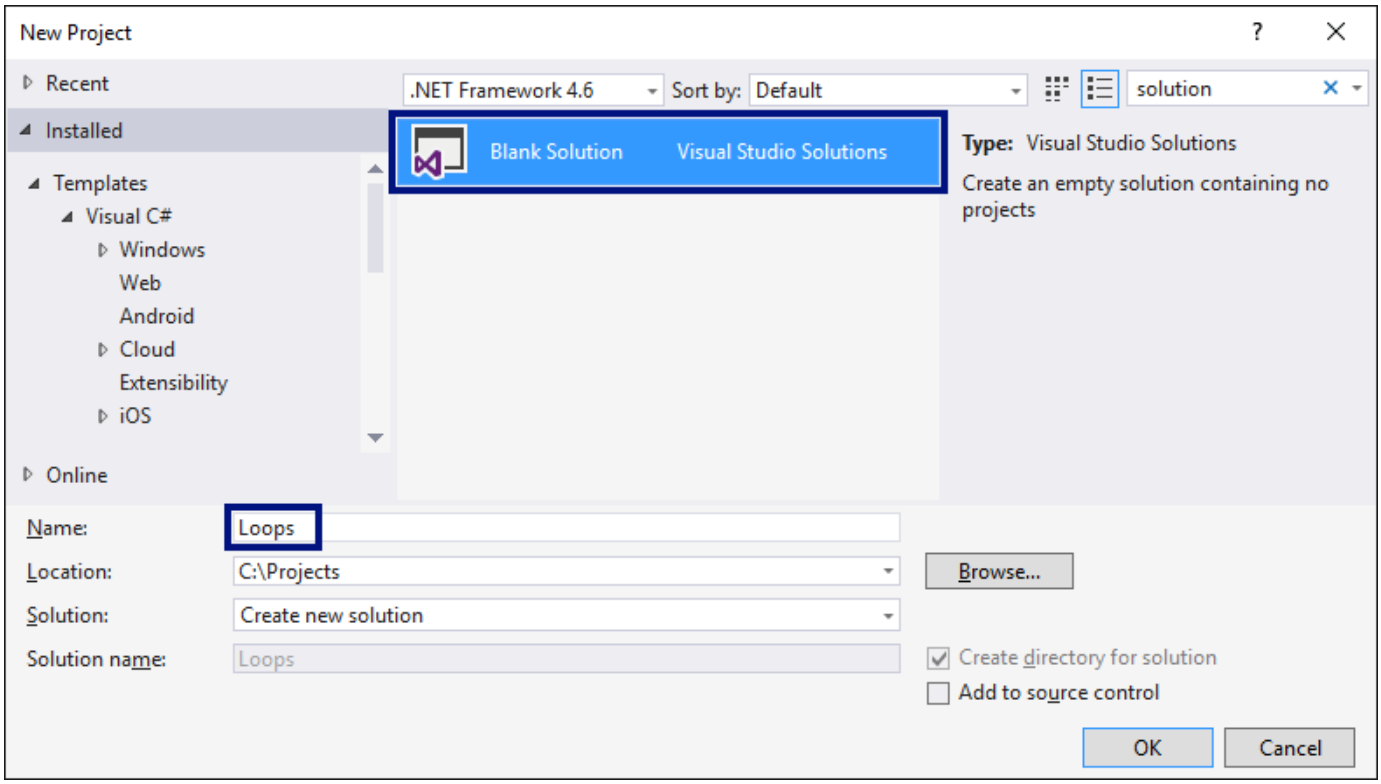
Упражнения: Повторения (цикли)

Практически упражнения към курса "Увод в програмирането" за ученици.
Тествайте решенията си от тази тема в Judge: https://judge.softuni.bg/Contests/2638/Повторения

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 'Loops'** → **[Set StartUp Projects…]** → **[Current selection]**.

1. Числа от 1 до 100

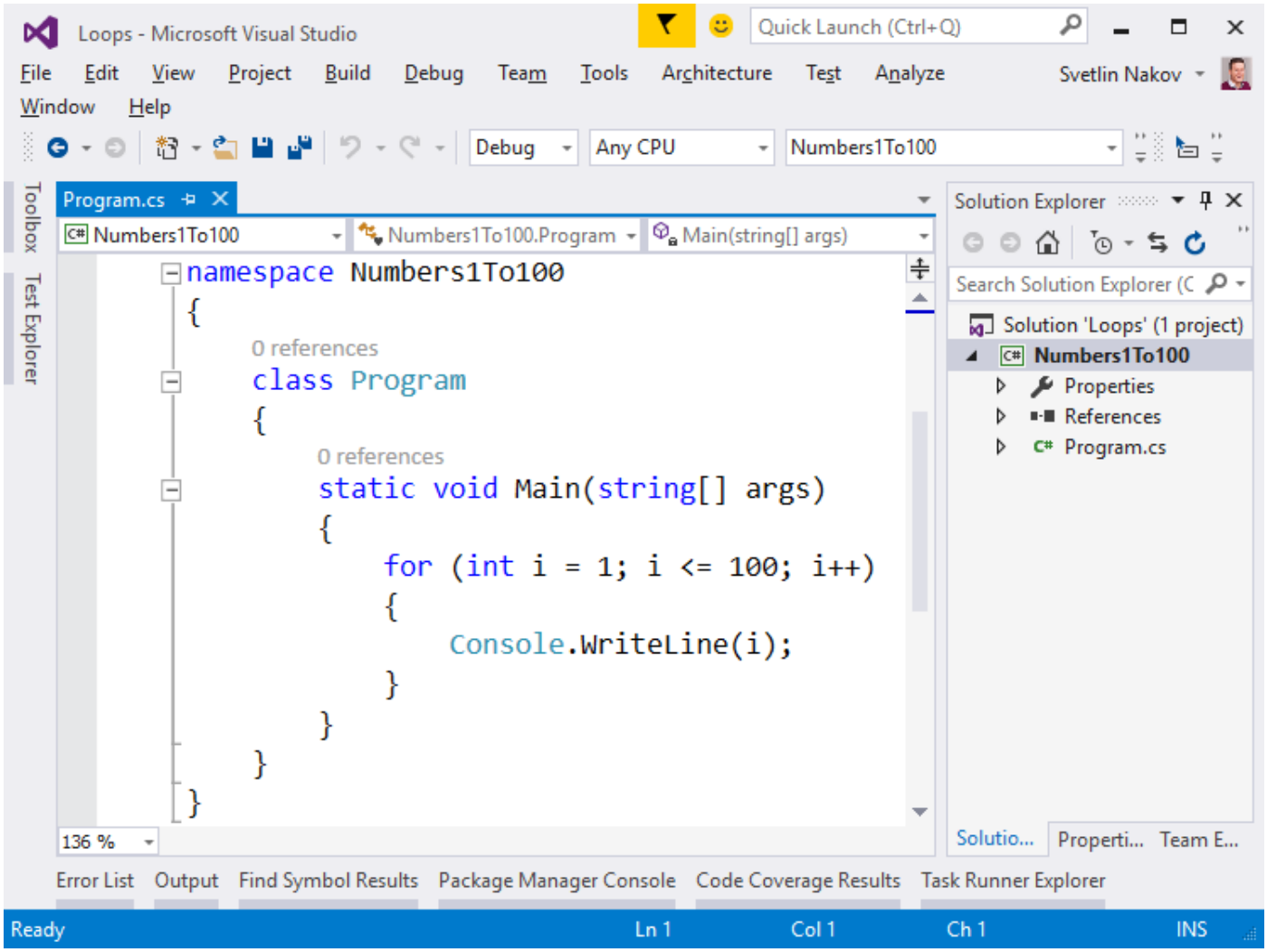
Напишете програма, която отпечатва числата от 1 до 100, по едно на ред.

вход	изход
(няма)	1 2 3 … 98 99 100

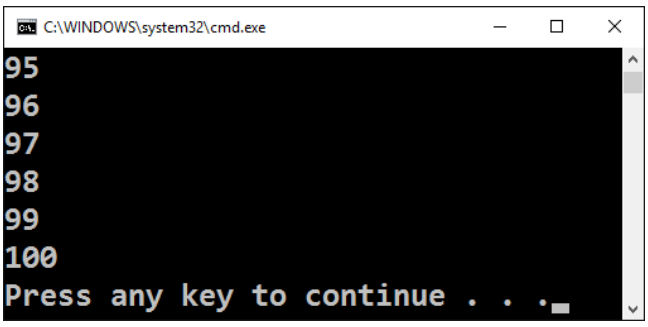
Подсказки:

1. Създайте **нов проект** в съществуващото Visual Studio решение – конзолна C# програма. Задайте подходящо име на проекта, например "**`Numbers1To100`**".

2. Отидете в тялото на метода **`Main(string[] args)`** и напишете решението на задачата. Можете да си помогнете с кода от картинката по-долу:



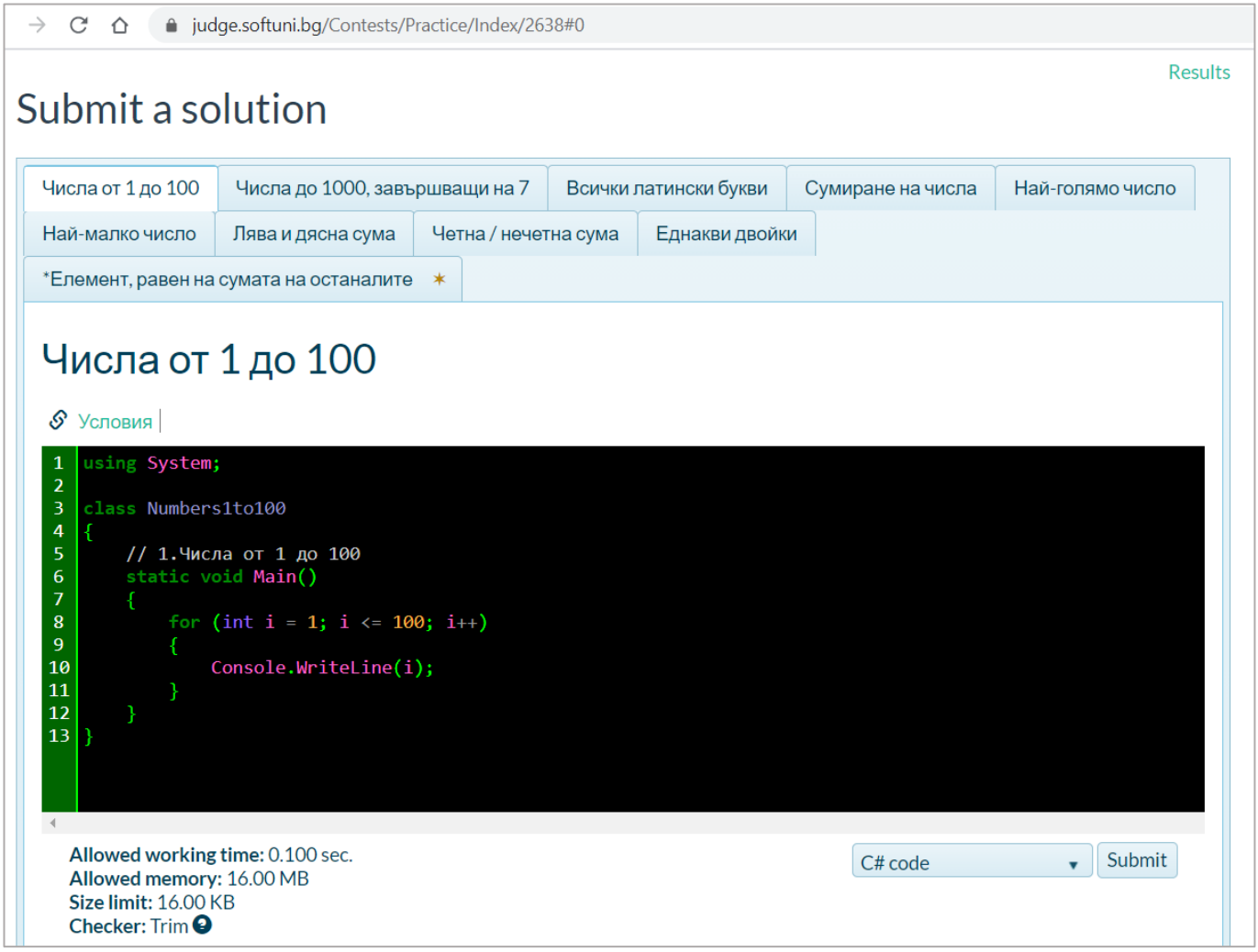
3. **Стартирайте** програмата с [Ctrl+F5] и я **тествайте**:



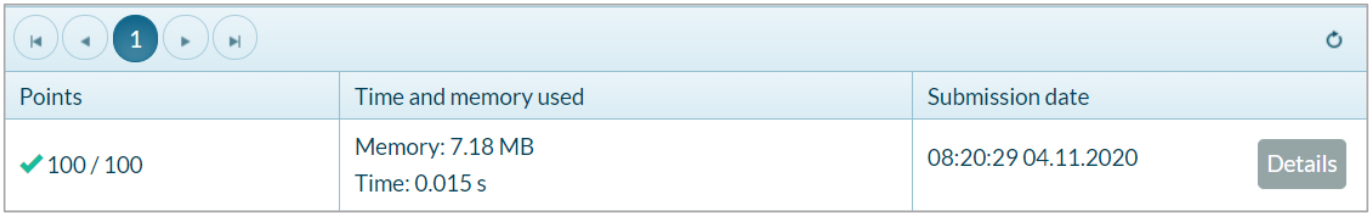
4. **Проверете** решението си в **judge системата**.

Отворете страницата в judge за този урок: https://judge.softuni.bg/Contests/2638/Повторения.

Изберете задачата “ **Числа от 1 до 100**”. Копирайте и поставете в тъмното поле **сорс кода**. Натиснете бутона за изпращане **[Submit]**:



Трябва да получите **100 точки** (напълно вярна задача):



2. Числа до 1000, завършващи на 7

Напишете програма, която отпечатва числата в диапазона **[1…1000]**, които **завършват на 7**.

вход	изход
(няма)	7 17 27 … 997

Подсказка: можете да завъртите **`for`**-цикъл от 1 до 1000 и да проверите всяко число дали завършва на 7. Едно число **`num`** завършва на 7, когато **`(num % 10 == 7)`**.

3. Всички латински букви

Напишете програма, която отпечатва всички букви от латинската азбука: **a**, **b**, **c**, …, **z**.

Подсказка: можете да завъртите **`for`**-цикъл от **`'a'`** до **`'z'`** (освен числа може да въртите в цикъл и букви).

4. Сумиране на числа

Да се напише програма, която **чете n-на брой цели числа**, въведени от потребителя, **и ги сумира**.

- От първия ред на входа се въвежда броят числа **n**.
- От следващите **n** реда се въвежда по едно цяло число.

Програмата трябва да прочете числата, да ги сумира и да отпечата сумата им. Примери:

вход	изход
2 10 20	30

вход	изход
3 -10 -20 -30	-60

вход	изход
4 45 -20 7 11	43

вход	изход
1 999	999

вход	изход
0	0

Подсказки:

- Първо въведете едно число **n** (броят числа, които предстои да бъдат въведени).
- Инициализирайте **`sum = 0`** (в началото няма още прочетени числа, и съответно сумата е празна).
- В цикъл **n пъти** прочетете по едно цяло число **`num`** и го прибавете към сумата (**`sum = sum + num`**).
- Накрая в **`sum`** трябва да се е запазила сумата на прочетените числа. Отпечатайте я.

5. Най-голямо число

Напишете програма, която чете **n-на брой цели числа** (**n** > 0), въведени от потребителя, и намира **най-голямото** измежду тях. Първо се въвежда броят числа **n**, а след това самите **n** числа, по едно на ред. Примери:

вход	изход
2 100 99	100

вход	изход
3 -10 20 -30	20

вход	изход
4 45 -20 7 99	99

вход	изход
1 999	999

вход	изход
2 -1 -2	-1

Подсказки:

- Първо въведете едно число **n** (броят числа, които предстои да бъдат въведени).
- Въведете от конзолата първото число. Сложете текущият максимум **`max`** да е прочетеното число.
- В цикъл **n-1 пъти** прочетете по едно цяло число **`num`**. Ако прочетеното число **`num`** е по-голямо от текущият максимум **`max`**, запомнете **`num`** в **`max`**.
- Накрая в **`max`** трябва да се е запазило най-голямото число. Отпечатайте го.

6. Най-малко число

Напишете програма, която чете **n-на брой цели числа** (**n** > 0), въведени от потребителя, и намира **най-малкото** измежду тях. Първо се въвежда броят числа **n**, а след това самите **n** числа, по едно на ред. Примери:

вход	изход
2	99

вход	изход
3	-30

вход	изход
4	-20

вход	изход
1	999

вход	изход
2	-2

100 99		-10 20 -30		45 -20 7 99		999		-1 -2	

Подсказки: задачата е абсолютно аналогична с предходната.

7. Лява и дясна сума

Да се напише програма, която чете **2*n-на брой** цели числа, подадени от потребителя, и проверява дали **сумата на първите n числа** (лява сума) е равна на **сумата на вторите n числа** (дясна сума). При равенство печата "**Yes**" + **сумата**; иначе печата "**No**" + **разликата**. Разликата се изчислява като положително число (по абсолютна стойност). Примери:

вход	изход	коментар
2 10 90 60 40	Yes, sum = 100	10+90 = 60+40 = 100

вход	изход	коментар
2 90 9 50 50	No, diff = 1	90+9 ≠ 50+50 Difference = |99-100| = 1

Подсказки:

- Въведете **n**.
- Въведете първите **n** числа (**лявата** половина) и ги сумирайте.
- Въведете още **n** числа (**дясната** половина) и ги сумирайте.
- Изчислете **разликата** между сумите по абсолютна стойност: **`Math.Abs(leftSum - rightSum)`**.
- Ако разликата е **0**, отпечатайте "**Yes**" + **сумата**; иначе отпечатайте "**No**" + **разликата**.

8. Четна / нечетна сума

Да се напише програма, която чете **n-на брой** цели числа, подадени от потребителя, и проверява дали **сумата от числата на четни позиции** е равна на **сумата на числата на нечетни позиции**. При равенство да се отпечата "**Yes**" + **сумата**; иначе да се отпечата "**No**" + **разликата**. Разликата се изчислява по абсолютна стойност. Примери:

вход	изход	коментар
4 10 50 60 20	Yes Sum = 70	10+60 = 50+20 = 70

вход	изход	коментар
4 3 5 1 -2	No Diff = 1	3+1 ≠ 5-2 Diff = |4-3| = 1

вход	изход	коментар
3 5 8 1	No Diff = 2	5+1 ≠ 8 Diff = |6-8| = 2

Подсказки: Въведете числата едно по едно и изчислете двете **суми** (числа на **четни** позиции и числа на **нечетни** позиции). Както в предходната задача, изчислете абсолютна стойност на разликата и отпечатайте резултата ("**Yes**" + **сумата** при разлика 0 или "**No**" + **разликата** в противен случай).

9. Еднакви двойки

Дадени са 2*n-на брой числа. Първото и второто формират **двойка**, третото и четвъртото също и т.н. Всяка двойка има **стойност** – сумата от съставящите я числа. Напишете програма, която проверява **дали всички двойки имат еднаква стойност** или печата **максималната разлика** между две последователни

двойки. Ако всички двойки имат еднаква стойност, отпечатайте "**Yes, value={Value}**" + **стойността**. В противен случай отпечатайте "**No, maxdiff={Difference}**" + **максималната разлика**. Примери:

вход	изход	коментари
3 1 2 0 3 4 -1	Yes, value=3	стойности = {3, 3, 3} еднакви стойности
4 1 1 3 1 2 2 0 0	No, maxdiff=4	стойности = {2, 4, 4, 0} разлики = {2, 0, 4} макс. разлика = 4
2 -1 0 0 -1	Yes, value=-1	стойности = {-1, -1} еднакви стойности

вход	изход	коментари
2 1 2 2 2	No, maxdiff=1	стойности = {3, 4} разлики = {1} макс. разлика = 1
1 5 5	Yes, value=10	стойности = {10} една стойност еднакви стойности
2 -1 2 0 -1	No, maxdiff=2	стойности = {1, -1} разлики = {2} макс. разлика = 2

Подсказки:

- Прочитайте входните числа **по двойки**. За всяка двойка пресмятайте **сумата**.
- Докато четете входните двойки, за всяка двойка без първата пресмятайте **разликата с предходната**. За целта пазете в отделна променлива сумата на предходната двойка.
- Намерете **най-голямата разлика** между две двойки. Ако е **0**, печатайте "**Yes**" иначе "**No**" + разликата.

10. * Елемент, равен на сумата на останалите

Да се напише програма, която чете **n-на брой** цели числа, въведени от потребителя, и проверява дали сред тях съществува число, което е равно на сумата на всички останали. Ако има такъв елемент, печата "**Yes**", "**Sum = **" + **неговата стойност**; иначе печата "**No**", "**Diff = **" + **разликата между най-големия елемент и сумата на останалите** (по абсолютна стойност).

Примери:

вход	изход	коментари
7 3 4 1	Yes Sum = 12	3 + 4 + 1 + 2 + 1 + 1 = 12

1 2 **12** 1		
4 **6** 1 2 3	Yes Sum = 6	1 + 2 + 3 = 12
3 1 1 10	No Diff = 8	|10 - (1 + 1)| = 8
3 5 5 1	No Diff = 1	|5 - (5 + 1)| = 1
3 1 1 1	No Diff = 1	

Подсказка: изчислете **сумата** на всички елементи и **най-големият** от тях и проверете търсеното условие.

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".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).