

# Методи

Дефиниране, предефиниране и  
използване на методи

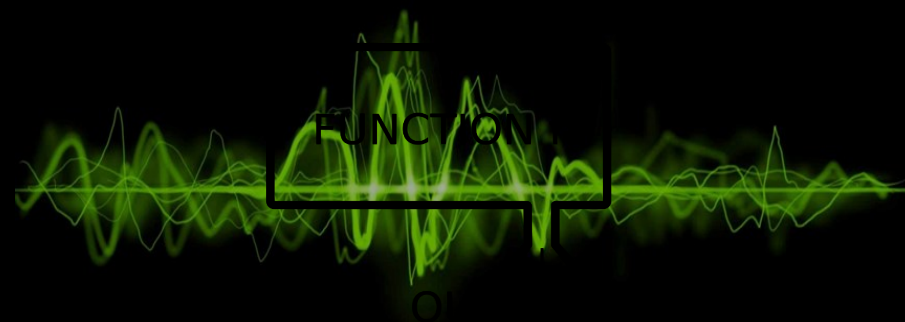


Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>

<https://github.com/BG-IT-Edu/School-Programming/tree/main/Courses/Applied-Programmer/Programming-Fundamentals>



# Съдържание

SoftUni Foundation

```
static void PrintHyphens(int count) ←  
{  
    Console.WriteLine(  
        new string('-', count));  
}  
  
static void Main()  
{  
    for (int i = 1; i <= 10; i++)  
    {  
        PrintHyphens(i); ←  
    }  
}
```

**Дефиниране и извикване на  
методи**

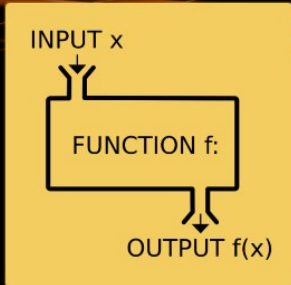
SoftUni Foundation

int long  
double float decimal



**Методи с параметри**

SoftUni Foundation



**Връщана стойност от метод**

SoftUni Foundation



**Предефиниране на методи**

```
static void PrintHyphens(int count) ←  
{  
    Console.WriteLine(  
        new string('-', count));  
}  
  
static void Main()  
{  
    for (int i = 1; i <= 10; i++)  
    {  
        PrintHyphens(i);  
    }  
}
```

# Дефиниране и извикване на методи



# Прости методи

- Метод е именована част от кода, която може да бъде извикана
- Примерна дефиниция на метод:

```
static void PrintHeader()  
{  
    Console.WriteLine("-----");  
}
```

Метод, наречен  
**PrintHeader**

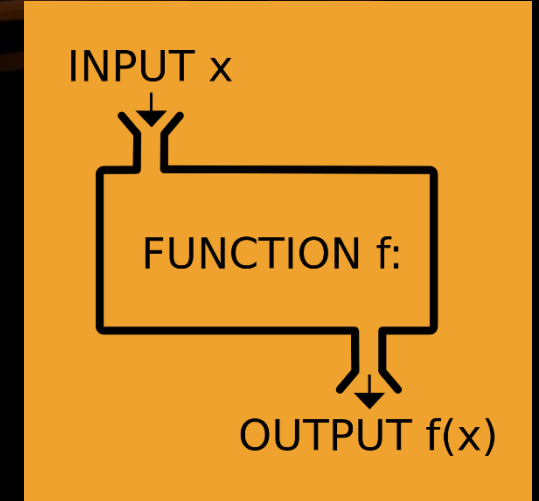
Тялото на метода се  
огражда с **{ }**

- Извикване на метода няколко пъти поред:

```
PrintHeader();  
PrintHeader();
```

# Защо да използваме методи?

- Програмирането става **по-обозримо**
  - Разделяме големите задачи на малки части
  - По-оптимална организация на програмата
  - Подобрява се четимостта на кода
  - Улеснява разбирането на кода
- Избягват се **повторенията в кода**
  - Улеснява поддръжката на кода
- **Повторно използване** на код
  - Използваме методите няколко пъти



# Дефиниране на методи

тип на връщания  
резултат

Име на метода

Параметри

```
static double GetSquare(double num)
```

```
{  
    return num * num;  
}
```

Тяло на  
метода

- Методите се дефинират в **класа**
- **Main()** също е метод
- Променливите в метода са **локални**

```
class Program  
{  
    static void Main()  
    {  
    }  
}
```

# Извикване на метод

- Методите първо се **дефинират**, а после **извикват** (многократно)

```
static void PrintHeader()  
{  
    Console.WriteLine("-----");  
}
```

Дефиниране  
на метода

- Методите могат да бъдат **извикани** чрез името им + **()**:

```
static void Main()  
{  
    PrintHeader();  
}
```

Извикване  
на метода

# Извикване на метод (2)

- Метод може да бъде извикан от:

- Метода Main – **Main()**

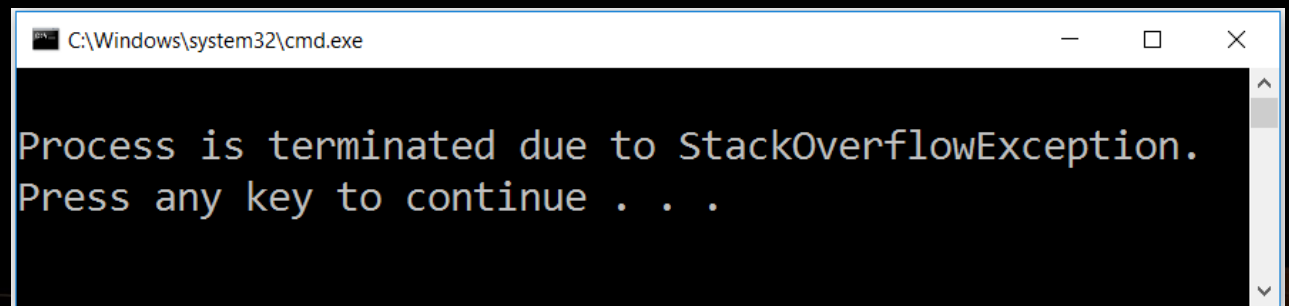
```
static void Main()  
{  
    PrintHeader();  
}
```

- Някой **друг метод**

```
static void PrintHeader()  
{  
    PrintHeaderTop();  
    PrintHeaderBottom();  
}
```

- **Своето тяло** – рекурсия

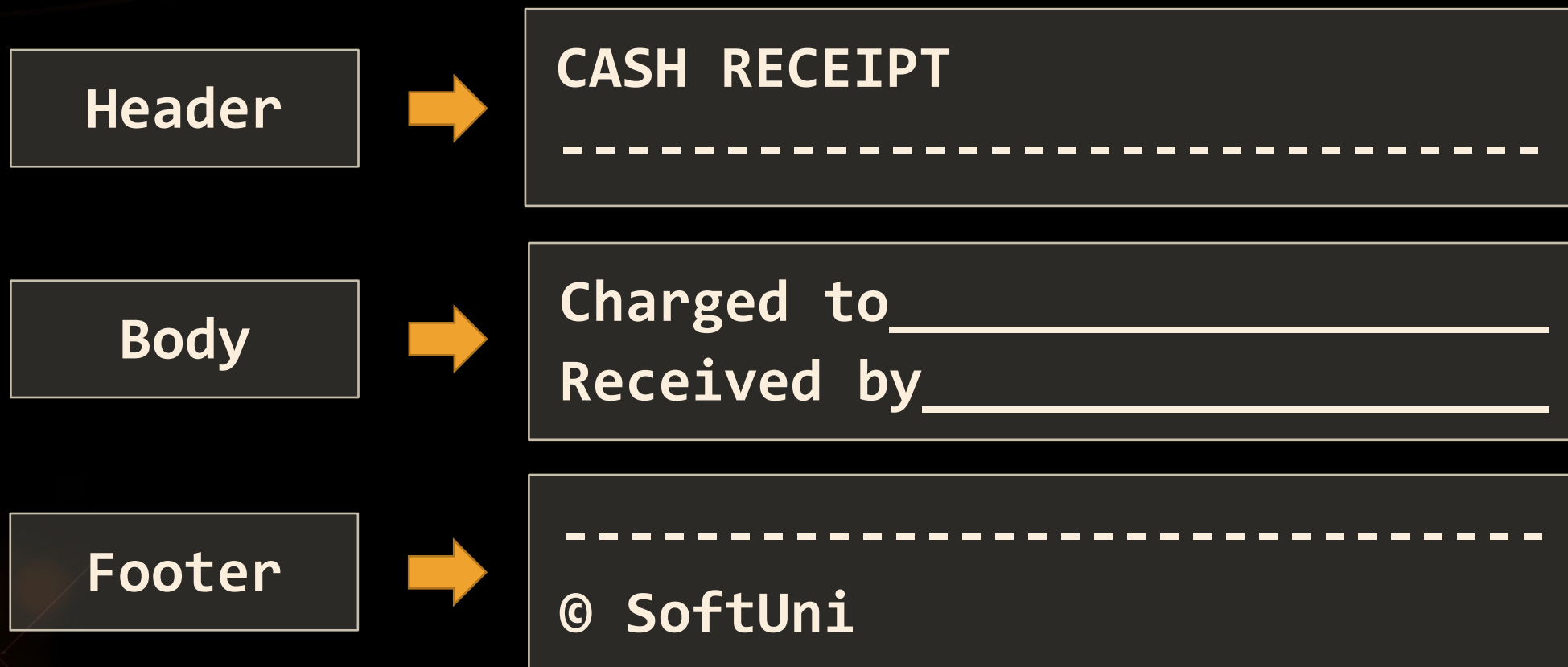
```
static void Crash()  
{ Crash(); }
```





# Задача: Празна касова бележка

- Създайте метод, който отпечатва празна касова бележка:

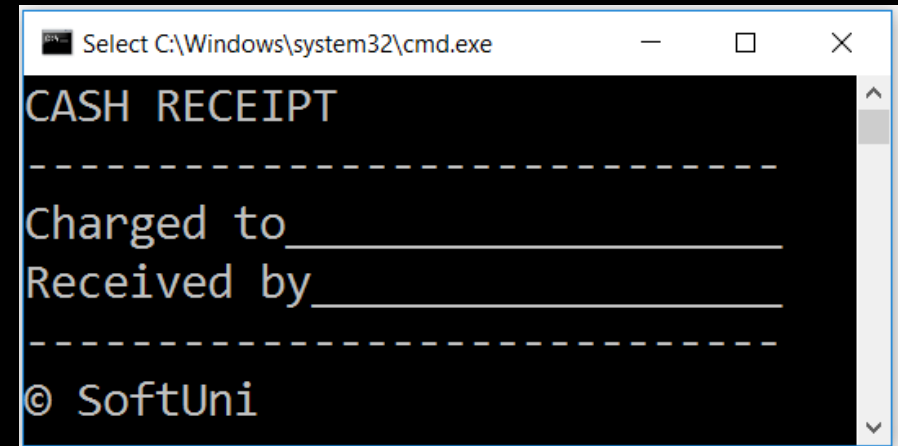


Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

# Решение: Празна касова бележка

- Създайте **3 метода** за печат на секциите (header + body + footer)
  - Копирайте съдържанието от слайда
  - За знака за копирайт използвайте Unicode "**\u00A9**"
- Създайте метод **PrintReceipt()**, извикващ тези 3 метода:

```
private static void PrintReceipt()
{
    PrintHeader();
    PrintBody();
    PrintFooter();
}
```



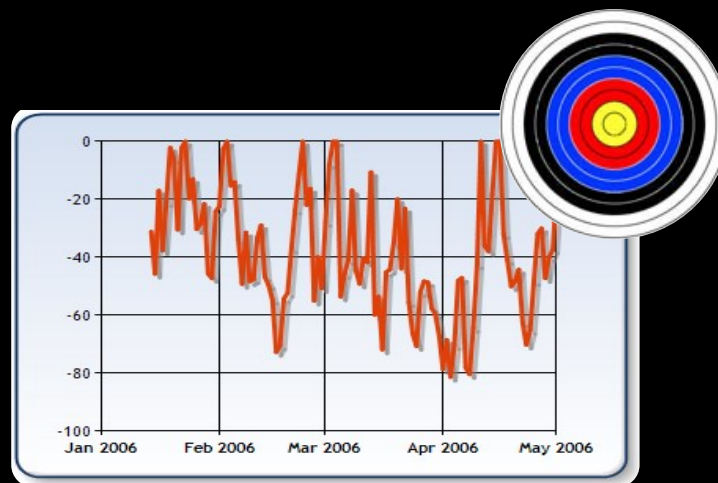
```
Select C:\Windows\system32\cmd.exe
CASH RECEIPT
-----
Charged to
Received by
-----
© SoftUni
```

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

**int**

**long**

**double**



**float**

**decimal**

**Методи с параметри**

# Параметри на методите

- Параметрите могат да са от всеки тип данни

```
static void PrintNumbers(int start, int end)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write("{0} ", i);
    }
}
```

Приема параметри **start** и **end** от тип **int**

Няколко параметъра, разделени със запетая

- Извикването на метода е с конкретни стойности (**аргументи**)

```
static void Main()
{
    PrintNumbers(5, 10);
}
```

При извикване подаваме аргументите



## Параметри на методите (2)

- Може да подадете **нула** или **повече** параметъра
- Може да подавате параметри **от различен тип**
- Всеки параметър има **име** и **тип**

Няколко параметъра  
от различен тип

Тип на  
параметъра

Име на  
параметъра

```
static void PrintStudent(string name, int age, double grade)
{
    Console.WriteLine("Student: {0}; Age: {1}, Grade: {2}",
        name, age, grade);
}
```

# Задача: Знака на цяло число

- Създайте метод, който отпечатва **знака** на цяло число **n**:

2



The number 2 is **positive**.

-5



The number -5 is **negative**.

0



The number 0 is **zero**.

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

# Решение: Знак на цяло число

```
static void PrintSign(int number)
{
    if (number > 0)
        Console.WriteLine("The number {0} is positive", number);
    else if (number < 0)
        Console.WriteLine("The number {0} is negative.", number);
    else
        Console.WriteLine("The number {0} is zero.", number);
}

static void Main()
{ PrintSign(int.Parse(Console.ReadLine())); }
```

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

# Опционални параметри

- Параметрите могат да имат стойности по подразбиране:

```
static void PrintNumbers(int start = 0, int end = 100)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write("{0} ", i);
    }
}
```

Стойности по  
подразбиране

- Методът по-горе може да бъде извикан по множество начини:

```
PrintNumbers(5, 10);
PrintNumbers(15);
PrintNumbers();
PrintNumbers(end: 40, start: 35);
```

Може да ги пропуснем при  
извикването на метода



# Задача: Отпечатване на триъгълник

- Създайте метод за отпечатване на триъгълници по начина, показан по-долу:

3



```
1
1 2
1 2 3
1 2
1
```

4



```
1
1 2
1 2 3
1 2 3 4
1 2 3
1 2
1
```

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

# Решение: Отпечатване на триъгълник

- Създайте метод за печат на един ред от триъгълника, извеждащ числата от подаден **start** до подаден **end**:

```
static void PrintLine(int start, int end)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write(i + " ");
    }
    Console.WriteLine();
}
```

## Решение: Отпечатване на триъгълник (2)

- Създайте метод, печатащ първата част (1..n) и друг за втората част (n-1...1) от триъгълника:

```
static void PrintTriangle(int n)
{
    for (int line = 1; line <= n; line++)
        PrintLine(1, line);

    for (int line = n - 1; line >= 1; line--)
        PrintLine(1, line);
}
```

Метод с  
параметър n

Редове 1...n

Редове n-1...1

# Задача: Начертайте запълнен квадрат

- Да се отпечати **запълнен квадрат** с размер **n** като в примера:

```
static void PrintHeaderRow(int n)
{
    Console.WriteLine(new
        string('-', 2 * n));
}
```

```
static void PrintMiddleRow(int n)
{
    Console.Write('-');
    for (int i = 1; i < n; i++)
        Console.Write("\\\\");
    Console.WriteLine('-');
}
```

Метод с  
параметър **n**

4

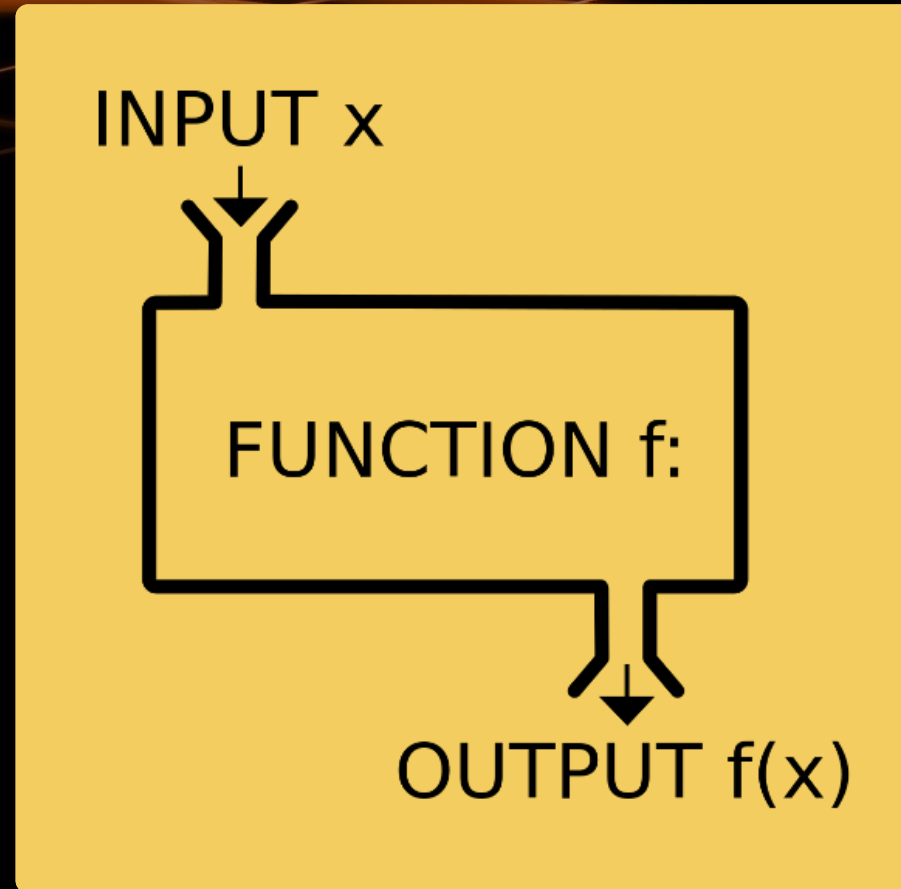


```
-----
-\\\\\\\\-
-\\\\\\\\-
-----
```

```
static void Main() {
    int n = // TODO: read n
    PrintHeaderRow(n);
    for (int i = 0; i < n - 2; i++)
        PrintMiddleRow(i);
    PrintHeaderRow(n);
}
```

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>





**Връщана стойност от метод**

# Типове връщана стойност

- Тип **void** – не връща стойност (само изпълнява код)

```
static void AddOne(int n)
{
    n += 1;
    Console.WriteLine(n);
}
```

Липсва  
команда **return**

- Други типове – връща стойности, от **типа**, връщан от метода

```
static int PlusOne(int n)
{
    return n + 1;
}
```

**връща** стойност  
от тип **int**

# Оператор return

- Ключовата дума **return** прекъсва изпълнението на метода
- Връща указаната стойност

```
static string ReadFullName()  
{  
    string firstName = Console.ReadLine();  
    string lastName = Console.ReadLine();  
    return firstName + " " + lastName;  
}
```

Връща **string**

- Void методите могат да бъдат **завършени** чрез команда **return**

```
return;
```

# Използването на връщана стойност

- Връщаната стойност може да бъде:
  - Присвоена на променлива:

```
int max = GetMax(5, 10);
```

- Използвана в израз:

```
decimal total = GetPrice() * quantity * 1.20m;
```

- Подадена на друг метод:

```
int age = int.Parse(Console.ReadLine());
```



# Конвертор на температури – пример

- Конвертира температури от Фаренхайт към Целзий:

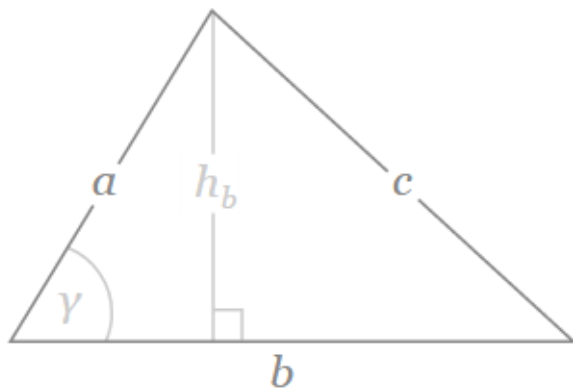
```
static double FahrenheitToCelsius(double degrees)
{
    double celsius = (degrees - 32) * 5 / 9;
    return celsius;
}
```

```
static void Main()
{
    Console.Write("Temperature in Fahrenheit: ");
    double fahrenheit = double.Parse(Console.ReadLine());
    double celsius = FahrenheitToCelsius(fahrenheit);
    Console.Write("Temperature in Celsius: {0:F2}", celsius);
}
```

# Задача: Пресмятане на лице на триъгълник

- Създайте метод който пресмята и връща **лицето** на **триъгълник** по дадени **основа** и **височина**

$$A = \frac{h_b b}{2}$$



$$\begin{aligned} b &= 3 \\ h_b &= 4 \end{aligned}$$



$$A = 6$$

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

# Решение: Пресмятане на лице на триъгълник

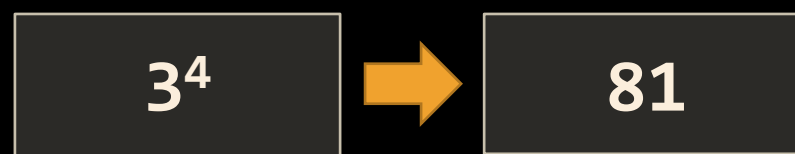
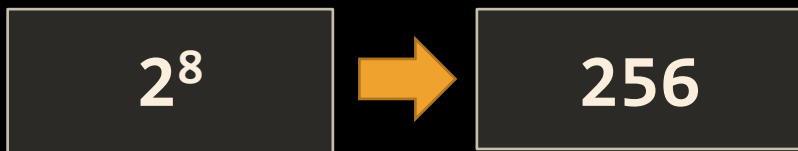
- Създайте метод с два параметъра от тип **double**, който връща **резултат** от тип **double**:

```
static double CalcTriangleArea(double width, double height)
{
    return width * height / 2;
}
```

```
static void Main()
{
    double width = double.Parse(Console.ReadLine());
    double height = double.Parse(Console.ReadLine());
    Console.WriteLine(CalcTriangleArea(width, height));
}
```

# Задача: Метод за повдигане на степен

- Създайте метод, който изчислява и връща стойността на число, повдигнато на степен:



```
static double RaiseToPower(double number, int power)
{
    double result = 1;
    for (int i = 0; i < power; i++)
        result *= number;
    return result;
}
```





# Предефиниране на методи

# Сигнатура на метод

- Комбинацията от **името** и **параметрите** на метод се нарича негов **сигнатура**

```
static void Print(string text)
{
    Console.WriteLine(text);
}
```

Method's  
signature

- Сигнатурата ни помага да **различим** методи с еднакви имена
- Когато два метода с **едно и също име** имат **различна сигнатура**, това се нарича „**предефиниране**“ на метод

# Предефиниране на методи

- Използване на едно и също име за множество методи с различни **сигнатури** (име и параметри на метода)

```
static void Print(string text)
{
    Console.WriteLine(text);
}
```

```
static void Print(int number)
{
    Console.WriteLine(number);
}
```

```
static void Print(string text, int number)
{
    Console.WriteLine(text + ' ' + number);
}
```

Методи с различни  
сигнатури



# Сигнатура и връщан тип данни

- Типът данни, връщани от метода **не е част** от сигнатурата му
- Разгледайте следния пример:

```
static void Print(string text)
{
    Console.WriteLine(text);
}
```

```
static string Print(string text)
{
    return text;
}
```

Грешка по време  
на компилиране!

- Как компилаторът да разбере **кой метод да извика?**



# Задача: По-голямото от две числа

- Създайте метод **GetMax()**, който връща по-голямата от две стойности (те могат да са от тип **int**, **char** or **string**)

int  
2  
16



16

string  
aaa  
bbb



aaa

char  
a  
z



z

Тествайте в Judge: <https://judge.softuni.bg/Contests/2662>

# Какво научихме този час?

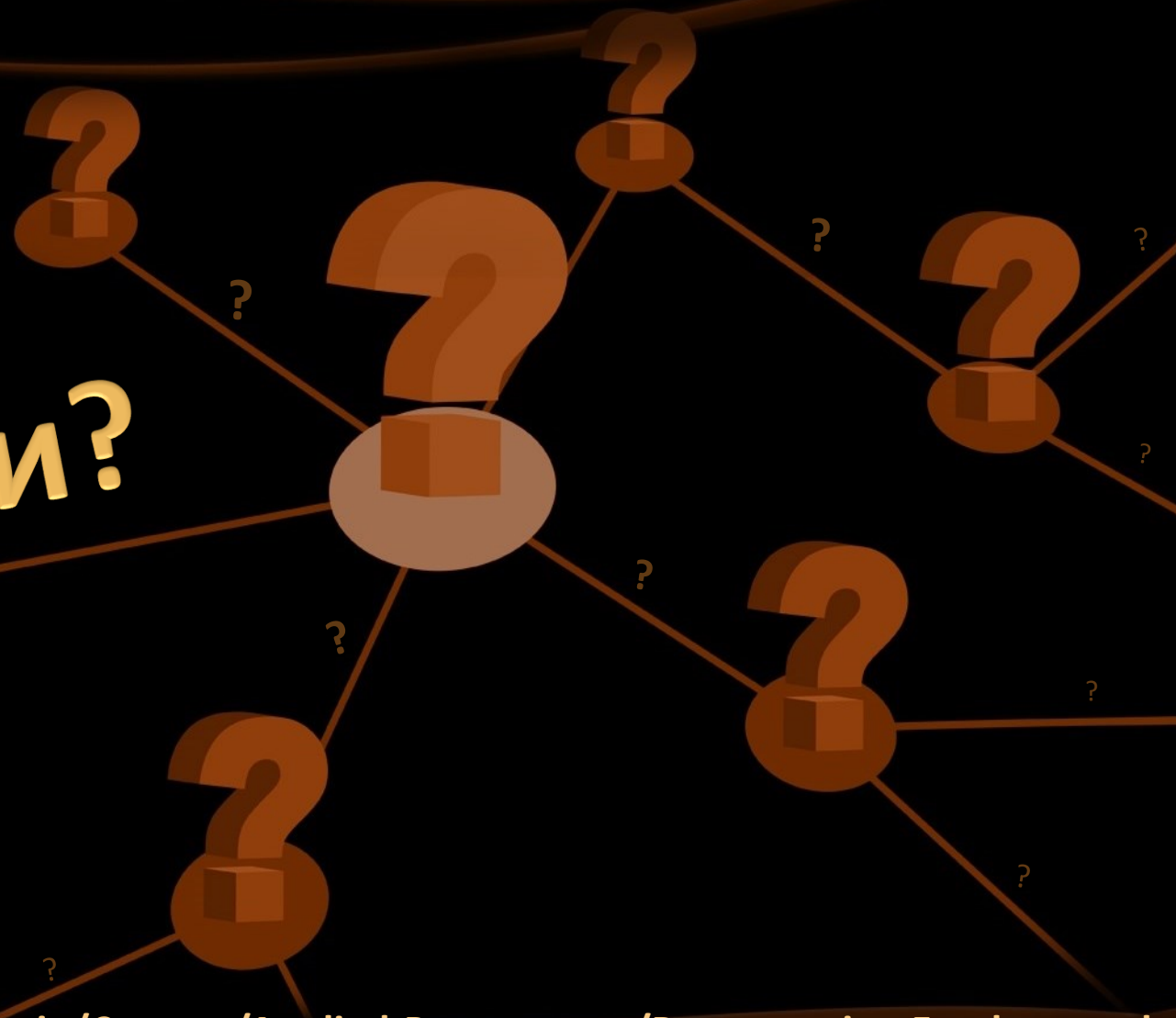
- Разделяме големите програми на прости **методи**, решаващи малки подзадачи
- Методът се състои от **декларация** и **тяло**
- Извиква се чрез **името** на метода + **()**
- Методът може да приема **параметри**
  - Параметрите получават **конкретни стойности** при **извикването** на метода
- Методите могат да **връщат** стойност или да не връщат нищо (**void**)



# Методи



## Въпроси?



# Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма **"Обучение за ИТ кариера"** на МОН за подготовка по професия "Приложен програмист"



Министерство  
на образованието  
и науката



Национална  
програма  
„Обучение за  
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni  
Foundation

