

Работа с по-сложни цикли

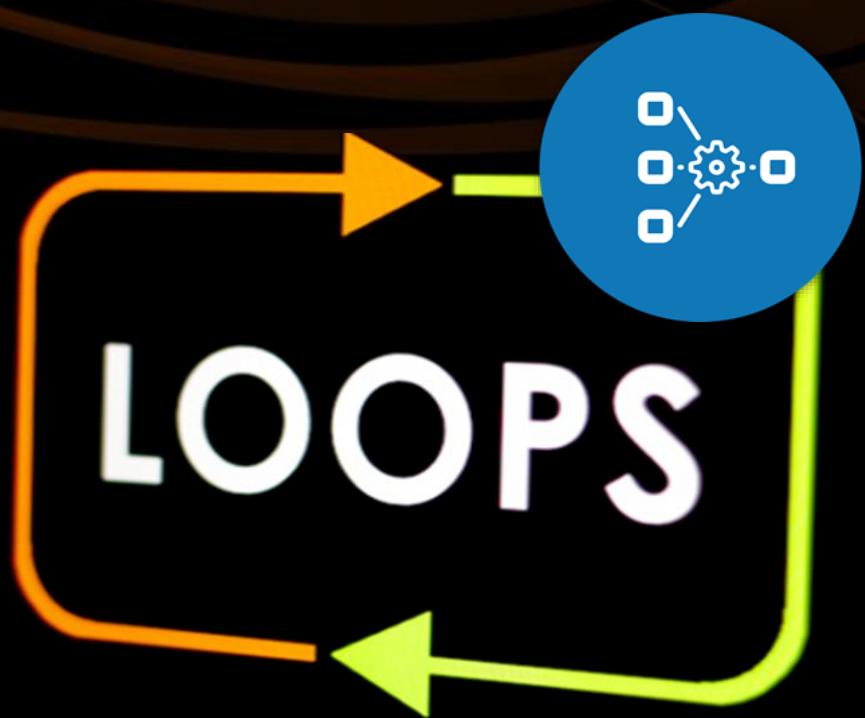


Софтуни
трейнърски екип
Софтуерен университет
<http://softuni.bg>

Цикли със стъпка, While, Do...While



сложни
цикли



Have a Question?

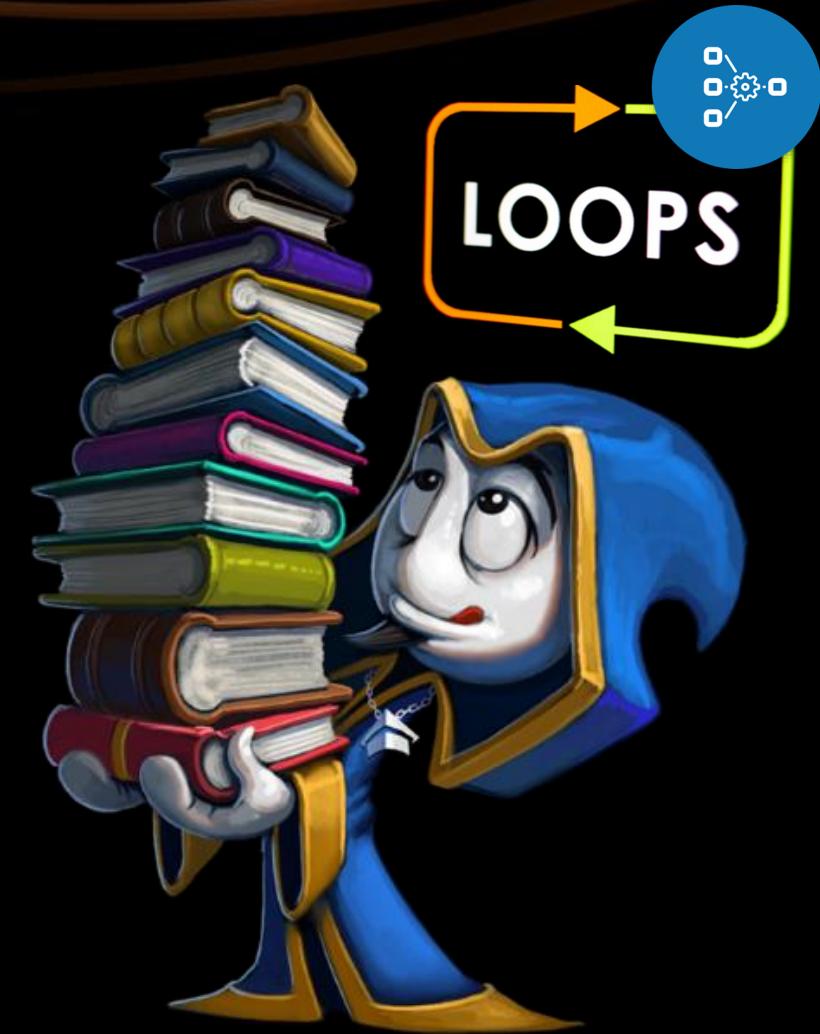


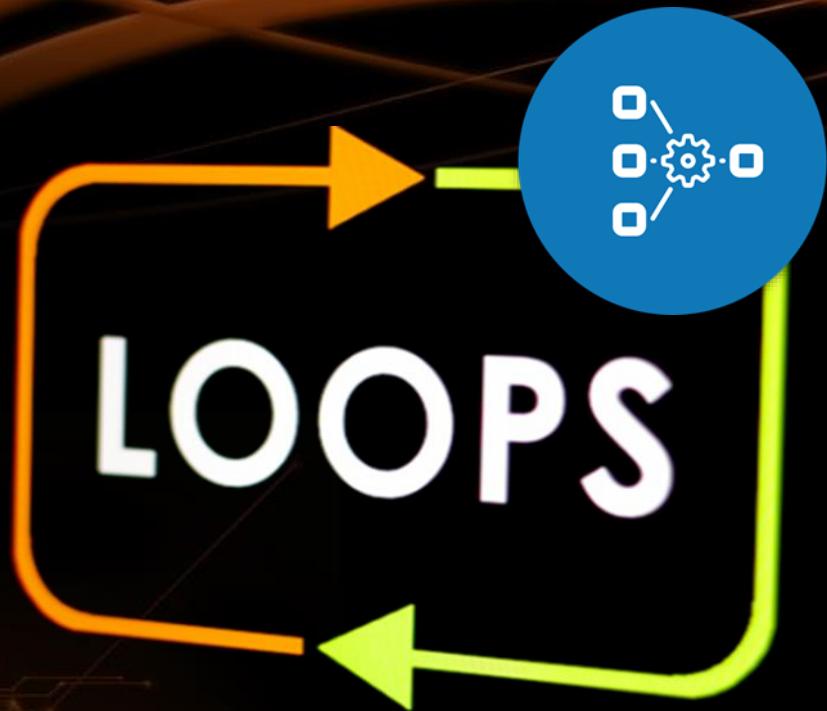
sli.do

#TODO

Съдържание

- По-сложни конструкции за цикъл:
 - цикъл със стъпка
 - цикъл с намаляваща стъпка
 - **while** цикъл
 - **do-while** цикъл
 - безкраен цикъл и оператор **break**
- По-сложни задачи с вложени цикли
- Създаване на уеб игра с ASP.NET MVC





Цикли със стъпка

Работа с по-сложни `for`-цикли

Числата от 1 до N през 3

- Да се отпечатат числата от **1** до **n** със стъпка **3**
- При **n** = 100: **1, 4, 7, 10, ..., 94, 97, 100**

```
int n = int.Parse(Console.ReadLine());  
for (var i = 1; i <= n; i+=3)  
{  
    Console.WriteLine(i);  
}
```



Задаване
на стъпка

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#0>

Числата от N до 1 в обратен ред

- Да се отпечатат числата от **n** до **1** в обратен ред (**стъпка -1**)
- При **n = 100**: **100, 99, 98, ..., 3, 2, 1**

```
int n = int.Parse(Console.ReadLine());  
for (var i = n; i >= 1; i-=1)  
{  
    Console.WriteLine(i);  
}
```

Обърнато
условие:
i >= 1

Отрицателна
стъпка: **-1**



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#1>

Числата от 1 до 2^n с for-цикъл

- Да се отпечатат числата от 1 до 2^n
- При $n = 10$: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024

```
int num = 1;  
for (var i = 0; i <= n; i++)  
{  
    Console.WriteLine(num);  
    num = num * 2;  
}
```

8	2	8	
16	128	4	2
2	4	2	4
2	32	8	

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#2>

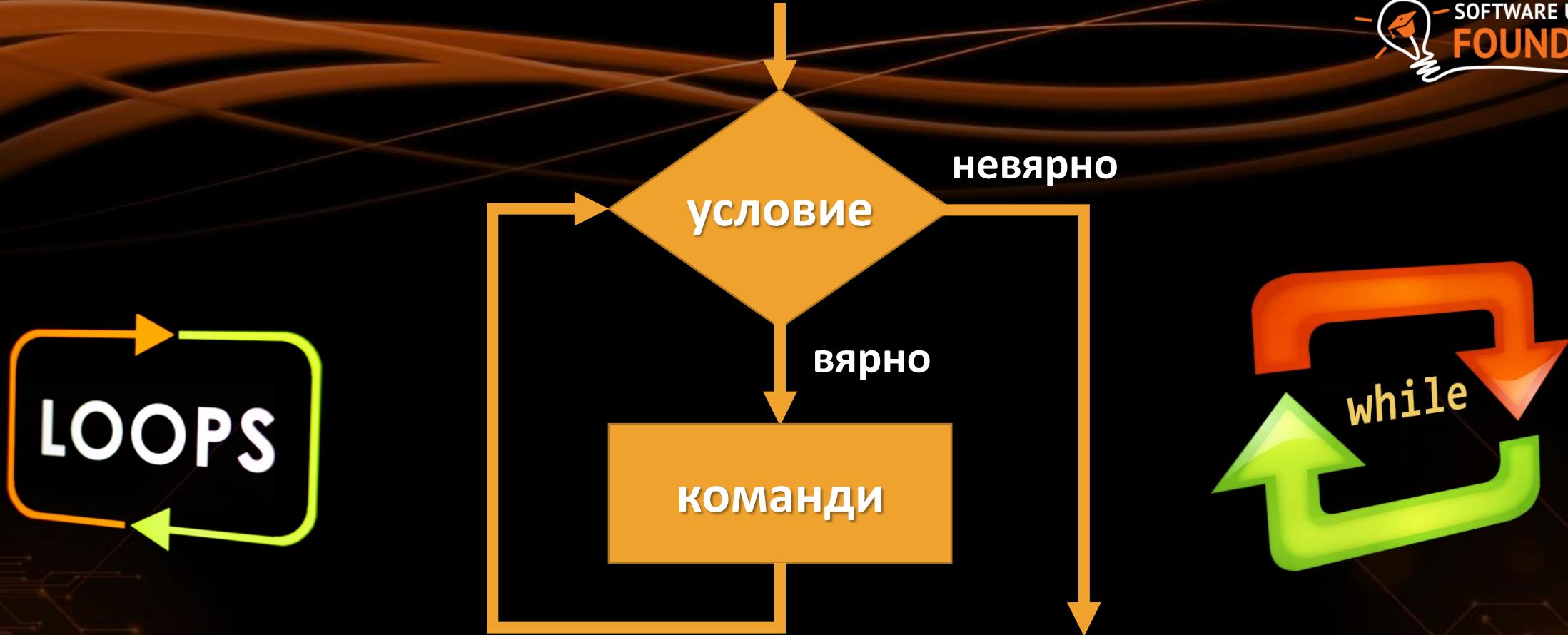
Четни степени на 2

- Да се отпечатат четните степени на 2 до 2^n : $2^0, 2^2, 2^4, 2^8, \dots, 2^n$
- При $n = 10$: 1, 4, 16, 64, 256, 1024

```
int num = 1;  
for (var i = 0; i <= n; i+=2)  
{  
    Console.WriteLine(num);  
    num = num * 2 * 2;  
}
```

Ползваме
стъпка 2

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#3>



While цикъл

Повторение докато е в сила дадено условие

Редица числа $2^k + 1$

- Да се отпечатат всички числа $\leq n$ от редицата: 1, 3, 7, 15, 31, ...
 - Всяко следващо число = предишно число * 2 + 1

```
int num = 1;  
while (num <= n)  
{  
    Console.WriteLine(num);  
    num = 2 * num + 1;  
}
```

Повтаряй докато е в
сила условието $num \leq n$

1, 3, 7, 15, 31, 63, ...

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#4>

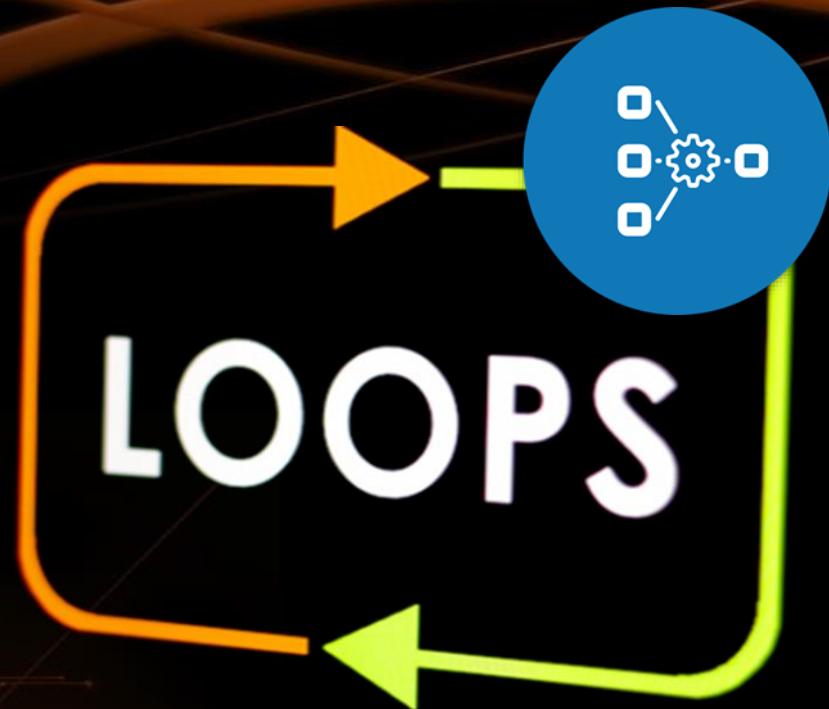
Число в диапазона [1...100]

- Да се въведе число в диапазона [1...100]
 - При невалидно число да се въведе отново

```
var num = int.Parse(Console.ReadLine());
while (num < 1 || num > 100)
{
    Console.WriteLine("Invalid number!");
    num = int.Parse(Console.ReadLine());
}
Console.WriteLine("The number is: {0}", num);
```

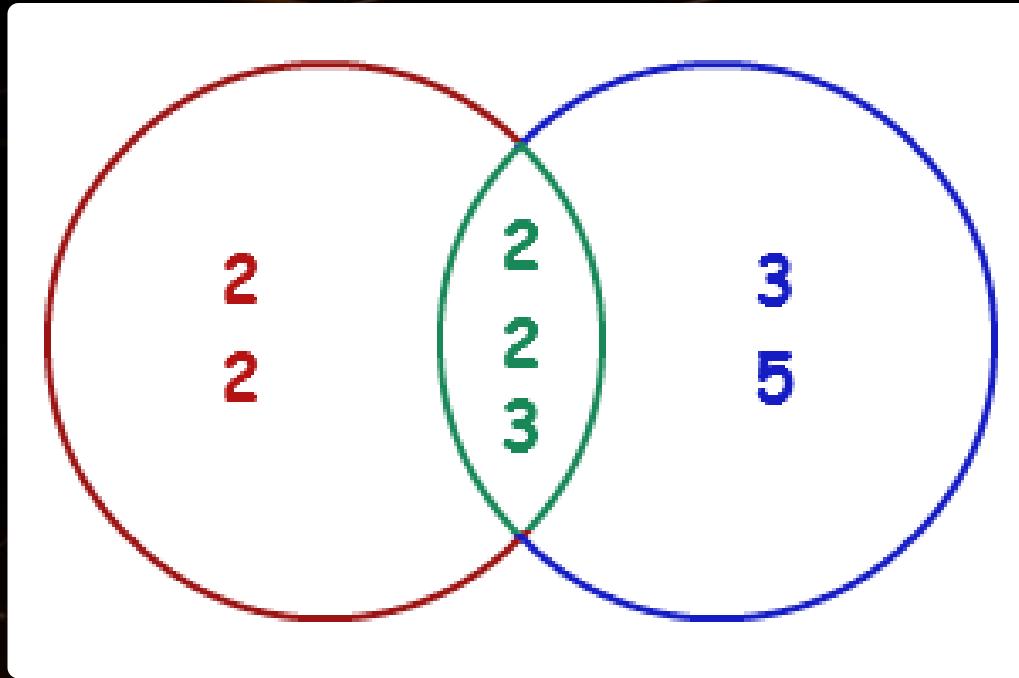


Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#5>



Цикли със стъпка и while цикъл

Работа на живо в клас (лаб)



Най-голям общ делител (НОД)

Алгоритъм на Евклид

Най-голям общ делител (НОД)

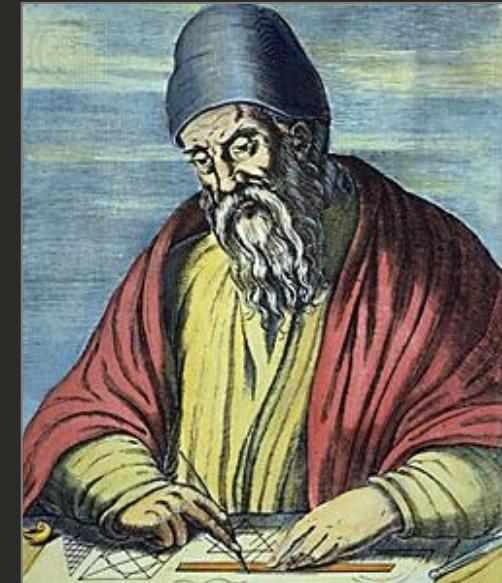
- Най-голям общ делител (НОД) на две естествени числа **a** и **b** е най-голямото число, което дели едновременно **a** и **b** без остатък
 - НОД(24, 16) = 8
 - НОД(12, 24) = 12
 - НОД(10, 10) = 10
 - НОД(67, 18) = 1
 - НОД(15, 9) = 3
 - НОД(100, 88) = 4
- Алгоритъм на Евклид за намиране на НОД:
 - Докато не достигнем остатък 0
 - Делим по-голямото число на по-малкото
 - Взимаме остатъка от делението

```
while b ≠ 0
  var oldB = b;
  b = a % b;
  a = oldB;
print a;
```

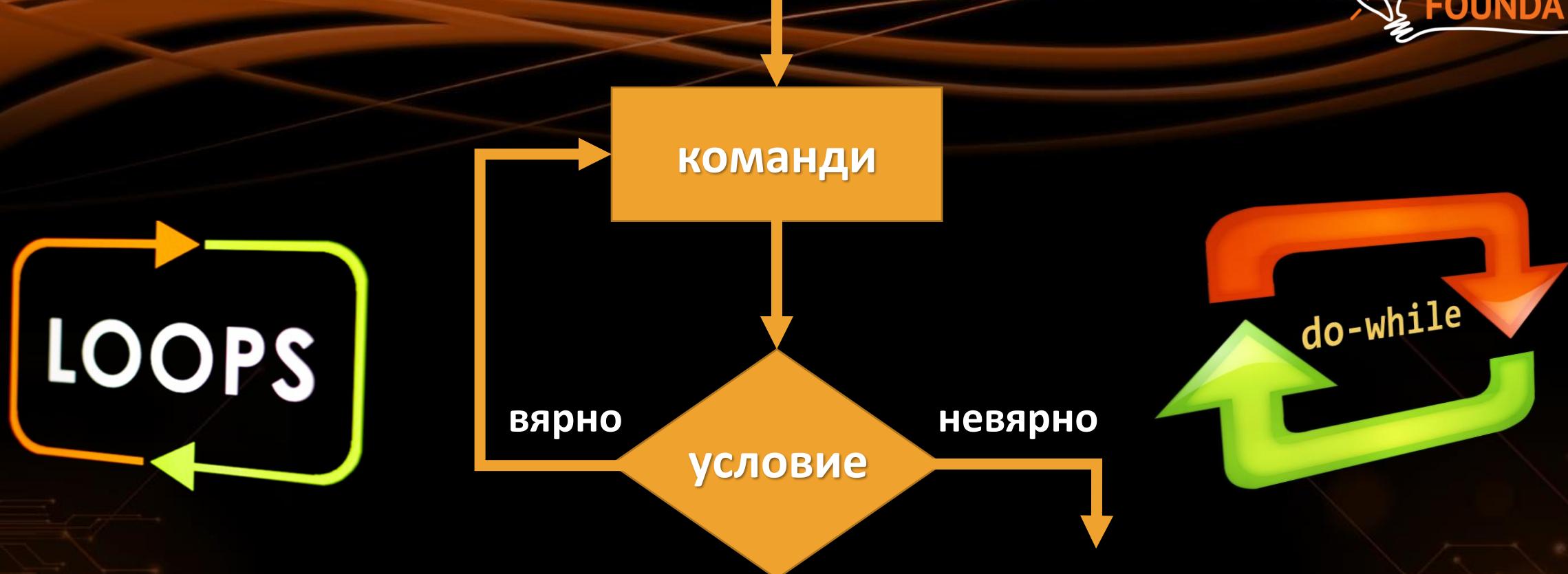
Алгоритъм на Евклид за НОД

- Да се въведат цели числа **a** и **b** и да се намери **НОД(a, b)**

```
var a = int.Parse(Console.ReadLine());  
var b = int.Parse(Console.ReadLine());  
while (b != 0)  
{  
    var oldB = b;  
    b = a % b;  
    a = oldB;  
}  
  
Console.WriteLine("GCD = {0}", a);
```



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#6>



Do...While цикъл

Повторение докато е изпълнено условието

Изчисляване на факториел

- За естествено число n да се изчисли $n! = 1 * 2 * 3 * \dots * n$
- Пример: $5! = 1 * 2 * 3 * 4 * 5 = 120$

```
var n = int.Parse(Console.ReadLine());
var fact = 1;
do
{
    fact = fact * n;
    n--;
} while (n > 1);
Console.WriteLine(fact);
```

$n!$

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#7>

Сумиране на цифрите на число

- Да се сумират цифрите на цяло положително число n
- При $n = 5634$: $5 + 6 + 3 + 4 = 18$

```
var n = int.Parse(Console.ReadLine());  
var sum = 0;  
do  
{  
    sum = sum + (n % 10);  
    n = n / 10;  
} while (n > 0);  
Console.WriteLine("Sum of digits: {0}", sum);
```

$n \% 10$ връща последната цифра на числото n

$n / 10$ изтрива последната цифра на n

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#8>



Безкрайни цикли и оператор break

Безкраен цикъл

- Безкраен цикъл е когато повтаряме нещо до безкрайност:

```
while(true)
{
    Console.WriteLine("Infinite loop");
}
```



```
for (;;)
{
    Console.WriteLine("Infinite loop");
}
```



Прости числа

- Едно число n е просто, ако се дели единствено на **1 и n**
 - Прости числа: **2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, ...**
 - Непрости (композитни) числа: $10 = 2 * 5$, $21 = 3 * 7$, $143 = 13 * 11$
- Едно число n е просто, ако се дели на число между **2 и $n-1$**
- Алгоритъм за проверка дали число е просто:
 - Проверяваме дали n се дели на **2, 3, ..., $n-1$**
 - Ако се раздели, значи е композитно
 - Ако не се раздели, значи е просто
- Оптимизация: вместо до $n-1$ да се проверяват делители до \sqrt{n}

PRIME NUMBERS			
2	3	5	7
Any number under 100 which can not be divided by one of the above numbers is prime.			
11	13	17	19
Any number under 400 which can not be divided by one of the above numbers is prime.			

Проверка за просто число. Оператор break

```
var n = int.Parse(Console.ReadLine());  
var prime = true;  
for (var i = 2; i <= Math.Sqrt(n); i++)  
    if (n % i == 0) {  
        prime = false;  
        break; → break излиза от цикъла  
    }  
if (prime) Console.WriteLine("Prime");  
else Console.WriteLine("Not prime");
```



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#9>

Оператор break в безкраен цикъл

- Да се напише програма, която въвежда четно число
 - При невалидно число да връща към повторно въвеждане

```
var n = 0;
while (true)
{
    Console.Write("Enter even number: ");
    n = int.Parse(Console.ReadLine());
    if (n % 2 == 0)
        break; // even number -> exit from the Loop
    Console.WriteLine("The number is not even.");
}
Console.WriteLine("Even number entered: {0}", n);
```



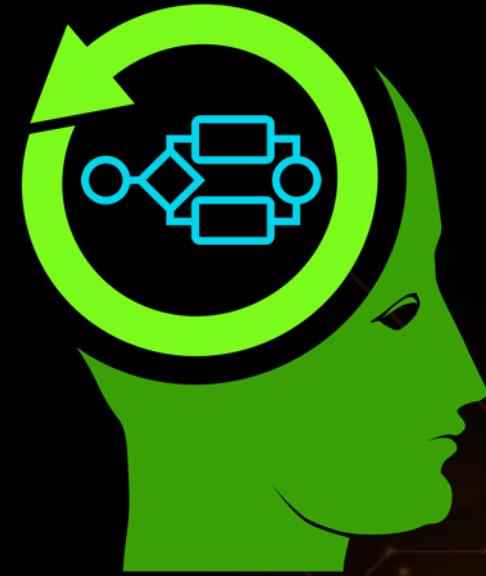
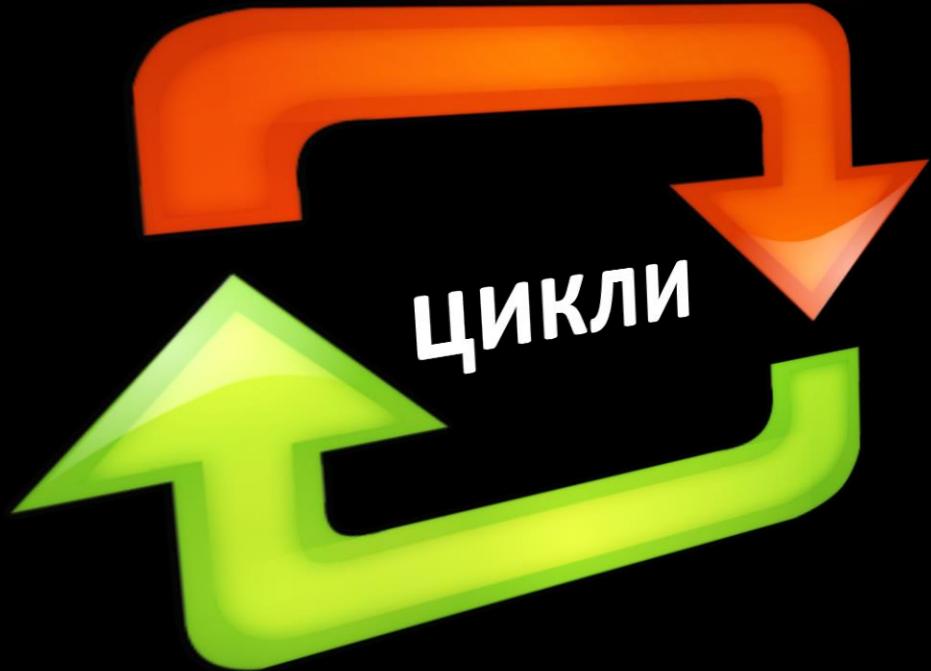
Справяне с грешни числа: try ... catch

```
try
{
    Console.WriteLine("Enter even number: ");
    n = int.Parse(Console.ReadLine());
    if (n % 2 == 0)
        break;
    Console.WriteLine("The number is not even.");
}
catch
{
    Console.WriteLine("Invalid number.");
}
```

Ако `int.Parse(...)` гърмне, ще се изпълни `catch { ... }` блокът



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#10>



Задачи с цикли

Числа на Фибоначи

- Числата на Фибоначи са следните: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
 - $F_0 = 1$
 - $F_1 = 1$
 - $F_n = F_{n-1} + F_{n-2}$
 - Пример: $F(15) = 987$
 - Да се въведе **n** и да се пресметна **n**-тото число на Фибоначи
- ```
var n = int.Parse(Console.ReadLine());
var f0 = 1;
var f1 = 1;
for (var i = 0; i < n-1; i++)
{
 var fNext = f0 + f1;
 f0 = f1;
 f1 = fNext;
}
Console.WriteLine(f1);
```
- 
 A circular portrait of Leonardo Fibonacci, an Italian mathematician, in historical attire.

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#11>

# Пирамида от числа

- Да се отпечатат числата 1...n в пирамида като в примерите:

n = 7



|       |
|-------|
| 1     |
| 2 3   |
| 4 5 6 |
| 7     |

n = 10



|          |
|----------|
| 1        |
| 2 3      |
| 4 5 6    |
| 7 8 9 10 |

n = 12



|          |
|----------|
| 1        |
| 2 3      |
| 4 5 6    |
| 7 8 9 10 |
| 11 12    |

n = 15



|                |
|----------------|
| 1              |
| 2 3            |
| 4 5 6          |
| 7 8 9 10       |
| 11 12 13 14 15 |



Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#12>

# Пирамида от числа – решение

```
var n = int.Parse(Console.ReadLine());
var num = 1;
for (var row = 1; row <= n; row++)
{
 for (var col = 1; col <= row; col++)
 {
 if (col > 1) Console.Write(" ");
 Console.Write(num);
 num++;
 if (num > n) break;
 }
 Console.WriteLine();
 if (num > n) break;
}
```



|          |
|----------|
| 1        |
| 2 3      |
| 4 5 6    |
| 7 8 9 10 |
| 11 12    |

# Таблица с числа

- Да се отпечатат числата  $1\dots n$  в таблица като в примерите:

$n = 2$



|   |   |
|---|---|
| 1 | 2 |
| 2 | 1 |



$n = 3$



|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 3 | 2 |
| 3 | 2 | 1 |

$n = 4$



|   |   |   |   |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 3 |
| 3 | 4 | 3 | 2 |
| 4 | 3 | 2 | 1 |

$n = 5$



|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 4 |
| 3 | 4 | 5 | 4 | 3 |
| 4 | 5 | 4 | 3 | 2 |
| 5 | 4 | 3 | 2 | 1 |

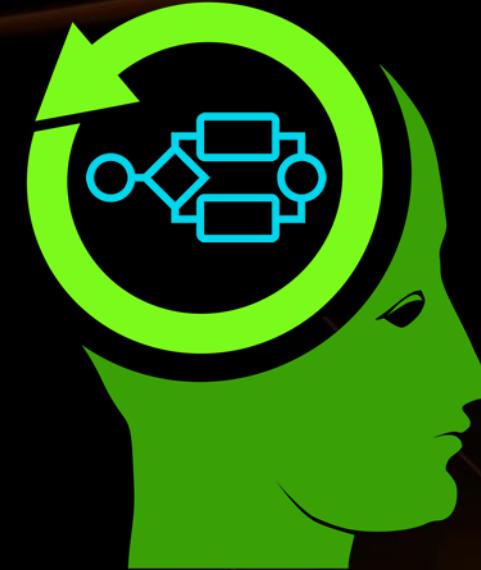
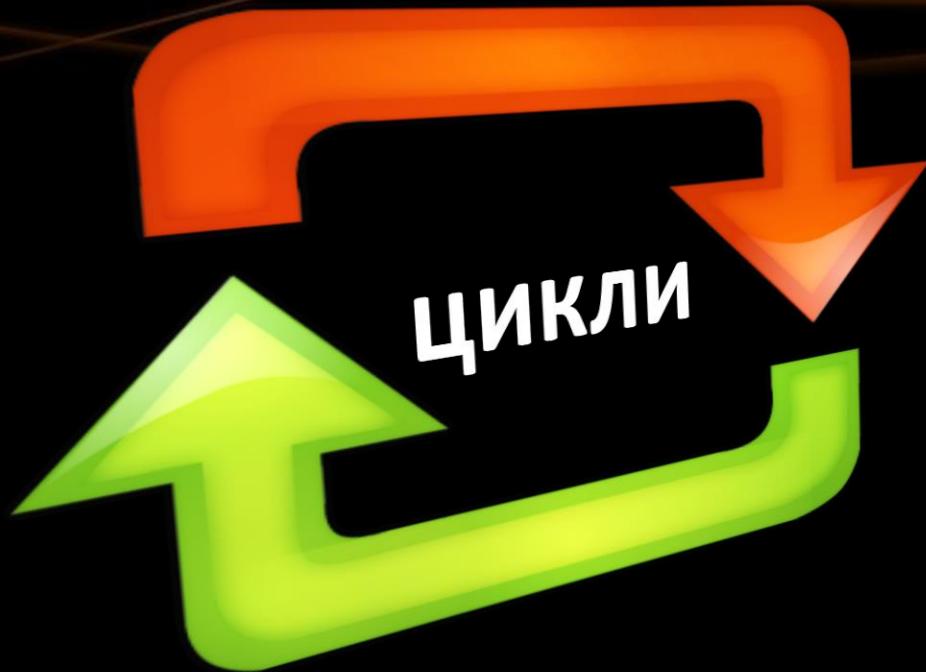
Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#13>

# Таблица с числа – решение

```
var n = int.Parse(Console.ReadLine());
for (int row = 0; row < n; row++)
{
 for (int col = 0; col < n; col++)
 {
 var num = row + col + 1;
 if (num > n) num = 2 * n - num;
 Console.Write(num + " ");
 }
 Console.WriteLine();
}
```

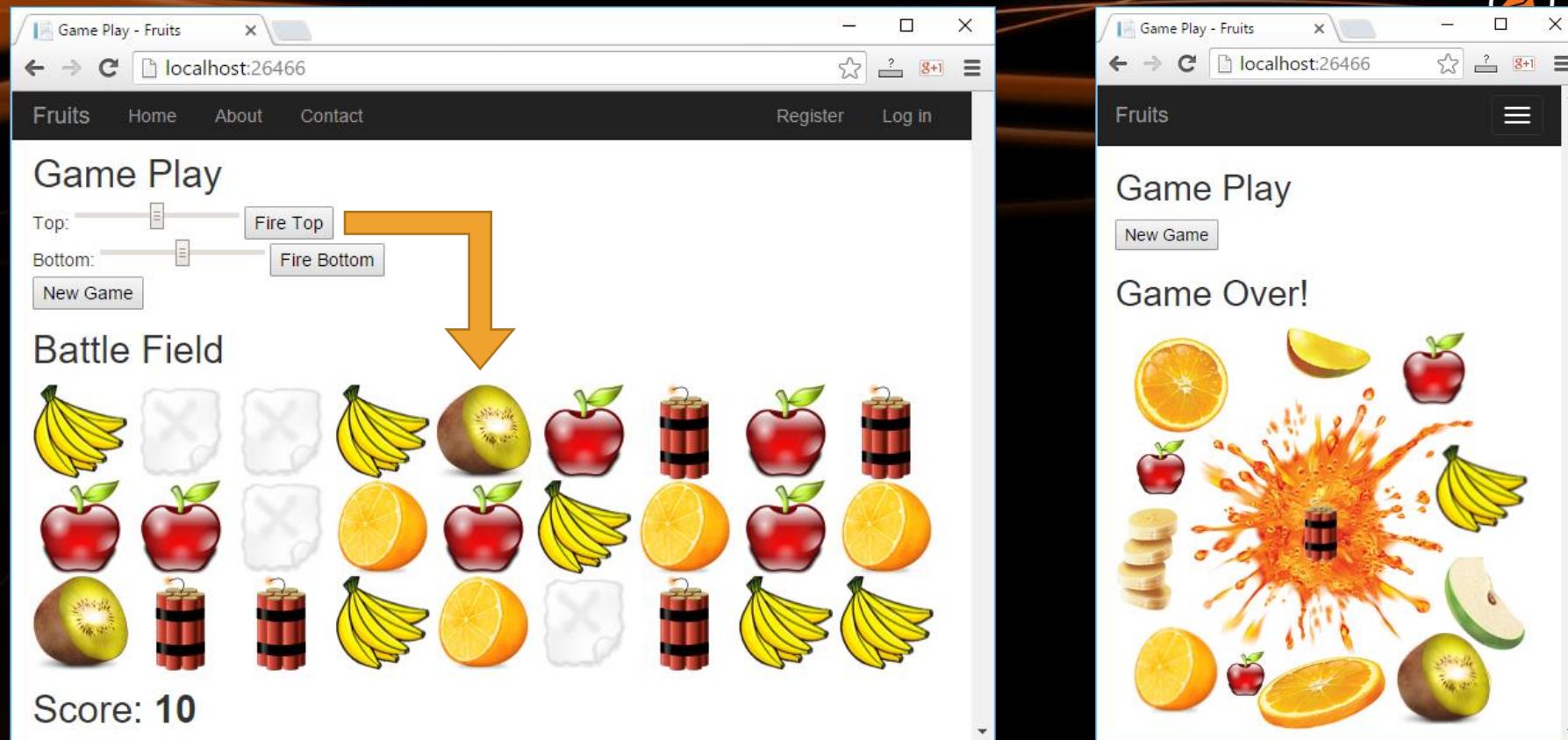
|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 4 |
| 3 | 4 | 5 | 4 | 3 |
| 4 | 5 | 4 | 3 | 2 |
| 5 | 4 | 3 | 2 | 1 |

Тестване на решението: <https://judge.softuni.bg/Contests/Practice/Index/156#13>



# По-сложни задачи с цикли

Работа на живо в клас (лаб)



# Уеб игра с ASP.NET MVC

## Обстреляй плодовете!

# Уеб игра "Обстреляй плодовете!"

### Game Play

Top:  Fire Top

Bottom:  Fire Bottom

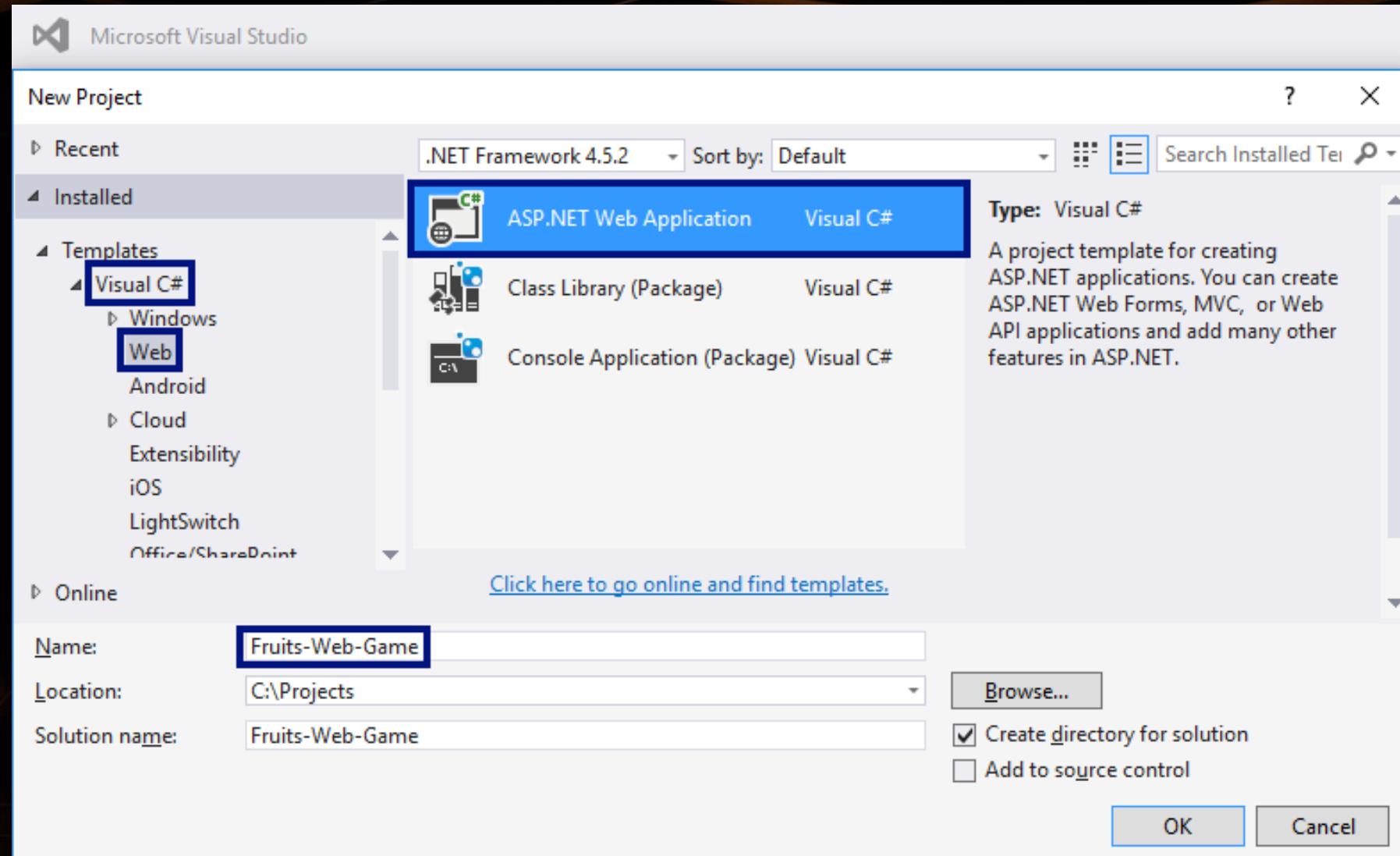
New Game

### Battle Field

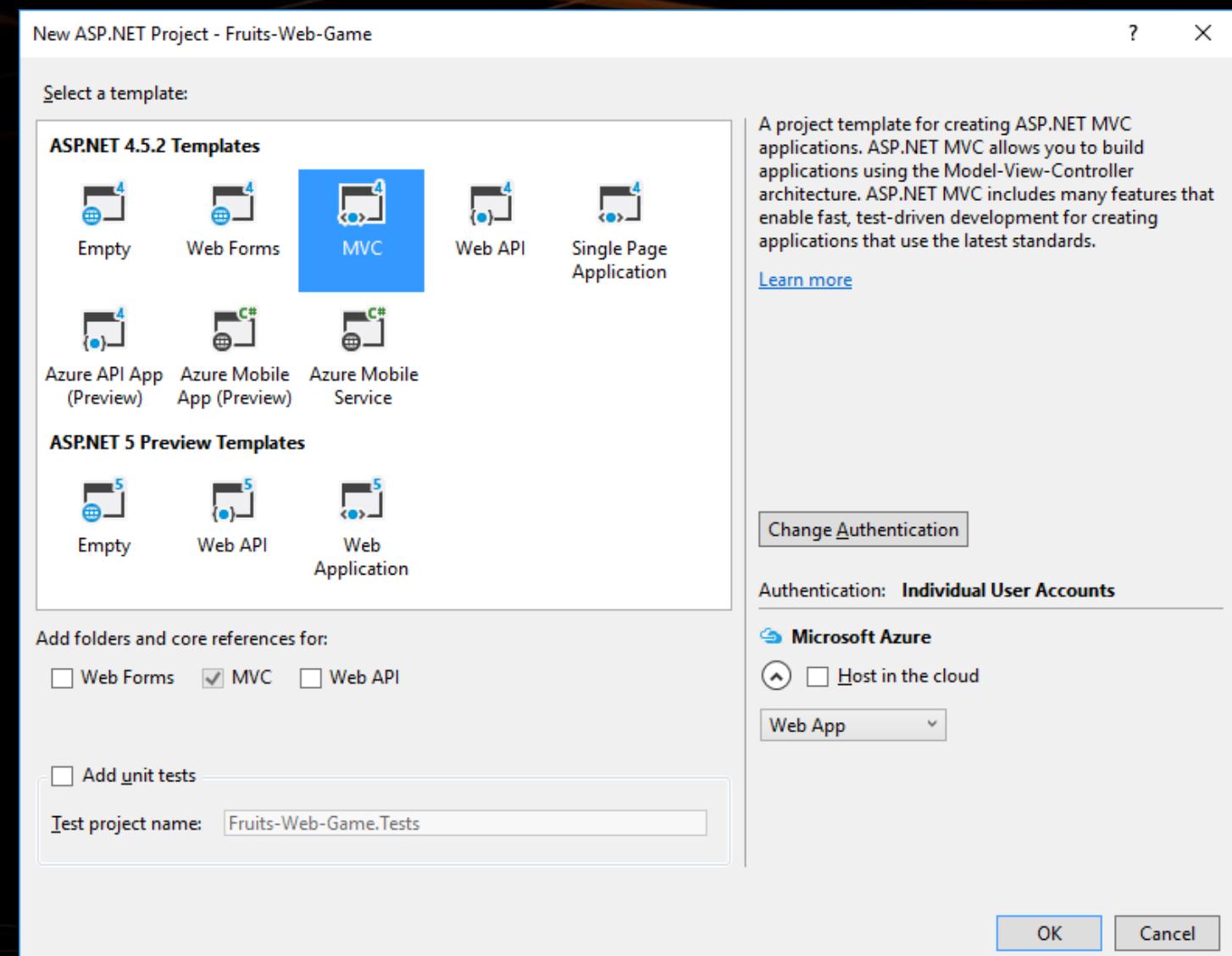


Score: 10

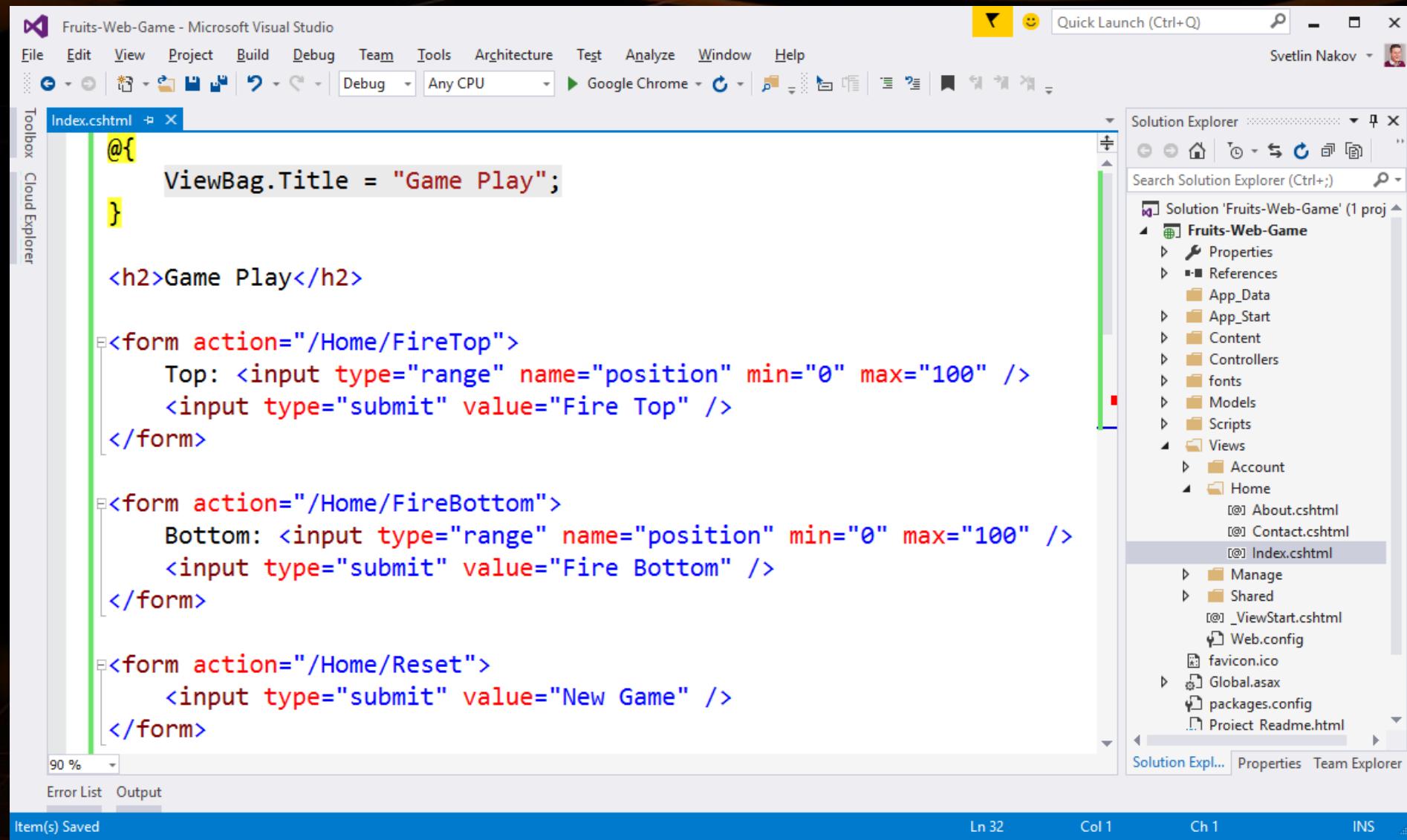
# Създаваме уеб приложение във VS



# Тип на уеб проекта → MVC



# Създаване на контролите за играта



The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** Fruits-Web-Game - Microsoft Visual Studio
- Menu Bar:** File, Edit, View, Project, Build, Debug, Team, Tools, Architecture, Test, Analyze, Window, Help
- Toolbox:** Standard icons for file operations.
- Solution Explorer:** Shows the project structure:
  - Solution 'Fruits-Web-Game' (1 proj)
    - Fruits-Web-Game
      - Properties
      - References
      - App\_Data
      - App\_Start
      - Content
      - Controllers
      - fonts
      - Models
      - Scripts
    - Views
      - Account
      - Home
        - About.cshtml
        - Contact.cshtml
        - Index.cshtml
      - Manage
      - Shared
  - Web.config
  - favicon.ico
  - Global.asax
  - packages.config
  - Project Readme.html
- Search Solution Explorer (Ctrl+;):** Search bar for the solution explorer.
- Code Editor:** The Index.cshtml file contains the following code:

```
ViewBag.Title = "Game Play";

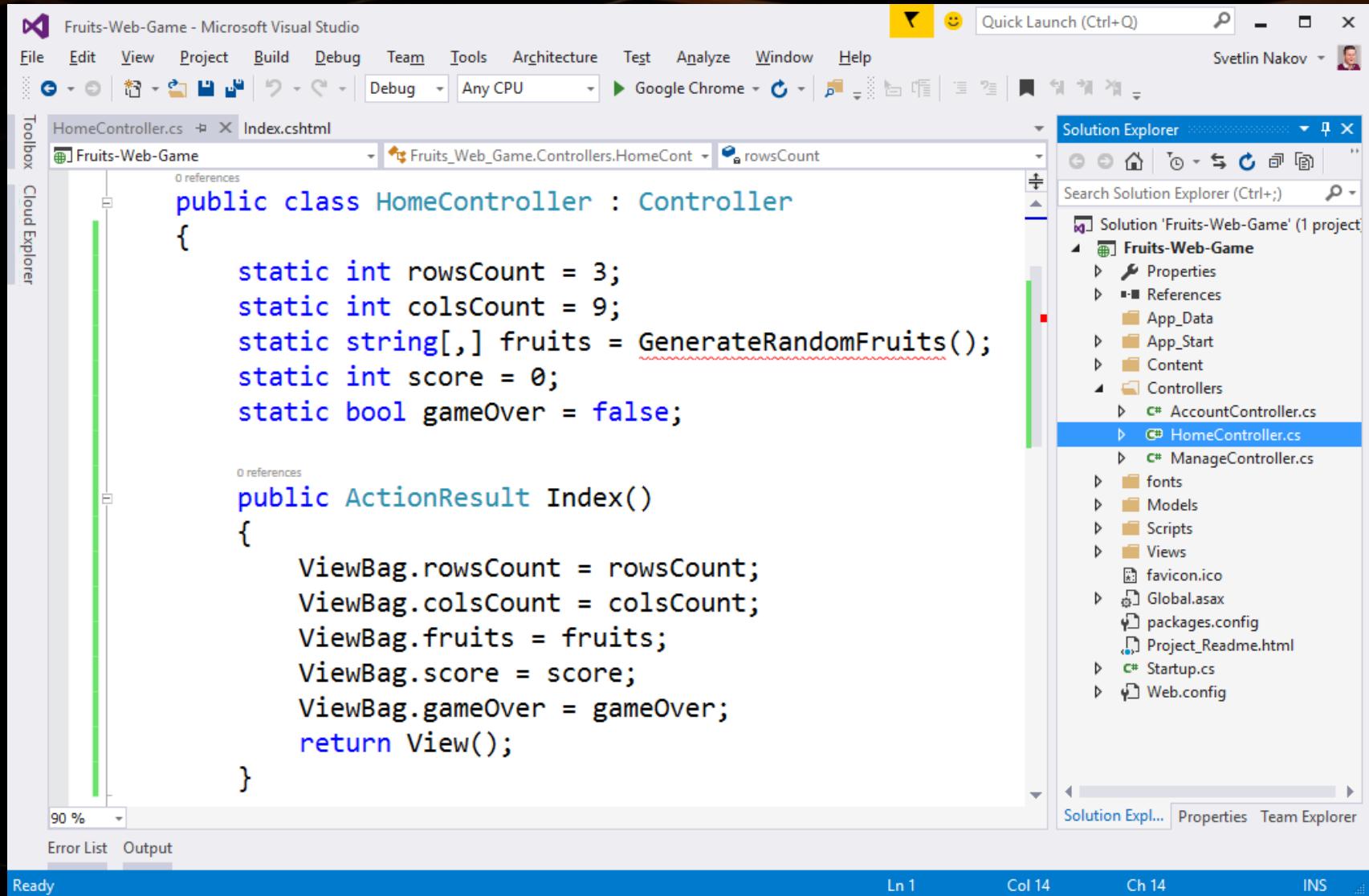
<h2>Game Play</h2>

<form action="/Home/FireTop">
 Top: <input type="range" name="position" min="0" max="100" />
 <input type="submit" value="Fire Top" />
</form>

<form action="/Home/FireBottom">
 Bottom: <input type="range" name="position" min="0" max="100" />
 <input type="submit" value="Fire Bottom" />
</form>

<form action="/Home/Reset">
 <input type="submit" value="New Game" />
</form>
```
- Status Bar:** Item(s) Saved, Ln 32, Col 1, Ch 1, INS

# Подготовка на плодовете в HomeController.cs



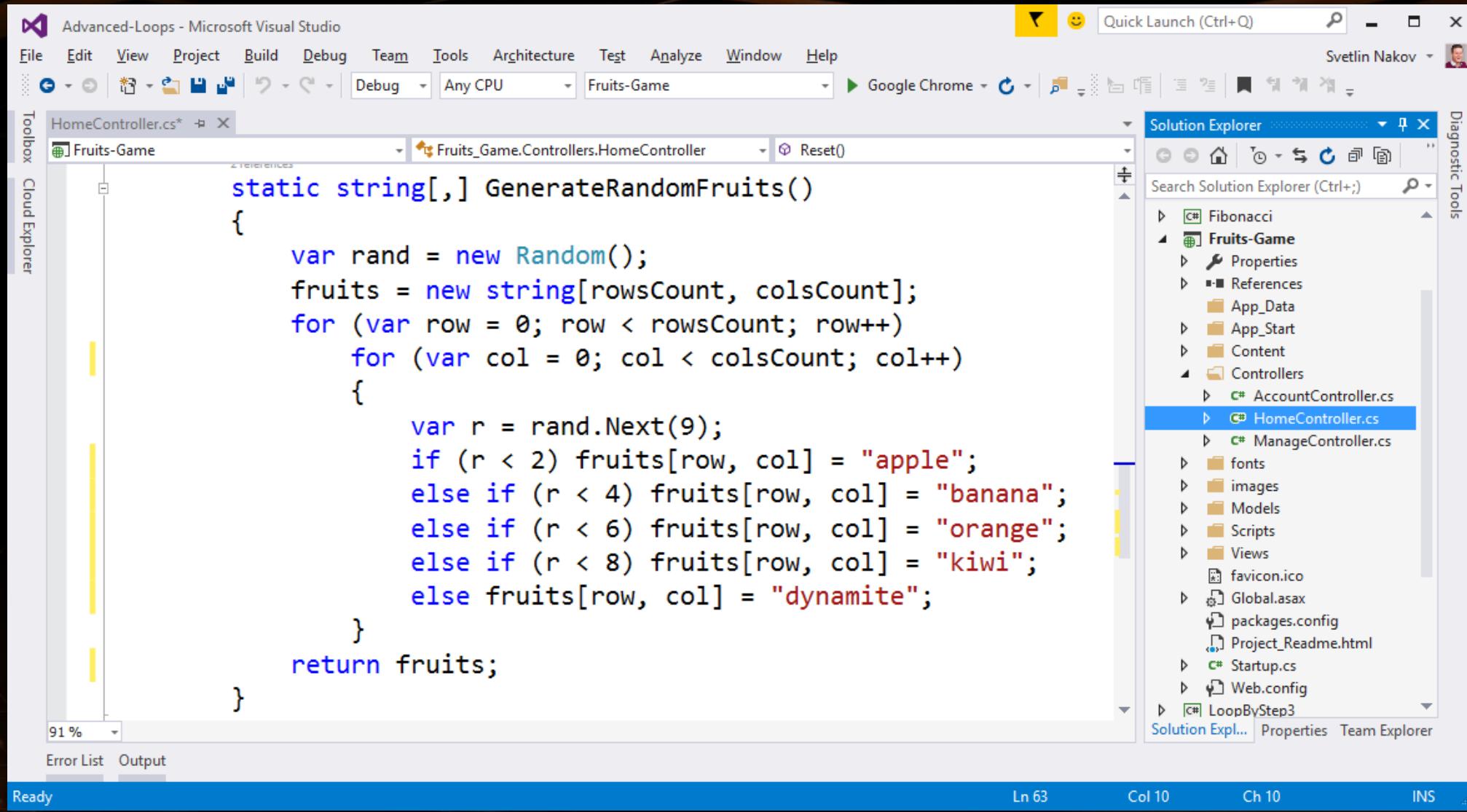
The screenshot shows the Microsoft Visual Studio interface with the project "Fruits-Web-Game" open. The code editor displays the `HomeController.cs` file, which contains the following C# code:

```
public class HomeController : Controller
{
 static int rowsCount = 3;
 static int colsCount = 9;
 static string[,] fruits = GenerateRandomFruits();
 static int score = 0;
 static bool gameOver = false;

 public ActionResult Index()
 {
 ViewBag.rowsCount = rowsCount;
 ViewBag.colsCount = colsCount;
 ViewBag.fruits = fruits;
 ViewBag.score = score;
 ViewBag.gameOver = gameOver;
 return View();
 }
}
```

The `GenerateRandomFruits()` method is highlighted with a red dotted underline, indicating a potential issue or warning. The Solution Explorer on the right shows the project structure, including files like `AccountController.cs`, `HomeController.cs` (which is selected), `ManageController.cs`, and `Startup.cs`.

# Генериране на случаини плодове

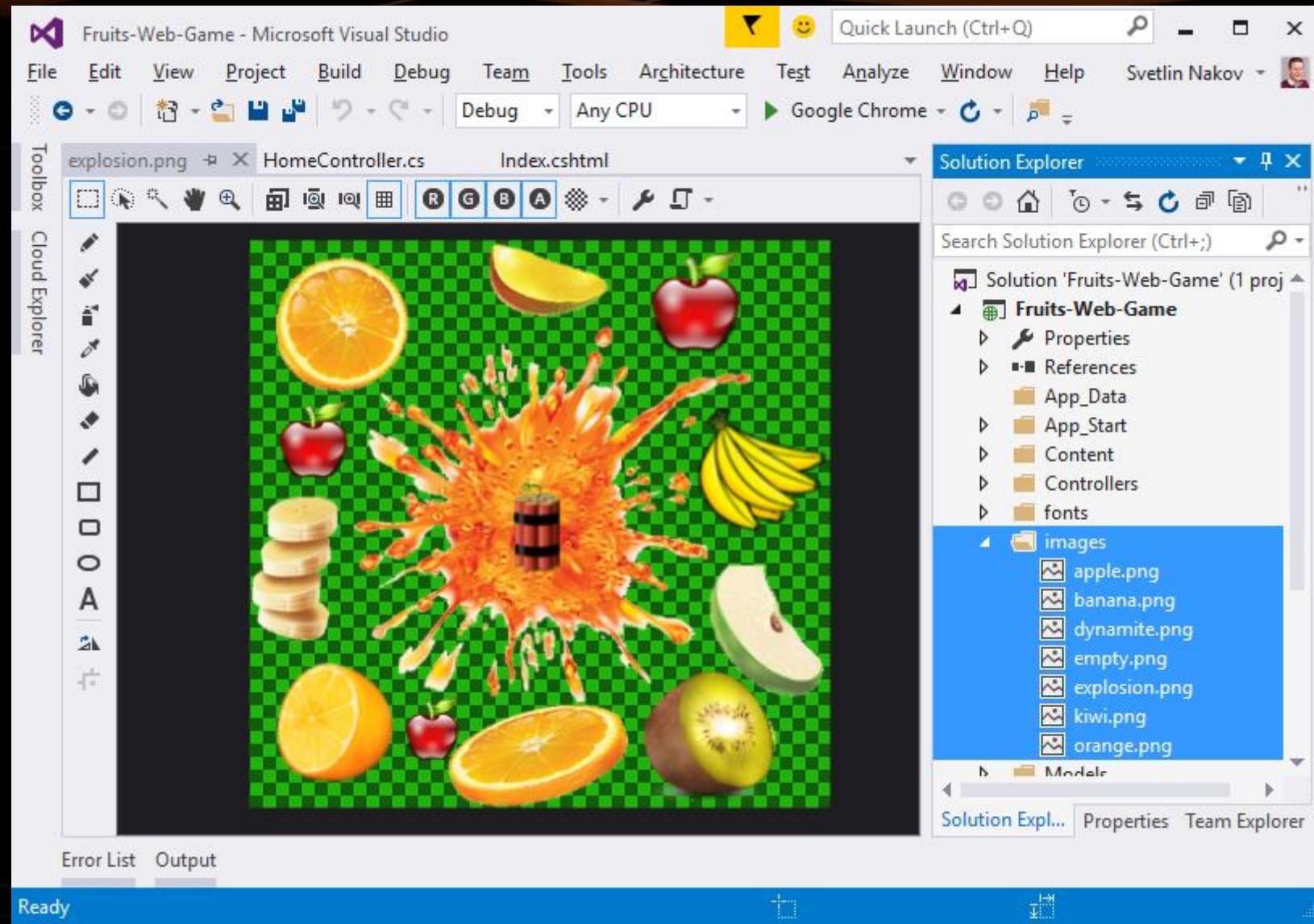


The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** Advanced-Loops - Microsoft Visual Studio
- MenuBar:** File, Edit, View, Project, Build, Debug, Team, Tools, Architecture, Test, Analyze, Window, Help
- Toolbars:** Standard, Debug, Task List, Solution Explorer, Properties, Team Explorer
- Status Bar:** Ready, Ln 63, Col 10, Ch 10, INS
- Solution Explorer:** Shows the project structure:
  - Fibonacci
  - Fruits-Game
    - Properties
    - References
    - App\_Data
    - App\_Start
    - Content
    - Controllers
      - AccountController.cs
      - HomeController.cs**
      - ManageController.cs
    - fonts
    - images
    - Models
    - Scripts
    - Views
    - favicon.ico
  - Global.asax
  - packages.config
  - Project\_Readme.html
  - Startup.cs
  - Web.config
  - LoopByStep3
- Code Editor:** HomeController.cs (selected)

```
static string[,] GenerateRandomFruits()
{
 var rand = new Random();
 fruits = new string[rowsCount, colsCount];
 for (var row = 0; row < rowsCount; row++)
 for (var col = 0; col < colsCount; col++)
 {
 var r = rand.Next(9);
 if (r < 2) fruits[row, col] = "apple";
 else if (r < 4) fruits[row, col] = "banana";
 else if (r < 6) fruits[row, col] = "orange";
 else if (r < 8) fruits[row, col] = "kiwi";
 else fruits[row, col] = "dynamite";
 }
 return fruits;
}
```
- Toolbox:** Fruits-Game
- Cloud Explorer:** (empty)
- Task List:** (empty)
- Output:** (empty)

# Добавяне на картинките в проекта



# Чертане на плодовете в Index.cshtml

Fruits-Web-Game - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Debug Any CPU Google Chrome

Index.cshtml\* Solution Explorer

```
<h2>Battle Field</h2>
@for (var row = 0; row < ViewBag.rowsCount; row++)
{
 for (int col = 0; col < ViewBag.colsCount; col++)
 {

 }

}
<h2>Score: @ViewBag.score</h2>
```

Search Solution Explorer (Ctrl+F)

Fruits-Web-Game

- Properties
- References
- App\_Data
- App\_Start
- Content
- Controllers
- fonts
- Models
- Scripts
- Views
  - Account
  - Home
    - About.cshtml
    - Contact.cshtml
    - Index.cshtml

90 %

Error List Output

Item(s) Saved

Ln 19 Col 8 Ch 8 INS

# Нагласяме текстовете в \_Layout.cshtml

Fruits-Web-Game - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Debug Any CPU Google Chrome

Svetlin Nakov

\_Layout.cshtml\* HomeController.cs Index.cshtml

```
@Html.ActionLink("Home", "Index", "Home")
@Html.ActionLink("About", "About", "Home")
@Html.ActionLink("Contact", "Contact", "Home")

 @Html.Partial("_LoginPartial")
</div>
</div>
<div class="container body-content">
 @RenderBody()
 <hr />
 <footer>
 <p>© @DateTime.Now.Year - My ASP.NET Application</p>
 </footer>
</div>

@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
<!-- bundle -->
```

82 %

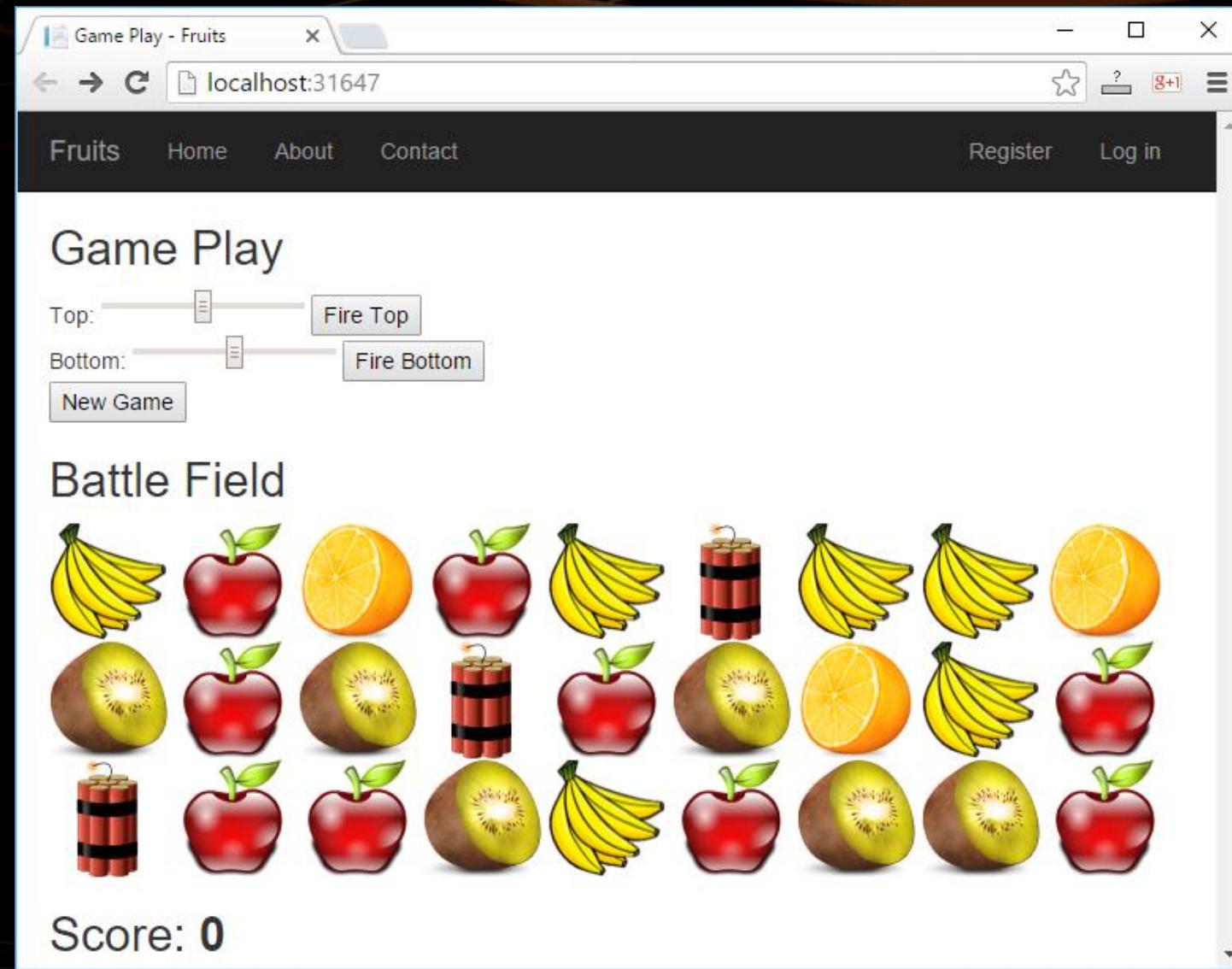
Error List Output

Ln 36 Col 66 Ch 66 INS

**Fruits**

The screenshot shows the Microsoft Visual Studio interface with the \_Layout.cshtml file open. The code displays a navigation menu with three items: Home, About, and Contact. Below the menu is a partial view for login. The main content area contains a placeholder for the body content, a horizontal line, and a footer section. The footer section contains a copyright notice: '&copy; @DateTime.Now.Year - My ASP.NET Application'. A blue oval highlights the word 'Fruits' in the footer text, which is crossed out with a large red X. A blue arrow points from the word 'Fruits' to the crossed-out text. The Solution Explorer on the right shows the project structure, including Views, Scripts, and various cshtml files.

# Стартираме с [Ctrl+F5] и тестваме



# Добавяме действията "Reset" и "Fire"



Microsoft Visual Studio window showing the 'Fruits-Web-Game' project.

**Solution Explorer:**

- Fruits-Web-Game (1 project)
  - Properties
  - References
  - App\_Data
  - App\_Start
  - Content
  - Controllers
    - AccountController.cs
    - HomeController.cs** (selected)
    - ManageController.cs
  - fonts
  - images
  - Models
  - Scripts
  - Views
  - favicon.ico
  - Global.asax
  - packages.config
  - Project\_Readme.html
  - Startup.cs
  - Web.config

**HomeController.cs:**

```

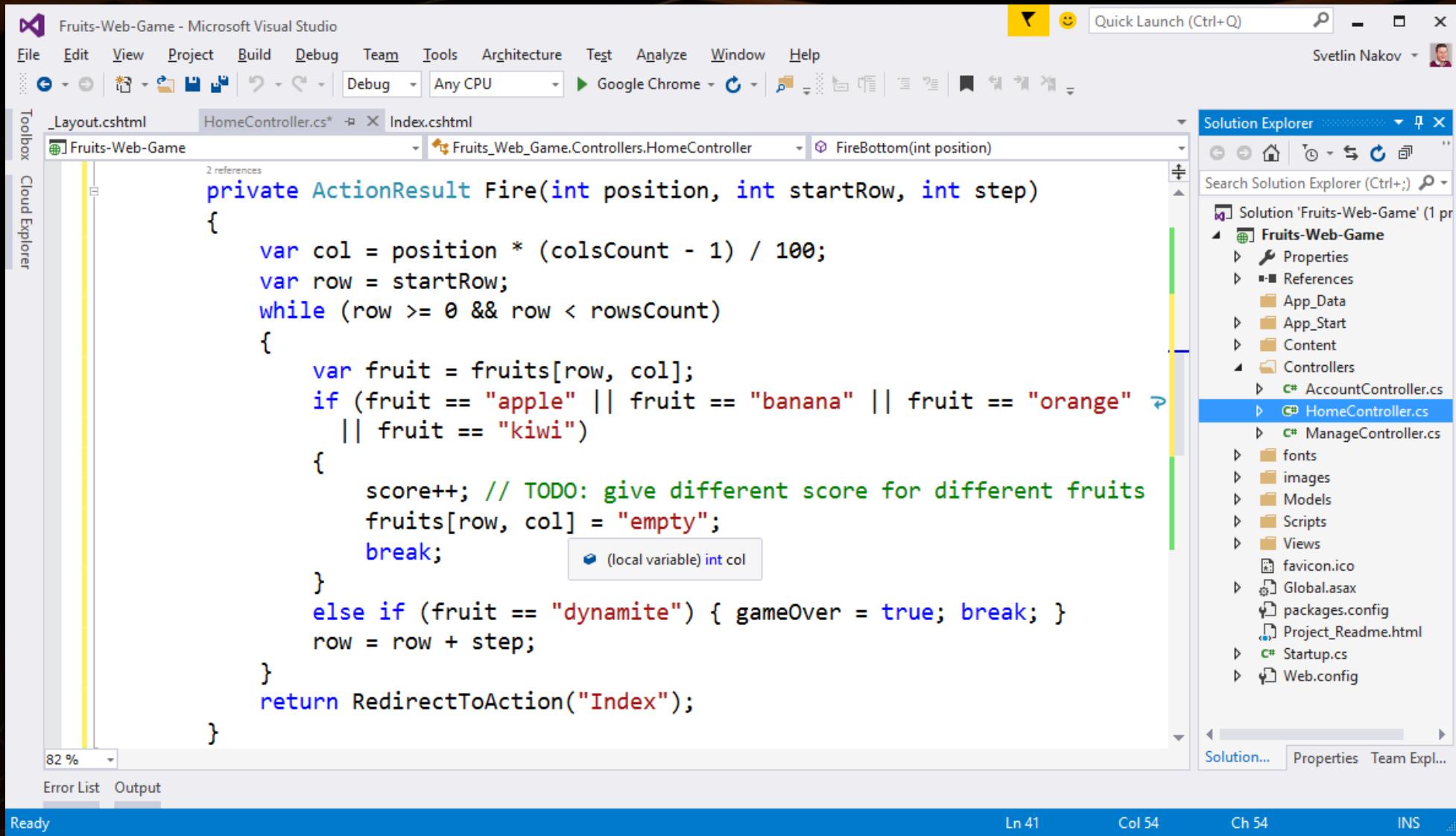
public ActionResult Reset()
{
 fruits = GenerateRandomFruits();
 gameOver = false;
 return RedirectToAction("Index");
}

public ActionResult FireTop(int position)
{
 return Fire(position, 0, 1);
}

public ActionResult FireBottom(int position)
{
 return Fire(position, rowsCount - 1, -1);
}

```

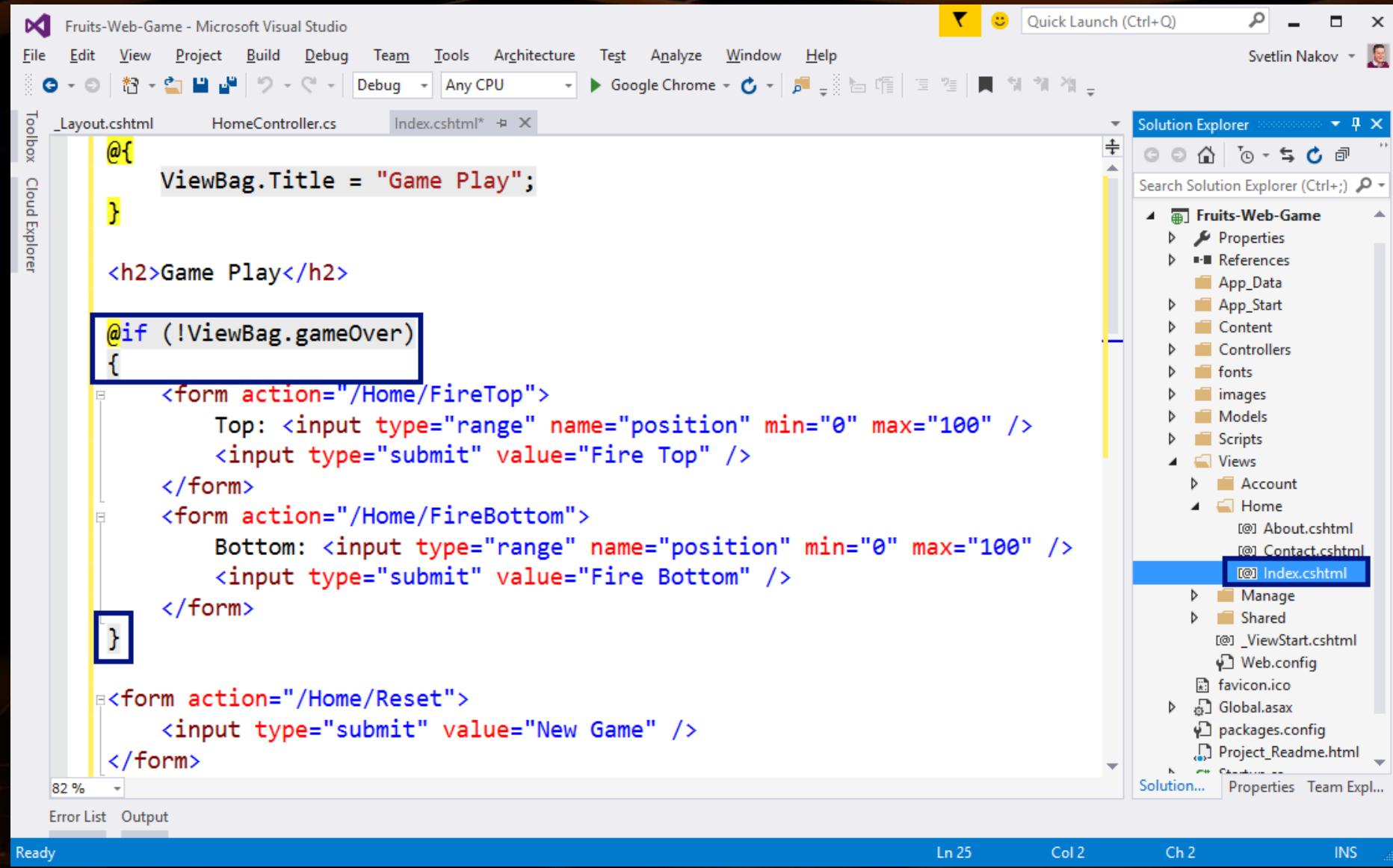
# Имплементираме "стрелянето"



The screenshot shows the Microsoft Visual Studio interface with the project 'Fruits-Web-Game' open. The code editor displays the `HomeController.cs` file, specifically the `Fire` method. The method takes three parameters: `int position`, `int startRow`, and `int step`. It iterates through rows, checking for specific fruits ('apple', 'banana', 'orange', 'kiwi'). If found, it increments the score, marks the fruit as 'empty', and breaks out of the loop. If a 'dynamite' fruit is found, it sets the game over flag and breaks. The row index is updated by the step value. Finally, it returns a redirect to the 'Index' action.

```
private ActionResult Fire(int position, int startRow, int step)
{
 var col = position * (colsCount - 1) / 100;
 var row = startRow;
 while (row >= 0 && row < rowsCount)
 {
 var fruit = fruits[row, col];
 if (fruit == "apple" || fruit == "banana" || fruit == "orange" ||
 || fruit == "kiwi")
 {
 score++; // TODO: give different score for different fruits
 fruits[row, col] = "empty";
 break;
 }
 else if (fruit == "dynamite") { gameOver = true; break; }
 row = row + step;
 }
 return RedirectToAction("Index");
}
```

# Имплементираме "Край на играта"



The screenshot shows the Microsoft Visual Studio interface for a web application named "Fruits-Web-Game". The main window displays the code for the "Index.cshtml" view. The code includes logic to set the ViewBag.Title to "Game Play" and to render two forms for firing at the top or bottom. A third form for resetting the game is also present. The Solution Explorer on the right shows the project structure, including the "Views/Home/Index.cshtml" file which is currently selected.

```
ViewBag.Title = "Game Play";

<h2>Game Play</h2>

@if (!ViewBag.gameOver)
{
 <form action="/Home/FireTop">
 Top: <input type="range" name="position" min="0" max="100" />
 <input type="submit" value="Fire Top" />
 </form>
 <form action="/Home/FireBottom">
 Bottom: <input type="range" name="position" min="0" max="100" />
 <input type="submit" value="Fire Bottom" />
 </form>
}

<form action="/Home/Reset">
 <input type="submit" value="New Game" />
</form>
```

# Имплементираме "Край на играта" (2)

Fruits-Web-Game - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Debug Any CPU Google Chrome

\_Layout.cshtml HomeController.cs Index.cshtml

Toolbox Cloud Explorer

```
@if (ViewBag.gameOver)
{
 <h2>Game Over!</h2>

}
else
{
 <h2>Battle Field</h2>
 for (var row = 0; row < ViewBag.rowsCount; row++)
 {
 for (int col = 0; col < ViewBag.colsCount; col++)
 {

 }

 }
 <h2>Score: @ViewBag.score</h2>
}
```

Svetlin Nakov

Solution Explorer

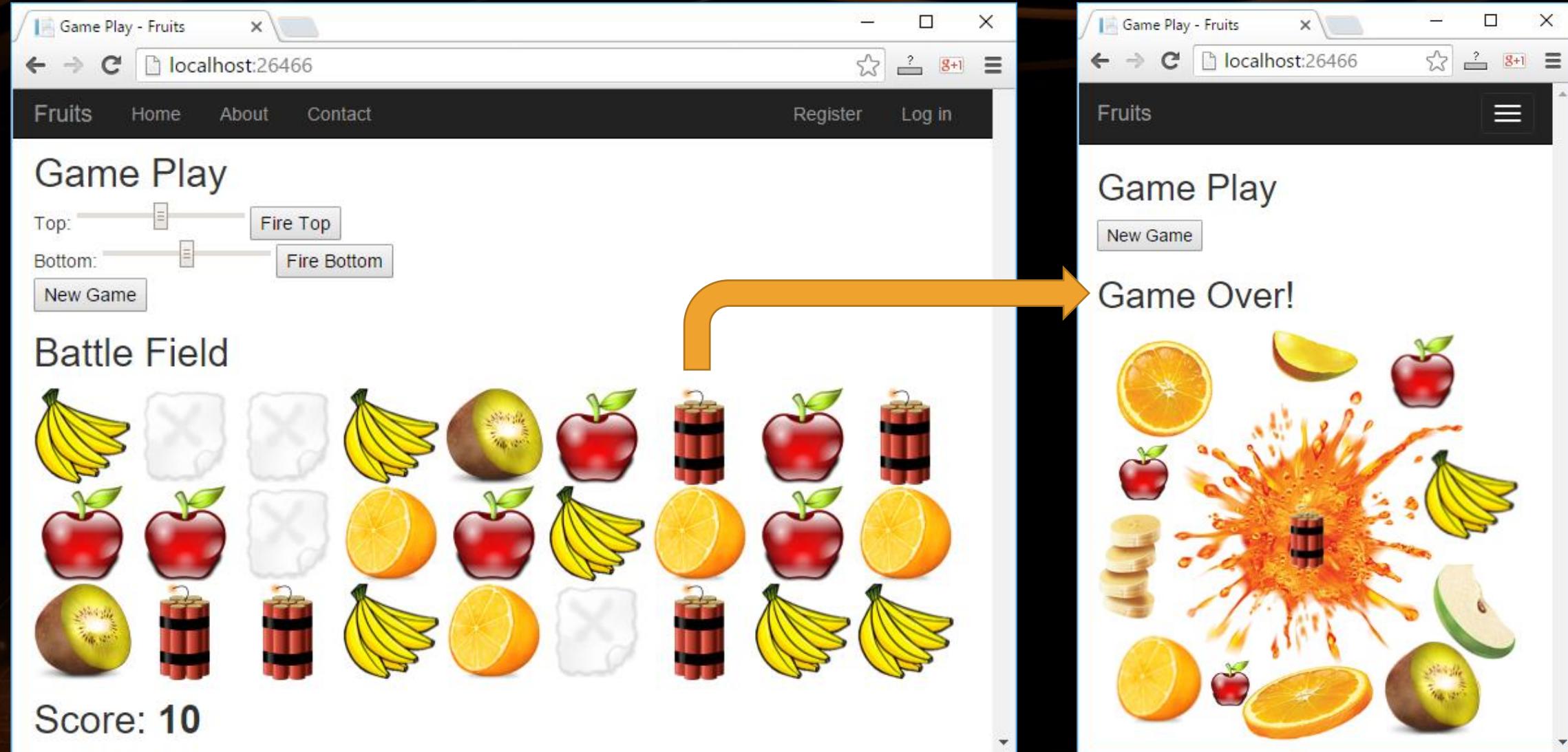
Search Solution Explorer (Ctrl+Shift+F)

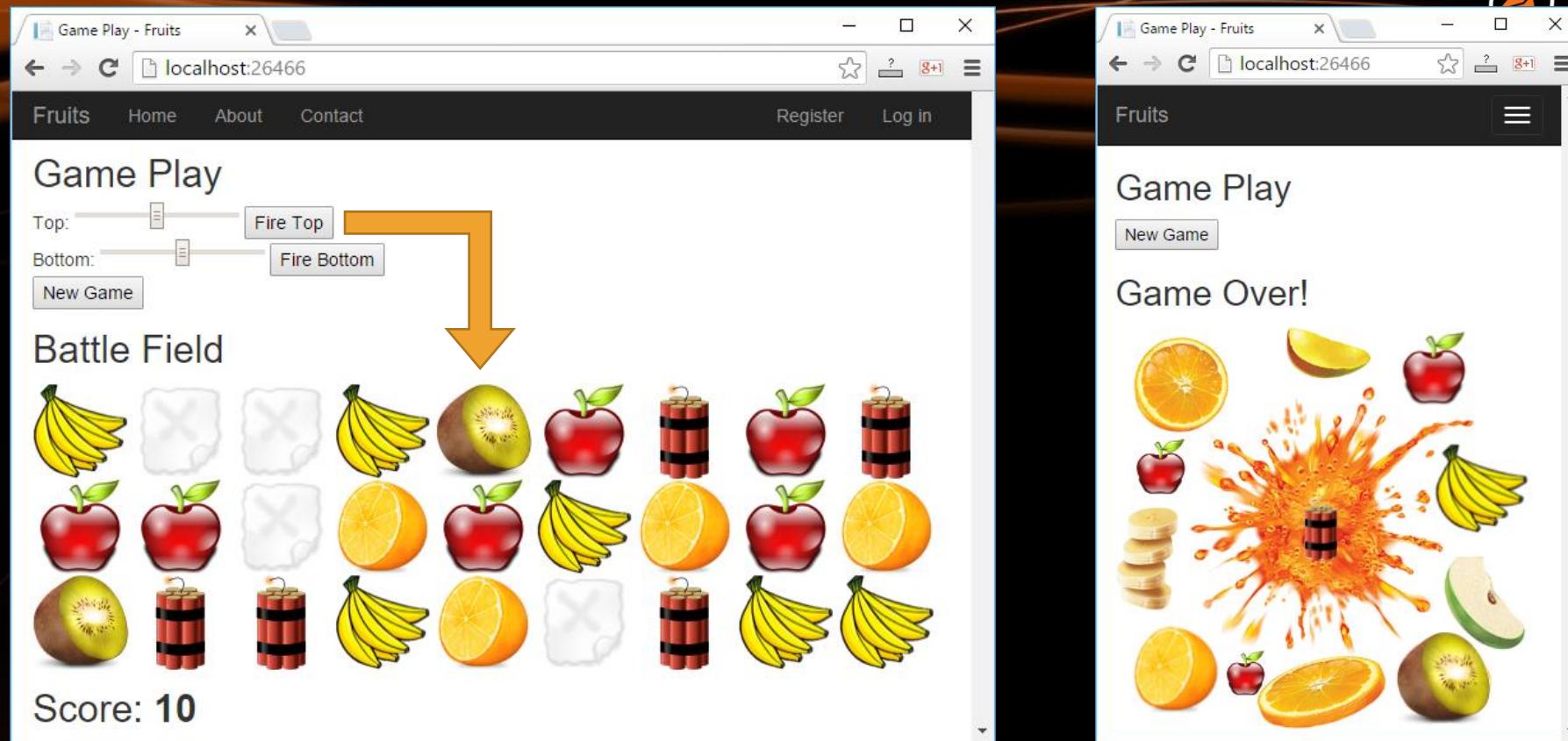
- Fruits-Web-Game
  - Properties
  - References
  - App\_Data
  - App\_Start
  - Content
  - Controllers
  - fonts
  - images
  - Models
  - Scripts
  - Views
    - Account
    - Home
      - About.cshtml
      - Contact.cshtml
      - Index.cshtml
    - Manage
    - Shared
  - \_ViewStart.cshtml
  - Web.config
- favicon.ico
- Properties
- Team Expl...

Error List Output

Ready Ln 20 Col 6 Ch 6 INS

# Стартираме с [Ctrl+F5] и тестваме





# Уеб игра "Обстреляй плодовете!"

Работа на живо в клас (лаб)

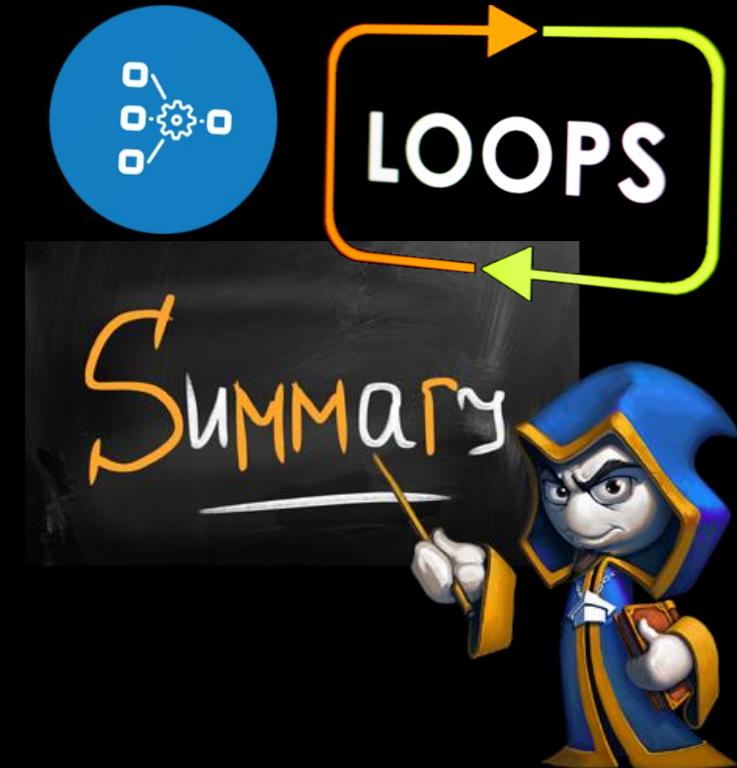
# Какво научихме днес?

- Можем да ползваме **for**-цикли със стъпка:

```
for (var i = 1; i <= n; i+=3)
 Console.WriteLine(i);
```

- Цикли **while** / **do-while** повтаря докато е в сила дадено условие:

```
int num = 1;
while (num <= n)
 Console.WriteLine(num++);
```



# Работа с по-сложни цикли



Въпроси?

SUPERHOSTING.BG



# Лиценз

- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"



- Благодарности: настоящият материал може да съдържа части от следните източници
  - Книга "Основи на програмирането със C#" от Светлин Наков и колектив с лиценз CC-BY-SA

# Бесплатни обучения в СофтУни

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

