***Lab 03 – Formatted output***

**WriteLine formatting in C#**

WriteLine is used to output a line of string in C#. It is often required to output a integer, string or other variable in a certain way. We need to use formatting in such cases.
The format parameter in formatting is embedded with zero or more format specifications of the form

**"{N [, M ][: formatString ]}", arg1, ... argN,**

**{index[,alignment][:formatString]}**

where:

* N is a zero-based integer indicating the argument to be formatted.
* M is an optional integer indicating the width of the region to contain the formatted value, padded with spaces. If M is negative, the formatted value is left-justified; if M is positive, the value is right-justified.
* formatString is an optional string of formatting codes.
* argN is the expression to use at the equivalent position inside the quotes in the string.

Consider the following example to understand it

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

 class Program

 {

 static void Main(string[] args)

 {

 Console.WriteLine("{0,5} {1,5}", 123, 456); // Right-aligned

 Console.WriteLine("{0,-5} {1,-5}", 123, 456); // Left-aligned

 Console.WriteLine("{0,-10:D6} {1,-10:D6}", 123, 456); // D6 means 6 decimal digits

 Console.ReadLine();

 }

 }

}

**Format Specifiers**

Standard numeric format strings are used to return strings in commonly used formats. They take the form X0, in which X is the format specifier and 0 is the precision specifier. The format specifier can be one of the nine built-in format characters that define the most commonly used numeric format types, as shown in Table 10-1.

***Table 10-1****- String and WriteLine Format Specifiers*

|  |  |
| --- | --- |
| **Character** | **Interpretation** |
| C or c | Currency |
| D or d | Decimal (decimal integer�don�t confuse with the .NET Decimal type) |
| E or e | Exponent |
| F or f | Fixed point |
| G or g | General |
| N or n | Currency |
| P or p | Percentage |
| R or r | Round-trip (for floating-point values only); guarantees that a numeric value converted to a string will be parsed back into the same numeric value |
| X or x | Hex |

|  |
| --- |
| using System;using System.Collections.Generic;using System.Linq;using System.Text;namespace ConsoleApplication1{ class Program { static void Main(string[] args) {  int i = 123456; Console.WriteLine("{0:C}", i); // $123,456.00 Console.WriteLine("{0:D}", i); // 123456 Console.WriteLine("{0:E}", i); // 1.234560E+005 Console.WriteLine("{0:F}", i); // 123456.00 Console.WriteLine("{0:G}", i); // 123456 Console.WriteLine("{0:N}", i); // 123,456.00 Console.WriteLine("{0:P}", i); // 12,345,600.00 % Console.WriteLine("{0:X}", i); // 1E240 Console.ReadLine(); } }} |

If you run above you should see the following command

|  |
| --- |
| $123,456.001234561.234560E+005123456.00123456123,456.0012,345,600.00 %1E240 |

To format output in C#, let us see examples to format date and double type.

Set formatted output for Double type.

using System;

class Demo {

   public static void Main(String[] args) {

      Console.WriteLine("Three decimal places...");

      Console.WriteLine(String.Format("{0:0.000}", 987.383));

      Console.WriteLine(String.Format("{0:0.000}", 987.38));

      Console.WriteLine(String.Format("{0:0.000}", 987.7899));

      Console.WriteLine("Thousands Separator...");

      Console.WriteLine(String.Format("{0:0,0.0}", 54567.46));

      Console.WriteLine(String.Format("{0:0,0}", 54567.46));

   }

}

Set formatted output for DateTime

using System;

static class Demo {

   static void Main() {

      DateTime d = new DateTime(2018, 2, 8, 12, 7, 7, 123);

      Console.WriteLine(String.Format("{0:y yy yyy yyyy}", d));

      Console.WriteLine(String.Format("{0:M MM MMM MMMM}", d));

      Console.WriteLine(String.Format("{0:d dd ddd dddd}", d));

   }

}

### 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);

 }

 }

}

**Lab Tasks**

1. Execute all example code and attach output in your lab file.
2. Create a small form which take following input from an employee of any company
	1. Name
	2. Father name
	3. Phone no
	4. Designation
	5. Date of hiring
	6. Salary

 Using above information display user profile in a formatted manner.

1. Create a simple c# program in which you will display the restaurant payment bill in formatted manner

