Output, Operators, Types & Variables and Input

Output

In this first example, you will learn about how to write some code to display **Hello World**, this is a classic programming example that's traditional start to learning any programming language!

Visit **dotnetfiddle.net** and in the large box define which **namespace** to use entering the following:

```
using System;
```

System is used to allow access to the **Console** to output **"Hello World"**, a **namespace** is a way of organising groups of programming elements that can be used in the programme and the using statement should end with a semicolon like most statements in **C#**.

The next thing to do is to define the **class** for the example called **Demo** by entering in to **dotnetfiddle.net** the following:

```
public class Demo
{

}
```

Public is a Keyword that defines the access level of a **class** and how it may be used by other parts of the code and in this case, anything can access it. The **class** Keyword is used to define classes and this also uses curly braces to define the scope of the **class**.

The next thing to define is the **Main** Method which is the entry point for the example and is the first code to be run - this Method should be defined as below:

```
public static void Main()
{

}
```

Like a **class**, a method can also have an access level and again this is **public** so anything can access it, **static** defines this Method as global but will go into this concept in more detail later and the **void** Keyword means the Method doesn't return a value and the brackets define any Parameters or none in this case, we'll learn about those later too.

Within the **Main** Method in **dotnetfiddle.net** the following should be entered:

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

Console is a type in the **System namespace** that can be used, here the **WriteLine** method is called and is like the **Main** Method in the example as it is also declared as being **static**. Inside the brackets is a value being passed as a Parameter, this is a **string** which is a set of text characters, in this case **"Hello World"**, the whole application in **dotnetfiddle.net** should look as below:

```
using System;

public class Demo
{
	public static void Main()
	{
		Console.WriteLine("Hello World");
	}
}
```

You can then use the **Run** option to start the programme which will display the text "**Hello World"** in the lower large box on **dotnetfiddle.net** or if **Auto Run** is **Yes** then it will run automatically.

This programme structure will be used in all subsequent examples so they'll always start with the **using** statements, followed by defining the **class** then the **Main** Method the example will use as its entry point which is the first point that will be reached when the example is started.

Operators

C# supports many kinds of Operators which are symbols that specify which Operations such as mathematics. We'll cover some of the basic ones here such as addition and multiplication.

The first thing is to define the basic structure of a programme by entering the following in **dotnetfiddle.net** in the main window:

```
using System;

public class Demo
{
	public static void Main()
	{
		// Code
	}
}
```

// Code is a Comment which is defined by writing **//** before anything that can be included in the programme but won't be run as part of it such as reminders or explanations of what the code should do.

The first Operator to use is **+** which is used for addition so enter the following below **// Code** in **dotnetfiddle.net**:

```
Console.WriteLine(4 + 2);
```

Then select **Run** in **dotnetfiddle.net** and this will display the answer as the output, it is also possible to use other mathematical Operators so can change **+** to ***** for multiply, **-** for subtraction or **/** for division to see what happens such as entering the following in **dotnetfiddle.net**:

```
Console.WriteLine(4 * 2);
Console.WriteLine(4 - 2);
Console.WriteLine(4 / 2);
```

You should see the appropriate answers appear in the output for each Operator that has been used whether it is addition, multiplication, subtraction or division.

Types & Variables

C# is what's known as a strongly typed language where every Variable has a **type** as does every expression that has a value, there are many types available, some of which will be covered here.

The first thing is to define the basic structure of the code by entering the following in **dotnetfiddle.net** in the main window:

```
using System;

public class Demo
{
	public static void Main()
	{
		// Code
	}
}
```

The first way to use types is like algebra in mathematics, so enter the following below **// Code**:

```
int a = 5;
int b = 2;
Console.WriteLine(a + b);
```

int is the **type** which is an Integer which can contain large or small whole numbers, **a** is the name of the variable and **=** is used to assign the value to the type, you can try changing the operator to ***** for multiply, **-** for subtraction and finally **/** for division and check the answer it gives when use **Run** in **dotnetfiddle.net**

In mathematics, it of course it is possible to get answers that aren't whole numbers so will need to use a **type** that allows this so replace the code from before with the following:

```
double a = 5;
double b = 2;
Console.WriteLine(a / b);
```

double is a **type** that does support decimal places in numbers so need to remember to use the correct **type** and you can use **Run** to see the correct answer.

Variables aren't just used to do mathematics - they can be used to store anything for use in the application either one or many times and can be changed with new values.

You can try this with **string** which represents a series of characters by entering in to **dotnetfiddle.net** the following:

```
using System;

public class Demo
{
	public static void Main()
	{
		string message = "Hello World";
		Console.WriteLine(message);
	}
}
```

Text and Numbers relate to things you encounter day to day and can relate to but there are also Types that are a bit more specific to programming, enter in to **dotnetfiddle.net** the following example:

```
using System;

public class Demo
{
	public static void Main()
	{
		bool a = true;
		bool b = true;
		Console.WriteLine(a && b);
	}
}
```

You can then run this in **dotnetfiddle.net**, **bool** is a **type** to store Boolean values which can either be **true** or **false**, in this example it will output **true** and uses another operator **&&** which means "and" so if both **a** and **b** are **true** the output will be **true**, it's also possible to use **||** which means "or" to produce the same output as if **a** or **b** are true, change the "**Console.WriteLine**" line to the following:

```
Console.WriteLine(!(a && b));
```

When run in **dotnetfiddle.net** this will output the opposite value as **!** means not so if something is **true** it becomes **false** and if it's **false** it becomes **true**.

Input

In the previous examples everything has been output to the **Console** but it is also possible to allow something to be Input from the **Console**.

To create a simple input example by using **dotnetfiddle.net** and entering the following:

```
using System;

public class Demo
{
	public static void Main()
	{
		string value = Console.ReadLine();
		Console.WriteLine(value);
	}
}
```

When Run in **dotnetfiddle.net** a "prompt" will appear below the code window **>** and can then type anything there and end input by typing enter or return, this will then be output.

Another example is to enter a number to be used in a calculation by entering in to **dotnetfiddle.net** the following:

```
using System;

public class Demo
{
	public static void Main()
	{
		int a = int.Parse(Console.ReadLine());
		int b = int.Parse(Console.ReadLine());
		Console.WriteLine(a + b);
	}
}
```

By typing in a value for **a** or **b** can then output the sum of these values, another feature shown is that types often have static Methods, in this case **Parse** which allows something to be converted to that **type** so when Run in **dotnetfiddle.net** you can enter **1** then **2** to get the answer **3**, it's possible to change the operator to ones used for **int** such as ***** for multiply, **-** for subtraction or **/** for divide – for that last one try changing **int** to **double** to avoid losing anything after a decimal point like when doing **5** divided by **2**.