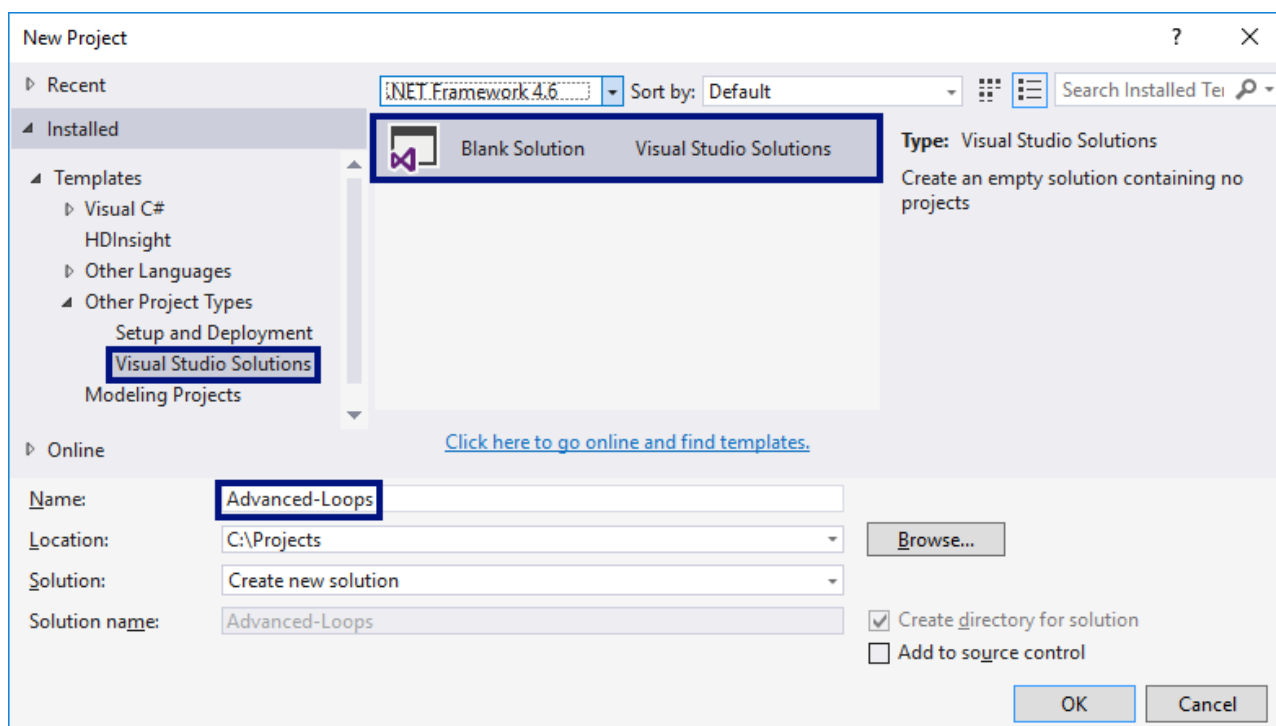


Упражнения: Работа с по-сложни цикли

Задачи за упражнение в клас и за домашно към курса „Основи на програмирането“ @ СофтУни.

0. Празно Visual Studio решение (Blank Solution)

1. Създайте празно решение (**Blank Solution**) във Visual Studio за да организирате кода от задачите за упражнение. Целта на този **blank solution** е да съдържа **по един проект за всяка задача** от упражненията.



2. Задайте **да се стартира по подразбиране текущия проект** (не първият в решението). Кликнете с десен бутон на мишката върху **Solution 'Advanced-Loops' → [Set StartUp Projects...] → [Current selection]**.

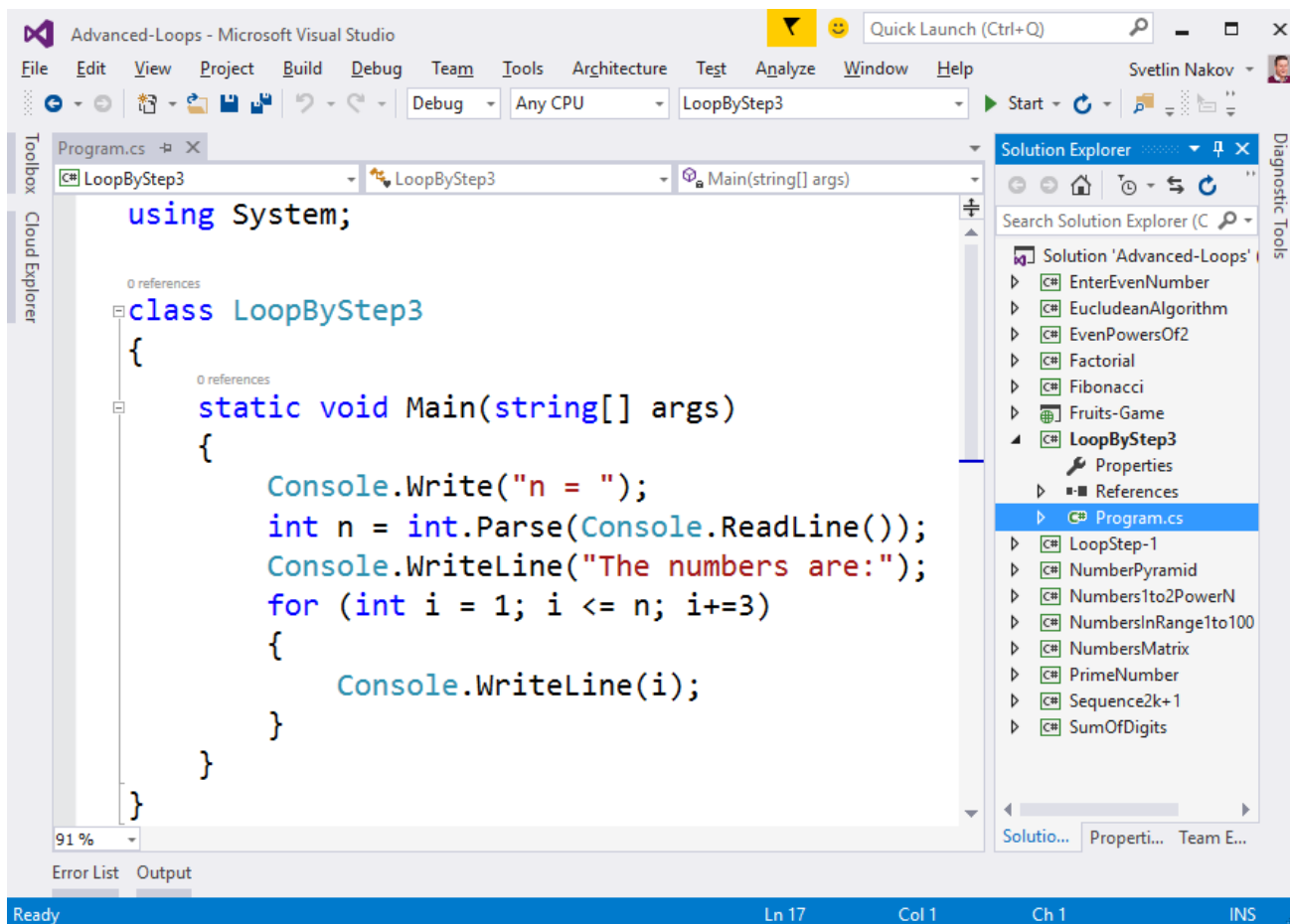
1. Числата от 1 до N през 3

Напишете програма, която въвежда число **n** и отпечатва **числата от 1 до n през 3** (със стъпка 3). Примери:

ВХОД	ИЗХОД	ВХОД	ИЗХОД	ВХОД	ИЗХОД
10	1 4 7 10	7	1 4 7	15	1 4 7 10 13

Подсказки:

1. Създайте **нов проект** в съществуващото Visual Studio решение – конзолна C# програма. Задайте подходящо име на проекта, например **“LoopByStep3”**.
2. Можете да завъртите **for-цикъл със стъпка 3** по следния начин: **for (var i = 1; i <= n; i+=3)**.
3. Отидете в тялото на метода **Main(string[] args)** и напишете решението на задачата. Можете да си помогнете с кода от картинката по-долу:



4. **Стартирайте** програмата с [Ctrl+F5] и я **тествайте**:

```

C:\WINDOWS\system32\cmd.exe
n = 10
The numbers are:
1
4
7
10
Press any key to continue . . .
  
```

```

C:\WINDOWS\system32\cmd.exe
n = 8
The numbers are:
1
4
7
Press any key to continue . . .
  
```

5. **Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#0>. Трябва да получите **100 точки** (напълно коректно решение).

2. Числата от N до 1 в обратен ред

Напишете програма, която въвежда цяло положително число **n** и печата **числата от n до 1 в обратен ред** (от най-голямото към най-малкото). Примери:

ВХОД	ИЗХОД	ВХОД	ИЗХОД	ВХОД	ИЗХОД
2	2 1	3	3 2 1	5	5 4 3 2 1

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#1>.

Подсказка: отпечатайте **n** звездички в цикъл **n** пъти, точно както в предната задача.

3. Числа от 1 до 2^n

Напишете програма, която чете от конзолата цяло число **n** и **печата числата от 1 до 2^n** . Примери:

вход	изход	вход	изход	вход	изход
3	1 2 4 8	4	1 2 4 8 16	5	1 2 4 8 16 32

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#2>.

Подсказка: завъртете **for**-цикъл от **0** до **n** и започвайки от **num = 1** на всяка стъпка умножавайте **num** по **2**.

4. Четни степени на 2

Да се напише програма, която въвежда **n** и **печата четните степени на 2** $2 \leq 2^n$: $2^0, 2^2, 2^4, 2^8, \dots, 2^n$. Примери:

вход	изход	вход	изход	вход	изход	вход	изход	вход	изход
3	1 4	4	1 4 16	5	1 4 16	6	1 4 16 64	7	1 4 16 64

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#3>.

Подсказка: започнете от 1 и в цикъл умножавайте по 4 на всяка стъпка.

5. Редица числа $2k+1$

Напишете програма, която въвежда число **n** и отпечата **всички числа $\leq n$ от редицата**: 1, 3, 7, 15, 31, Всяко следващо число се изчислява като **предишното число * 2 + 1**. Примери:

вход	изход	вход	изход	вход	изход	вход	изход
3	1 3	8	1 3 7	17	1 3 7 15	31	1 3 7 15 31

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#4>.

Подсказки:

- Започнете от **num = 1**.
- В цикъл докато **num** не стигне **n**, печатайте, го умножавайте по **2** и прибавяйте **1**.

6. Число в диапазона [1...100]

Напишете програма, която въвежда цяло положително число **n** в диапазона [1...100]. При въвеждане на число извън посочения диапазон, да се отпечата съобщение за грешка и потребителят да се подкани **да въведе ново число**. Примери:

вход / изход
Enter a number in the range [1...100]: 35 The number is: 35
Enter a number in the range [1...100]: 105 Invalid number! Enter a number in the range [1...100]: 0 Invalid number! Enter a number in the range [1...100]: -200 Invalid number! Enter a number in the range [1...100]: 77 The number is: 77

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#5>.

Подсказки:

- Въведете число.
- Повтаряйте в цикъл докато числото е невалидно: отпечатайте грешка и въведете число отново.

7. Най-голям общ делител (НОД)

Напишете програма, която въвежда две цели положителни числа **a** и **b** и изчислява и отпечата **най-големият им общ делител (НОД)**. Примери:

вход	изход	вход	изход	вход	изход	вход	изход	вход	изход
24	8	67	1	15	3	100	4	10	10
16		18		9		88		10	

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#6>.

Подсказка: имплементирайте **алгоритъма на Евклид**: <https://bg.wikipedia.org/wiki/алгоритъм-на-Евклид>.

8. Факториел

Напишете програма, която въвежда цяло число **n** ($1 \leq n \leq 12$) и изчислява и отпечата $n! = 1 * 2 * \dots * n$ (**факториел**). Примери:

вход	изход	вход	изход	вход	изход	вход	изход	вход	изход
5	120	6	720	10	3628800	1	1	2	2

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#7>.

Подсказка: в цикъл умножете числата от 1 до n.

9. Сумиране на цифрите на число

Напишете програма, която въвежда цяло число **num** и отпечата **сумата от цифрите му**. Примери:

вход	изход	коментар	вход	изход	коментар	вход	изход	вход	изход
5634	18	6+6+3+4 = 18	19	10	1+9 = 10	5	5	17151	15

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#8>.

Подсказка: в цикъл докато не стигнете до 0 сумирайте последната цифра на числото ($num \% 10$) и го разделяйте след това на 10 (така изтривате последната му цифра).

10. Проверка за просто число

Напишете програма, която въвежда цяло число **n** и **проверява дали е просто число** (дали се дели само на себе си и на единица). Да се отпечата **"Prime"** или **"Not prime"**. Примери:

вход	изход	вход	изход	вход	изход	вход	изход	вход	изход
2	Prime	3	Prime	4	Not Prime	5	Prime	7	Prime

вход	изход	вход	изход	вход	изход	вход	изход
1	Not Prime	0	Not Prime	-1	Not Prime	149	Prime

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#9>.

Подсказки:

- Ако числото е по-малко от 2, значи не е просто.
- Всяко друго число първоначално се приема за **просто** и се проверява в цикъл дали се дели на числата 2, 3, 4, ..., до корен квадратен от **n** (цялата част).

11. Въвеждане на четно число (с обработка на грешен вход)

Напишете програма, която **въвежда четно число**. Ако потребителят въведе **грешно число** (нечетно число или стринг, който не е цяло число), трябва да му излиза **съобщение за грешка** и да **въвежда отново**. Примери:

вход / изход
Enter even number: 34 Even number entered: 35
Enter even number: 35 The number is not even. Enter even number: hello Invalid number! Enter even number: 12.85 Invalid number! Enter even number: 3464232636536513 Invalid number! Enter even number: 8 Even number entered: 8

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/156#10>.

Подсказки:

- В цикъл въвеждайте число и проверявайте дали е четно. При коректно число излезте от цикъла.
- С **try { ... } catch { ... }** конструкция прихванете грешните числа, които не могат да се обърнат в **int**.

12. Числа на Фибоначи

Напишете програма, която въвежда цяло число **n** и пресмята **n-тото число на Фибоначи**. Нулевото число на Фибоначи е 1, първото е също 1, а всяко следващо е сумата от предходните две. Примери:

вход	изход	вход	изход	вход	изход	вход	изход	вход	изход
0	1	1	1	2	2	5	8	10	89

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#11>.

Подсказка:

- При $n < 2$ отпечатайте 1.
- Започнете от $f0=1$ и $f1=1$ и в цикъл сумирайте последните две числа. Записвайте последните две числа след всяка стъпка в $f0$ и $f1$.

13. Пирамида от числа

Напишете програма, която въвежда цяло число n и отпечатва пирамида от числа като в примерите:

вход	изход	вход	изход	вход	изход	вход	изход
7	1 2 3 4 5 6 7	10	1 2 3 4 5 6 7 8 9 10	12	1 2 3 4 5 6 7 8 9 10 11 12	15	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#12>.

Подсказка:

- С два вложени цикъла печатайте пирамида от числа: на първия ред едно число, на втория ред 2 числа, на третия ред 3 числа и т.н.
- В отделен брояч пазете колко числа сте отпечатали до момента (и кое е текущото число). Когато стигнете n , излезте внимателно от двата вложени цикъла с **break** или **return**.

14. Таблица с числа

Напишете програма, която въвежда цяло число n и отпечатва таблица (матрица) от числа като в примерите:

вход	изход	вход	изход	вход	изход	вход	изход
2	1 2 2 1	3	1 2 3 2 3 2 3 2 1	4	1 2 3 4 2 3 4 3 3 4 3 2 4 3 2 1	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

Тествайте решението си в judge системата: <https://judge.softuni.bg/Contests/Practice/Index/156#13>.

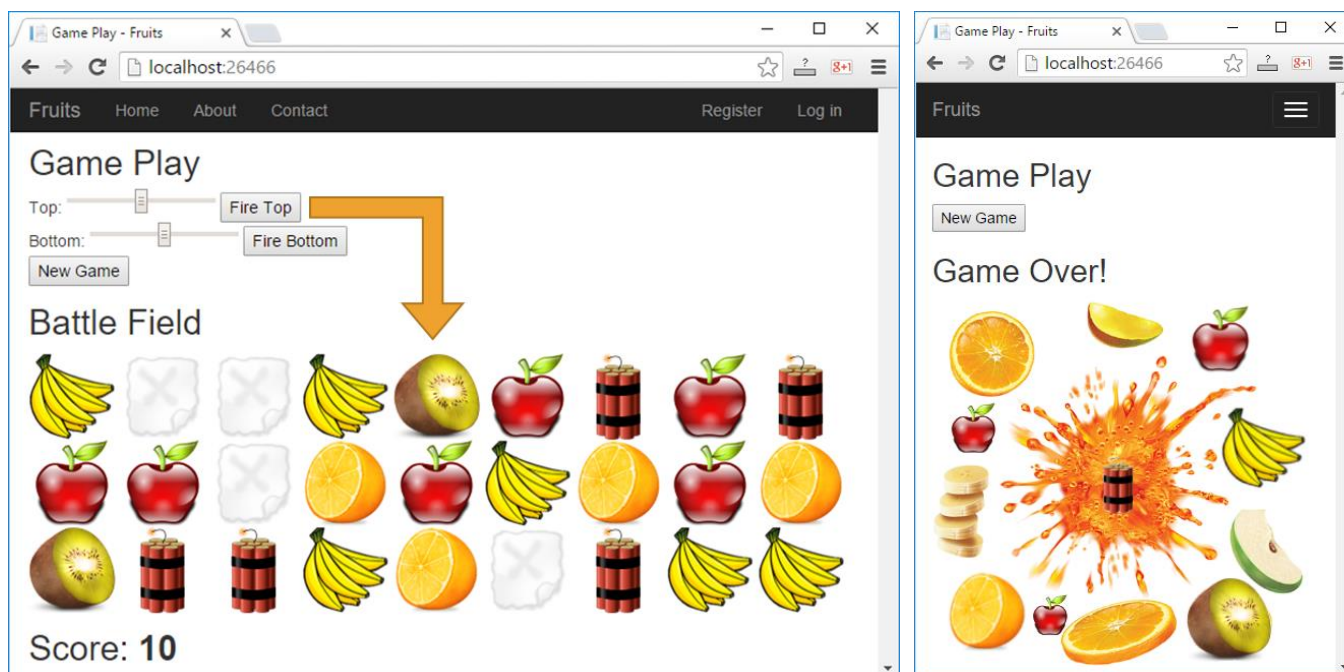
Подсказка:

- С два вложени цикъла за **row** (ред) и **col** (колона) печатайте число по формулата $num = row + col + 1$.
- За долната дясна половина на таблицата ще се получат грешни резултати. Там използвайте формулата $2*n - num$.
- Как се сетихме за тези формули? Математическа досетливост: наблюдаваме числата, предполагаме каква е формулата, тестваме и ако не се получи, измисляме друга формула и пак пробваме. В случая имаме различни формули за горната лява и долната дясна половина от матрицата.

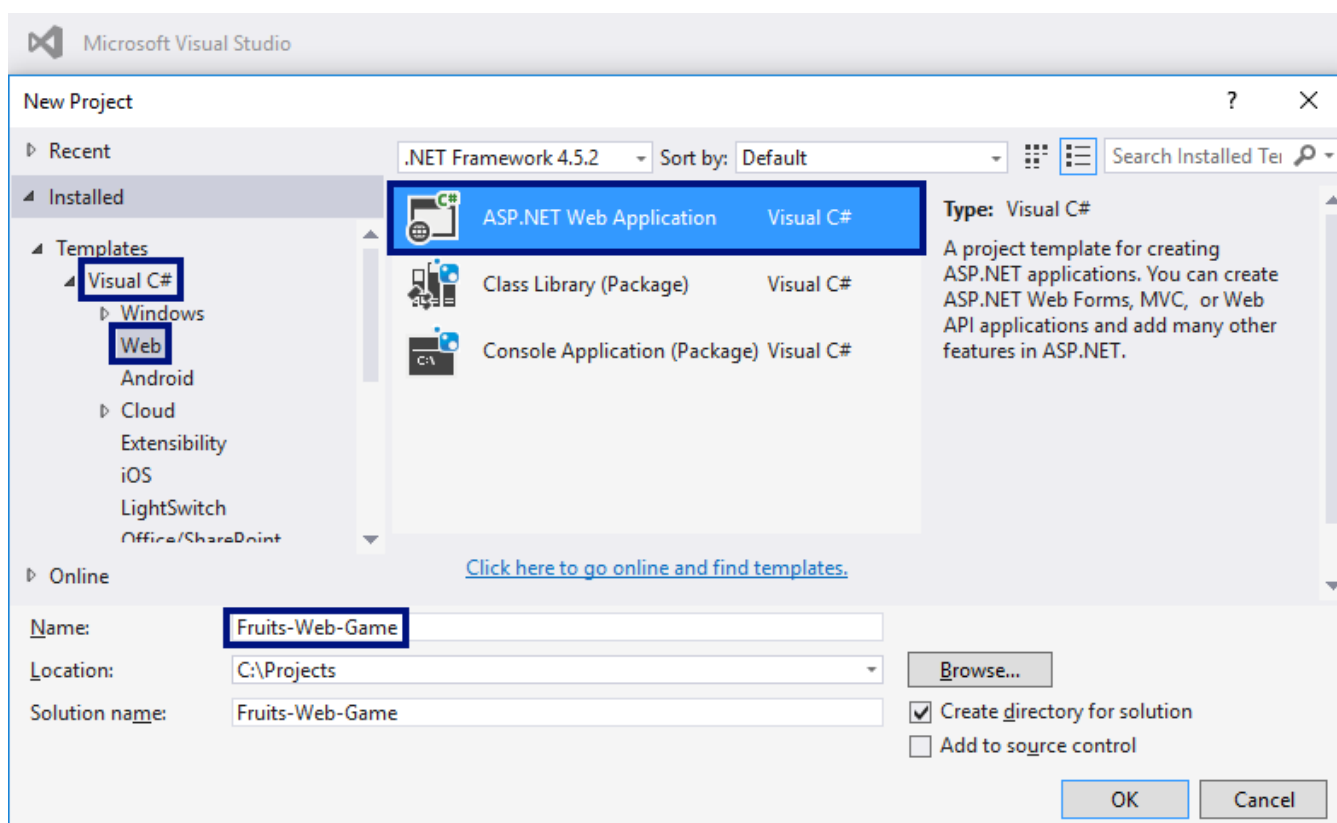
15. Уеб игра „Обстреляй плодовете!“

Да се разработи ASP.NET MVC уеб приложение – игра, в която играчът стреля по плодове, подредени в таблица. Успешно уцелените плодове изчезват, а играчът получава точки за всеки уцелен плод. При уцелване на динамит, плодовете се взривяват и играта свършва (като във Fruit Ninja).

Стрелбата се извършва по колони, отгоре надолу или отдолу нагоре, а местоположението на удара (колоната под обстрел) се задава чрез скролер (scroll bar). Заради неточността на скролера, играчът не е съвсем сигурен по коя колона ще стреля. Така при всеки изстрел има шанс да не улови и това прави играта по-интересна (подобно на прашката в Angry Birds).



1. Във Visual Studio създайте **ново ASP.NET MVC уеб приложение** с език **C#**. Добавете нов проект от [Solution Explorer] → [Add] → [New Project...]. Дайте смислено име, например **"Fruits-Web-Game"**:



Изберете тип на уеб приложението **"MVC"**:

New ASP.NET Project - Fruits-Web-Game

Select a template:

ASP.NET 4.5.2 Templates

Empty Web Forms **MVC** Web API Single Page Application

Azure API App (Preview) Azure Mobile App (Preview) Azure Mobile Service

ASP.NET 5 Preview Templates

Empty Web API Web Application

Add folders and core references for:

☐ Web Forms ☒ MVC ☐ Web API

☐ Add unit tests

Test project name: Fruits-Web-Game.Tests

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

[Learn more](#)

Change Authentication

Authentication: Individual User Accounts

Microsoft Azure

☐ Host in the cloud

Web App

OK Cancel

2. Сега създавайте контролите за играта.

Целта е да добавите **скролиращи ленти** (scroll bars), с които се играчът се прицелва, и бутон за старт на **нова игра**. Редактирайте файла **Views/Home/Index.cshtml**. Изтрийте всичко и въведете кода от картинката:

```

@{
    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>

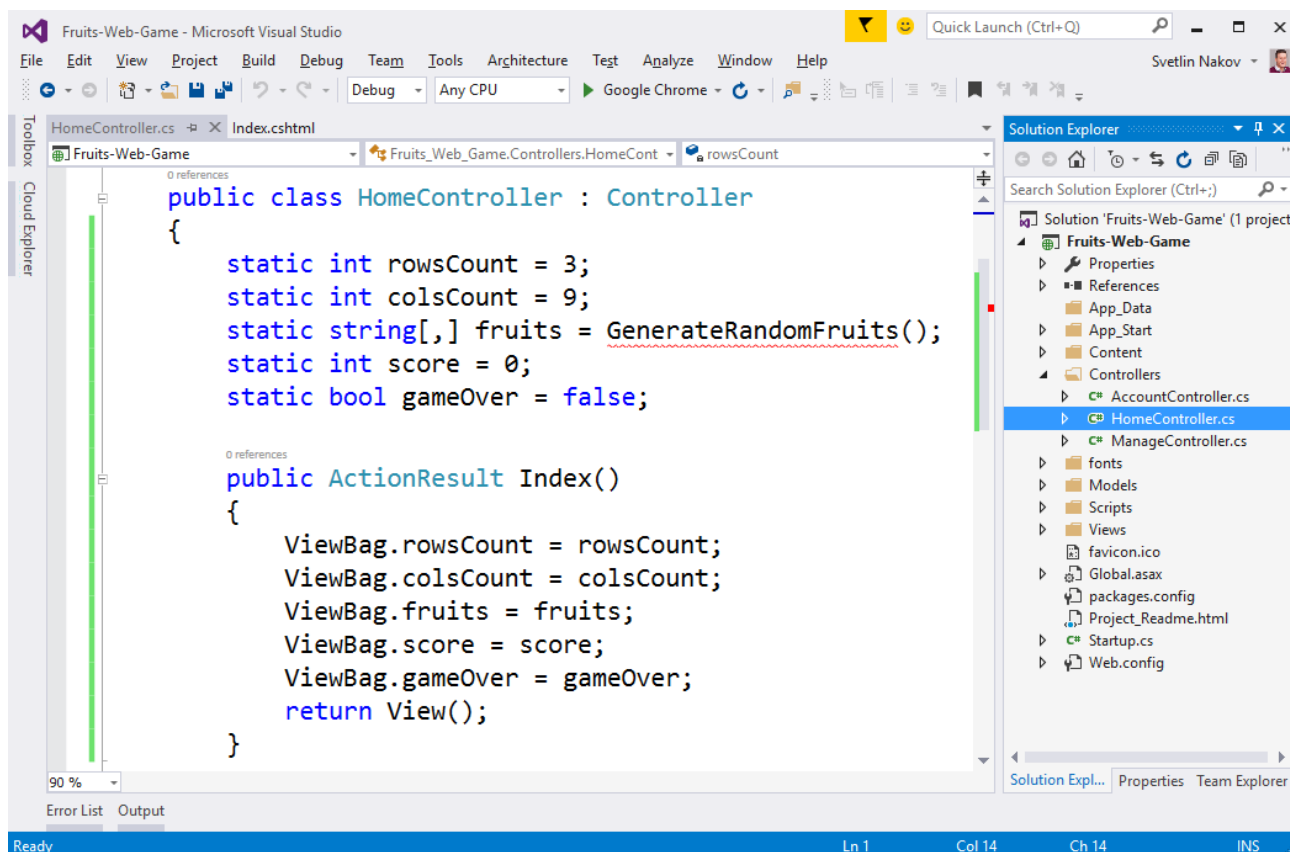
<form action="/Home/Reset">
    <input type="submit" value="New Game" />
</form>

```


Този код създава уеб форма <form> със скролер (поле) “**position**” за задаване на число в интервала [0...100] и бутон [Fire Top] за изпращане на данните от формата към сървъра. Действието, което ще обработи данните, се казва “/Home/FireTop”, което означава метод “**FireTop**” в контролер “**Home**”, който се намира във файла “**HomeController.cs**”. Следват още две подобни форми с бутони [Fire Bottom] и [New Game].

3. Сега трябва да подготвите плодовете за рисуване в изгледа.

Добавете кода от картинката в контролера **Controllers/HomeController.cs**:



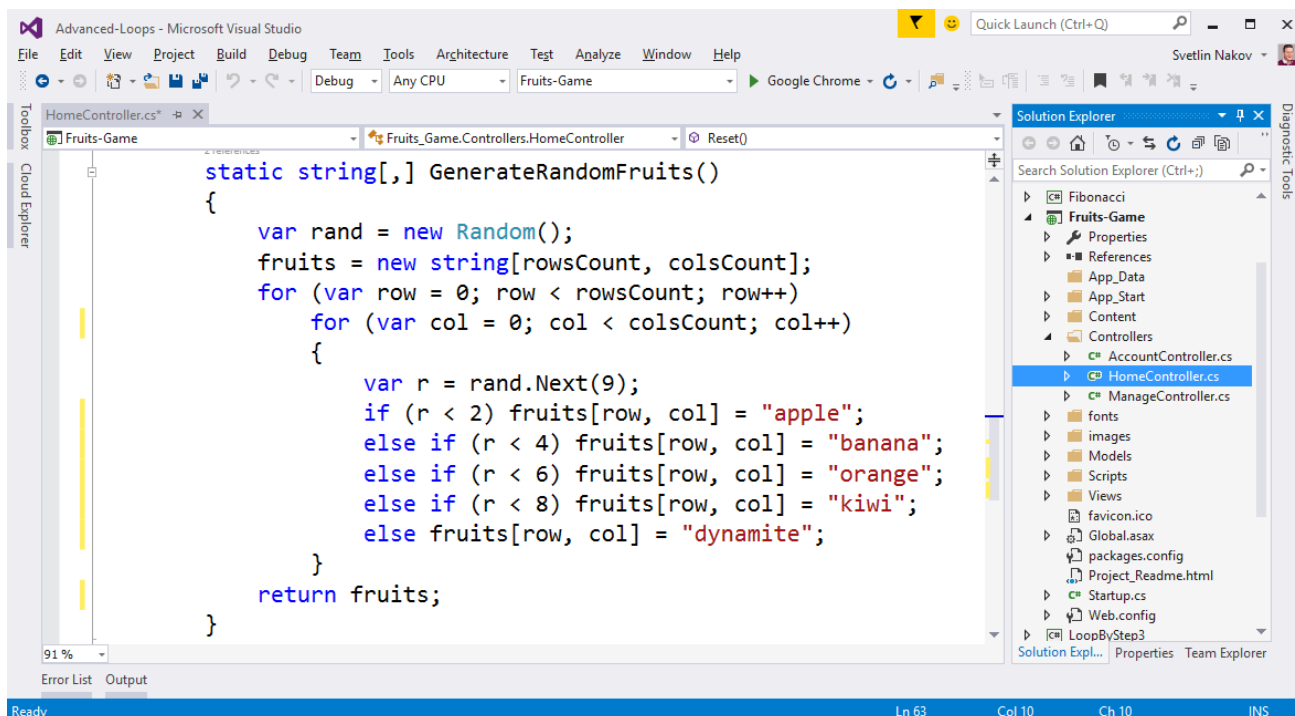
Горният код дефинира полета за **брой редове**, **брой колони**, за **таблицата с плодове** (игралното поле), за натрупаните от играча **точки** и информация дали играта е активна или е **свършила** (поле **gameOver**). Игралното поле е с размери 9 колони на 3 реда и съдържа за всяко поле текст какво има в него: **apple**, **banana**, **orange**, **kiwi**, **empty** или **dynamite**.

Главното действие **Index()** подготвя игралното поле за чертане като записва във **ViewBag** структурата елементите на играта и извиква изгледа, който ги чертае в страницата на играта в уеб браузъра като HTML.

4. Генерирайте **случайни плодове**.

За да генерирате случайни плодове, трябва да напишете метод **GenerateRandomFruits()** с кода от картинката по-долу. Този код записва в таблицата (матрицата) **fruits** имена на различни картинки и така изгражда игралното поле. Във всяка клетка от таблицата се записва една от следните стойности: **apple**, **banana**, **orange**, **kiwi**, **empty** или **dynamite**. След това, за да се нарисува съответното изображение в изгледа, към текста от таблицата ще се долепи “**.png**” и така ще се получи името на файла с картинката, която да се вмъкне в HTML страницата като част от игралното поле. Попълването на игралното поле (9 колони с по 3 реда) става в изгледа **Index.cshtml** с два вложени **for**-цикъла (за ред и за колона).

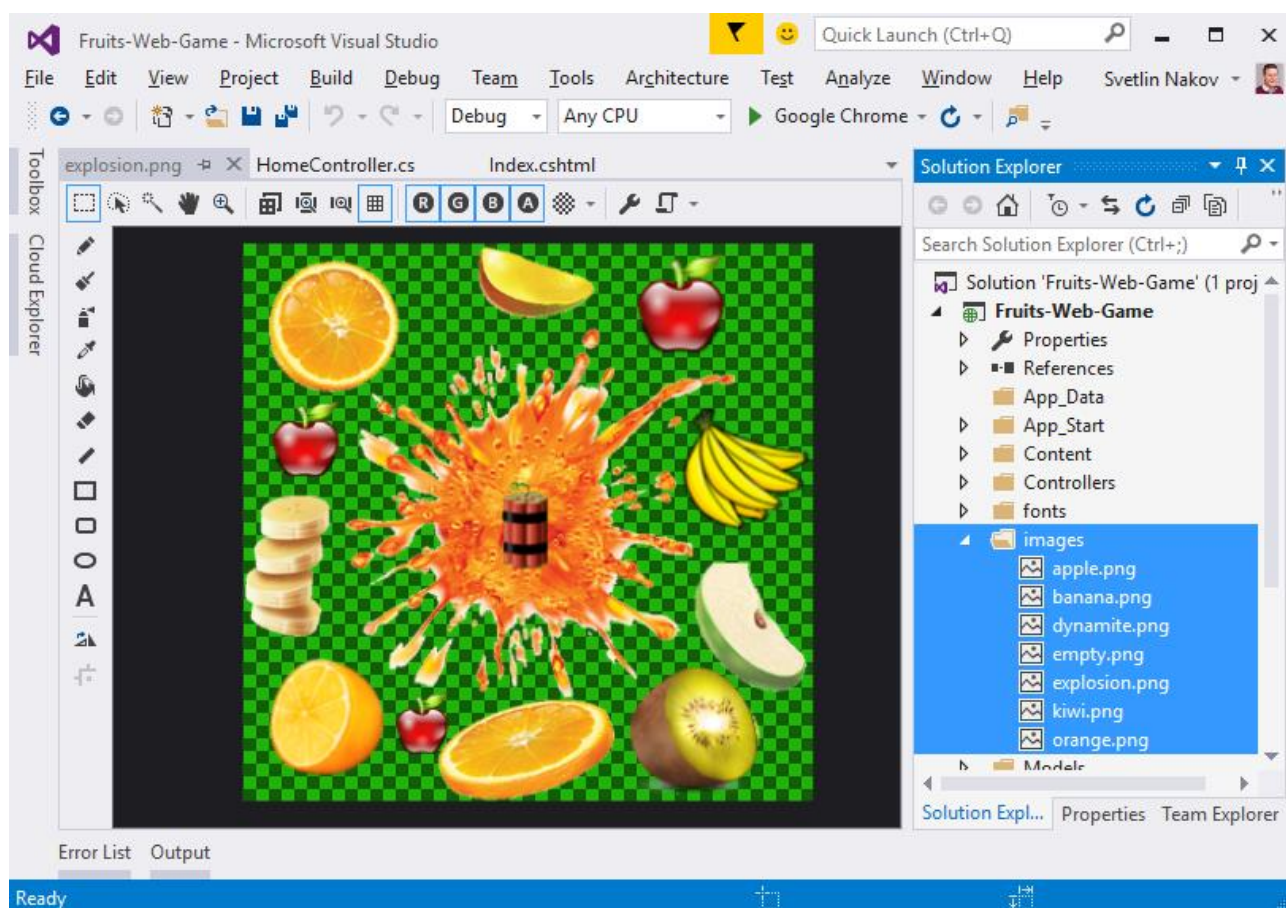
За да се генерират случайни плодове за всяка клетка се генерира **случайно число** между 0 и 8 (вж. класа **Random()** в .NET). Ако числото е 0 или 1, се слага **apple**, ако е между 2 и 3, се слага **banana** и т.н. Ако числото е 8, се поставя **dynamite**. Така плодовете се появяват 2 пъти по-често отколкото динамита. Ето и кода:



5. Добавете картинките за играта.

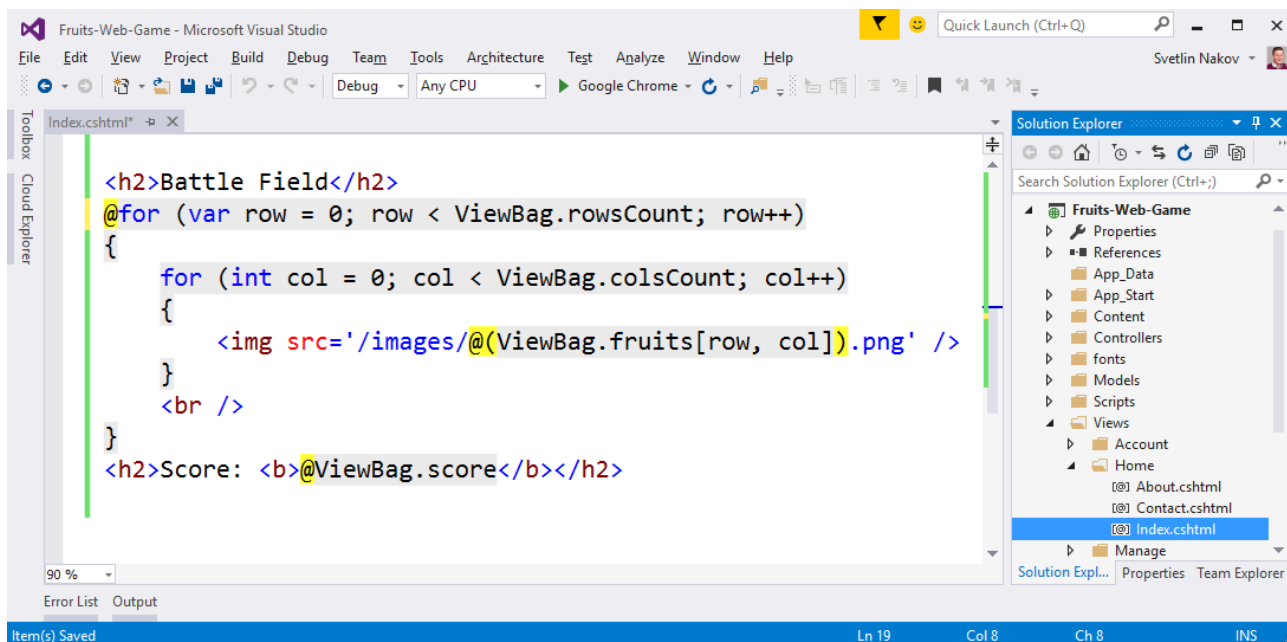
От [Solution Explorer] направете папка **"images"** в коренната директория на проекта. Използвайте менюто [Add] → [New Folder].

Сега добавете **картинките** за играта (те са част от файловете със заданието за домашно). Копирайте ги от Windows Explorer и ги поставете в папката **"images"** в [Solution Explorer] във Visual Studio с **copy / paste**.



6. Чертане на плодовете в Index.cshtml

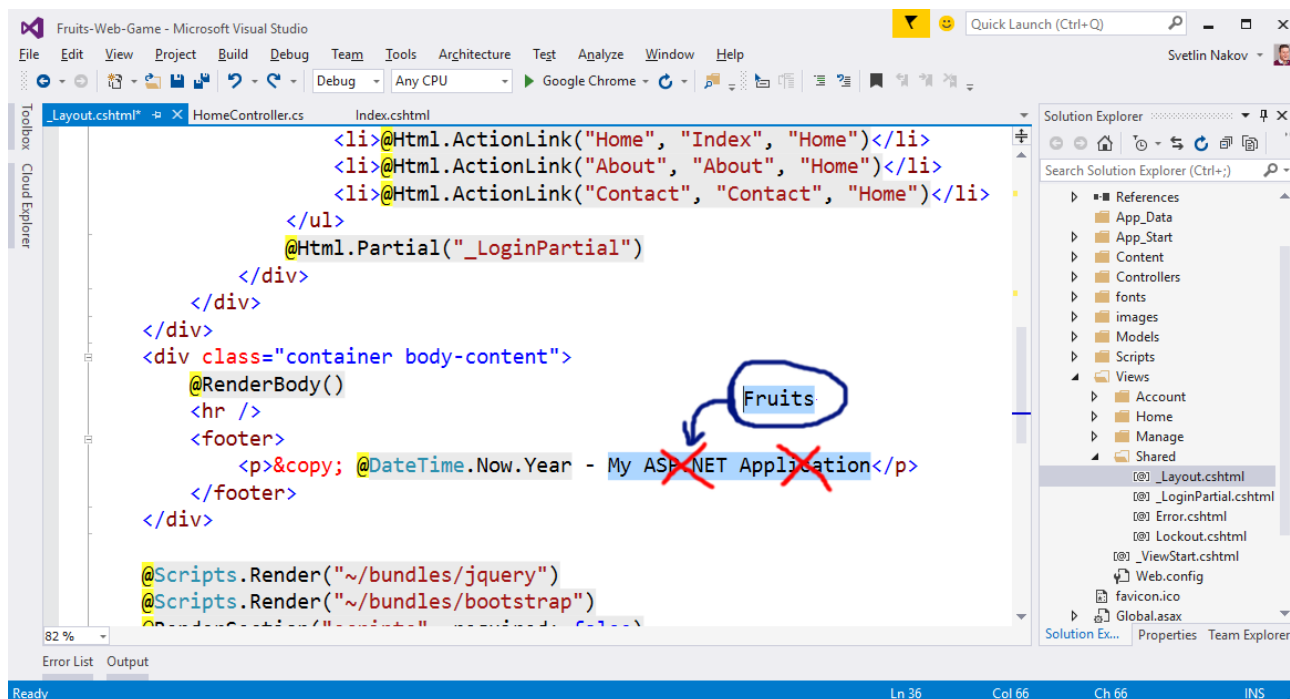
За да **начертаете игралното поле** с подовите, трябва да завъртите **два вложени цикъла** (за редовете и за колоните). Всеки ред се състои от 9 на брой картинки, всяка от които съдържа **apple**, **banana** или друг плод или празно (**empty**) или динамит (**dynamite**). Картинките се чертаят като се отпечата HTML таг за вмъкване на картинка от вида на ``. Девет картинки се подреждат една след друга на всеки от редовете, а след тях се преминава на нов ред с `
`. Това се повтаря три пъти за трите реда. Накрая се отпечатват точките на играча. Ето как изглежда **кодът** за чертане на игралното поле и точките:



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

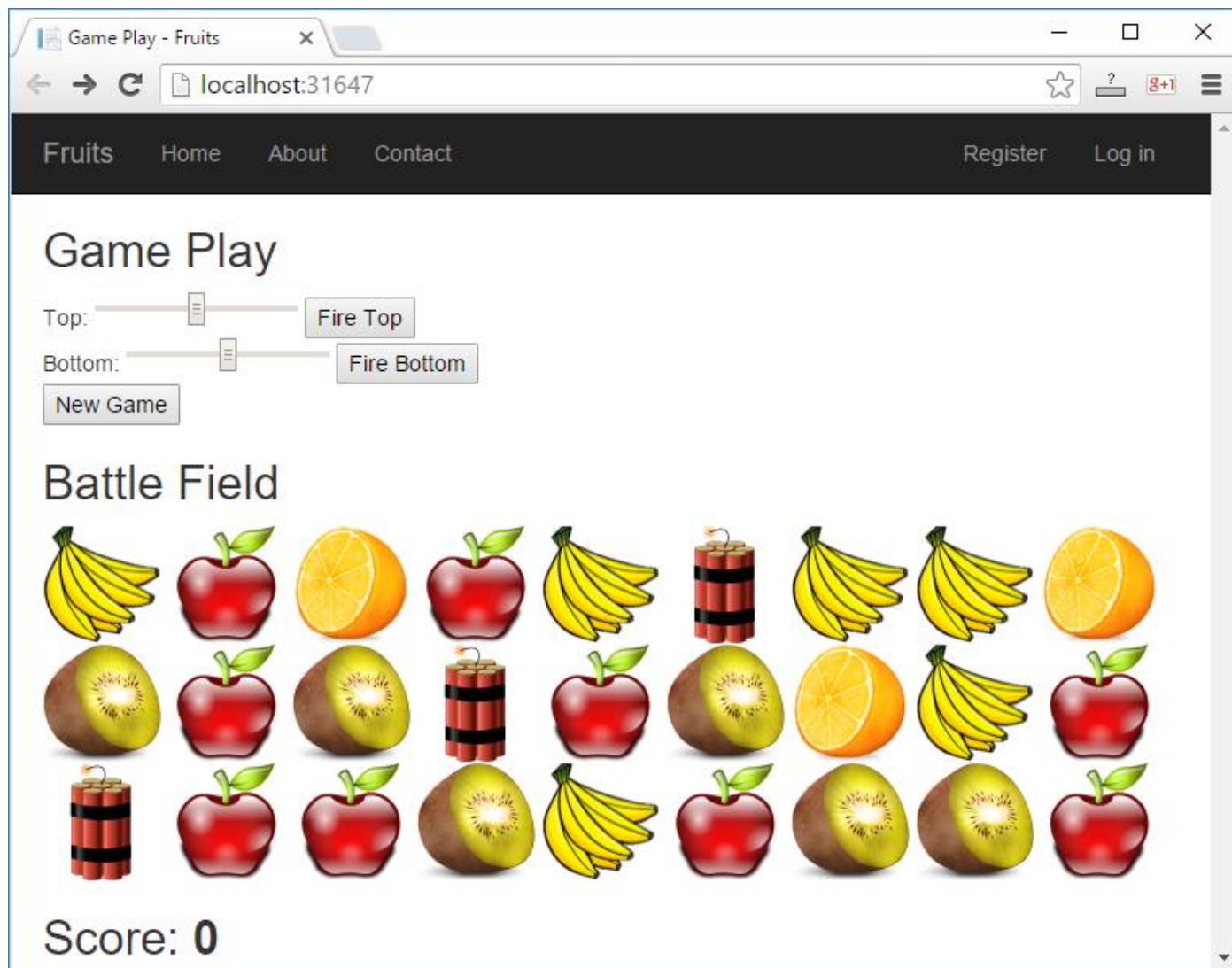
Обърнете внимание на жълтите символи **@** – те служат за превключване между езика **C#** и езика **HTML** и идват от **Razor** синтаксиса за рисуване на динамични уеб страници.

- Нагласете текстовете във файла `/Views/Shared/_Layout.cshtml`. Заменете **“My ASP.NET Application”** с по-подходящи текстове, например **“Fruits”**:



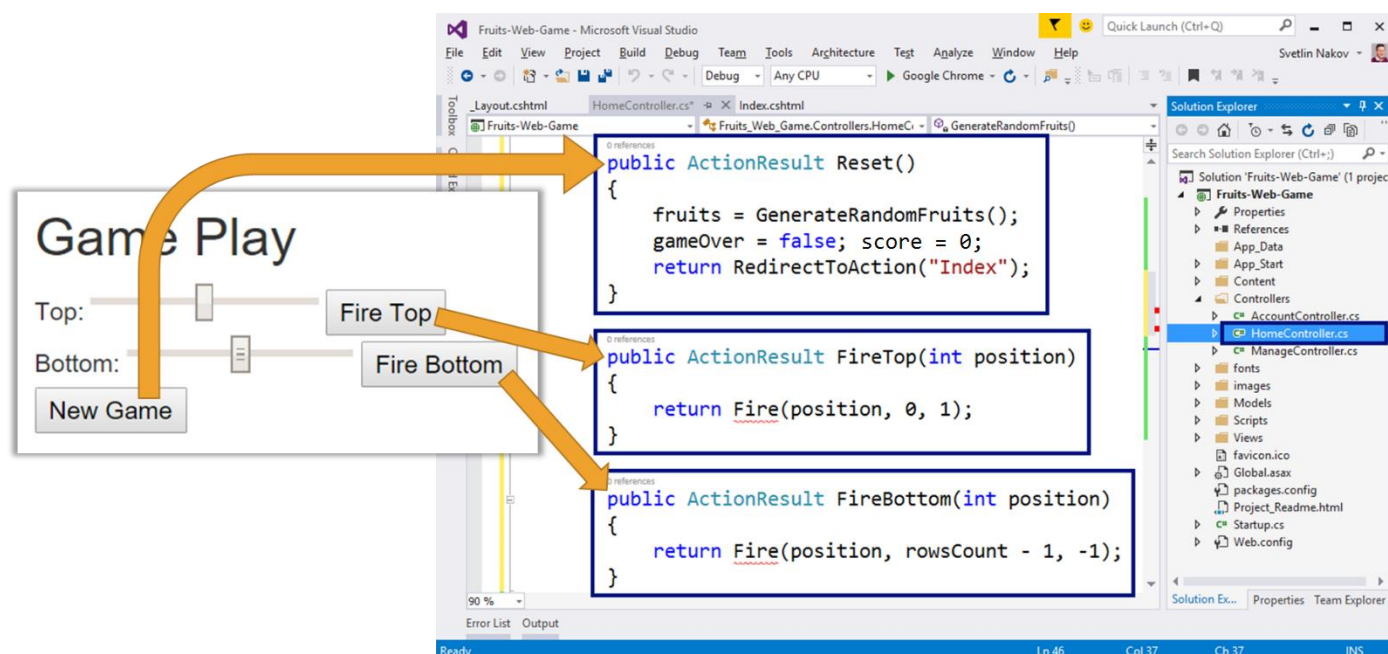
```
<li>@Html.ActionLink("Home", "Index", "Home")</li>
<li>@Html.ActionLink("About", "About", "Home")</li>
<li>@Html.ActionLink("Contact", "Contact", "Home")</li>
</ul>
@Html.Partial("_LoginPartial")
</div>
</div>
<div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
        <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
    </footer>
</div>
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("Scripts", required: false)
```

- Стартирайте проекта с **[Ctrl+F5]** и му се порадвайте. Очаква се да бъде генерирано случайно игрово поле с плодове с размери 9 на 3 и да се визуализира в уеб страницата чрез поредица картинки:



Сега играта е донякъде направена: игралното поле се генерира случайно и се визуализира успешно (ако не сте допуснали грешка някъде). Остава да се реализира същината на играта: **стрелянето по плодовете**.

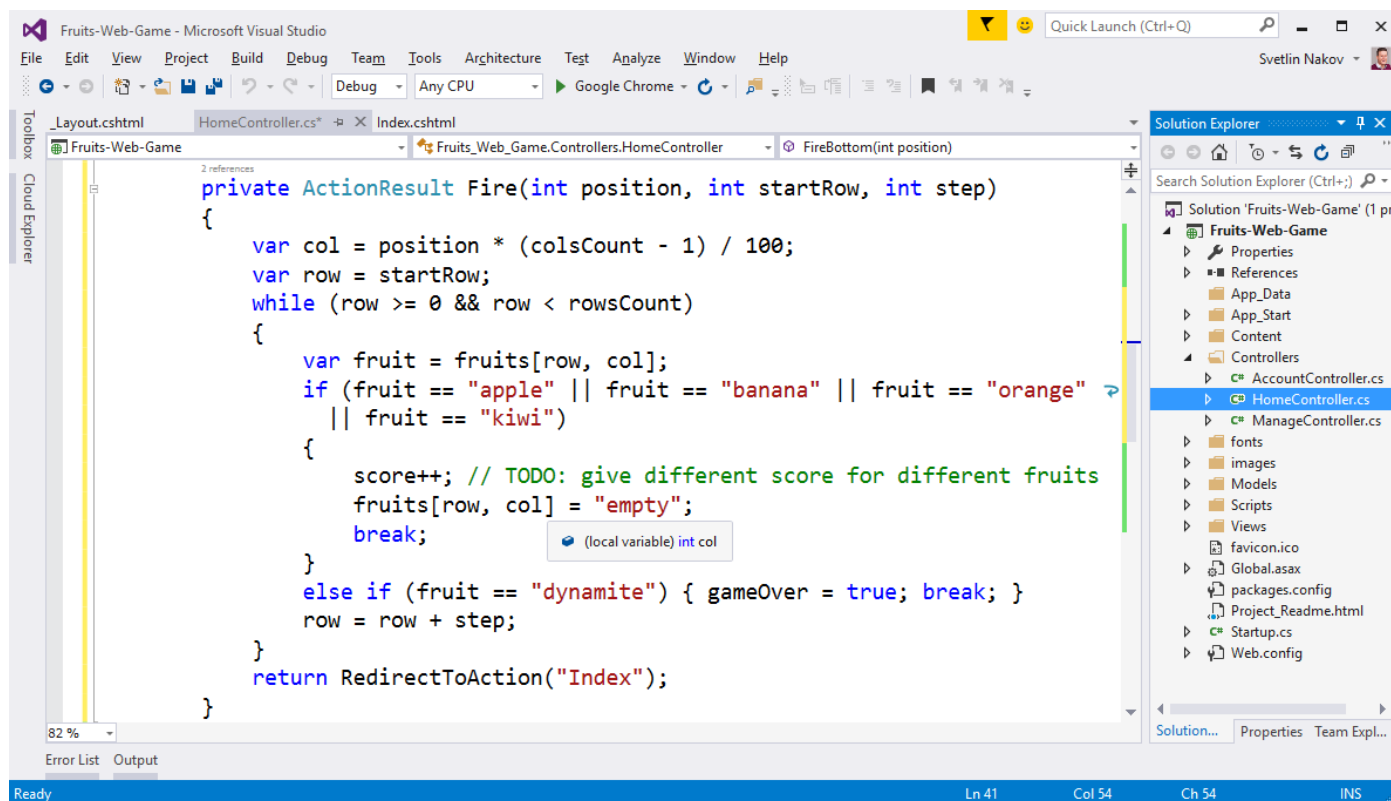
9. Добавете действията [New Game] и [Fire Top] / [Fire Bottom] в контролера "HomeController.cs":



Горният код дефинира три действия:

- **Reset()** – стартира нова игра, като генерира ново случайно игрално поле с плодове и експлозиви, нулира точките на играча и прави играта валидна (**gameOver = false**). Това действие е доста просто и може да се тества веднага с [Ctrl+F5], преди да се напишат другите.
- **FireTop(position)** – стреля по ред **0** на позиция **position** (число от 0 до 100). Извиква се стреляне в посока **надолу** (+1) от ред **0** (най-горния). Самото стреляне е по-сложно като логика и ще бъде разгледано след малко.
- **FireBottom(position)** – стреля по ред **2** на позиция **position** (число от 0 до 100). Извиква се стреляне в посока **нагоре** (-1) от ред **2** (най-долния).

10. Имплементирайте "стрелянето" – метода **Fire(position, startRow, step)**:



```
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");
}
```

Стрелянето работи по следния начин: първо се изчислява номера на колоната **col**, към която играчът се е прицелил. Входното число от скролера (между 0 и 100) се намалява до число между 0 и 8 (за всяка от 9-те колони). Номерът на реда **row** е или 0 (ако изстрелът е отгоре) или броят редове минус едно (ако изстрелът е отдолу). Съответно посоката на стрелба (стъпката) е **1** (надолу) или **-1** (нагоре).

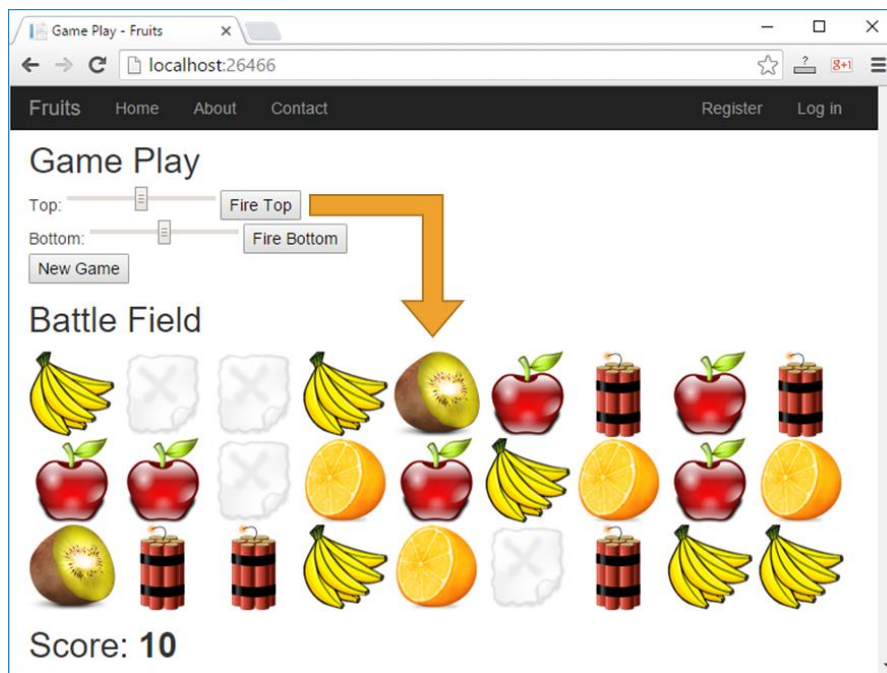
За да се намери къде изстрелът поразява плод или динамит, се преминава в цикъл през всички клетки от игралното в прицелената колона и от първия до последния атакуван ред. Ако се срещне плод, той изчезва (замества се с **empty**) и се дават точки на играча. Ако се срещне **dynamite**, играта се отбелязва като свършила.

Оставаме на по-запалените да имплементират по-сложно поведение, например да се дават различни точки при уцелване на различен плод, да се реализира анимация с експлозия (това не е твърде лесно), да се взимат точки при излишно стреляне в празна колона и подобни.

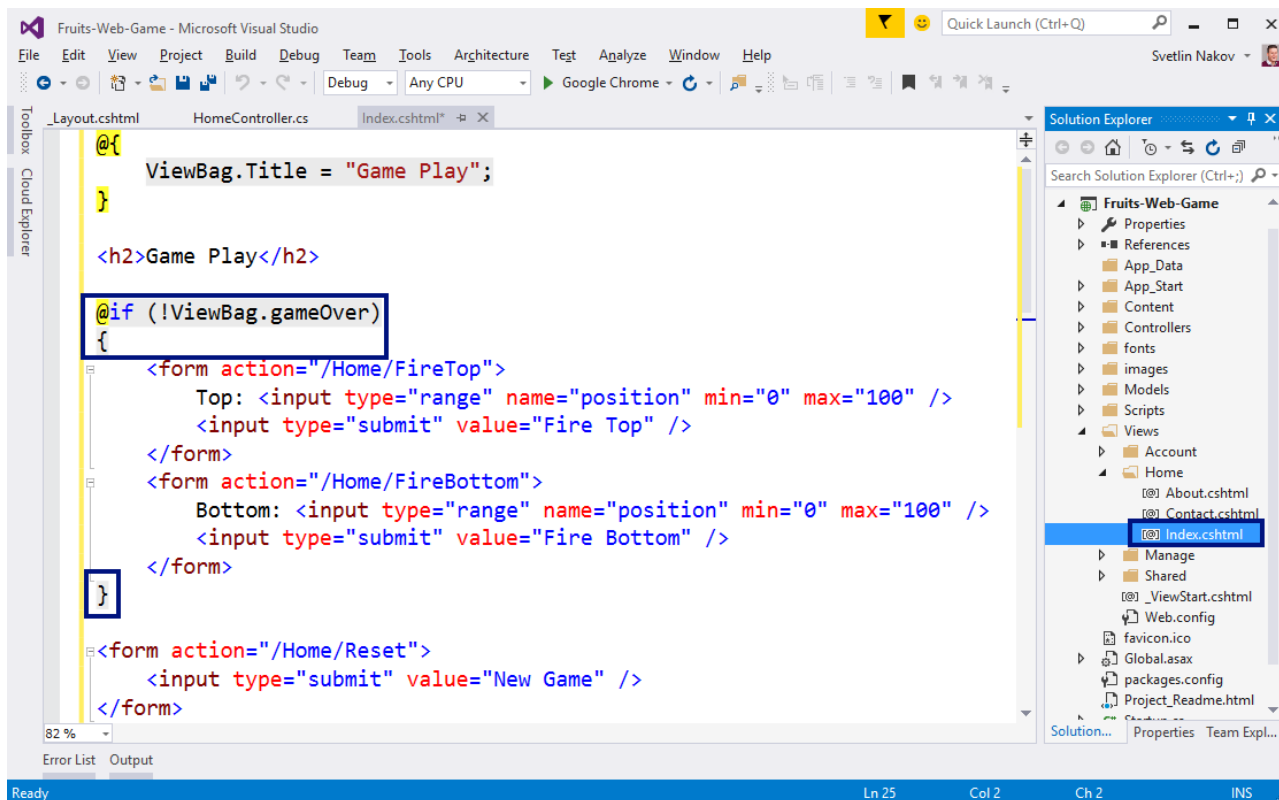
11. Тествайте какво работи до момента като стартирате с [Ctrl+F5]:

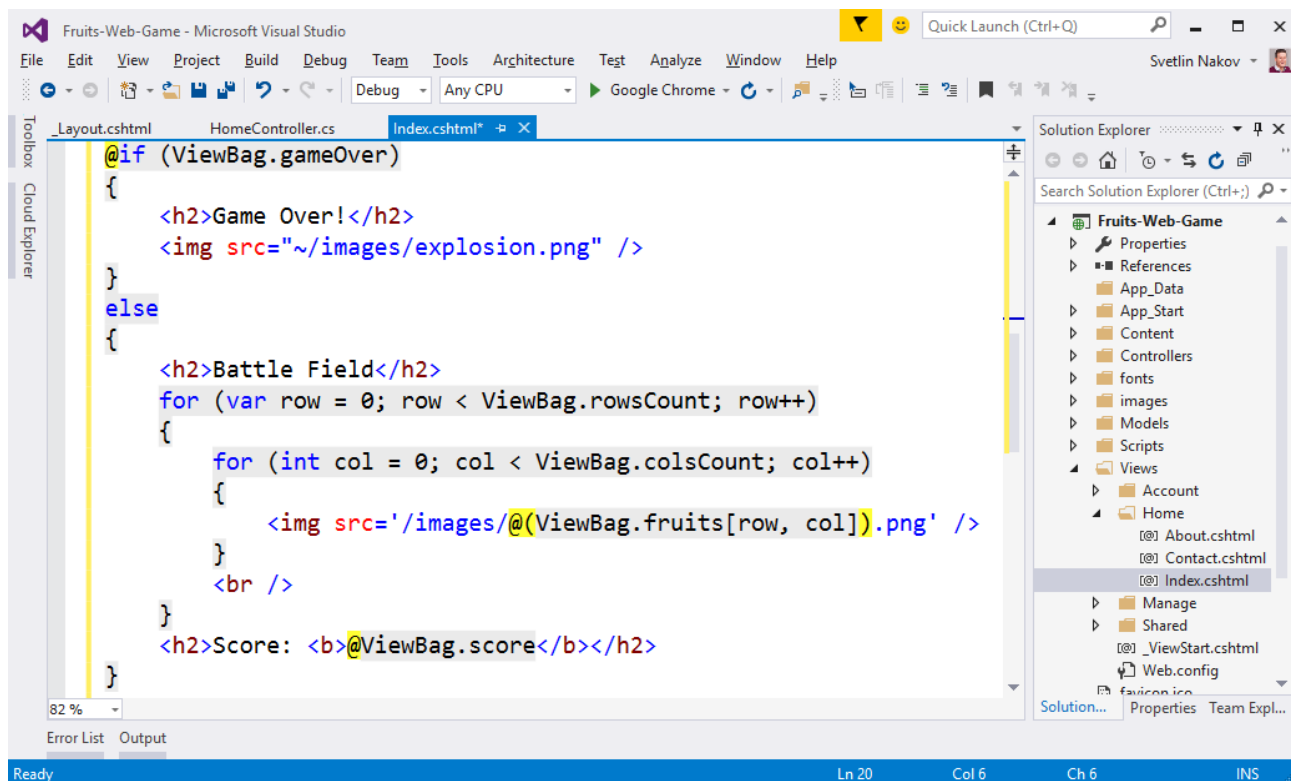
- **Нова игра** → бутонът за нова игра трябва да генерира ново игрално поле със случайно разположени плодове и експлозиви и да нулира точките на играча.

- **Стреляне отгоре** → стрелянето отгоре трябва да премахва най-горният плод в уцелената колона или да предизвиква край на играта при динамит. Всъщност при край на играта все още нищо няма да се случва, защото в изгледа този случай още не се разглежда.
- **Стреляне отдолу** → стрелянето отдолу трябва да премахва най-долния плод в уцелената колона или да прекратява играта при уцелване на динамит.



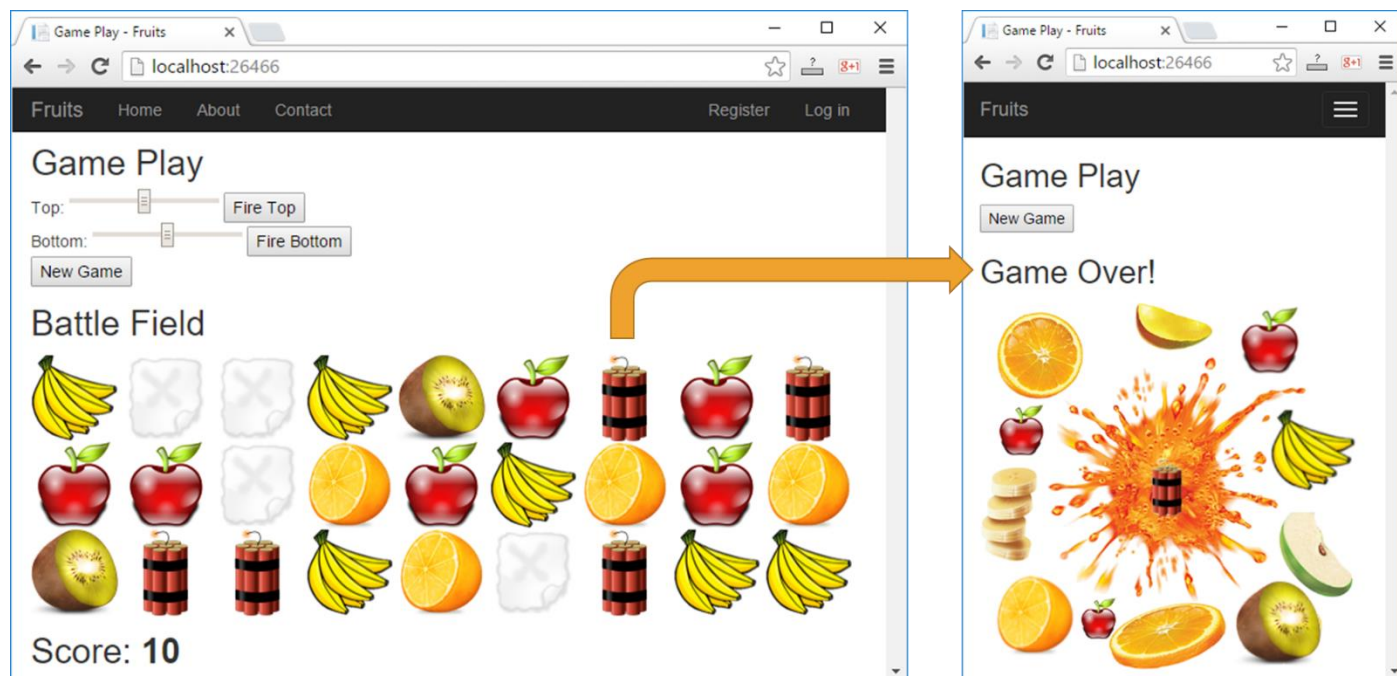
12. Имплементирайте "Край на играта". За момента при край на играта нищо не се случва. Ако играчът уцели динамит, в контролера се отбелязва, че играта е свършила (`gameOver = true`), но този факт не се визуализира по никакъв начин. За да заработи свършването на играта, е необходимо да добавим няколко проверки в изгледа:





Кодът по-горе проверява дали е свършила играта и показва съответно контролите за стреляне и игралното поле (при активна игра) или картинка с експлодирали плодове при край на играта.

13. След промяната в кода на изгледа стартирайте с [Ctrl+F5] и **тествайте** играта отново:



Този път при уцелване на динамит, трябва да се появи дясната картинка и да се позволява единствено действието "нова игра" (бутонът [New Game]).