# Задачи за прости изчисления

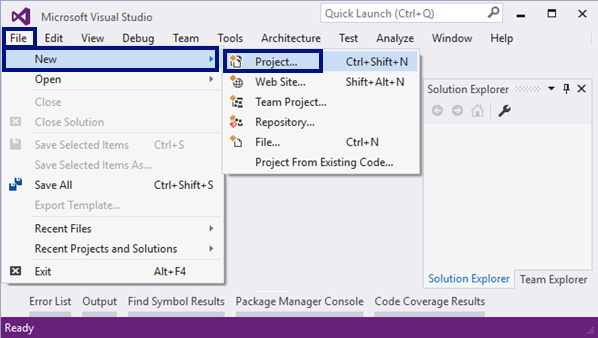
Задачи за упражнения и домашно върху прости изчисления, оператори и изрази.

## Празно Visual Studio решение (Blank Solution)

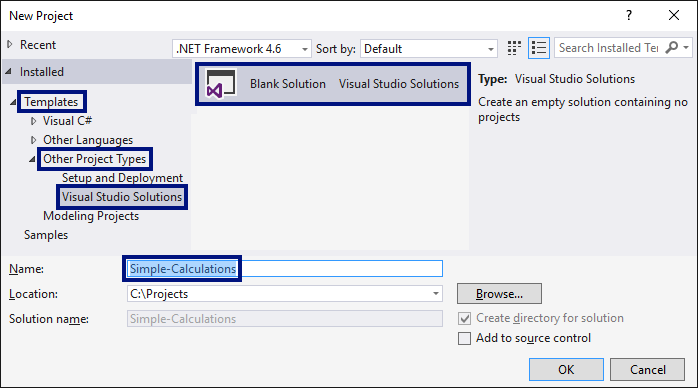
В настоящото практическо занимание ще използваме **Blank Solution с няколко проекта**, за да организираме решенията на задачите от упражненията – всяка задача в отделен проект и всички проекти в общ solution.

Стартираме **Visual Studio**.

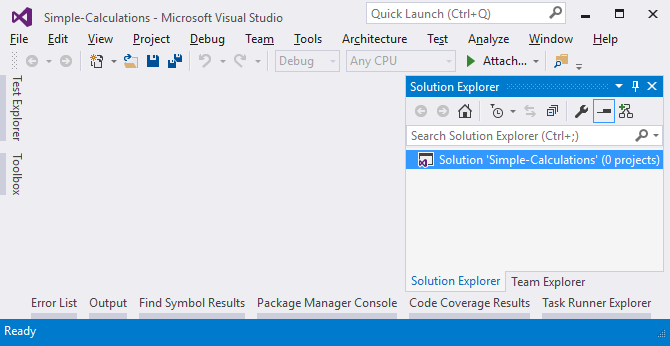
Създаваме нов **Blank Solution:** [**File**] -> [**New**] -> [**Project**]:



Избираме от диалоговия прозорец [**Templates**] -> [**Other Project Types**] -> [**Visual Studio Solutions**] -> [**Blank Solution**] и даваме подходящо име на проекта, например “Simple-Calculations”:



Сега имаме създаден **празен Visual Studio Solution** (с 0 проекта в него):



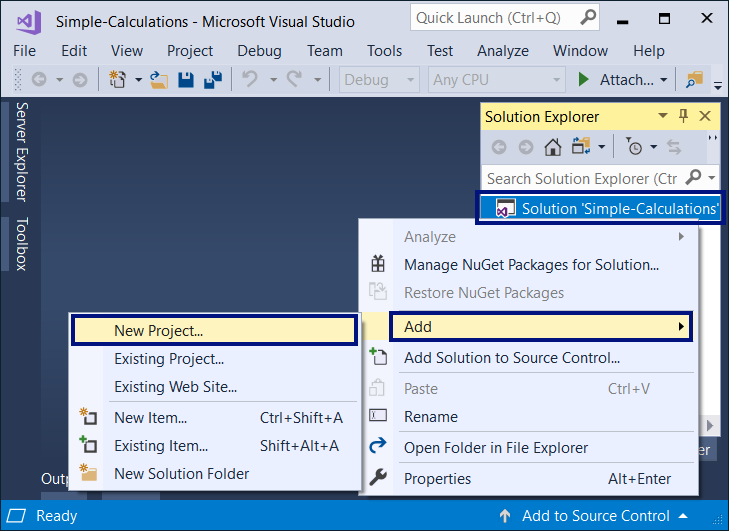
Целта на този **blank solution** e да добавяме в него **по един проект за всяка задача** от упражненията.

## Пресмятане на лице на квадрат

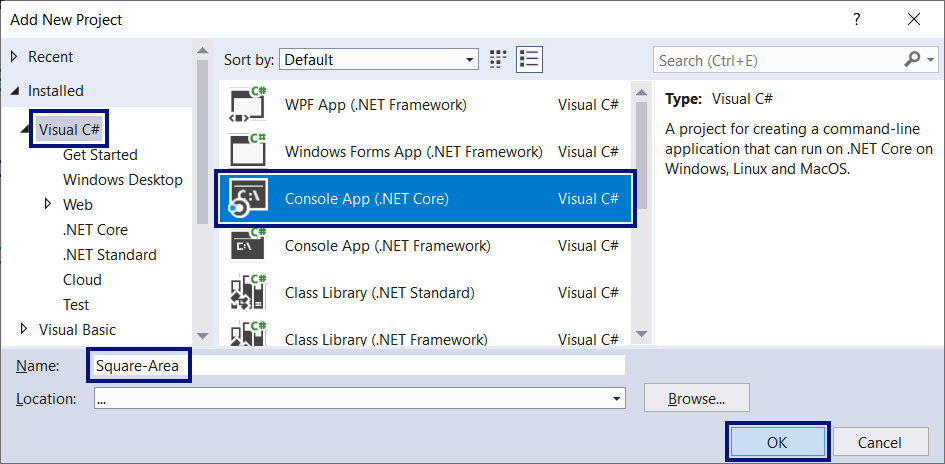
Първата задача от тази тема е следната: да се напише конзолна програма, която **въвежда цяло число a и пресмята лицето** на квадрат със страна **a**. Задачата е тривиално лесна: **въвеждате число** от конзолата, **умножавате го само по себе си** и **печатате получения резултат** на конзолата.

### Упътване

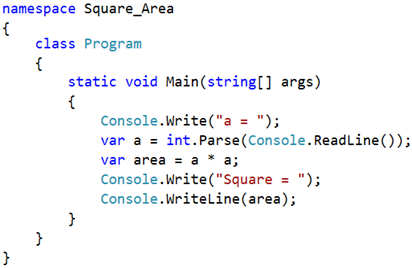
Създаваме **нов проект** в съществуващото Visual Studio решение. В **Solution Explorer** кликнете с десен бутон на мишката върху **Solution 'Simple-Calculations'**. Изберете [**Add**] -> [**New Project…**]:



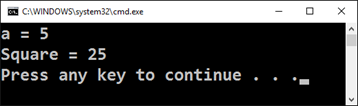
Ще се отвори **диалогов прозорец** за избор на **тип проект** за създаване. Избираме **C# конзолно приложение** с име “Square-Area”:



Вече имаме solution с едно конзолно приложение в него. Остава да напишем **кода за решаване на задачата**. За целта отиваме в тялото на метода **Main(string[] args)** и пишем следния код:

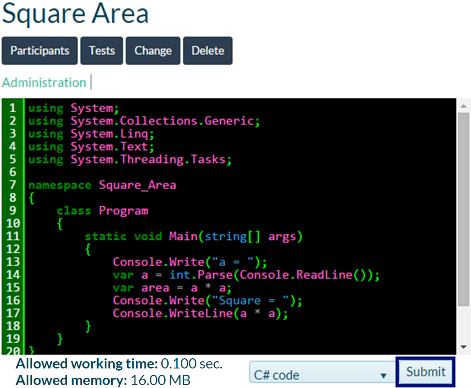


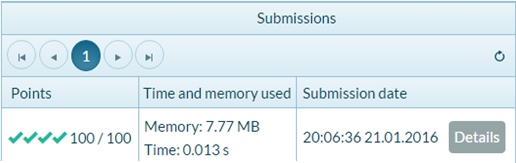
Кодът въвежда цяло число чрез a = int.Parse(Console.ReadLine()), след това изчислява **area = a \* a** и накрая печата стойността на променливата area. **Стартираме** програмата с [**Ctrl+F5**] и я **тестваме** с различни входни стойности:



### Тестване в Judge системата

Тествайте решението си тук: <https://judge.softuni.bg/Contests/Practice/Index/504#0>. Трябва да получите 100 точки (напълно коректно решение):



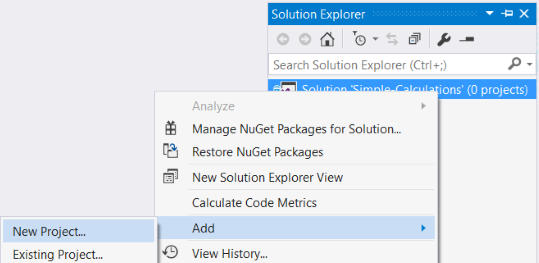


## Задача: от инчове към сантиметри

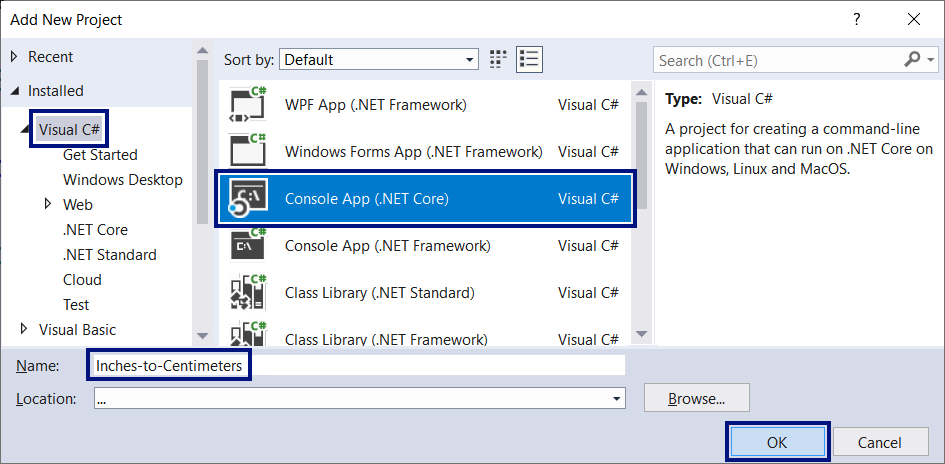
Да се напише програма, която **чете от конзолата число** (не непременно цяло) и преобразува числото от **инчове в сантиметри.** За целта **умножава инчовете по 2.54** (защото 1 инч = 2.54 сантиметра).

### Упътване

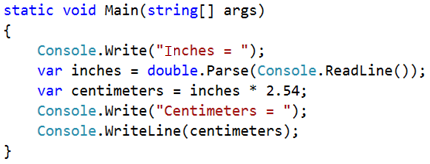
Първо създаваме **нов C# конзолен проект** в решението “Simple-Calculations”. Кликаме с мишката върху решението в **Solution Explorer** и избираме [**Add**] -> [**New Project…**]:



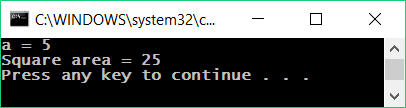
Избираме [**Visual C#**] -> [**Windows**] -> [**Console Application**] и задаваме име “Inches-to-Centimeters”:



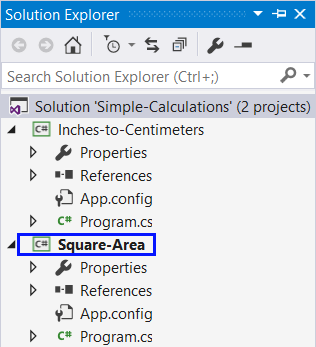
Следва да напишем **кода на програмата**:



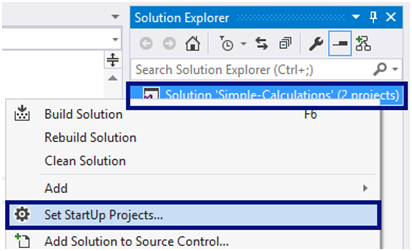
**Стартираме програмата** с [**Ctrl+F5**]:



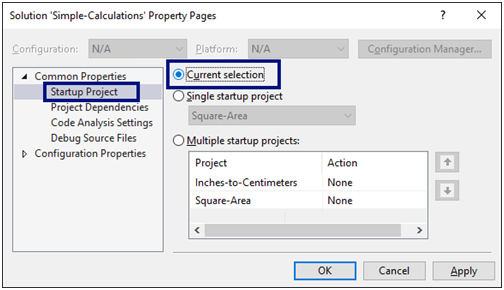
Изненада! Какво става? Програмата не работи правилно… Всъщност това не е ли предходната програма? Във Visual Studio **текущият активен проект** в един solution е маркиран в получерно и може да се сменя:



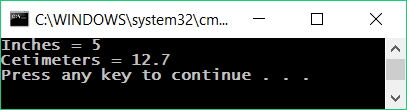
За да включим режим на **автоматично преминаване към текущия проект**, кликаме върху главния solution с десния бутон на мишката и избираме [**Set StartUp Projects…**]:



Ще се появи диалогов прозорец, от който трябва да се избере [**Startup Project**] -> [**Current Selection**]:



Сега отново **стартираме програмата**, както обикновено с [**Ctrl+F5**]. Този път ще се стартира **текущата отворена програма**, която преобразува инчове в сантиметри. Изглежда работи коректно:



## Преобразуване на масив в число

Напишете програма, която въвежда масив от цели числа и го преобразува чрез сумиране на съседни двойки елементи, докато се получи едно цяло число. Например, ако имаме 3 елемента {2,10,3}, то събираме първите два и вторите два елемента и получаваме {2+10, 10+3} = {12, 13}, после събираме всички съседни елементи и получаваме {12+13} = {25}.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 2 10 3 | 25 | 2 10 3 🡪 2+10 10+3 🡪 12 13 🡪 12 + 13 🡪 25 |
| 5 0 4 1 2 | 35 | 5 0 4 1 2 🡪 5+0 0+4 4+1 1+2 🡪 5 4 5 3 🡪 5+4 4+5 5+3 🡪 9 9 8 🡪 9+9 9+8 🡪 18 17 🡪 18+17 🡪 35 |
| 1 | 1 | Масивът {1} е вече свит в число |

### Упътване

Докато имаме повече от един елемент в масива nums[], повтаряй следното:

* Създай нов масив condensed[] с размер nums.Length-1.
* Събирай числата от nums[] в condensed[]:
  + condensed[i] = nums[i] + nums[i+1]
* nums[] = condensed[]

Процесът е илюстриран по-долу:





## Обръщане на масив от символни низове

Напишете програма, която да прочете масив от симовлни низове, обръща масива и печата на неговите елементи. Входът се състои от поредица от низове, разделени с интервал. Отпечатва резултата на един ред с разделител интервал.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| a b c d e | e d c b a |
| -1 hi ho w | w ho hi -1 |

### Упътване

* Въведете масив от символни низове
* Разменете първият елемент (с индекс 0) с последния елемент (с индекс n--1)
* Продължете с тези размени с останалите елементи докато стигнете средата на масива



* Друг, по-кратък подход е да се ползва готовия разширяващ метод .Reverse() от “System.Linq”.