други активи) представлява **защитено авторско съдържание**
- Нерегламентирано копиране, разпространение или използване е незаконно
- © СофтУни – <https://softuni.org>
- © Софтуерен университет – <https://softuni.bg>



- Софтуерен университет – качествено образование, професия и работа за софтуерни инженери
  - [softuni.bg](http://softuni.bg)
- Фондация "Софтуерен университет"
  - [softuni.foundation](http://softuni.foundation)
- Софтуерен университет @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Дискуссионни форуми на СофтУни
  - [forum.softuni.bg](http://forum.softuni.bg)



Software University

