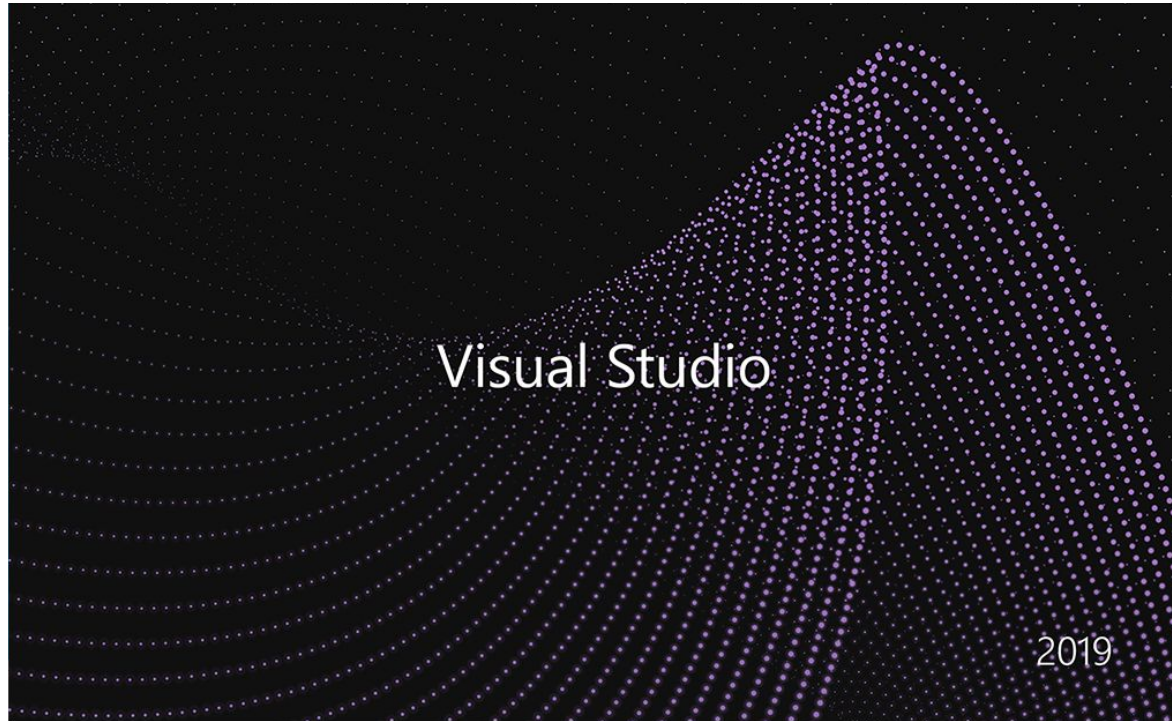




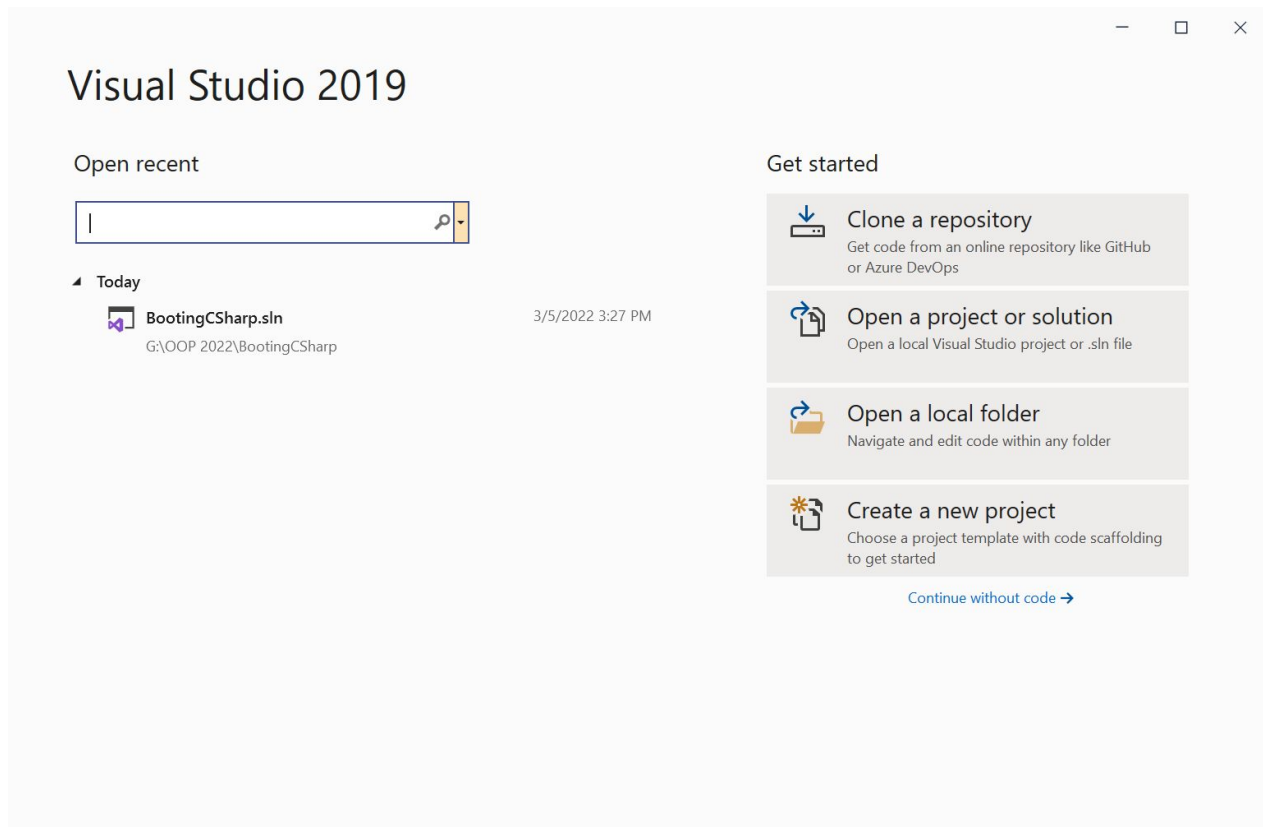
# Booting on C#



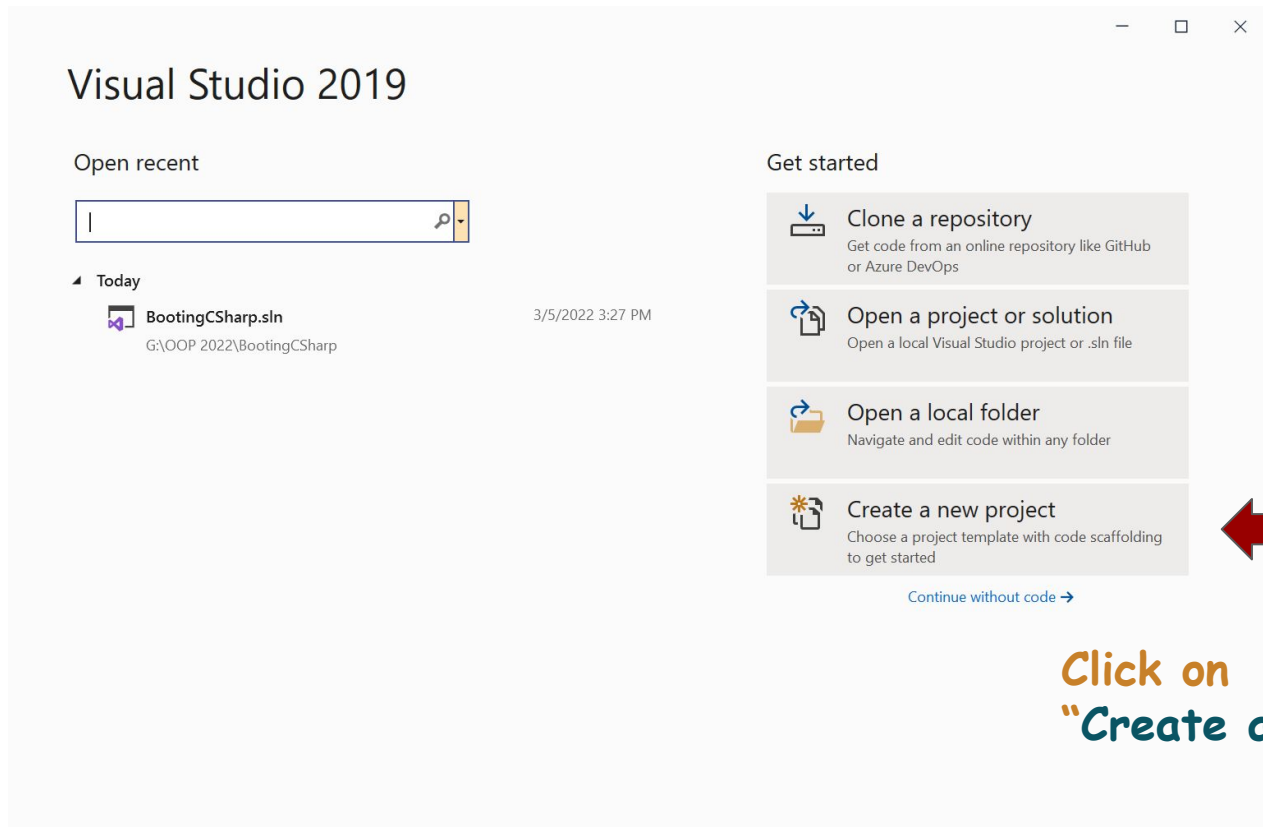
# Visual Studio 2019 Community Edition



# Visual Studio 2019 Community Edition

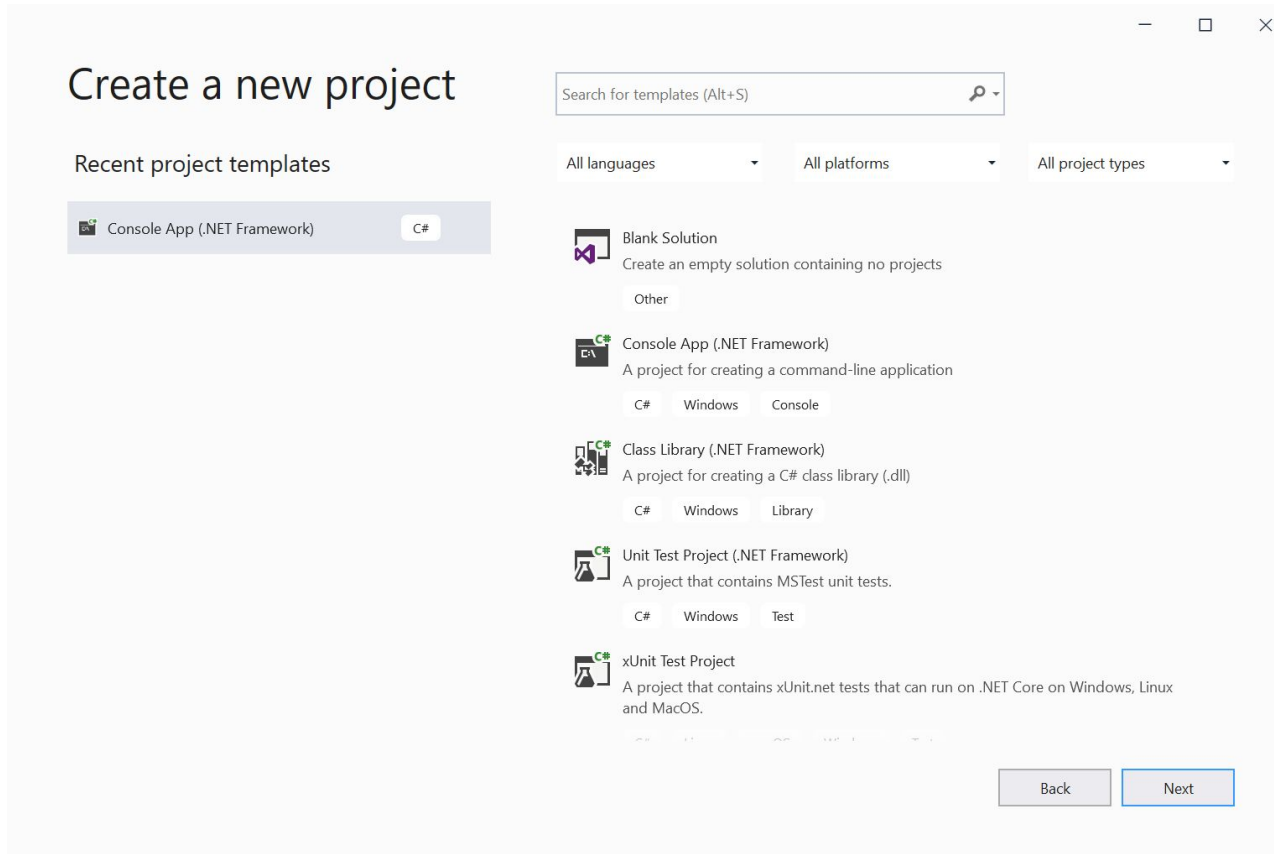


# Visual Studio 2019 Community Edition

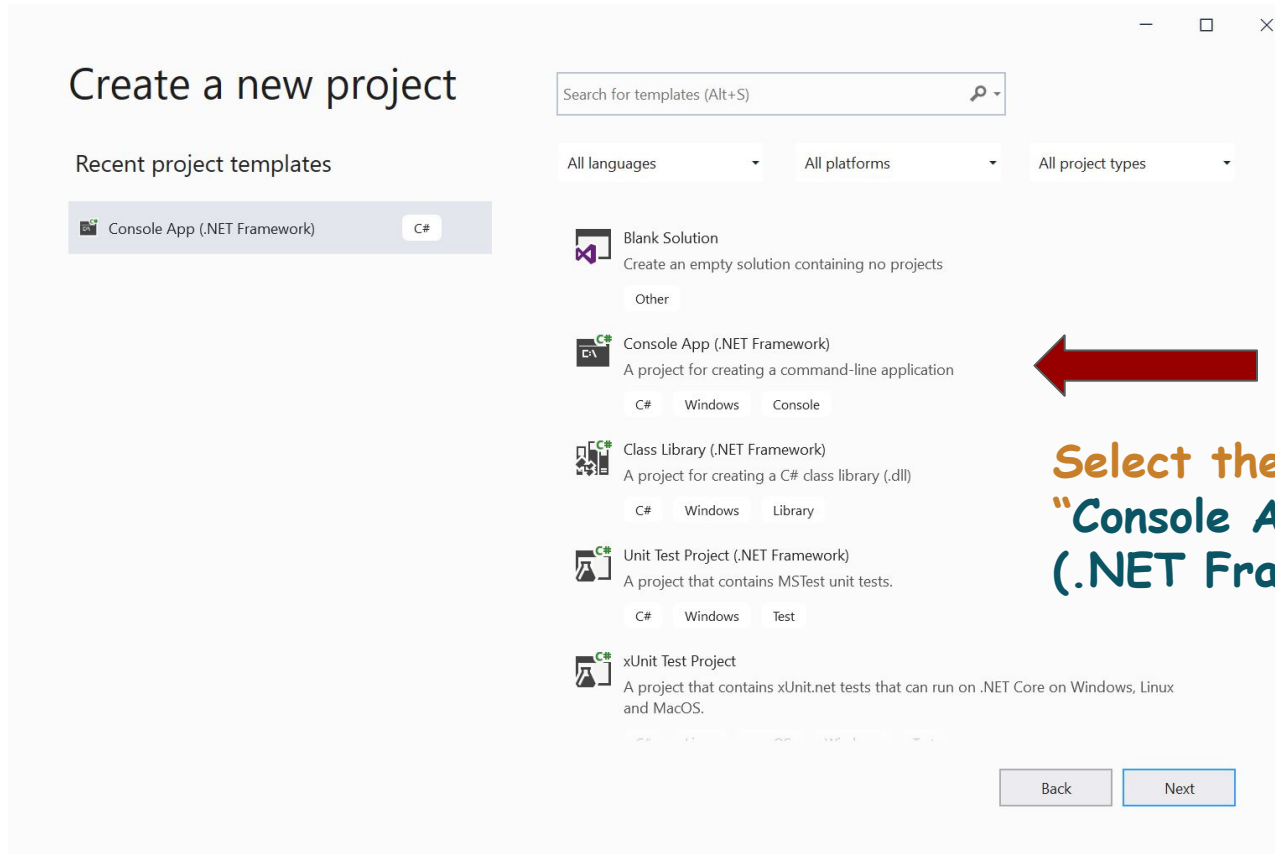


Click on  
"Create a new project"

# Select the Project

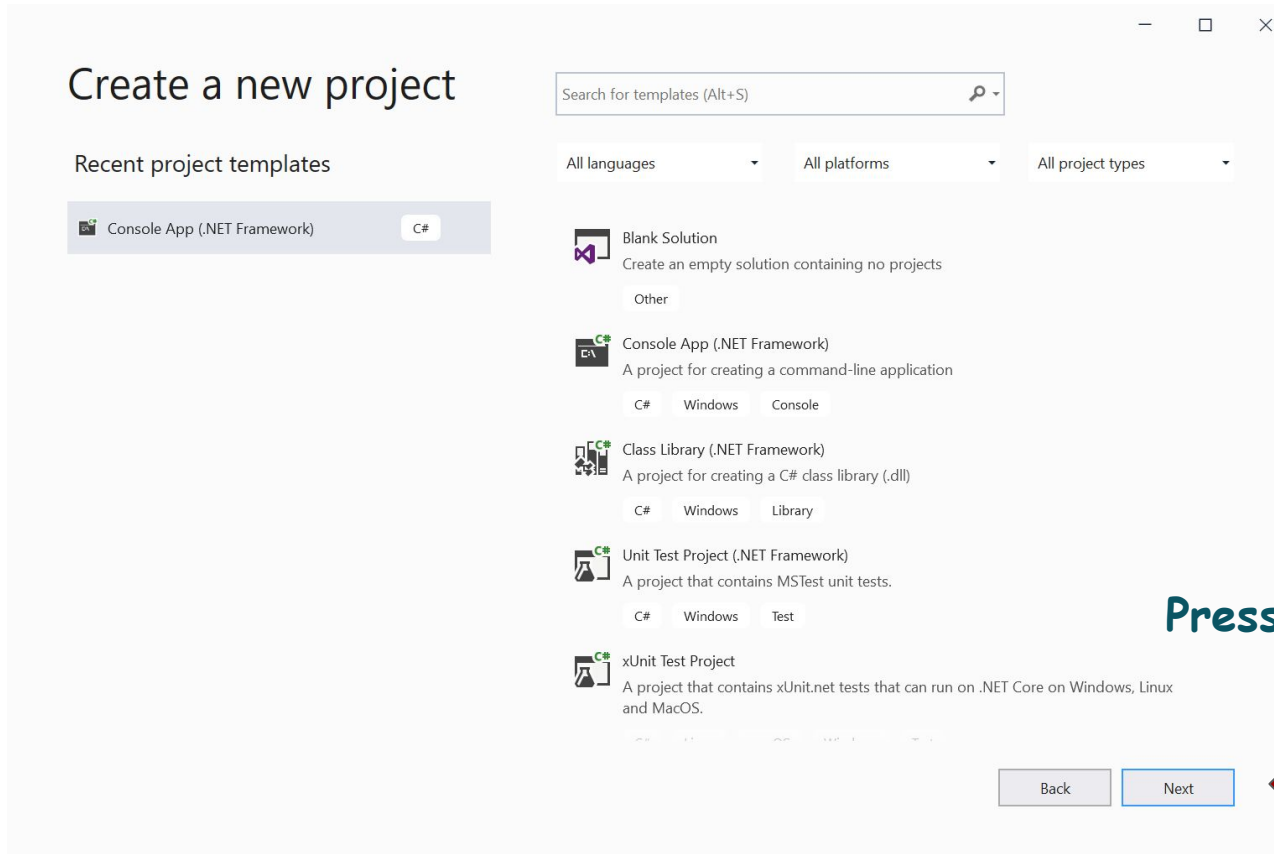


# Select the Project



Select the Project  
"Console App  
(.NET Framework)"

# Select the Project



Press **NEXT** button

# Configure the Project

— □ ×

## Configure your new project

Console App (.NET Framework) C# Windows Console

Project name

ConsoleApp1

Location

C:\Users\Hp\source\repos

...

Solution name ⓘ

ConsoleApp1

☐ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Back

Create



# Configure the Project

Configure your new project

Console App (.NET Framework) C# Windows Console

Project name

ConsoleApp1

Location

C:\Users\Hp\source\repos

Solution name ⓘ

ConsoleApp1

☐ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Back Create

Write the Project Name



# Configure the Project

Configure your new project

Console App (.NET Framework) C# Windows Console

Project name

ConsoleApp1

Location

C:\Users\Hp\source\repos

Solution name ⓘ

ConsoleApp1

☐ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Back Create

Select the directory where you want to save your project



# Configure the Project

— □ ×

## Configure your new project

Console App (.NET Framework) C# Windows Console

Project name

ConsoleApp1

Location

C:\Users\Hp\source\repos

Solution name ⓘ

ConsoleApp1

☐ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

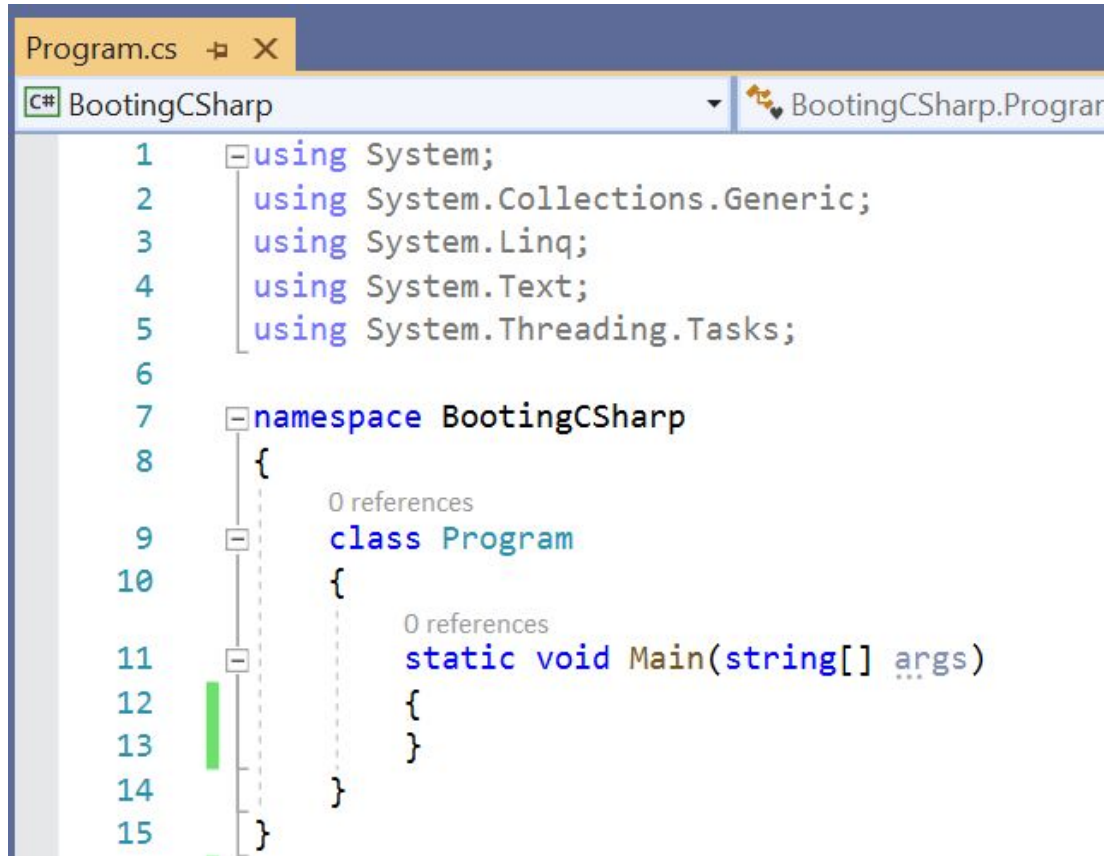
Back

Create

Press Create



# C# Program Basic Structure



The image shows a screenshot of a C# program in a code editor. The file is named `Program.cs` and is part of a project named `BoottingCSharp`. The code defines a `namespace BoottingCSharp` containing a `class Program` with a `Main` method. The code is as follows:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BoottingCSharp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
```

The code is displayed with line numbers 1 through 15 on the left. The `using` statements are grouped together. The `namespace` and `class` definitions are indented. The `Main` method is a static method. The code is color-coded: `using` is blue, `namespace` is blue, `class` is blue, `static` is blue, `void` is blue, `Main` is blue, `string` is blue, `args` is blue, and `args` is blue.

# C# Program Basic Structure

```
Program.cs  [icon] X
C# BootingCSharp BootingCSharp.Program
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
```



By default it is including some libraries. Just like we did

#include<iostream>  
In C++

# C# Program Basic Structure

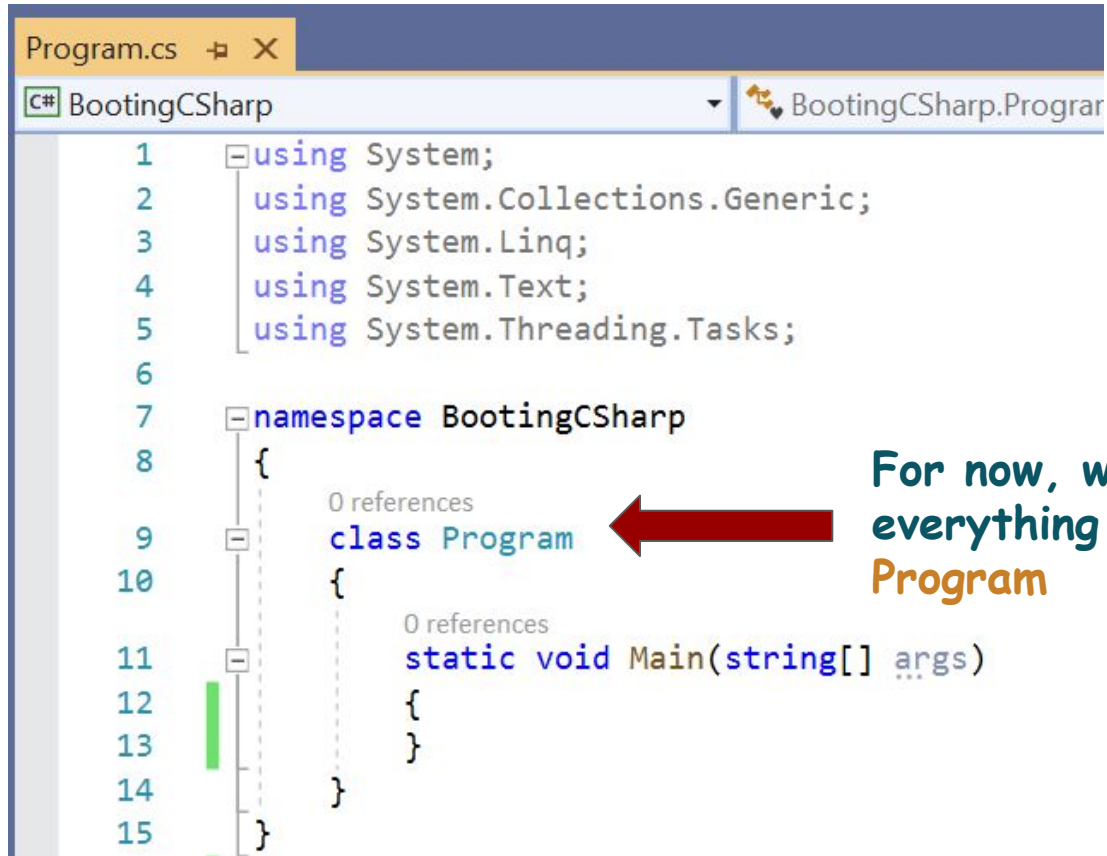
```
Program.cs [X]
[+] BootingCSharp BootingCSharp.Program
1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Text;
5      using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
```



Just like we wrote  
using namespace std;

Here namespace is  
the name of the project that we  
wrote.

# C# Program Basic Structure



```
Program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BootingCSharp
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

The screenshot shows a C# program in Visual Studio. The file is named Program.cs and is part of a project named BootingCSharp. The code defines a namespace BootingCSharp containing a class Program. The class Program has a static method Main that takes an array of strings as an argument. The code is written in a standard C# style with line numbers on the left.

For now, we will write everything inside the class **Program**

# Visual Studio: Error Window

The screenshot shows the Visual Studio IDE with a C# program named `Program.cs` open. The code defines a `namespace BootingCSharp` containing a `class Program` with a `Main` method. Inside the `Main` method, there is a `cout` statement, which is underlined in red, indicating an error. The Error List window at the bottom shows two errors: `CS1002 ; expected` and `CS0103 The name 'cout' does not exist in the current context`. The status bar at the bottom indicates 2 errors and 0 warnings.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace BootingCSharp
8 {
9     class Program
10     {
11         static void Main(string[] args)
12         {
13             cout
14         }
15     }
16 }
17
```

100 % 2 0 Ln: 17 Ch: 1 SPC CRLF

Error List

Entire Solution 2 Errors 0 Warnings 0 Messages Build + IntelliSense Search Error List

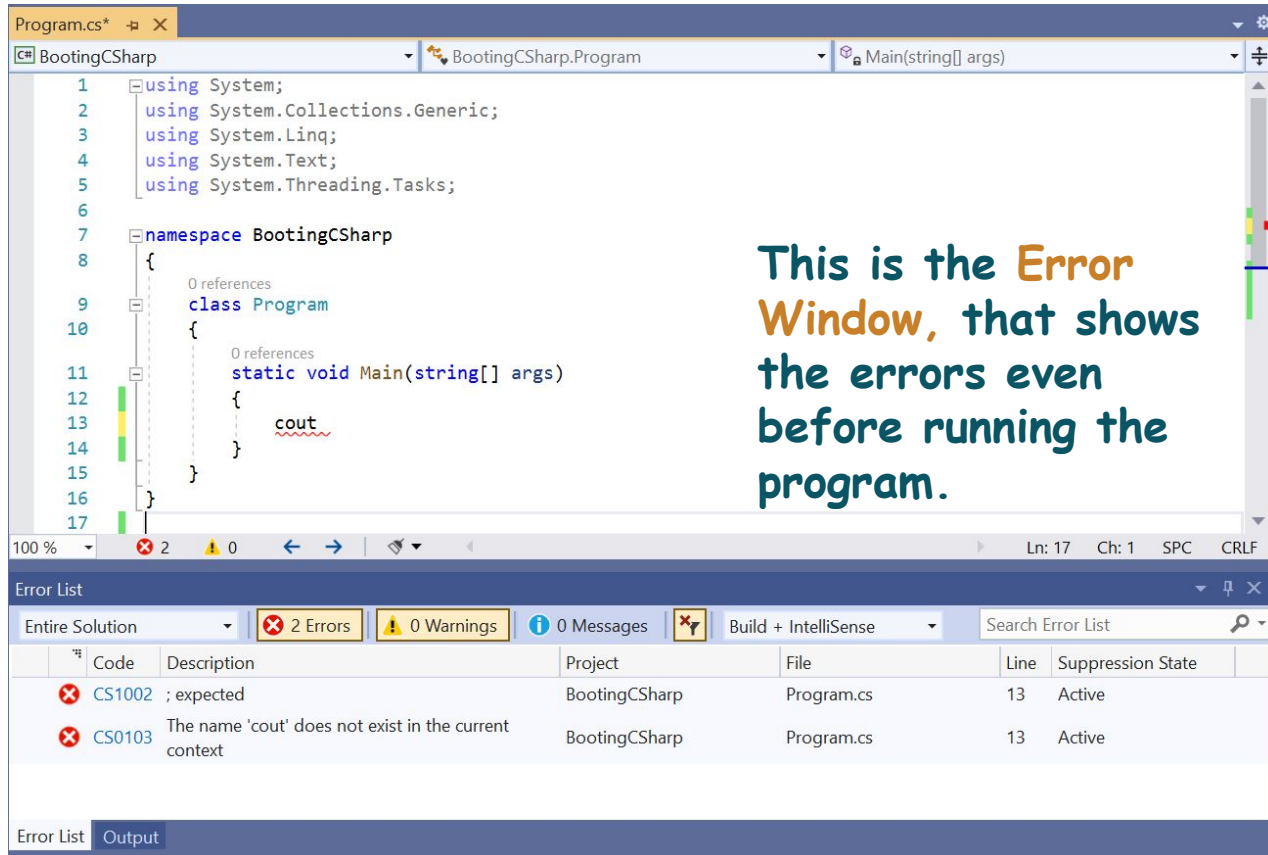
	Code	Description	Project	File	Line	Suppression State
	CS1002	; expected	BootingCSharp	Program.cs	13	Active
	CS0103	The name 'cout' does not exist in the current context	BootingCSharp	Program.cs	13	Active

Error List Output



# Visual Studio: Error Window

This is the **Error Window**, that shows the errors even before running the program.



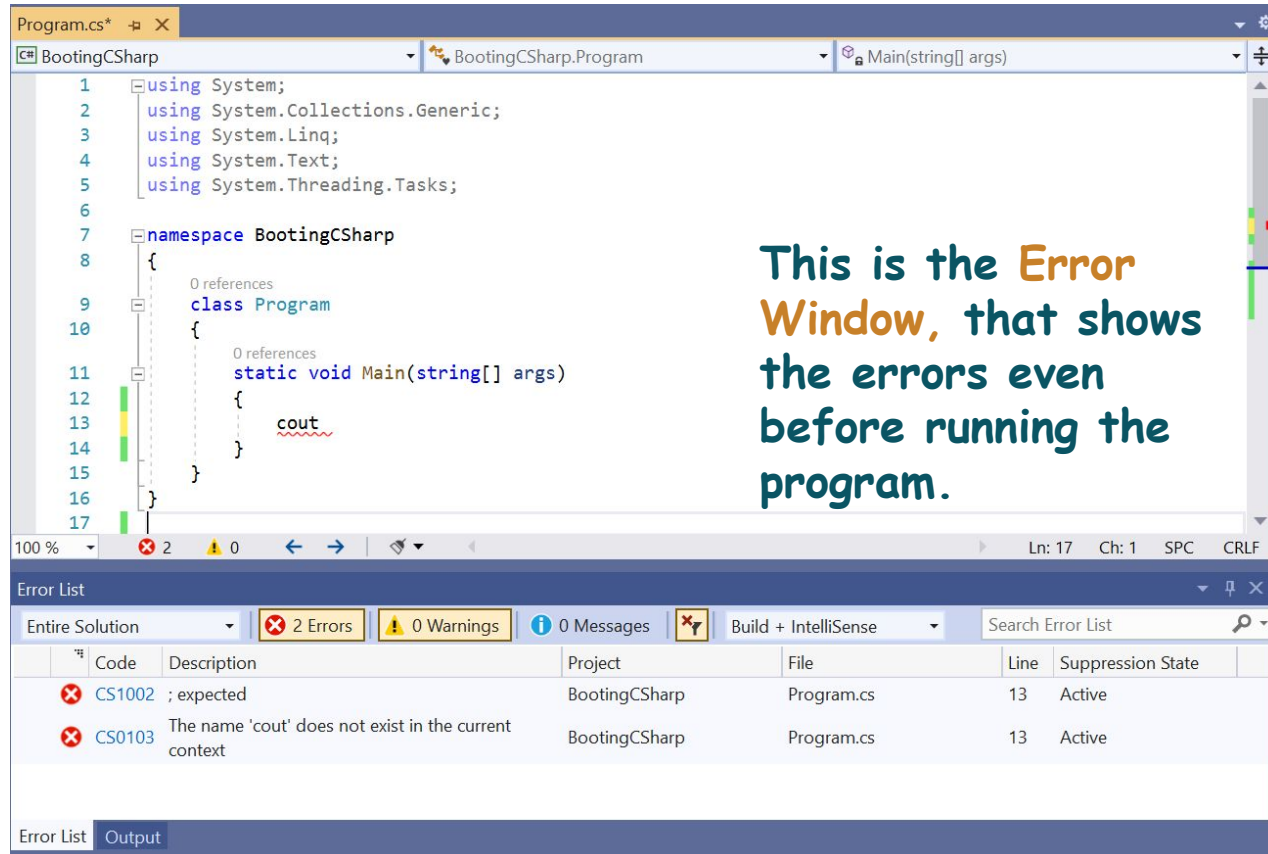
The screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code defines a namespace BootingCSharp and a class Program with a static void Main method. The Main method contains a line of code: `cout`. The Error List window at the bottom shows two errors:

Code	Description	Project	File	Line	Suppression State
CS1002	; expected	BootingCSharp	Program.cs	13	Active
CS0103	The name 'cout' does not exist in the current context	BootingCSharp	Program.cs	13	Active

A red arrow points from the left towards the Error List window.

# Visual Studio: Error Window

This is the **Error Window**, that shows the errors even before running the program.



The screenshot shows the Visual Studio IDE with a C# file named Program.cs. The code defines a namespace BootingCSharp and a class Program with a static void Main method. A red squiggly line under the word 'cout' on line 13 indicates an error. The Error List window at the bottom shows two errors: CS1002 ('; expected') and CS0103 ('The name 'cout' does not exist in the current context'). A red arrow points from the left towards the Error List window.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace BootingCSharp
8 {
9     class Program
10     {
11         static void Main(string[] args)
12         {
13             cout
14         }
15     }
16 }
17
```

100 % 2 0 Ln: 17 Ch: 1 SPC CRLF

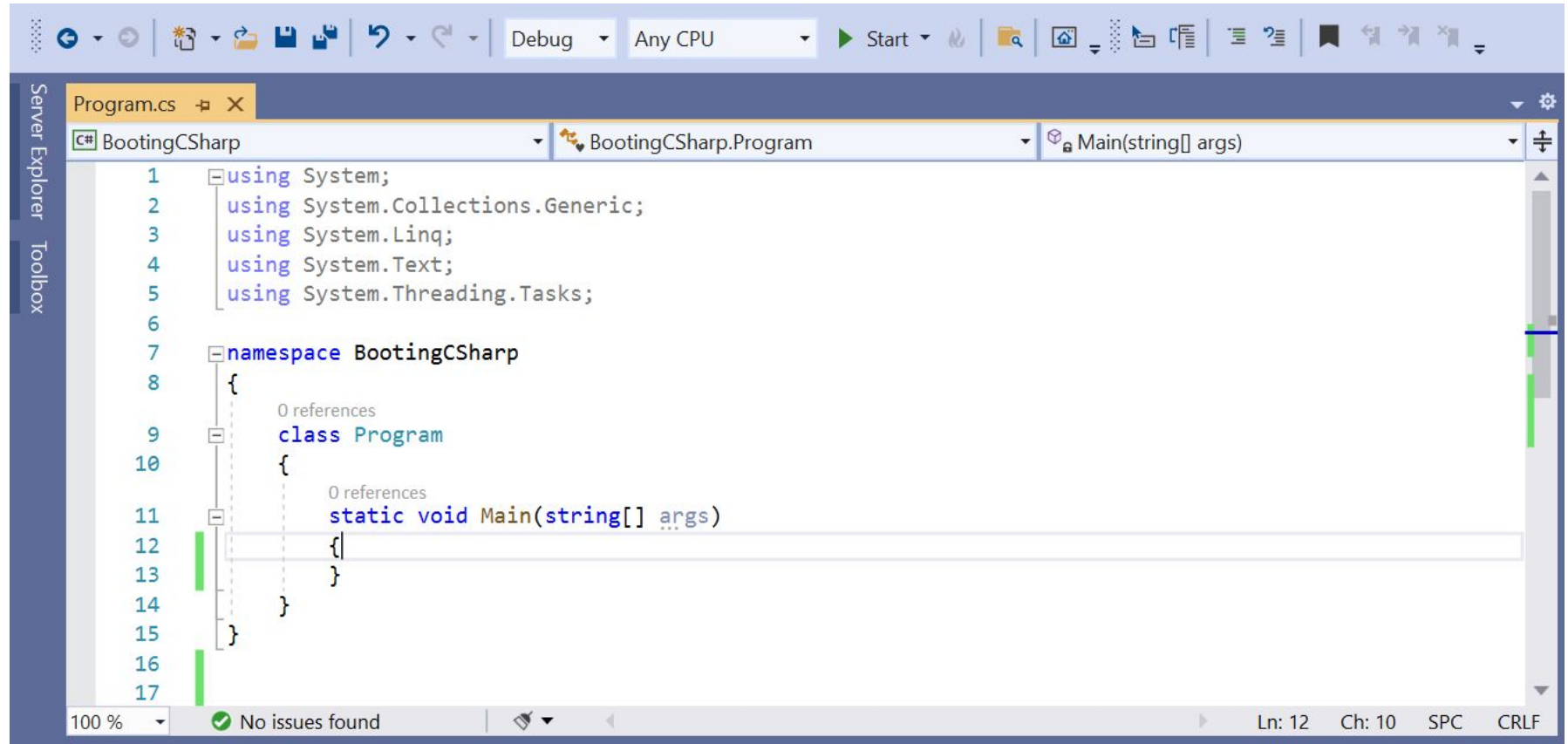
Error List

Entire Solution 2 Errors 0 Warnings 0 Messages Build + IntelliSense Search Error List

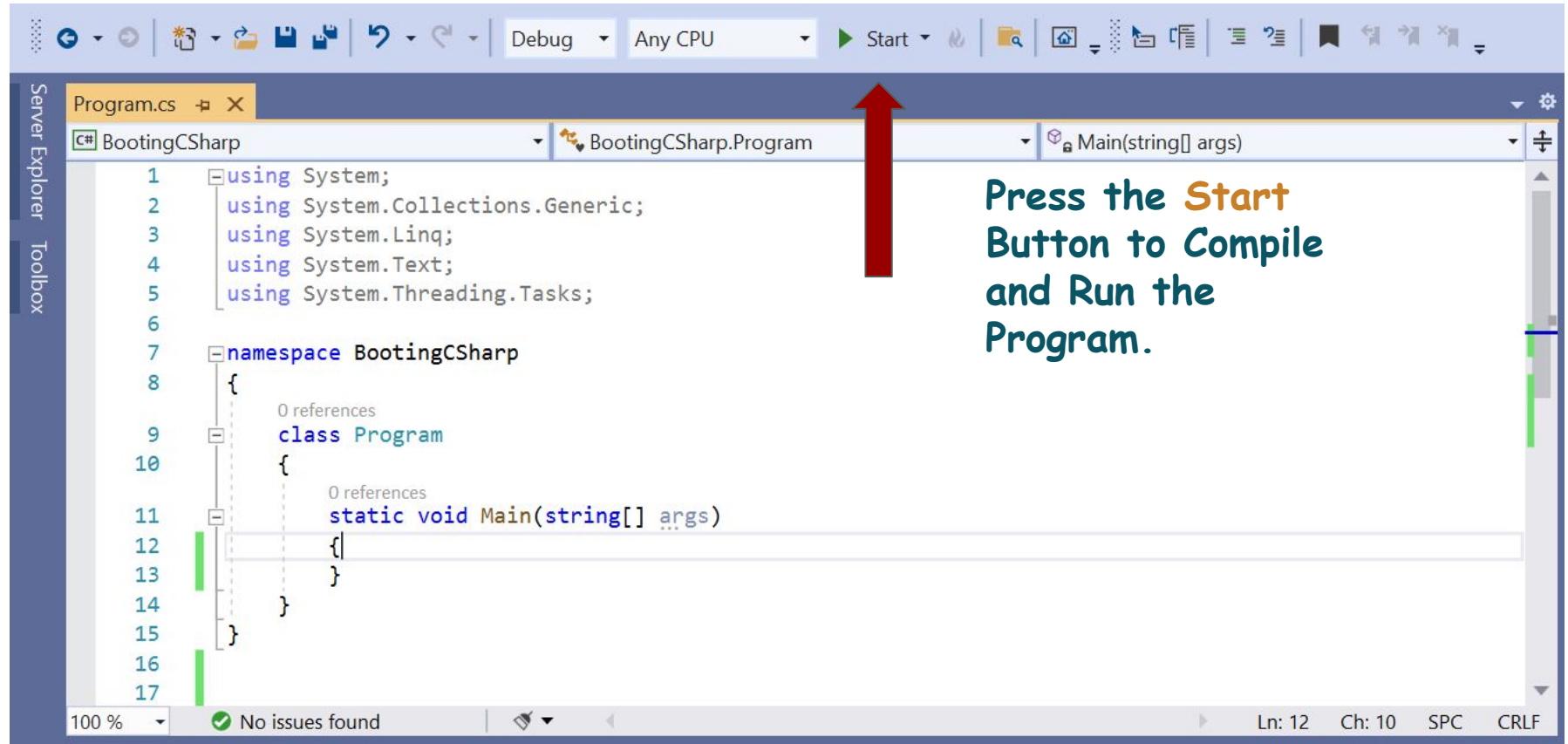
	Code	Description	Project	File	Line	Suppression State
	CS1002	; expected	BootingCSharp	Program.cs	13	Active
	CS0103	The name 'cout' does not exist in the current context	BootingCSharp	Program.cs	13	Active

Error List Output

# Visual Studio: Compile and Run



# Visual Studio: Compile and Run



The screenshot shows the Visual Studio IDE with a C# program open. The file explorer on the left shows 'Program.cs'. The code editor displays the following code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace BootingCSharp
8 {
9     0 references
10     class Program
11     {
12         0 references
13         static void Main(string[] args)
14         {
15         }
16     }
17 }
```

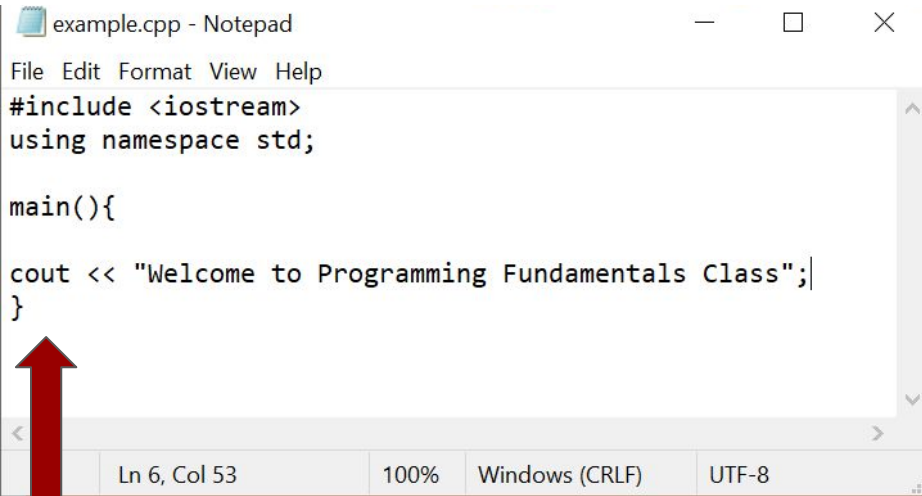
A red arrow points to the 'Start' button in the top toolbar, which is labeled 'Start' and has a green play icon. The status bar at the bottom indicates '100 %' zoom, 'No issues found', and the current line is 'Ln: 12'.

Press the **Start** Button to Compile and Run the Program.

# Displaying Output in C#



# Displaying Output on Console



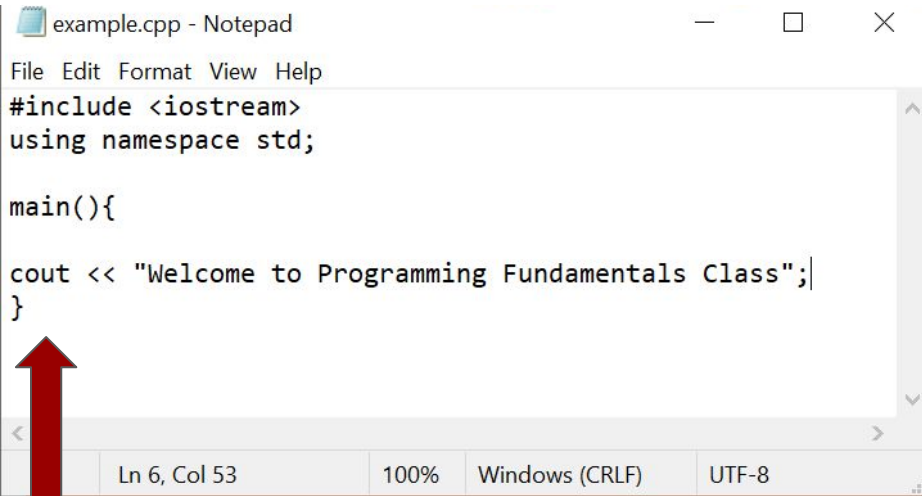
```
example.cpp - Notepad
File Edit Format View Help
#include <iostream>
using namespace std;

main(){
cout << "Welcome to Programming Fundamentals Class";
}

Ln 6, Col 53 100% Windows (CRLF) UTF-8
```

In C++, we used **cout** to display the output on the console.

# Displaying Output on Console

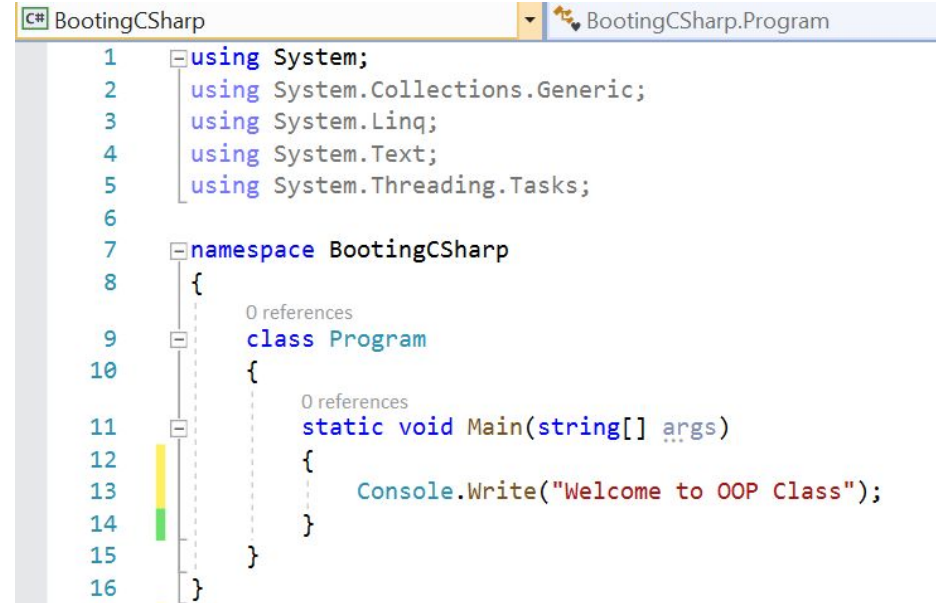


```
example.cpp - Notepad
File Edit Format View Help
#include <iostream>
using namespace std;

main(){
cout << "Welcome to Programming Fundamentals Class";
}
```

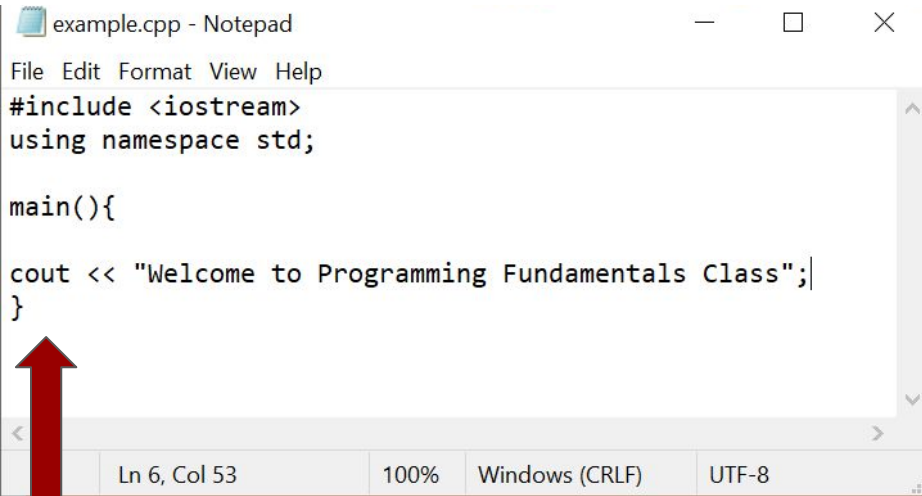
Ln 6, Col 53    100%    Windows (CRLF)    UTF-8

In C++, we used **cout** to display the output on the console.



```
BootingCSharp    BootingCSharp.Program
1    using System;
2    using System.Collections.Generic;
3    using System.Linq;
4    using System.Text;
5    using System.Threading.Tasks;
6
7    namespace BootingCSharp
8    {
9       0 references
10       class Program
11       {
12           0 references
13           static void Main(string[] args)
14           {
15               Console.WriteLine("Welcome to OOP Class");
16           }
17       }
```

# Displaying Output on Console

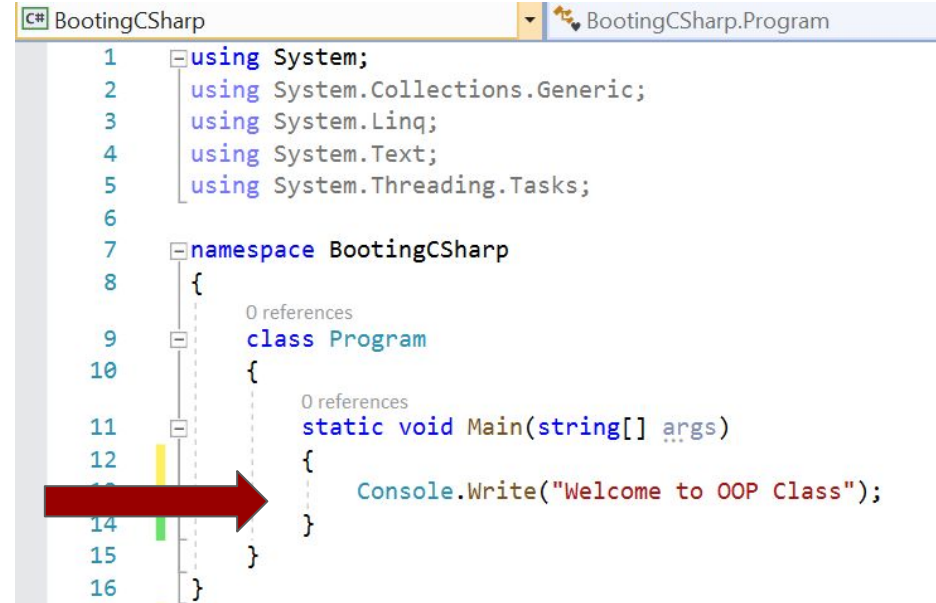


```
example.cpp - Notepad
File Edit Format View Help
#include <iostream>
using namespace std;

main(){
cout << "Welcome to Programming Fundamentals Class";
}
```

Ln 6, Col 53    100%    Windows (CRLF)    UTF-8

In C++, we used **cout** to display the output on the console.



```
BootlingCSharp    BootlingCSharp.Program
1    using System;
2    using System.Collections.Generic;
3    using System.Linq;
4    using System.Text;
5    using System.Threading.Tasks;
6
7    namespace BootlingCSharp
8    {
9       0 references
10       class Program
11       {
12           0 references
13           static void Main(string[] args)
14           {
15               Console.WriteLine("Welcome to OOP Class");
16           }
17       }
```

In C#, we use **Console.WriteLine(" ")** to display the output on the console.



# Displaying Output on Console

```
C# BootingCSharp BootingCSharp.Program
1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Text;
5      using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      0 references
10     class Program
11     {
12         0 references
13         static void Main(string[] args)
14         {
15             Console.WriteLine("Welcome to OOP Class");
16         }
17     }
18 }
```

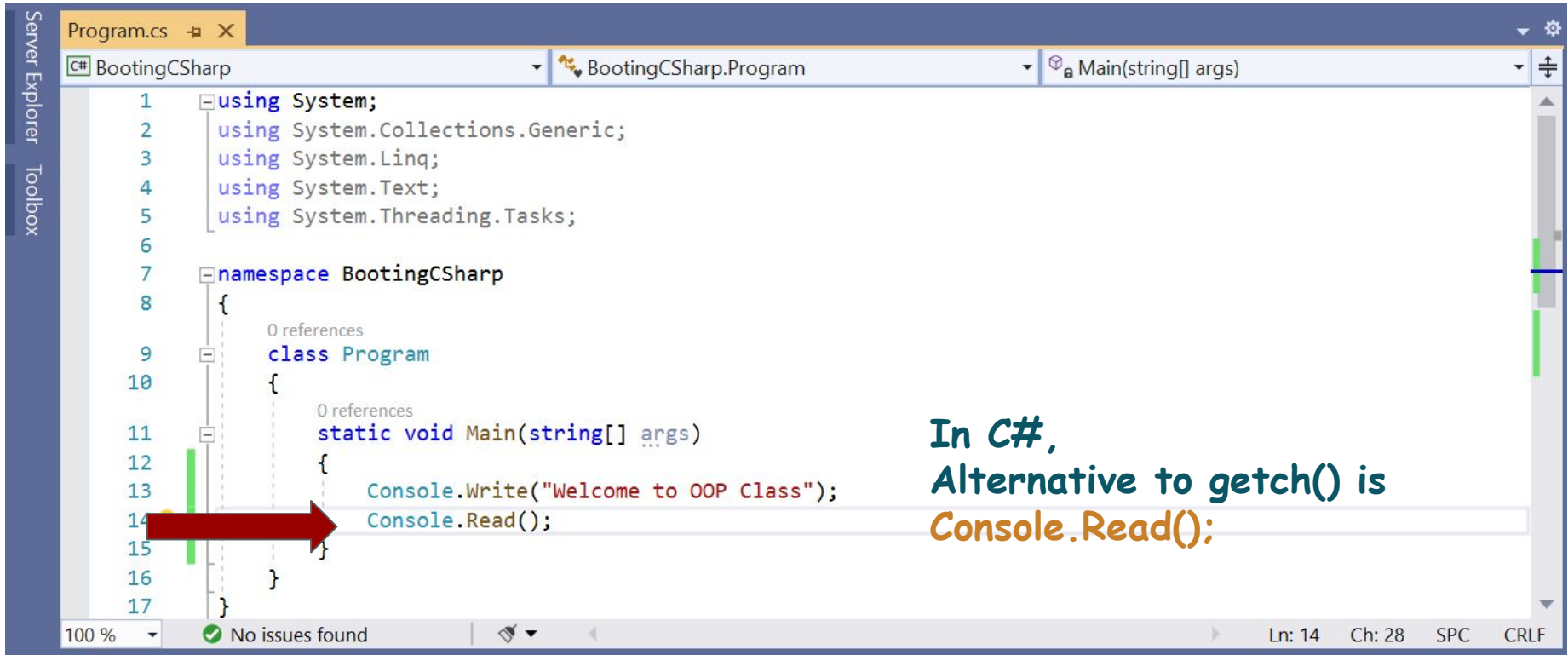
This code will run and terminate without **stopping** on the console.

# Displaying Output on Console

```
C# BootingCSharp BootingCSharp.Program
1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Text;
5      using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      0 references
10     class Program
11     {
12         0 references
13         static void Main(string[] args)
14         {
15             Console.WriteLine("Welcome to OOP Class");
16         }
17     }
18 }
```

This code will run and terminate without **stopping** on the console.

# Stopping at the Console



The screenshot shows a Visual Studio IDE with a C# project named 'BootigCSharp'. The code in 'Program.cs' is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace BootigCSharp
8 {
9     0 references
9     class Program
10    {
11        0 references
11        static void Main(string[] args)
12        {
13            Console.WriteLine("Welcome to OOP Class");
14            Console.ReadLine();
15        }
16    }
17 }
```

A red arrow points to the `Console.ReadLine();` line (line 14). To the right of the code, an annotation states:

In C#,  
Alternative to getch() is  
**Console.ReadLine();**

The status bar at the bottom indicates '100 %', 'No issues found', and line information 'Ln: 14 Ch: 28 SPC CRLF'.

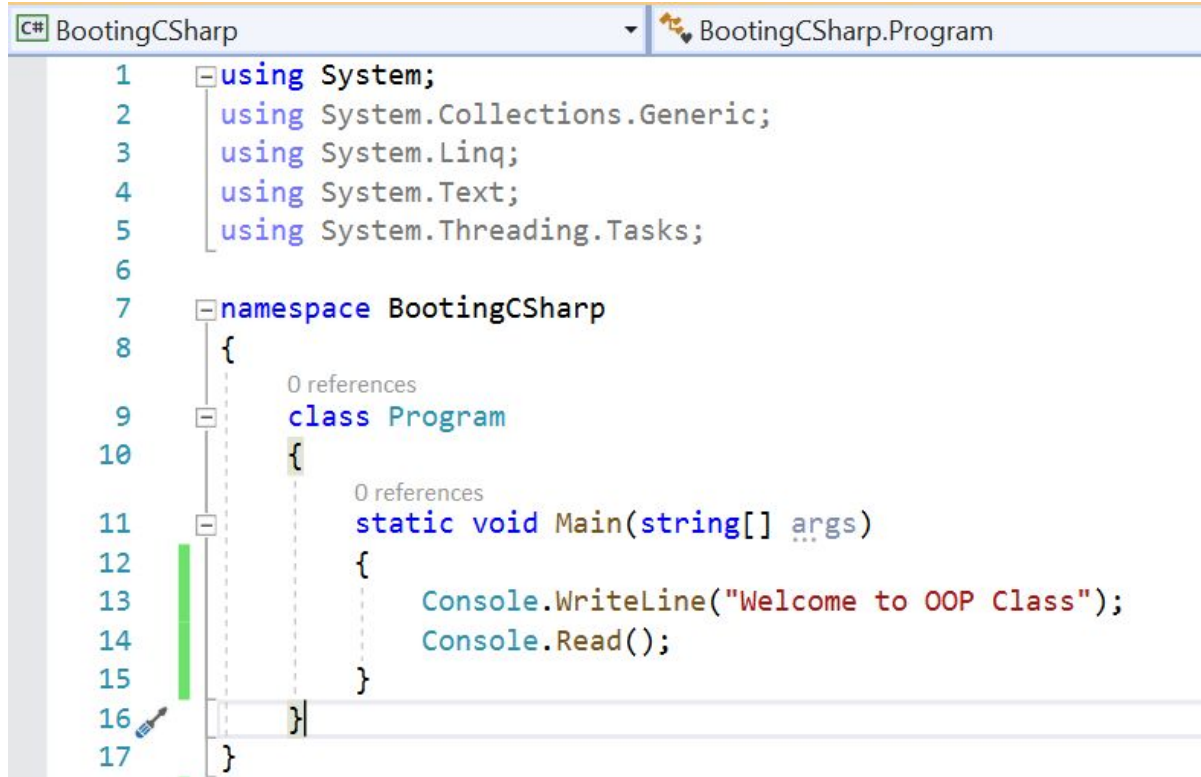
# Output on the Console



A screenshot of a Windows console window. The title bar at the top shows the file path "G:\OOP 2022\BootingCSharp\BootingCSharp\bin\Debug\BootingCSharp.exe" and standard window controls (minimize, maximize, close). The main area of the console is light gray and contains the text "Welcome to OOP Class" in a black, monospaced font. A vertical scrollbar is visible on the right side of the console area.

```
G:\OOP 2022\BootingCSharp\BootingCSharp\bin\Debug\BootingCSharp.exe
Welcome to OOP Class
```

# Displaying Output on Console with endl



```
C# BootingCSharp
BootnCSharp.Program

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Welcome to OOP Class");
14             Console.Read();
15         }
16     }
17 }
```

The image shows a C# code editor window titled 'C# BootingCSharp'. The file 'BootnCSharp.Program' is open. The code is as follows:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Welcome to OOP Class");
14             Console.Read();
15         }
16     }
17 }
```

The code uses `Console.WriteLine` to output the string "Welcome to OOP Class" to the console. The `Console.Read()` method is used to keep the console window open until the user presses a key.

# Displaying Output on Console with endl

```
C# BootingCSharp BootingCSharp.Program
1  using System;
2      using System.Collections.Generic;
3      using System.Linq;
4      using System.Text;
5      using System.Threading.Tasks;
6
7  namespace BootingCSharp
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Welcome to OOP Class");
14             Console.Read();
15         }
16     }
17 }
```

In C#,  
Alternative to cout with endl is  
**Console.WriteLine(" ");**



# Declaring Variables in C#



# Declaring Variables: **int** datatype

```
c# BootingCSharp BootingCS
1  + using ...
6
7  - namespace BootingCSharp
8      {
9      - 0 references
10     - class Program
11         {
12         - 0 references
13         - static void Main(string[] args)
14             {
15                 int number = 7;
16                 Console.Write("Number: ");
17                 Console.WriteLine(number);
18                 Console.Read();
19             }
20         }
21     }
```

Same as in C++



# Declaring Variables: **string** datatype

```
C# BootingCSharp BootingCSharp
1  + using ...
6
7  namespace BootingCSharp
8  {
9      0 references
10     class Program
11     {
12         0 references
13         static void Main(string[] args)
14         {
15             string word = "Hello";
16             Console.Write("Word: ");
17             Console.WriteLine(word);
18             Console.Read();
19         }
20     }
21 }
```

Same as in C++


# Declaring Variables: **char** datatype

```
C# BootingCSharp
1  + using ...
6
7  namespace BootingCSharp
8  {
9      0 references
10     class Program
11     {
12         0 references
13         static void Main(string[] args)
14         {
15             char letter = 'A';
16             Console.Write("Letter: ");
17             Console.WriteLine(letter);
18             Console.Read();
19         }
20     }
21 }
```

Same as in C++

# Declaring Variables: float datatype

```
C# BootingCSharp BootingCS
1  + using ...
6
7  - namespace BootingCSharp
8  {
9      0 references
10     - class Program
11     {
12         0 references
13         - static void Main(string[] args)
14         {
15             float num = 6.5F;
16             Console.WriteLine("Decimal: ");
17             Console.WriteLine(num);
18             Console.ReadLine();
19         }
20     }
21 }
```



Only difference is that you have to write **F** at the end of the number to explicitly mention that it is a **floating point number**



# Taking Input in C#



# Taking Input

```
C# BootingCSharp BootingCSharp.F
1  + using ...
6
7  - namespace BootingCSharp
8  {
9      0 references
10     - class Program
11     {
12         0 references
13         - static void Main(string[] args)
14         {
15             string input;
16             Console.Write("Enter a Name: ");
17             input = Console.ReadLine();
18             Console.WriteLine(input);
19             Console.Read();
20         }
21     }
22 }
```

Alternative to cin is  
**Console.ReadLine();**

# Taking Input

```
C# BootingCSharp BootingCSharp.F
1  + using ...
6
7  - namespace BootingCSharp
8  {
9      0 references
      - class Program
10     {
11         0 references
        - static void Main(string[] args)
12         {
13             string input;
14             Console.WriteLine("Enter a Name: ");
15             input = Console.ReadLine();
16             Console.WriteLine(input);
17             Console.Read();
18         }
19     }
20 }
```

Input is always in a **string**. Therefore if we want to take a **number as input** then we have to do **typecasting**.

# Taking Input an **int** Number

```
C# BootingCSharp BootingCSharp.Program
1  + using ...
6
7  namespace BootingCSharp
8  {
9      0 references
      class Program
10     {
11         0 references
        static void Main(string[] args)
12         {
13             string input;
14             int num;
15             Console.Write("Enter a Number: ");
16             input = Console.ReadLine();
17             num = int.Parse(input);
18             Console.WriteLine(num);
19             Console.Read();
20         }
21     }
22 }
```

TypeCasting,  
Converting **string** into  
integer.



# Taking Input a float Number

```
C# BootingCSharp BootingCSharp.Program
1  + using ...
6
7  - namespace BootingCSharp
8  {
9      0 references
      - class Program
10     {
11         0 references
        - static void Main(string[] args)
12         {
13             string input;
14             float num;
15             Console.Write("Enter a Number: ");
16             input = Console.ReadLine();
17             num = float.Parse(input);
18             Console.WriteLine(num);
19             Console.Read();
20         }
21     }
22 }
```

TypeCasting,  
Converting string into  
float.





# Working Example: Vision

Write a program that takes **length of side** of a square as input and calculate its area using following formula:  
$$\text{Area} = \text{Length} * \text{Length}$$



```
G:\OOP 2022\BootingCSharp\BootingCSharp\bin\Debug\BootingCSharp.exe
Enter the Length: 5
Area is:
25
```



# Working Example: Solution

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        string input;
        float length;
        float area;
        Console.Write("Enter the Length: ");
        input = Console.ReadLine();
        length = float.Parse(input);
        area = length * length;
        Console.WriteLine("Area is: ");
        Console.WriteLine(area);
        Console.Read();
    }
}
```



# Conditional Statements, Comparison and Logical Operators in C#



# Working Example: Vision

We want to print “You are Passed” when Marks are greater than 50 but print “You are Failed” when Marks are less than or equal to 50.



# Working Example: Solution

```
static void Main(string[] args)
{
    string input;
    float marks;
    Console.WriteLine("Enter the Marks: ");
    input = Console.ReadLine();
    marks = float.Parse(input);
    if (marks > 50)
    {
        Console.WriteLine("You are Passed");
    }
    else
    {
        Console.WriteLine("You are Failed");
    }
    Console.Read();
}
```

Same as in C++





# Loops in C#



# Loops in C#

We have two type of loops

- Counter Loops
- Conditional Loops

## Counter Loop

Count the steps and execute until specific number of times.

## Condition Loop

Execute the steps until a specific condition is not met.

# Working Example: Vision

We want to print "Welcome Jack" 5 times.





# For Loop: Solution

```
static void Main(string[] args)
{
    for (int x = 0; x < 5; x++)
    {
        Console.WriteLine("Welcome Jack");
    }
    Console.Read();
}
```

Same as in C++



# Working Example: Vision

Write a program that keeps taking input from the user and sum up all these input until he enters **-1**. When user enters **-1** the program should print sum of all values.



# While Loop: Solution

```
static void Main(string[] args)
{
    int num;
    int sum = 0;
    Console.Write("Enter Number: ");
    num = int.Parse(Console.ReadLine());
    while (num != -1)
    {
        sum = sum + num;
        Console.Write("Enter Number: ");
        num = int.Parse(Console.ReadLine());
    }
    Console.WriteLine("The total sum is {0}", sum);
    Console.Read();
}
```

Same as in C++



# Do While Loop: Solution

```
static void Main(string[] args)
{
    int num;
    int sum = 0;
    do
    {
        Console.Write("Enter Number: ");
        num = int.Parse(Console.ReadLine());
        sum = sum + num;
    }
    while (num != -1);
    sum = sum + 1;
    Console.WriteLine("The total sum is {0}", sum);
    Console.Read();
}
```

Same as in C++



# Working Example: Vision

Write a program that takes 3 numbers as input and print the largest of them.



# Solution with Variables

Why do not we declare multiple variables and use a for loop ?

```
int fnum, snum, tnum;
Console.Write("Enter First Numbers:");
fnum = int.Parse(Console.ReadLine());
Console.Write("Enter Secon Numbers:");
snum = int.Parse(Console.ReadLine());
Console.Write("Enter Third Numbers:");
tnum = int.Parse(Console.ReadLine());
if (fnum > snum && fnum > tnum){
    Console.WriteLine("First Number is Greater");
}
if (snum > fnum && snum > tnum) {
    Console.WriteLine("Second Number is Greater");
}
if (tnum > fnum && tnum > snum) {
    Console.WriteLine("Third Number is Greater");
}
```



# Solution with Variables

It would be easier if we have same variable name for same type of data and if it require to access any data we just give its number

```
int fnum, snum, tnum;
Console.Write("Enter First Numbers:");
fnum = int.Parse(Console.ReadLine());
Console.Write("Enter Secon Numbers:");
snum = int.Parse(Console.ReadLine());
Console.Write("Enter Third Numbers:");
tnum = int.Parse(Console.ReadLine());
if (fnum > snum && fnum > tnum){
    Console.WriteLine("First Number is Greater");
}
if (snum > fnum && snum > tnum) {
    Console.WriteLine("Second Number is Greater");
}
if (tnum > fnum && tnum > snum) {
    Console.WriteLine("Third Number is Greater");
}
```







# Arrays in C#





# Arrays

You can access variables through **same name** with different index value [0]

```
int[] value = new int[5];
```



# Solution with Arrays

```
static void Main(string[] args)
{
    //Taking input
    int[] numbers=new int[3];
    for (int idx = 0; idx < 3; idx++) {
        Console.Write("Enter the Number {0}:",idx);
        numbers[idx] = int.Parse(Console.ReadLine());
    }
}
```



```
int fnum, snum, tnum;
Console.Write("Enter First Numbers:");
fnum = int.Parse(Console.ReadLine());
Console.Write("Enter Secon Numbers:");
snum = int.Parse(Console.ReadLine());
Console.Write("Enter Third Numbers:");
tnum = int.Parse(Console.ReadLine());
if (fnum > snum && fnum > tnum){
    Console.WriteLine("First Number is Greater");
}
if (snum > fnum && snum > tnum) {
    Console.WriteLine("Second Number is Greater");
}
if (tnum > fnum && tnum > snum) {
    Console.WriteLine("Third Number is Greater");
}
```

# Solution with Arrays

```
static void Main(string[] args)
{
    //Taking input
    int[] numbers=new int[3];
    for (int idx = 0; idx < 3; idx++) {
        Console.Write("Enter the Number {0}:",idx);
        numbers[idx] = int.Parse(Console.ReadLine());
    }
    //Finding the Largest
    int largest = -1;
    for (int idx = 0; idx < 3; idx++) {
        if (numbers[idx] > largest) {
            largest = numbers[idx];
        }
    }
    Console.WriteLine("Largest is: {0}", largest);
    Console.Read();
}
```



```
int fnum, snum, tnum;
Console.Write("Enter First Numbers:");
fnum = int.Parse(Console.ReadLine());
Console.Write("Enter Secon Numbers:");
snum = int.Parse(Console.ReadLine());
Console.Write("Enter Third Numbers:");
tnum = int.Parse(Console.ReadLine());
if (fnum > snum && fnum > tnum){
    Console.WriteLine("First Number is Greater");
}
if (snum > fnum && snum > tnum) {
    Console.WriteLine("Second Number is Greater");
}
if (tnum > fnum && tnum > snum) {
    Console.WriteLine("Third Number is Greater");
}
```

# Learning Objective

Know the programming constructs of **C#** and Solve Complex problems in **C#** language.



# Self Assessment

1. Lilly is  $N$  years old. For each birthday she receives a present. For each odd birthday (1, 3, 5, ...,  $n$ ) she receives toys for each even birthday (2, 4, 6, ...,  $n$ ) she receives money. For her second birthday she received 10.00 USD, and the amount is increased by 10.00 USD for each following even birthday (2 -> 10, 4 -> 20, 6 -> 30 etc.).

Over the years Lilly has secretly saved her money. Lilly's brother, in the years when she received money, took 1.00 USD from each of the amounts.

Lilly has sold the toys, received over the years, each one for  $P$  USD and added the sum to the amount of saved money.

With the money she wanted to buy a washing machine for  $X$  USD. Write a Function that takes lily's age, price of washing machine and price of toy and calculates how much money she has saved and if it is enough to buy a washing machine.



# Self Assessment

## Input:

We read from the console 3 numbers, each on a separate line:

1. Lilly's age - integer in the range of [1 ... 77].
2. Price of the washing machine - number in the range of [1.00 ... 10,000.00].
3. Unit price of each toy - integer in the range of [0 ... 40].

## Output

Print on the console one single line:

1. If Lilly's money is enough:  
"Yes! {N}" - where N is the remaining money after the purchase
2. If the money is not enough:  
"No! {M}" - where M is the insufficiency amount



# Self Assessment

Input	Output	Comments
10 170.00 6	Yes! 5.00	<p>For the first birthday she gets a toy; 2nd -&gt; 10 USD; 3rd -&gt; toy; 4th -&gt; 10 + 10 = 20 USD; 5th -&gt; toy; 6th -&gt; 20 + 10 = 30 USD; 7th -&gt; toy; 8th -&gt; 30 + 10 = 40 USD; 9th -&gt; toy; 10th -&gt; 40 + 10 = 50 USD. She has saved: 10 + 20 + 30 + 40 + 50 = 150 USD. She sold 5 toys for 6 USD each = 30 USD. Her brother took 1 USD 5 times = 5 USD. Remaining amount: 150 + 30 - 5 = 175 USD. 175 &gt;= 170 (price of the washing machine): she managed to buy it and is left with 175-170 = 5 USD.</p>
21 1570.98 3	No! 997.98	<p>She has saved 550 USD. She has sold 11 toys, 3 USD each = 33 USD. Her brother has taken for 10 years 1 USD each year = 10 USD. Remaining amount: 550 + 33 - 10 = 573 USD. 573 &lt; 1570.98: she did not manage to buy a washing machine. The insufficiency amount is: 1570.98 - 573 = 997.98 USD.</p>

