**C# Basic Input and Output**

In this tutorial, we will learn how to take input from user and display output in C# using various methods.

**C# Output**

In order to output something in C#, we can use

System.Console.WriteLine() OR

System.Console.Write()

Here, System is a [namespace](https://www.programiz.com/csharp-programming/namespaces), Console is a class within namespace System and WriteLine and Write are methods of class Console.

Let's look at a simple example that prints a string to output screen.

**Example 1: Printing String using WriteLine()**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

Console.WriteLine("C# is cool");

}

}

}

When we run the program, the output will be

C# is cool

**Difference between WriteLine() and Write() method**

The main difference between WriteLine() and Write() is that the Write() method only prints the string provided to it, while the WriteLine() method prints the string and moves to the start of next line as well.

Let's take a look at the following examples to understand the difference between these methods.

**Example 2: How to use WriteLine() and Write() method?**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

Console.WriteLine("Prints on ");

Console.WriteLine("New line");

Console.Write("Prints on ");

Console.Write("Same line");

}

}

}

When we run the program, the output will be

Prints on

New line

Prints on Same line

**Printing Variables and Literals using WriteLine() and Write()**

The WriteLine() and Write() method can be used to print variables and literals. Here's an example.

**Example 3: Printing Variables and Literals**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

int value = 10;

// Variable

Console.WriteLine(value);

// Literal

Console.WriteLine(50.05);

}

}

}

When we run the program, the output will be

10

50.05

**Combining (Concatenating) two strings using + operator and printing them**

Strings can be combined/concatenated using the + operator while printing.

**Example 4: Printing Concatenated String using + operator**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

int val = 55;

Console.WriteLine("Hello " + "World");

Console.WriteLine("Value = " + val);

}

}

}

When we run the program, the output will be

Hello World

Value = 55

**Printing concatenated string using Formatted String [Better Alternative]**

A better alternative for printing concatenated string is using formatted string. Formatted string allows programmer to use placeholders for variables. For example, the following line,

Console.WriteLine("Value = " + val);

can be replaced by,

Console.WriteLine("Value = {0}", val);

{0} is the placeholder for variable val which will be replaced by value of val. Since only one variable is used so there is only one placeholder.

Multiple variables can be used in the formatted string. We will see that in the example below.

**Example 5: Printing Concatenated string using String formatting**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

int firstNumber = 5, secondNumber = 10, result;

result = firstNumber + secondNumber;

Console.WriteLine("{0} + {1} = {2}", firstNumber, secondNumber, result);

}

}

}

When we run the program, the output will be

5 + 10 = 15

Here, {0} is replaced by firstNumber, {1} is replaced by secondNumber and {2} is replaced by result. This approach of printing output is more readable and less error prone than using + operator. To know more about string formatting, visit *C# string formatting*.

**C# Input**

In C#, the simplest method to get input from the user is by using the ReadLine() method of the Console class. However, Read() and ReadKey() are also available for getting input from the user. They are also included in Console class.

**Example 6: Get String Input From User**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

string testString;

Console.Write("Enter a string - ");

testString = Console.ReadLine();

Console.WriteLine("You entered '{0}'", testString);

}

}

}

When we run the program, the output will be:

Enter a string - Hello World

You entered 'Hello World'

**Difference between ReadLine(), Read() and ReadKey() method:**

The difference between ReadLine(), Read() and ReadKey() method is:

* ReadLine(): The ReadLine() method reads the next line of input from the standard input stream. It returns the same string.
* Read(): The Read() method reads the next character from the standard input stream. It returns the ascii value of the character.
* ReadKey(): The ReadKey() method obtains the next key pressed by user. This method is usually used to hold the screen until user press a key.

If you want to know more about these methods, here is an interesting discussion on StackOverflow on: [Difference between Console.Read() and Console.ReadLine()?](https://stackoverflow.com/questions/6825943/difference-between-console-read-and-console-readline).

**Example 7: Difference between Read() and ReadKey() method**

using System;

namespace Sample

{

class Test

{

public static void Main(string[] args)

{

int userInput;

Console.WriteLine("Press any key to continue...");

Console.ReadKey();

Console.WriteLine();

Console.Write("Input using Read() - ");

userInput = Console.Read();

Console.WriteLine("Ascii Value = {0}",userInput);

}

}

}

When we run the program, the output will be

Press any key to continue...

x

Input using Read() - Learning C#

Ascii Value = 76

From this example, it must be clear how ReadKey() and Read() method works. While using ReadKey(), as soon as the key is pressed, it is displayed on the screen.

When Read() is used, it takes a whole line but only returns the ASCII value of first character. Hence, 76 (ASCII value of L) is printed.

**Reading numeric values (integer and floating point types)**

Reading a character or string is very simple in C#. All you need to do is call the corresponding methods as required.

But, reading numeric values can be slightly tricky in C#. We’ll still use the same ReadLine() method we used for getting string values. But since the ReadLine() method receives the input as string, it needs to be converted into integer or floating point type.

One simple approach for converting our input is using the methods of Convert class.

**Example 8: Reading Numeric Values from User using Convert class**

using System;

namespace UserInput

{

class MyClass

{

public static void Main(string[] args)

{

string userInput;

int intVal;

double doubleVal;

Console.Write("Enter integer value: ");

userInput = Console.ReadLine();

/\* Converts to integer type \*/

intVal = Convert.ToInt32(userInput);

Console.WriteLine("You entered {0}",intVal);

Console.Write("Enter double value: ");

userInput = Console.ReadLine();

/\* Converts to double type \*/

doubleVal = Convert.ToDouble(userInput);

Console.WriteLine("You entered {0}",doubleVal);

}

}

}

When we run the program, the output will be

Enter integer value: 101

You entered 101

Enter double value: 59.412

You entered 59.412

The ToInt32() and ToDouble() method of Convert class converts the string input to integer and double type respectively. Similarly, we can convert the input to other types. Here is a [complete list of available methods for Convert class](https://msdn.microsoft.com/en-us/library/system.convert(v=vs.110).aspx).

There are other ways to get numeric inputs from user. To learn more, visit [Reading an integer from user input](https://stackoverflow.com/questions/24443827/reading-an-integer-from-user-input).