

PROGRAMMING IN C#

Module 1: Introduction to C# and VS

Module 2: Variables and Data Types

Lab Guide for Lab1

Session Objectives

In this session, you will be practicing with

- Data type in C#
- Input and output data in C#
- Number and datetime format specifier
- Implicitly typed local variables
- Anonymous Types

Part 1 – Getting started (30 minutes)

Exercise 2: Using datatype

Step 1: Add a console based project 'DataType' to the solution

Step 2: Right click on project DataTypes -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'DataTypes.cs'

Step 4: Replace the code in 'DataTypes.cs' with the given code

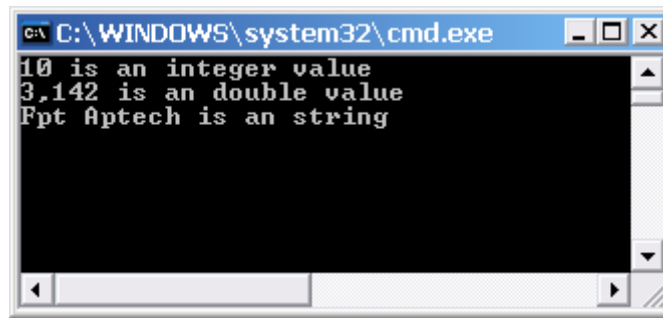
```
using System;
class Example
{
    static void Main(string[] args)
    {
        int intVal;
        double dblVal;
        string strVal;
        intVal = 10;
        dblVal = 3.142;
        strVal = "Fpt Aptech";
        Console.WriteLine("{0} is an integer value", intVal);
        Console.WriteLine("{0} is an double value", dblVal);
        Console.WriteLine("{0} is an string", strVal);
        Console.ReadLine();
    }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'DataTypes' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



Exercise 3: value type

Step 1: Add a console based project 'ValueType' to the solution

Step 2: Right click on project ValueType-> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'ValueType.cs'

Step 4: Replace the code in 'ValueType.cs' with the given code

```
using System;
using System.Text;

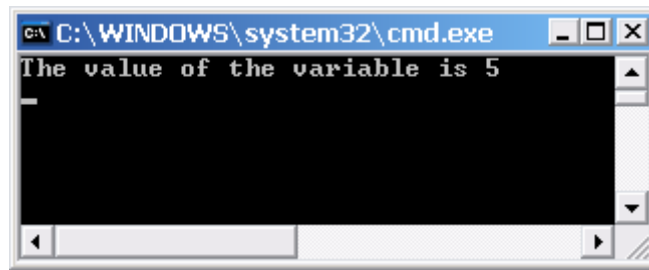
class Example1
{
    static void Main(string[] args)
    {
        int valueVal = 5;
        Test(valueVal);
        Console.WriteLine("The value of the variable is {0}", valueVal);
        Console.ReadLine();
    }
    static void Test(int valueVal)
    {
        int temp = 5;
        valueVal = temp * 2;
    }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'ValueType' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



Exercise 3: Reference Type

Step 1: Add a console based project 'ReferenceType' to the solution

Step 2: Right click on project ReferenceType -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'ReferenceType.cs'

Step 4: Replace the code in 'ReferenceType.cs' with the given code

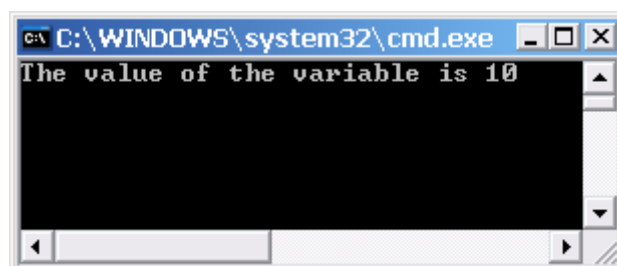
```
using System;
class ReferenceType
{
    public int valueVal;
}
class TestReference
{
    static void Main(string[] args)
    {
        ReferenceType refer = new ReferenceType();//Create instance
        refer.valueVal = 5;
        Test(refer);//call function
        Console.WriteLine("The value of the variable is {0}",
refer.valueVal);
        Console.ReadLine();
    }
    static void Test(ReferenceType refer)
    {
        int temp = 5;
        refer.valueVal = temp * 2;
    }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build ReferenceType option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



Exercise 4: Number format specifier

Step 1: Add a console based project 'NumberFormat' to the solution

Step 2: Right click on project NumberFormat -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'NumberFormat.cs'

Step 4: Replace the code in 'NumberFormat.cs' with the given code

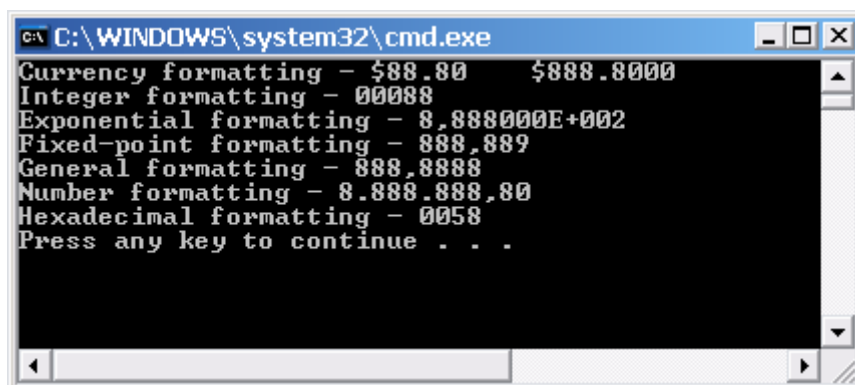
```
/*This program demonstrates the the numeric formatting in C#*/  
using System;  
class NumberFormat  
{  
    static void Main(string[] args)  
    {  
        Console.WriteLine("Currency formatting - {0:C} {1:C4}", 88.8, 888.8);  
        Console.WriteLine("Integer formatting - {0:D5}", 88);  
        Console.WriteLine("Exponential formatting - {0:E}", 888.8);  
        Console.WriteLine("Fixed-point formatting - {0:F3}", 888.8888);  
        Console.WriteLine("General formatting - {0:G}", 888.8888);  
        Console.WriteLine("Number formatting - {0:N}", 8888888.8);  
        Console.WriteLine("Hexadecimal formatting - {0:X4}", 88);  
    }  
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'NumberFormat' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



```
C:\WINDOWS\system32\cmd.exe  
Currency formatting - $88.80    $888.8000  
Integer formatting - 00088  
Exponential formatting - 8.888000E+002  
Fixed-point formatting - 888.888  
General formatting - 888.8888  
Number formatting - 8.888.888,80  
Hexadecimal formatting - 0058  
Press any key to continue . . .
```

Exercise 5: Datetime format specifier

Step 1: Add a console based project 'DateTimeFormat' to the solution

Step 2: Right click on project DateTimeFormat -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'DateTimeFormat.cs'

Step 4: Replace the code in 'DateTimeFormat.cs' with the given code

```
using System;
class MainClass
{
    public static void Main()
    {
        DateTime dt = DateTime.Now; // obtain current time

        Console.WriteLine("d format: {0:d}", dt);
        Console.WriteLine("D format: {0:D}", dt);

        Console.WriteLine("t format: {0:t}", dt);
        Console.WriteLine("T format: {0:T}", dt);

        Console.WriteLine("f format: {0:f}", dt);
        Console.WriteLine("F format: {0:F}", dt);

        Console.WriteLine("g format: {0:g}", dt);
        Console.WriteLine("G format: {0:G}", dt);

        Console.WriteLine("m format: {0:m}", dt);
        Console.WriteLine("M format: {0:M}", dt);

        Console.WriteLine("r format: {0:r}", dt);
        Console.WriteLine("R format: {0:R}", dt);

        Console.WriteLine("s format: {0:s}", dt);

        Console.WriteLine("u format: {0:u}", dt);
        Console.WriteLine("U format: {0:U}", dt);

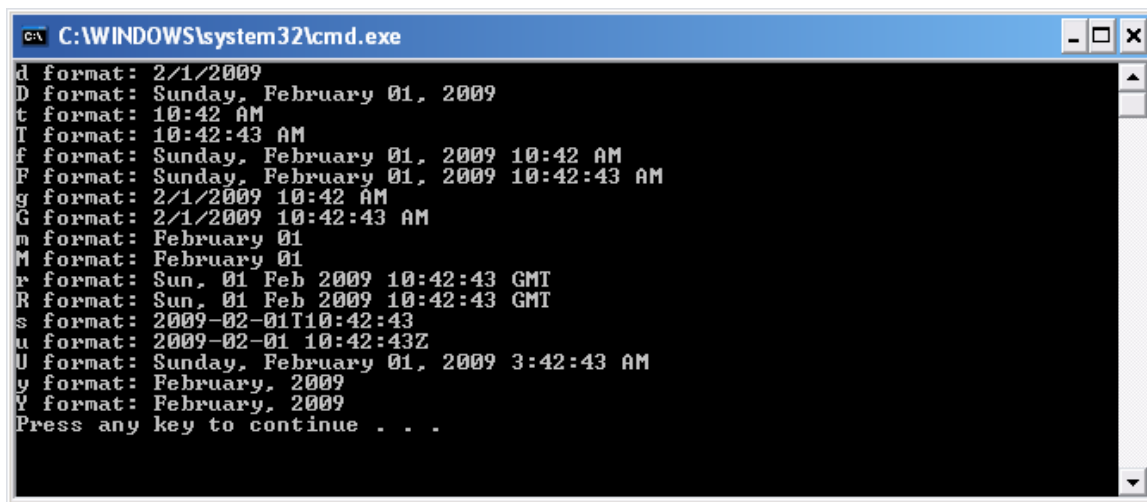
        Console.WriteLine("y format: {0:y}", dt);
        Console.WriteLine("Y format: {0:Y}", dt);
    }
}
```

Step 5: Select menu File -> Save to save the file

Step 6: Select Build -> Build 'DateTimeFormat' option to build the project

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



```
C:\WINDOWS\system32\cmd.exe
d format: 2/1/2009
D format: Sunday, February 01, 2009
t format: 10:42 AM
T format: 10:42:43 AM
f format: Sunday, February 01, 2009 10:42 AM
F format: Sunday, February 01, 2009 10:42:43 AM
g format: 2/1/2009 10:42 AM
G format: 2/1/2009 10:42:43 AM
m format: February 01
M format: February 01
r format: Sun, 01 Feb 2009 10:42:43 GMT
R format: Sun, 01 Feb 2009 10:42:43 GMT
s format: 2009-02-01T10:42:43
u format: 2009-02-01 10:42:43Z
U format: Sunday, February 01, 2009 3:42:43 AM
y format: February, 2009
Y format: February, 2009
Press any key to continue . . .
```

Exercise 6: Implicitly typed local variables

Step 1: Add a console based project 'ImplicitlyTypedLocal' to the solution

Step 2: Right click on project 'ImplicitlyTypedLocal' -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'ImplicitlyTypedLocal.cs'

Step 4: Replace the code in 'ImplicitlyTypedLocal.cs' with the given code

```
class ImplicitlyTypedLocal
{
    static void Main(string[] args)
    {
        var i = 5;
        var s = "hello";
        var d = 1.0;
        //i is an integer
        Console.WriteLine("i*i: " + i * i);
        //s is a string
        Console.WriteLine("s in upper case:" + s.ToUpper());
        //d is a double
        Console.WriteLine("type of d:" + d.GetType());
        Console.ReadLine();
    }
}
```

Step 5: Run the program

Exercise 7: Anonymous Types variables

Step 1: Add a console based project 'AnonTypes' to the solution

Step 2: Right click on project 'AnonTypes' -> set as Startup project

Step 3: Rename the class file 'Program.cs' to 'AnonTypes.cs'

Step 4: Replace the code in 'AnonTypes.cs' with the given code

```
class AnonTypes
{
    static void Main(string[] args)
    {
        var p1 = new { Name = "A", Price = 3 };
        Console.WriteLine("Name = {0}\nPrice = {1}",
            p1.Name.ToLower(), p1.Price);
    }
}
```

```
Console.ReadLine();
```

```
}  
}
```

Step 5: Run the program

Part 2 – Workshops (30 minutes)

- Quickly look at workshops of Module 1 and Module 2 on Onlinevarsity
- Try to compile, run and observe the output of sample code provided for related workshop. Discuss with your class-mate and your instructor if needed.

Part 3 – Lab Assignment (60 minutes)

Do the assignment for Module 1 and Module 2 carefully. Discuss with your class-mates and your instructor if needed.

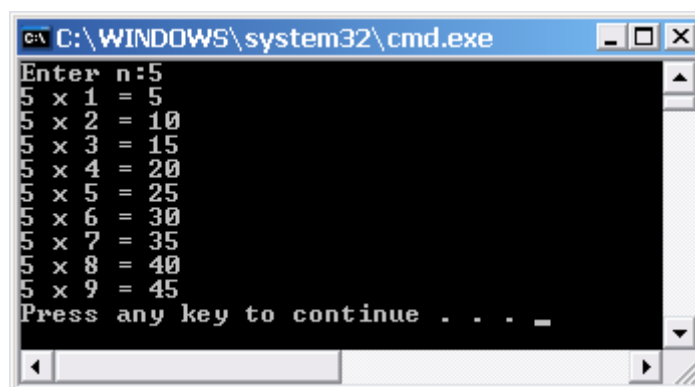
Part 4 – Do it yourself

Exercise 1: Write a program to enter: name, address, phone and display this information.

Exercise 2: Write a program to accept three integer numbers and find the maximum number from three integers.

Exercise 3: Write a program that accepts a number between 1 and 7 from the user and return the corresponding day of the week(1- Monday, 2- Tuesday and so on).

Exercise 4: Write a program to display the first 9 multiples of an integer. N entered from user



Exercise 5: Write a program to print the factorials of the integers from 1 to 20

References

- 1) Onlive varsity C# Programming, Aptech Education
- 2) MSDN Document