



Въведение във MS Visual Studio

Пано Панов

pano.panov@gmail.com

Sofia, 2022

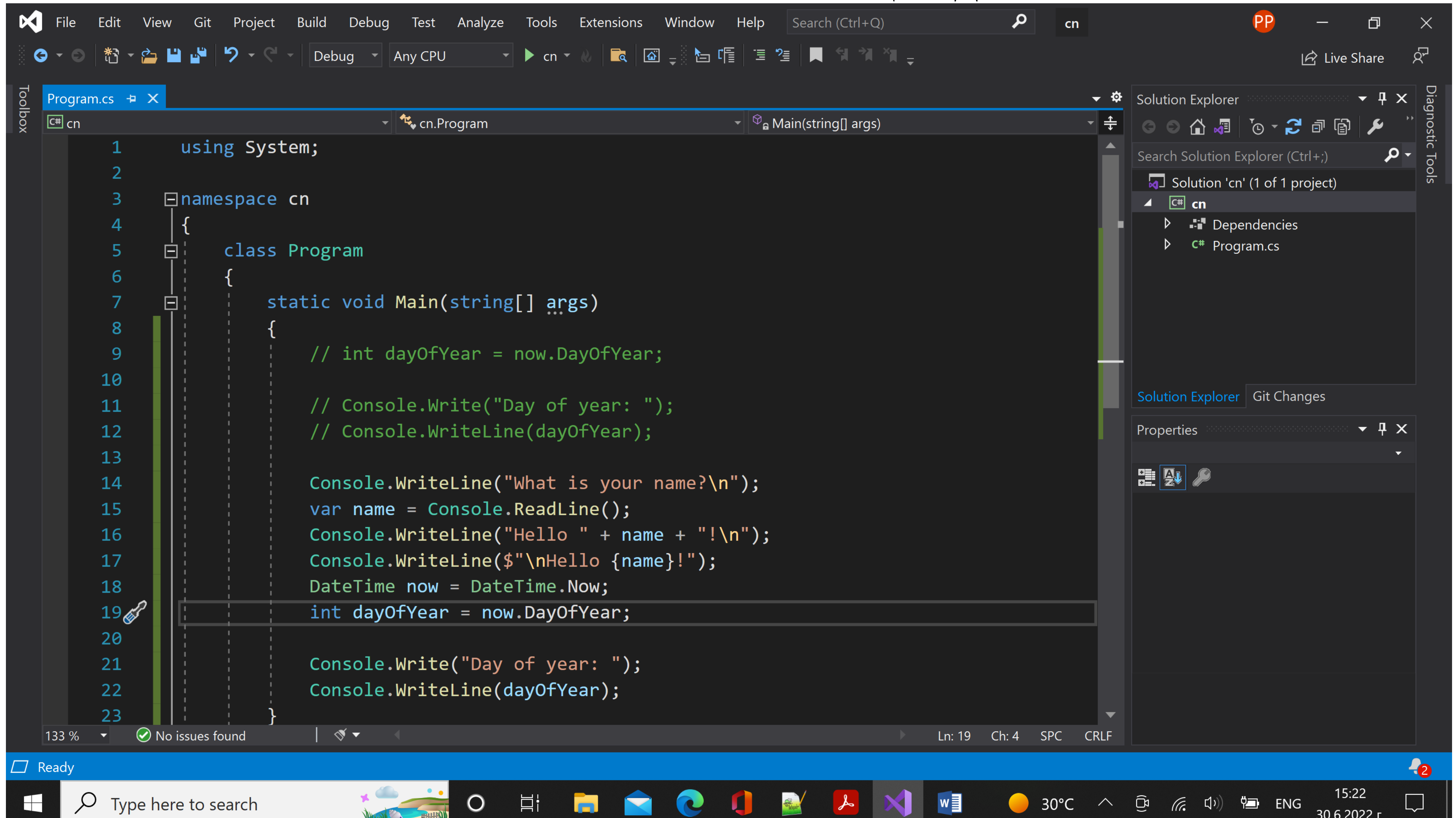
В тази лекция ще стане дума за:

- Средата за програмиране MS Visual Studio
- Езика за програмиране C#
- Елементи на програмата на C#
- Конзолно приложение
- Въвеждане и извеждане на данни.

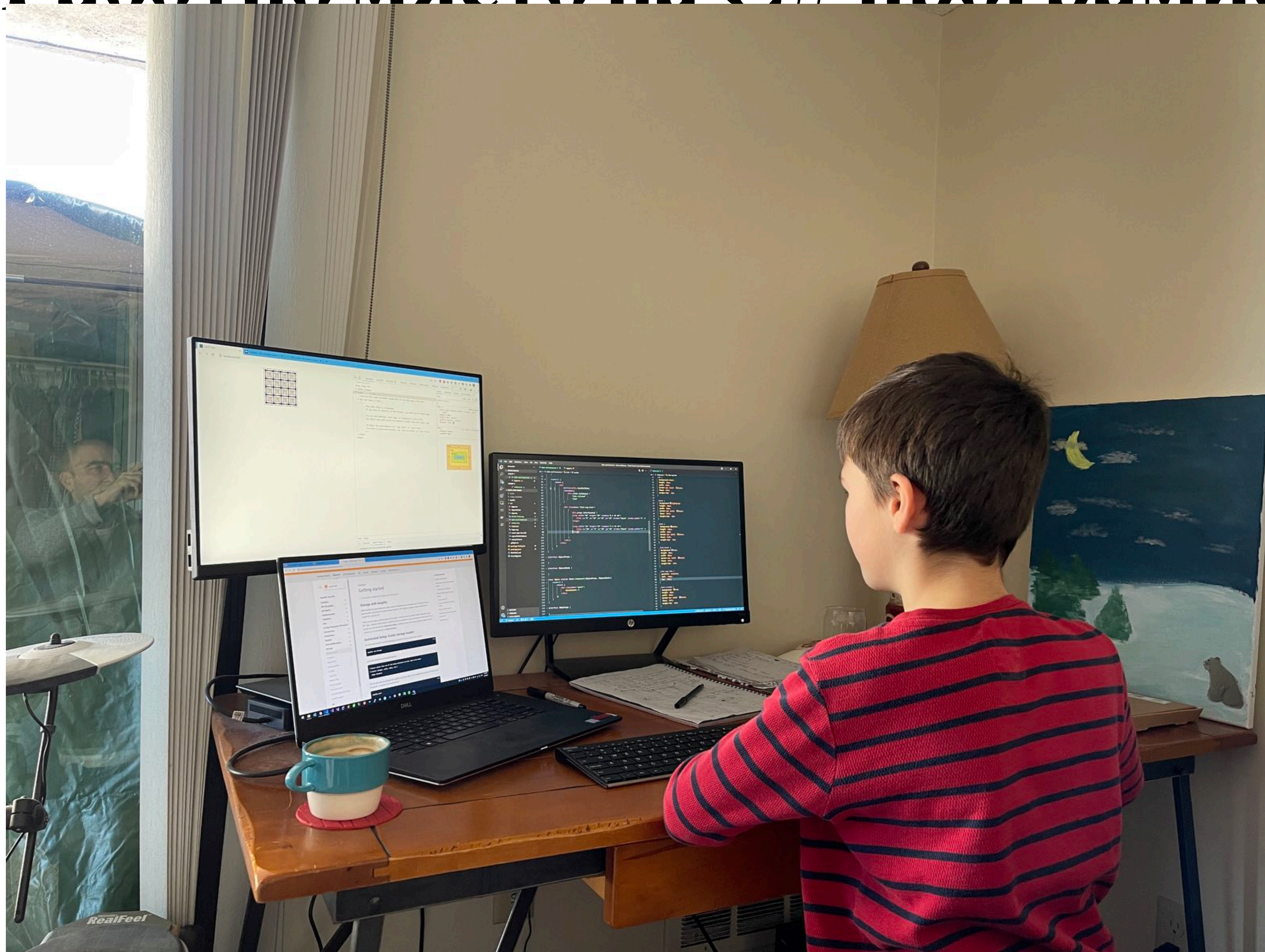
Добре дошли във Visual Studio IDE

- Visual Studio IDE е приложение, включващо голям брой инструменти, които поддържат разнообразни аспекти на разработката на софтуер.
- То е креативна стартова площадка, която служи за въвеждане, редактиране и форматиране на код, отстраняване на грешки, тестване и публикуване на готово за ползване приложение.
- Visual Studio включва интелигентен редактор на код, мощен дебъгер, компилатори, инструменти за завършване на код, графични дизайнери и много други функции за подобряване на процеса на разработка на софтуер.
- Налични са три версии на Visual Studio: Community, Professional и Enterprise. Ние ще работим с Community, която е безплатна.

Visual Studio – общ вид



Работно място на C# програмист



Project, Solution, Solution Explorer

- **Project:** Когато създаваме приложение или уебсайт във Visual Studio, започваме със създаването на *проект*. В логически смисъл проектът съдържа всички файлове, които трябва да се компилират до *изпълним файл*, *библиотека* или *уебсайт*. Тези файлове могат да съдържат програмен код, изображения, данни и т.н. Проектът също така съдържа настройки на компилатора и други конфигурационни файлове, които може да са необходими за ползване на различни услуги или за връзка с компоненти, с които вашата програма ще комуникира.
- **Solution:** Проектът се съдържа в *решение*. Въпреки името си, решението не е „отговор“. Това е просто контейнер за един или повече свързани проекти, заедно с информация за компилирането, настройките на прозореца на Visual Studio и всякакви други файлове, които не са свързани с конкретен проект.
- **Solution Explorer:** Дава възможност за преглеждане, навигиране и управление на наличните *кодovi* файлове. Включва библиотеки и компоненти на изпълнителната система. Solution Explorer служи за организиране на кода, като групира файловете в решения и проекти.

Вълнообразни подчертавания

Вълнообразните подчертавания ни предупреждават за грешки или потенциални проблеми в нашия код, докато пишем. Тези визуални подсказки ни помагат да отстраним проблемите незабавно, без да чакаме да открием грешките по време на „билдване“ или изпълнение. Ако задържим курсора на мишката върху вълничка, ще видим по-детайлна информация за грешката. В лявото поле може да се появи и крушка, показваща бързи действия, които можем да предприемем, за да коригираме грешката.

```
int prim=3*Math.Log8
```

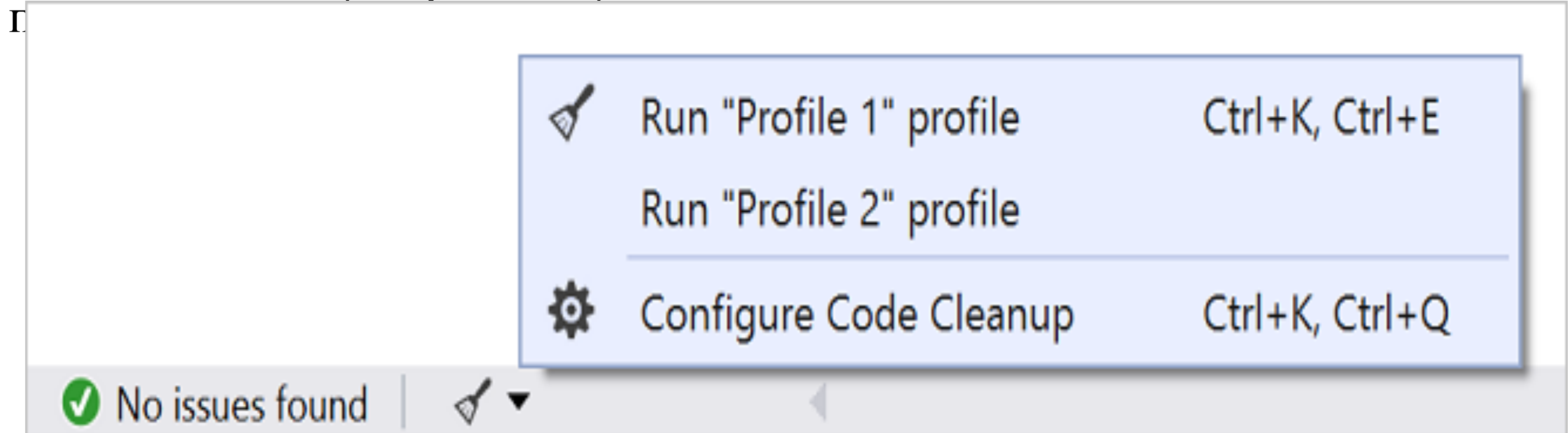


CS0117: 'Math' does not contain a definition for 'Log8'

Show potential fixes (Alt+Enter or Ctrl+.)

Почистване и форматиране на кода

С натискането на един бутон можем да форматираме кода си и да приложим всякакви корекции на кода, предложени от настройките на стила на кода, конвенциите за `.editorconfig` и анализаторите на Roslyn. Почистването на кода, което понастоящем е достъпно само за C# код, ни помага да разрешим проблеми в кода си, преди да го предоставим за



Още функционалности и удобства

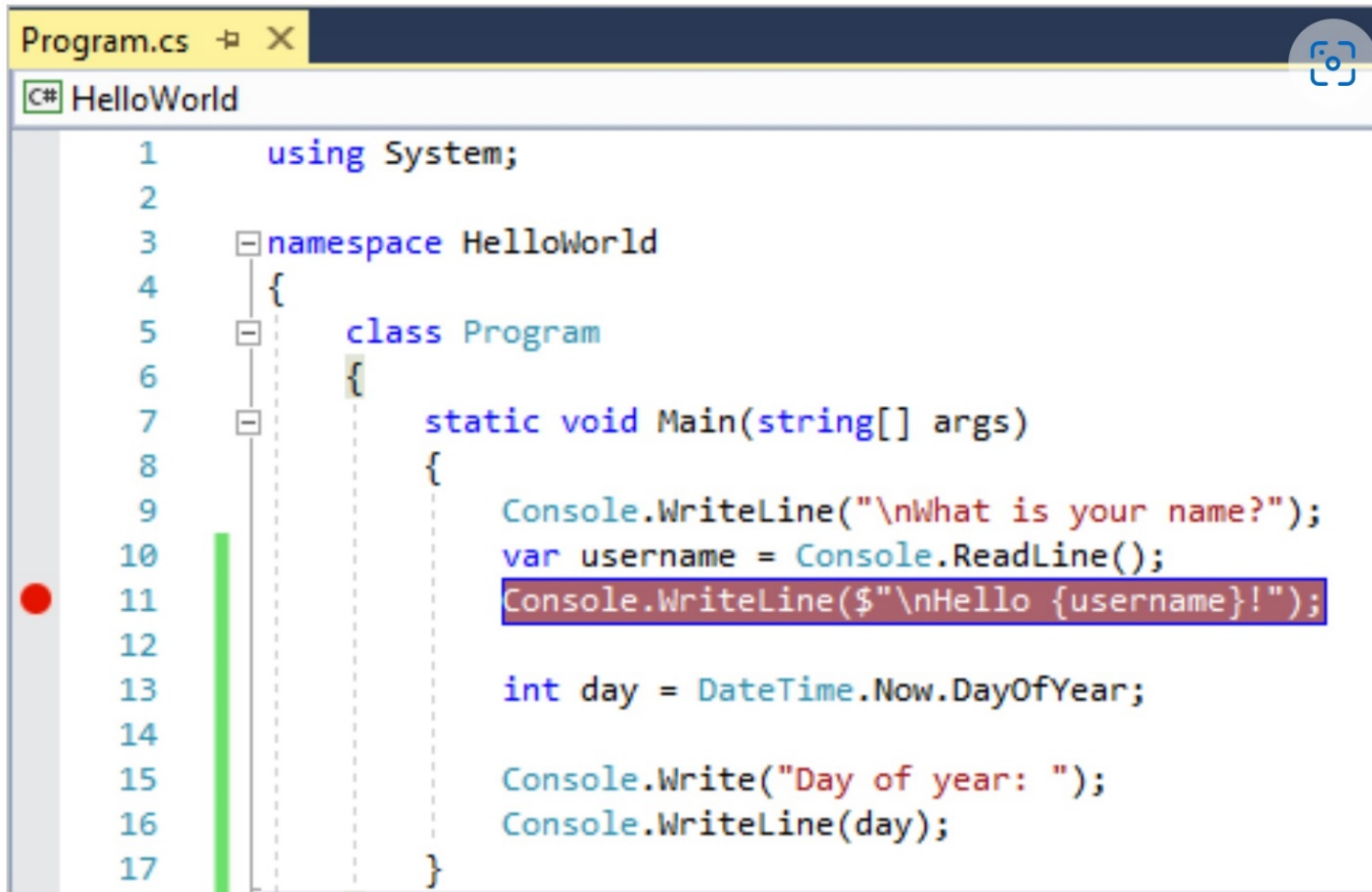
- **Refactoring:** Рефакторингът включва операции като интелигентно преименуване на променливи, извличане на един или повече реда код в нов метод и промяна на реда на параметрите на метода.
- **IntelliSense:** Набор от функции, които показват информация за нашия код директно в редактора и в някои случаи пишат малки части от код вместо нас. Това е като да имате вградена основна документация в редактора, така че не е нужно да търсите информация за типа другаде.
- **Visual Studio search:** Менютата, опциите и свойствата на Visual Studio понякога са твърде сложни и объркани. Търсенето във Visual Studio или Ctrl+Q е лесен начин за бързо намиране на код, функции и други компоненти в целия solution от едно място.
- **Live Share:** Споделяне на живо. Съвместно редактиране и отстраняване на грешки с други колеги в реално време, независимо от типа на нашето приложение или програмния език. Можем незабавно и сигурно да споделим своя проект. Можем също така да споделяме сесии за отстраняване на грешки, екземпляри на терминали, локални уеб приложения, гласови повиквания и други.
- **Call Hierarchy:** Йерархия на извикванията. Прозорецът на йерархията на извикванията показва методите, които извикват избран метод. Тази информация може да бъде полезна, когато обмисляме промяна или премахване на даден метод или когато се опитваме да проследим грешка.

Използване на вградения дебъгер

Написаният код, трябва да бъде стартиран и тестван за грешки. Системата за отстраняване на грешки на Visual Studio ни позволява да преминем през кода оператор след оператор, като проверяваме стойностите на променливите по време на изпълнението. Можем да зададем точки на прекъсване, които спират изпълнението на кода на определен ред и да наблюдаваме как стойностите на променливите се променят, докато кодът се изпълнява. Да зададем точка на прекъсване, за да видим стойността на променливата `username`, докато програмата работи. Поставяме точка на прекъсване на реда `Console.WriteLine($"\\nHello {username}!");` като кликнем в крайното ляво поле или натиснем F9. В полето се появява червен кръг и линията се маркира.

Започваме отстраняване на грешки, като избирем `Debug > Start Debugging` или F5. Когато се появи прозорецът на конзолата и поиска име, въвеждаме нашето име. Фокусът се връща към редактора на код на Visual Studio и редът от код с точката на прекъсване се маркира в жълто. Жълтото осветяване означава, че този ред от код предстои да се изпълни. Точката на прекъсване кара приложението да спре изпълнението на този ред. Задръжаме курсора на мишката върху променливата `username`, за да видим нейната стойност. Можете също така да щракнете с десния бутон върху `username` и да изберем `Добавяне на наблюдател`, за да добавим променливата към прозореца за наблюдение, където можем също да видим нейната стойност. Натискаме F9 за да премаънем точката на прекъсване.

Използване на вградения дебъггер



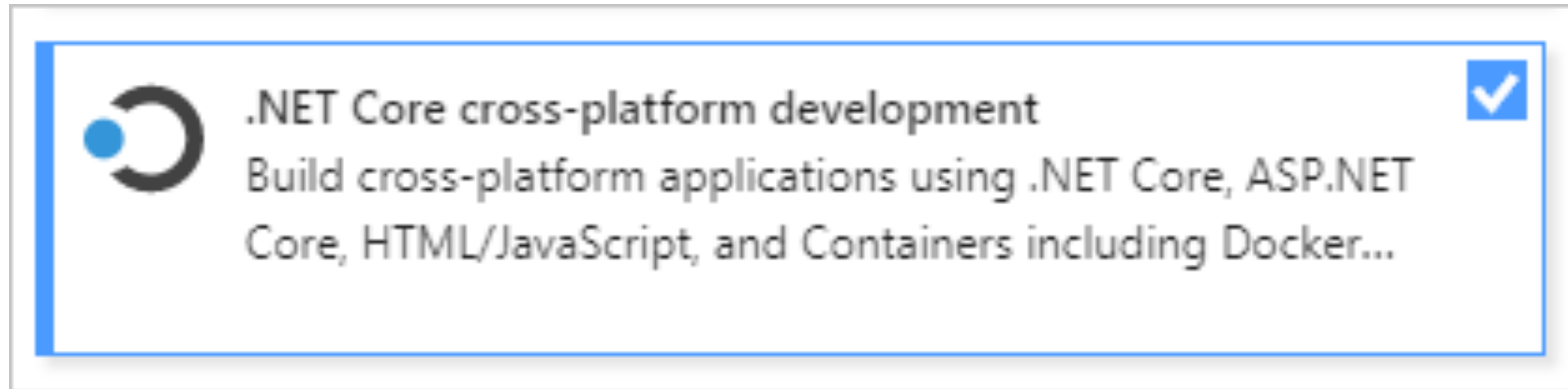
The image shows a screenshot of a C# code editor window. The title bar at the top indicates the file is 'Program.cs'. Below the title bar, the editor shows the code for a 'HelloWorld' application. The code is as follows:

```
1 using System;
2
3 namespace HelloWorld
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("\nWhat is your name?");
10            var username = Console.ReadLine();
11            Console.WriteLine($"Hello {username}!");
12
13            int day = DateTime.Now.DayOfYear;
14
15            Console.Write("Day of year: ");
16            Console.WriteLine(day);
17        }
18    }
19 }
```

A red dot on the left margin indicates a breakpoint is set on line 11. A green vertical bar highlights the current line of execution, which is also line 11. The code is color-coded: keywords are blue, strings are red, and comments are green. The line containing the breakpoint is highlighted with a blue background.

Инсталиране на Visual Studio 2019

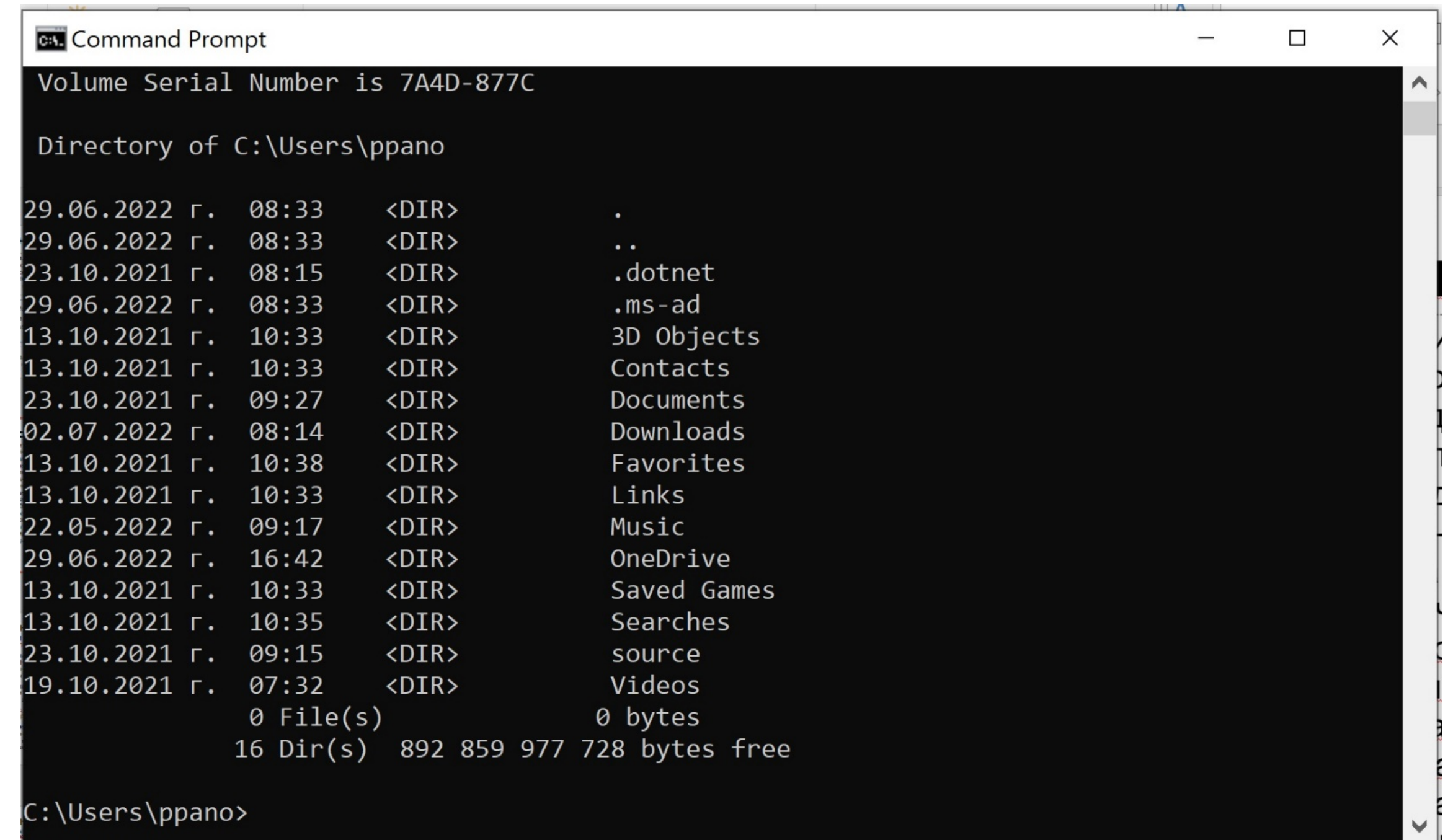
За да започнем инсталацията, изтегляме Visual Studio инсталатора и го инсталираме нашата система. Модулният инсталатор ни позволява да избираме и инсталираме работни модули, които са групи от функции, необходими за програмните езици или платформи, които искаме. За да следваме стъпките за създаване на програма, по време на инсталацията трябва да изберем приложението за кросплатформена разработка на .NET Core.



Когато отворим Visual Studio за първи път, трябва да влезем, като използваме нашия акаунт в Microsoft или нашия служебен или учебен акаунт.

Конзолно приложение

Конзолното приложение, в контекста на C#, е приложение, което приема входни данни и показва изходни данни в конзолата на командния ред с достъп до три основни потока от данни: стандартен вход, стандартен изход и стандартен изход за съобщения за грешка. Конзолното приложение улеснява четенето и писането на знаци от конзола - поотделно или като цял ред. Това е най-простата форма на C# програма и обикновено се извиква от командния ред на Windows Console.



```
Command Prompt
Volume Serial Number is 7A4D-877C

Directory of C:\Users\ppano

29.06.2022 г. 08:33 <DIR> .
29.06.2022 г. 08:33 <DIR> ..
23.10.2021 г. 08:15 <DIR> .dotnet
29.06.2022 г. 08:33 <DIR> .ms-ad
13.10.2021 г. 10:33 <DIR> 3D Objects
13.10.2021 г. 10:33 <DIR> Contacts
23.10.2021 г. 09:27 <DIR> Documents
02.07.2022 г. 08:14 <DIR> Downloads
13.10.2021 г. 10:38 <DIR> Favorites
13.10.2021 г. 10:33 <DIR> Links
22.05.2022 г. 09:17 <DIR> Music
29.06.2022 г. 16:42 <DIR> OneDrive
13.10.2021 г. 10:33 <DIR> Saved Games
13.10.2021 г. 10:35 <DIR> Searches
23.10.2021 г. 09:15 <DIR> source
19.10.2021 г. 07:32 <DIR> Videos
0 File(s) 0 bytes
16 Dir(s) 892 859 977 728 bytes free

C:\Users\ppano>
```

Конзолното приложение обикновено съществува под формата на самостоятелен изпълним файл с минимален или никакъв графичен потребителски интерфейс (GUI). Потребителят обикновено управлява конзолното приложение използвайки само клавиатура и монитор. В днешно време конзолните приложения се разработват в модерна програмна среда, като например тази на Visual Studio. По този начин се улеснява възприемането и научаването на нов програмен език, като се избягва сложността от възприемането и на графичен интерфейс. За обработка на информация и администриране, конзолното приложение е доста удобно, интегрирано в самата операционна система.

Конзолно приложение Hello world

C (език за програмиране)

```
#include <stdio.h>

int main(void) {
    printf("Hello, world!\n");
    return 0;
}
```

C++

```
#include <iostream>

Using namespace std;

int main() {
    cout<<"Hello, world"<< endl;
    return 0;
}
```

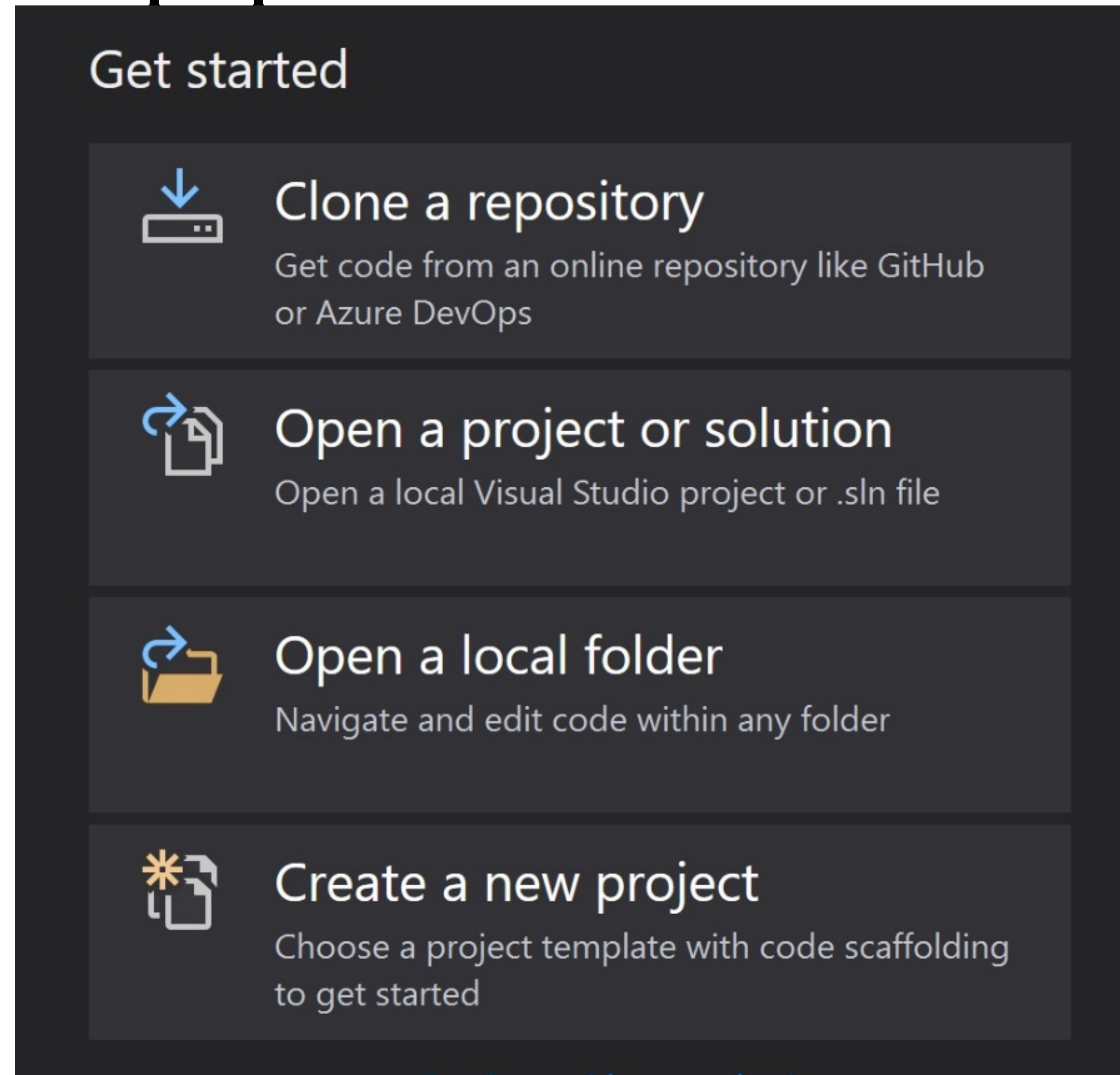
C#

```
using System;

class HelloWorldApp
{
    public static void Main()
    {
        Console.WriteLine("Hello,
world!");
    }
}
```

Създаване на нов проект с Visual Studio

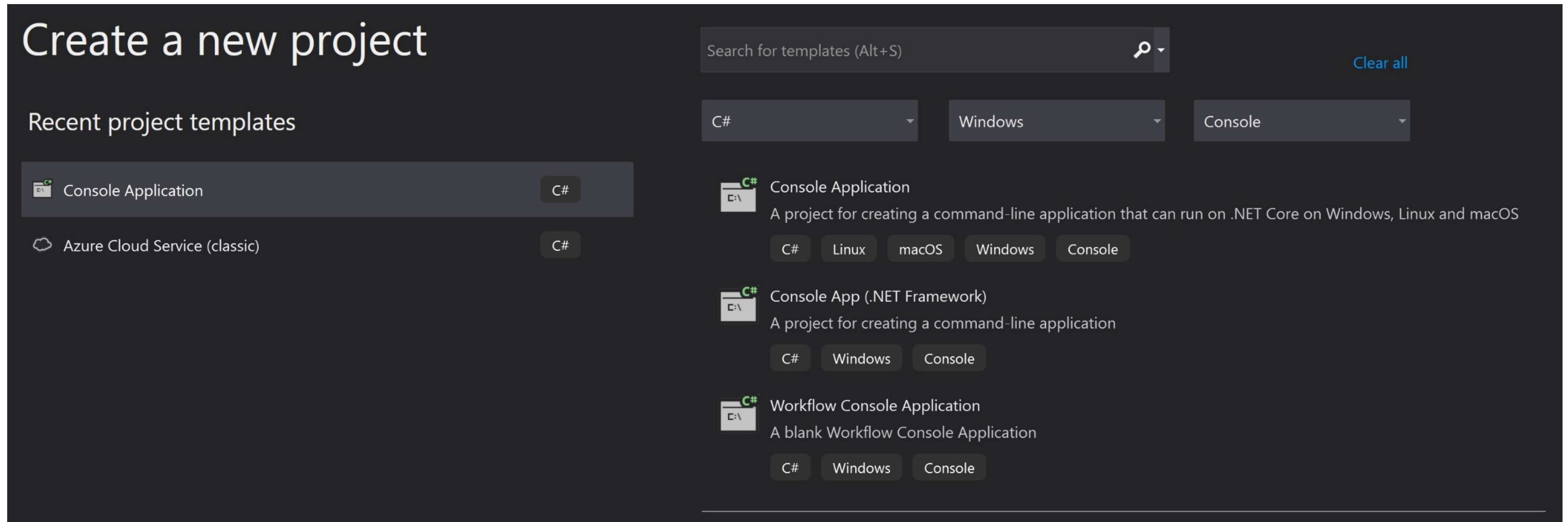
1. Стартираме Visual Studio



2. Избираме Create a new project

Създаване на нов проект с Visual Studio

Прозорецът Създаване на нов проект се отваря и показва няколко шаблона за проекти. Шаблонът съдържа основните файлове и настройки, необходими за даден тип проект.

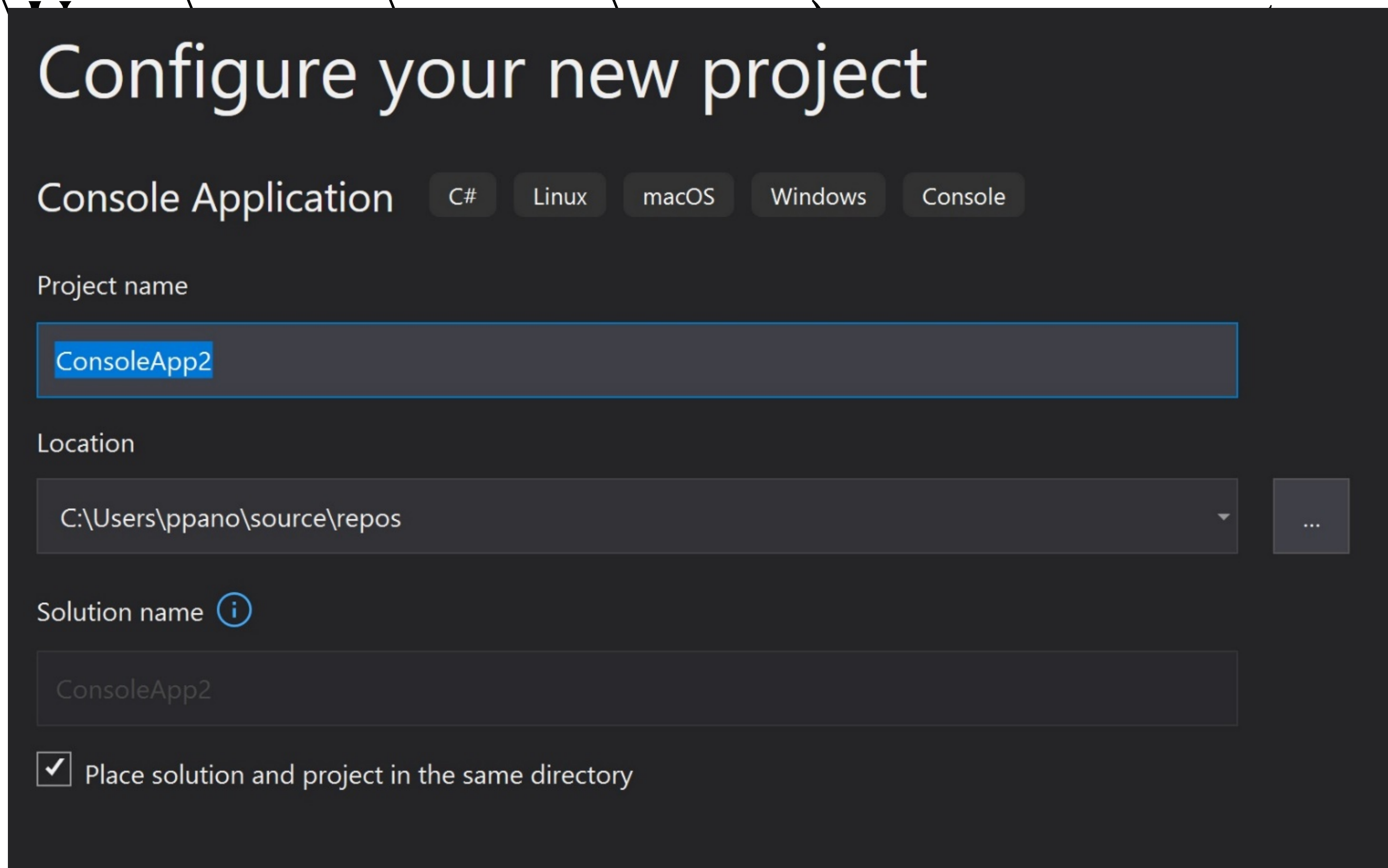


За да намерим шаблона, който искаме, въвеждаме `.net core console` в полето за търсене. Списъкът с налични шаблони се филтрира автоматично въз основа на въведените от нас ключови думи. Избираме `C#` от падащия списък Всички езици; `Windows` от списъка на Всички платформи и `Конзола` от списъка Всички типове проекти. Избираме шаблона за конзолно приложение и след това избираме Напред.

Създаване на нов проект с Visual Studio

4. В прозореца Конфигуриране на нашия нов проект въвеждаме HelloWorld в полето Име на проекта. По желание може да променим местоположението на директорията за файловете на нашия проект (пътят по подразбиране е C:

Напред.



Configure your new project

Console Application C# Linux macOS Windows Console

Project name

ConsoleApp2

Location

C:\Users\ppano\source\repos

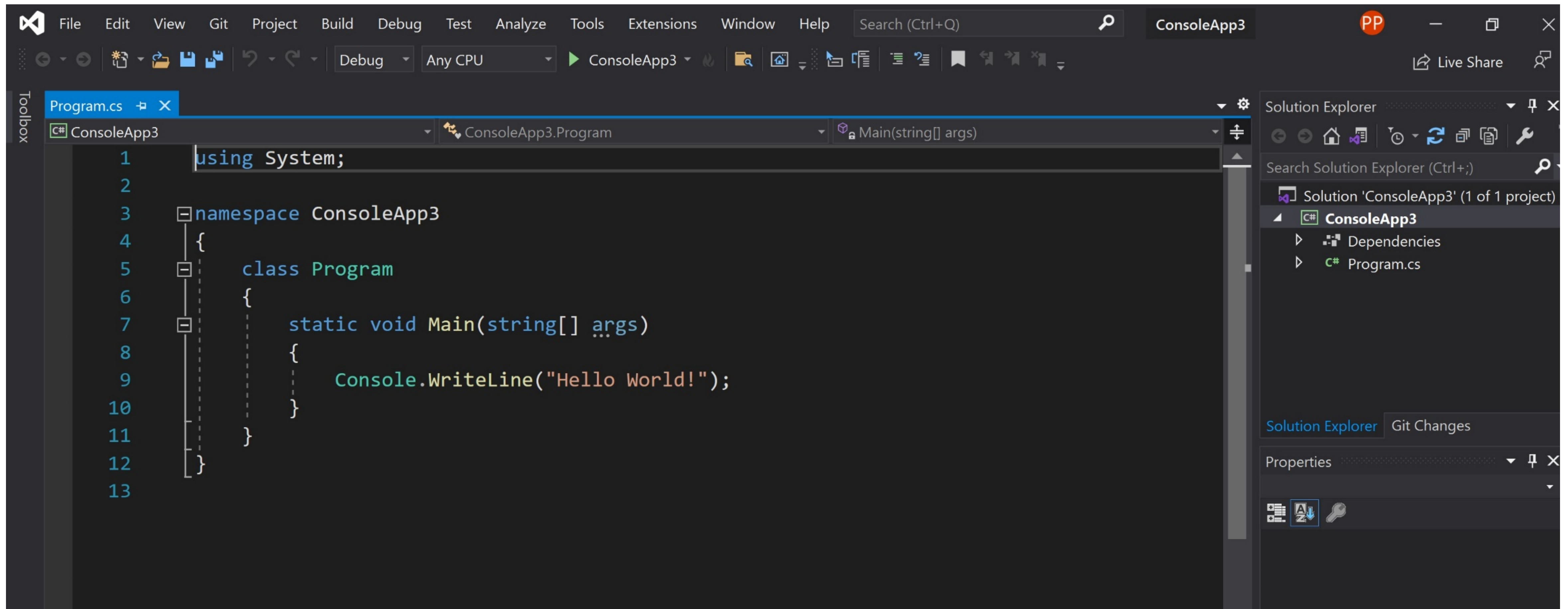
Solution name ⓘ

ConsoleApp2

☒ Place solution and project in the same directory

Създаване на нов проект с Visual Studio

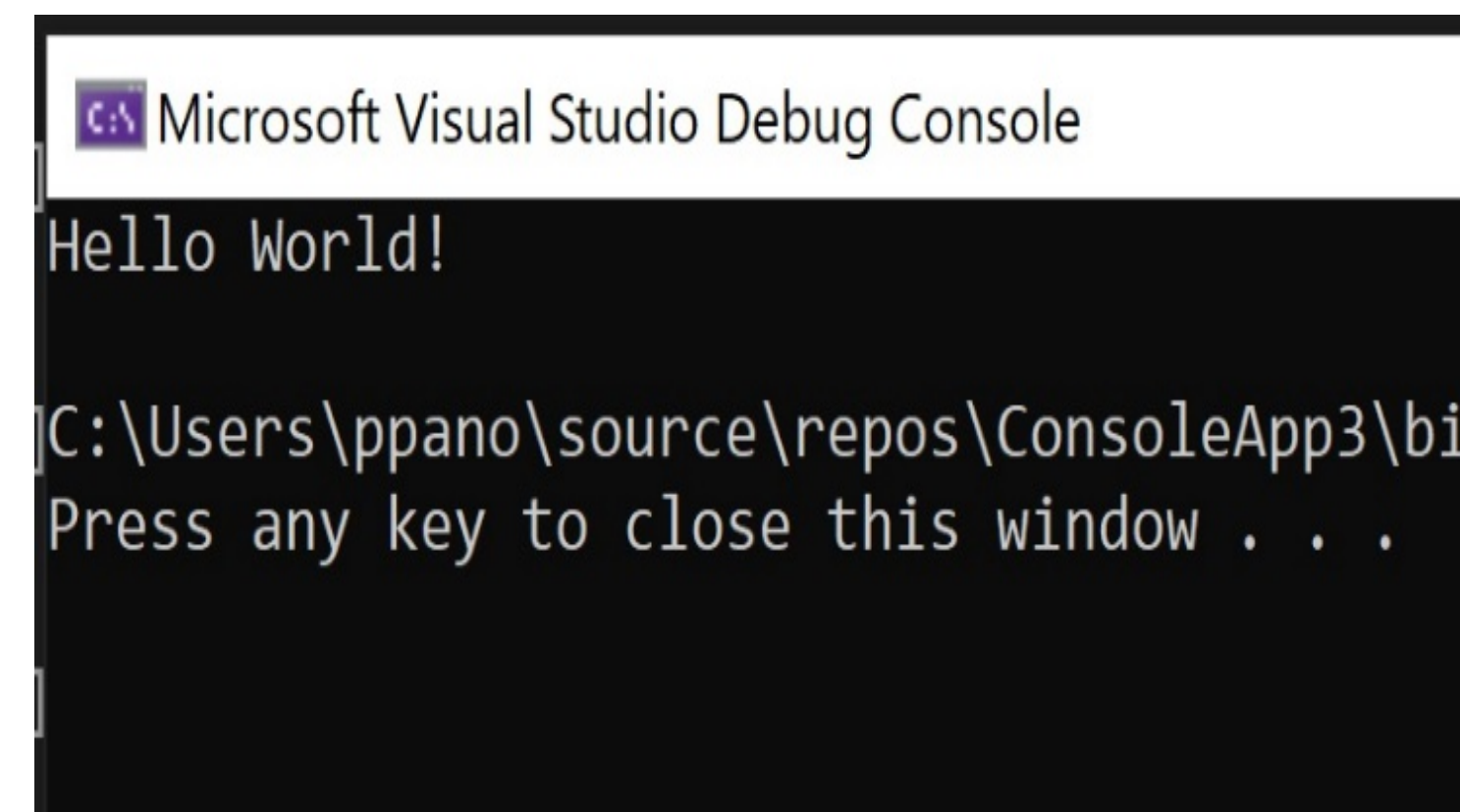
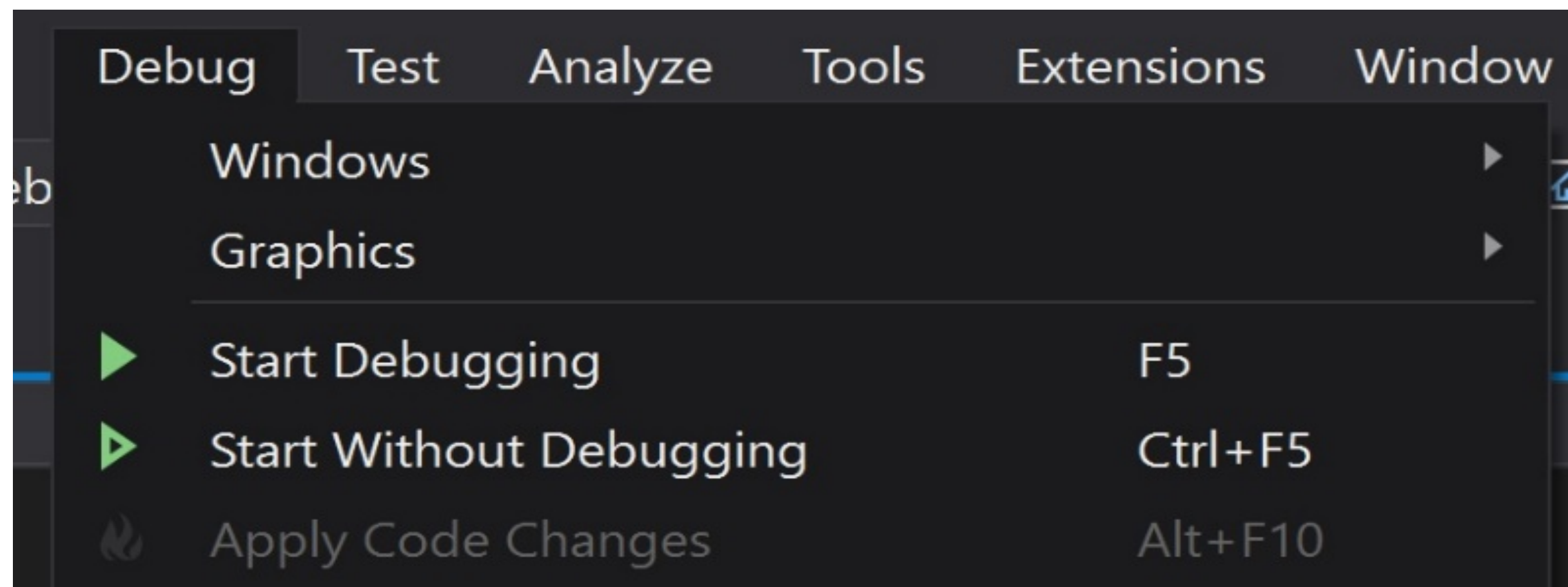
5. Visual Studio създава проекта. Това е просто приложение "Hello World", което извиква метода `Console.WriteLine()`, за да покаже низ "Hello World!" в прозореца на конзолата (изход на програмата).



6. Стартираме конзолното приложение от бутон Run

Стартиране на проекта с Visual Studio

Козолното приложение стартираме с или без дебъгинг, както е показано на картинката по-долу: Избираме Debug → Start с или без Debugging (за предпочитане като начало):



Visual Studio създава нашето конзолно приложение и го стартира. Появява се конзолен прозорец и текстът Hello World!

След като сме видели резултатът от изпълнението, по-добре е да затворим конзолата, защото при останала отворена конзола следващо компилиране няма да е възможно.

Спиране на изпълнението на нашия проект става, като се натисне бутона с изобразено червено квадратче, вдясно от зеленото триъгълниче за старт.

Елементарна структура на C# програма

```
using System;
namespace ConsoleApp3
{
    class Program
    {
        static void Main(string[] args)
        {
            ... C# КОД ...
        }
    }
}
```

Какво скрие зад служебните думи **using**, **namespace**, **class** и **static void Main**?

Елементарна структура на C# програма

using System ключовата дума `using` се използва за включване на пространството от имена на средата `System` в програмата. Програмата обикновено има множество `using` оператори.

Следващият ред съдържа декларацията **namespace** за пространство от имена `ConsoleApp3` на нашата програма. Пространствата от имена се използват за организиране на класовете. Те помагат да се контролира обхвата на методите и класовете в по-големи .Net проекти. `namespace` е начин да запазим един набор от имена (като имена на класове), различен от други набори от имена. Най-голямото предимство на използването на `namespace` е, че имената на класове, които са декларирани в едно пространство от имена, няма да се сблъскат със същите имена на класове, декларирани в друго пространство от имена.

Следващият ред съдържа декларация на клас, **class** `Program` включва данните и дефинициите на методите, които нашата програма използва. Класовете обикновено съдържат множество методи. Методите определят поведението на класа. Класът `ConsoleApp3` обаче има само един метод – задължителният за всяко конзолно приложение главен метод **Main**.

Следващият ред дефинира метода `Main`, който е входна точка за всяко конзолно приложение на C#. Методът `Main` определя какво прави класът `Program`, когато се изпълни.

Конзолно приложение с вход и изход

Нека променим нашата първа програма по следния начин:

Да подменим реда:

```
Console.WriteLine("Hello World!");
```

със C# кода:

```
Console.WriteLine("\nWhat is your name?");  
var name = Console.ReadLine(); // Implicit type definition  
Console.WriteLine($"Hello {name}!");
```

Този код извежда подсказка: Какво е вашето име? в прозореца на конзолата и след това изчаква, докато потребителят въведе някакъв текст, последван от Enter.

Приложението извежда текста:

Hello <въведения от потребителя текст> !

Стартирайте приложението отново, като изберете Debug > Start Without Debugging или като натиснете Ctrl+F5.

Visual Studio създава и стартира приложението, отваря се конзолен прозорец, който ви подканва да въведете вашето име. Въведете името си в прозореца на конзолата и натиснете Enter. Приложението извежда:

Hello <Въведеното име>!

Разлика между методите Console.Write() и Console.WriteLine()

Печат на число, въведено от потребител

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int number;
        Console.Write("Enter a number:");
        number = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("You entered :{0}",number);
        Console.ReadLine();
    }
}
```

Методът `Convert.ToInt32()` превръща низа, прочетен от конзолата в 32 битово цяло число, чиято стойност се присвоява на променливата `number`.

Събиране на две цели числа

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int num1, num2, sum;
        Console.WriteLine("Calculate the sum of two numbers:");
        Console.Write("Input number1:");
        num1 = Convert.ToInt32(Console.ReadLine());
        Console.Write("Input number2:");
        num2 = Convert.ToInt32(Console.ReadLine());
        sum = num1 + num2;
        Console.Write("Result:" + sum);
    }
}
```

Умножение на две дробни числа

```
using System;
class Program
{
    static void Main(string[] args)
    {
        float number1,number2,product;
        Console.Write("Enter a number1:");
        number1 = Convert.ToSingle(Console.ReadLine());
        Console.Write("Enter a number2:");
        number2 = Convert.ToSingle(Console.ReadLine());
        product = number1 * number2;
        Console.WriteLine("{0} * {1} = {2}",number1,number2,
product);
    }
}
```

Методът `Convert.ToSingle()` превръща низ в дробно число от тип `float`.

Лице на правоъгълник

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int area, length, width;
        Console.Write("Please write the length of your rectangle:
");
        length = Convert.ToInt32(Console.ReadLine());
        Console.Write("Please write the width of your rectangle: ");
        width = Convert.ToInt32(Console.ReadLine());
        area = length * width;
        Console.WriteLine("The area of rectangle is: {0}", area);
        Console.ReadKey();
    }
}
```

Въпроси и задачи

Въпроси:

1. Кой език за програмиране поддържа Visual Studio?
2. Каква е ролята на solution explorer?
3. Възможно ли е един solution да съдържа повече от един project?
4. Какво извършва свойството на Visual Studio - Refactoring?
5. Възможно ли е споделянето на код посредством Git във Visual Studio?

Задачи за програмиране на C#: Напишете конзолно приложение, което:

1. Пресмята лицето и периметъра на квадрат по зададена страна.
2. Пресмята дължината на окръжността и лицето на кръг с даден радиус.
3. Пита Петър каква кола кара, какъв е нейният цвят, колко литра изразходва на 100 км и каква е марката на гумите ѝ и извежда следния текст:
Колата на Петър е <марка на колта>, <цвет> на цвят, разход <V.V> на сто,
а гумите ѝ са марка <марка на гумите>.
4. Пресмята средно аритметичното на три числа.

Изгревът е близо

Благодаря за вниманието!

Готов съм да отговарям на вашите
въпроси

