

# **CO452 Programming Concepts**

Week 10 - C# Part 1

# 3 elements of programming

variables  
sequence      iteration  
                 arrays  
selection      objects

## Programming concepts



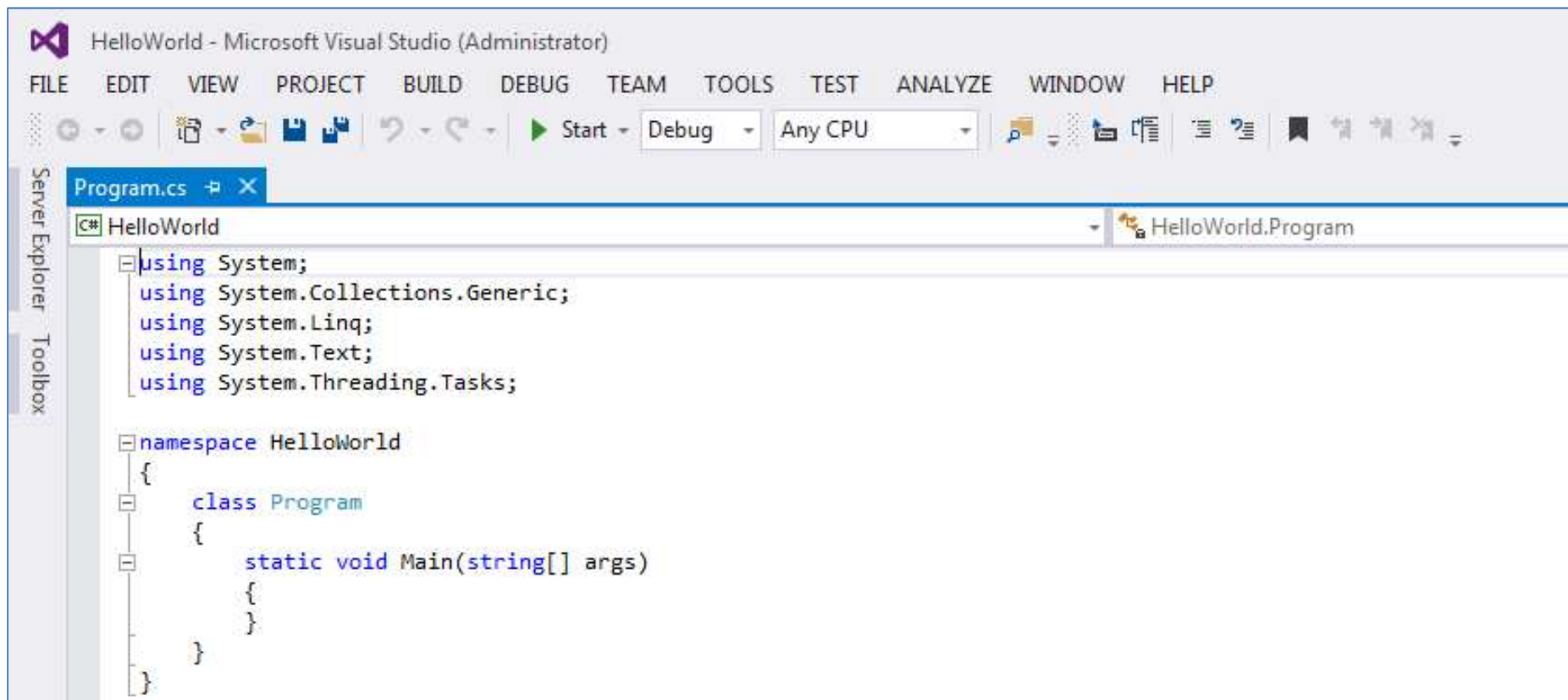
Syntax



Environment

# Microsoft Visual Studio

- An integrated development environment (IDE) to develop computer programs for MS Windows, as well as web sites, web applications and web services.



# Brief history of C#

- C# was announced in 2000
- Fully Object-Oriented language – everything is an object
- Some say that it is similar to Java, but there are differences which make them unique





# Simple C#

and how to write  
a short Program

# The **Main()** function/method

```
static void Main ()  
{  
  
    Console.WriteLine( "Hello World!");  
  
}
```

The **Main()** function (or method) is where execution starts

- But it needs to be embedded inside a **class**
- And the class is put inside a **namespace**

# A complete program

```
using System;
```

```
namespace Task1
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main ()
```

```
        {
```

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

```
        }
```

```
    }
```

```
}
```



Use **Ctrl + F5**  
to execute  
the program



braces { } act like begin and end

# The Main() Method

## static void Main()

- Every project must have one of these
- Main tells us this is the main method ..  
execution starts here
- void tells us that this method does not  
return a value
- static means that we do not have to create  
an object from this class in order to execute  
this method (normally we do)



# Important Note about C#

## C# is Case Sensitive

- **main()** is NOT the same as **Main()**
- **total** is NOT the same as **Total** etc.



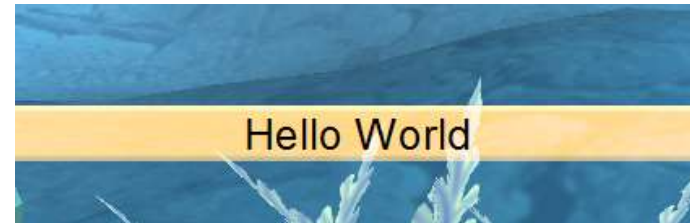
# Output in C#

**How to put  
stuff on the screen**

# Output - Comparison

Ceebot

```
message("Hello World");
```



C#

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



# Simple Output using Console.Write()

```
using System;
```

```
// author: B Ward Date: 23 July 2013
```

```
namespace Task2
```

```
{
```

```
    class Program
```

```
    {
```

```
        static void Main ()
```

```
        {
```

```
            Console.Clear();    // first clear the screen
```

```
            Console.Write( "High Wycombe" );
```

```
            Console.Write( "Gateway to Paradise" );
```

```
            Console.Write( "and Bucks New Uni" );
```

```
        }
```

```
    }
```

```
}
```

**Using statements** – bring into your program pre-created code that your program can use; these simplify your coding but

you

**using System;**

may or may not need all of them

*// author: B Ward Date: 23 July 2013*

namespace Task2

{

class Program

{

static void Main ()

{

Console.Clear(); *// first clear the screen*

Console.Write( "High Wycombe" );

Console.Write( "Gateway to Paradise" );

Console.Write( "and Bucks New Uni" );

}

}

}

# Running the previous Program

```
High WycombeGateway to Paradiseand Bucks New Uni_
```

Each output continues from the previous one.  
There are no line breaks.

# Using Console.WriteLine() and \n

```
using System;
// author: B Ward Date: 23 July 2013
namespace Task2
{
    class Program
    {
        static void Main ()
        {
            Console.Clear();    // first clear the screen
            Console.WriteLine( "High Wycombe" );
            Console.WriteLine( "Gateway to Paradise" );
            Console.WriteLine( "and \n\n Bucks New Uni");
        }
    }
}
```

# Running the Program again

```
High Wycombe  
Gateway to Paradise  
and  
  
Bucks New Uni  
-
```



# Output Control Characters

`\n` is an example of a control character.

There are many others that can be used in output statements and they all start with the special “escape” character backslash `\`

## Control Characters

<code>\n</code>	<b>newline</b>
<code>\t</code>	<b>tab</b>
<code>\b</code>	<b>backspace</b>
<code>\\</code>	<b>print <u>one</u> backslash</b>
<code>\"</code>	<b>print <u>one</u> double quote</b>

These can be used at any position in an output string

## Using some Control Characters : What will be the result?

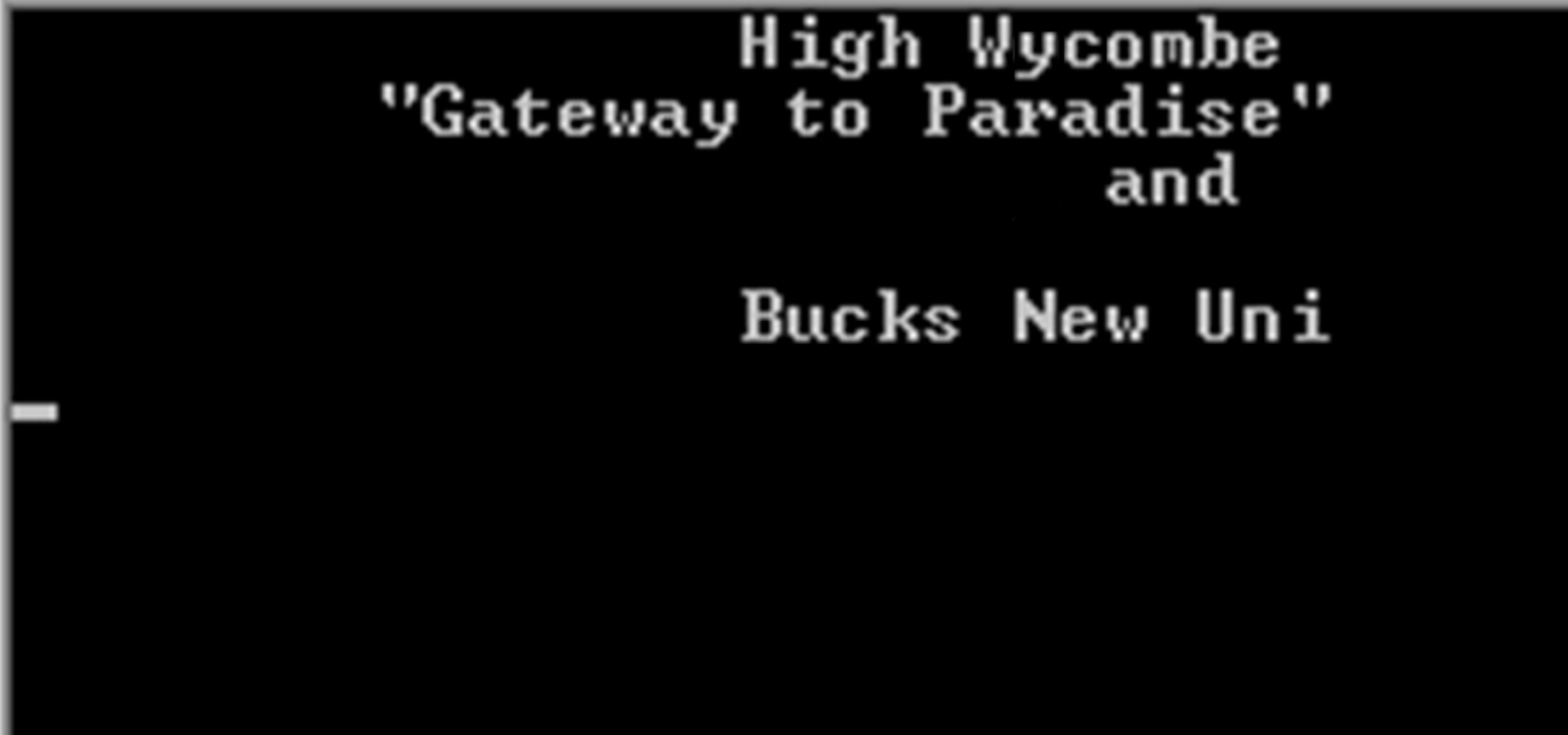
```
using System;
// author: B Ward   Date: 24 July 2013

namespace Task7
{
    class Program
    {
        static void Main ()

            Console.Clear();    // first clear the screen
            Console.WriteLine ( " \t\tHigh Wycombe" );
            Console.WriteLine ( "\t"Gateway to Paradise\t\t" );
            Console.WriteLine( "\t\t\tand\n\n\t\tBucks New Uni " );

    }
}
```

# Running the Program



```
High Wycombe  
"Gateway to Paradise"  
and  
Bucks New Uni
```



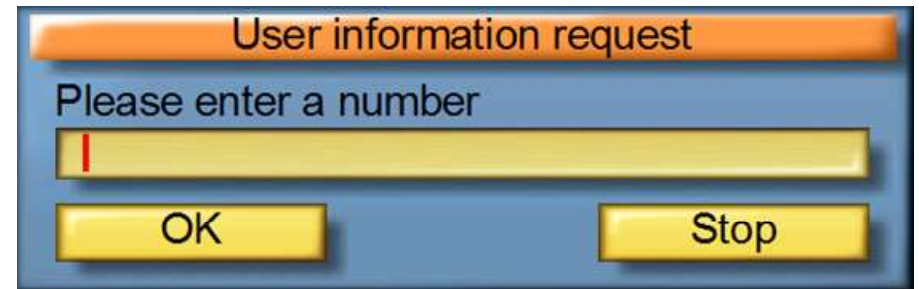
# Input in C#

How to enter  
stuff from  
the keyboard

# Input - Comparison

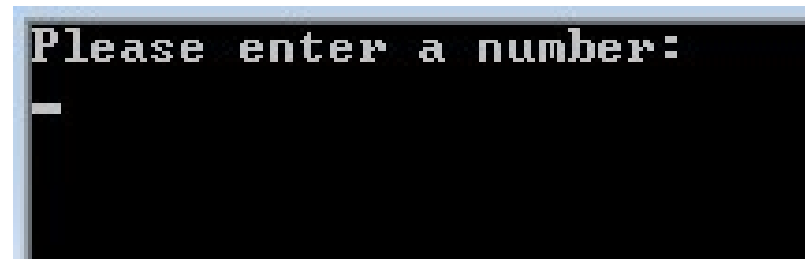
## Ceebot

```
input = dialog("Please enter...");
```



## C#

```
input = Console.ReadLine();
```



# Recap on Variables



Variables need a **data type**, and a name:

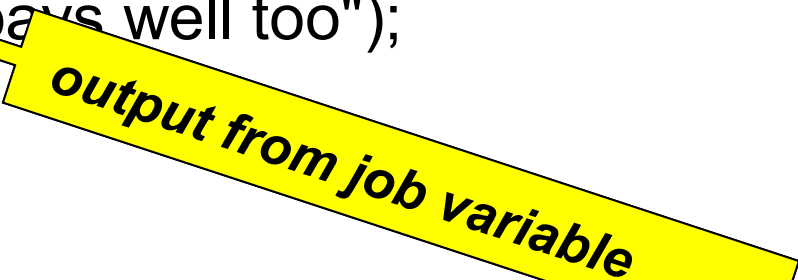
```
string name = "Nick";
```

```
char n = 'n';
```

```
int num = 70;
```

# Input using Console.ReadLine()

```
using System;
namespace Task3
{
    class Program
    {
        static void Main ()
        {
            string job ; 
            Console.WriteLine( "Hello There!" );
            Console.Write( "What is your occupation?" );
            job = Console.ReadLine(); 
            Console.WriteLine( job + "! That must be very satisfying\n");
            Console.WriteLine("I hope it pays well too");
        }
    }
}
```



# Running the program

```
job = Console.ReadLine();
```

```
Hello There  
What is your occupation? plumber * typed user input  
plumber! That must be very satisfying  
I hope it pays well too
```

```
Console.WriteLine(job + "! That must etc. " );
```

## Note:

- job is a variable and is NOT placed inside "quotes"
- Use **+** to join together the various parts of the output



# More Input in C#

## Conversion



# Input of string, int and double

```
using System;
namespace Task4
{
    class Program
    {
        static void Main ()
        {
            string input, name ;
            double height;
            int age;

            Console.Write( "What is your name?" );
            name = Console.ReadLine();

            Console.Write( "How old are you : ");
            input = Console.ReadLine();
            age = Convert.ToInt32(input);

            Console.Write( "And how tall are you in metres: ");
            input = Console.ReadLine();
            height = Convert.ToDouble(input);

            Console.WriteLine( "\n\n\tHello " + name);
            Console.WriteLine("\tYou are " + age + " years old and "
                + height + " metres tall" );
        }
    }
}
```

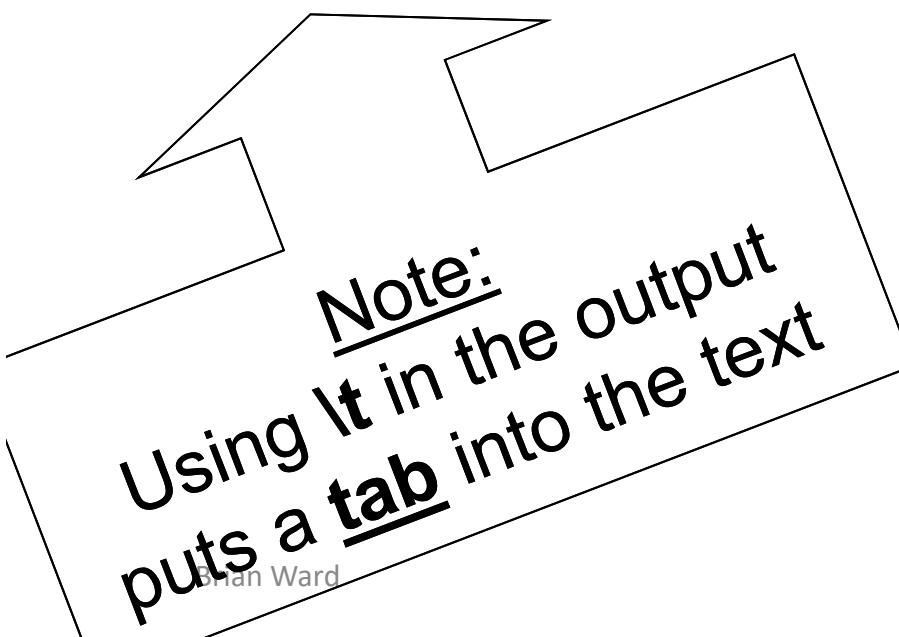
convert input  
string to int

convert input  
string to double

# Running the Program

```
What is your name? Brian
How old are you : 35
And how tall are you in metres: 1.65

    Hello Brian
    You are 35 years old and 1.65 metres tall
```



Note:  
Using `lt` in the output  
puts a tab into the text

# Compare C# with Ceebot

---

```
string input;
```

```
int age;
```

- **C#**

```
Console.Write("How old are you?");
```

```
input = Console.ReadLine();
```

```
age = Convert.ToInt32(input);
```

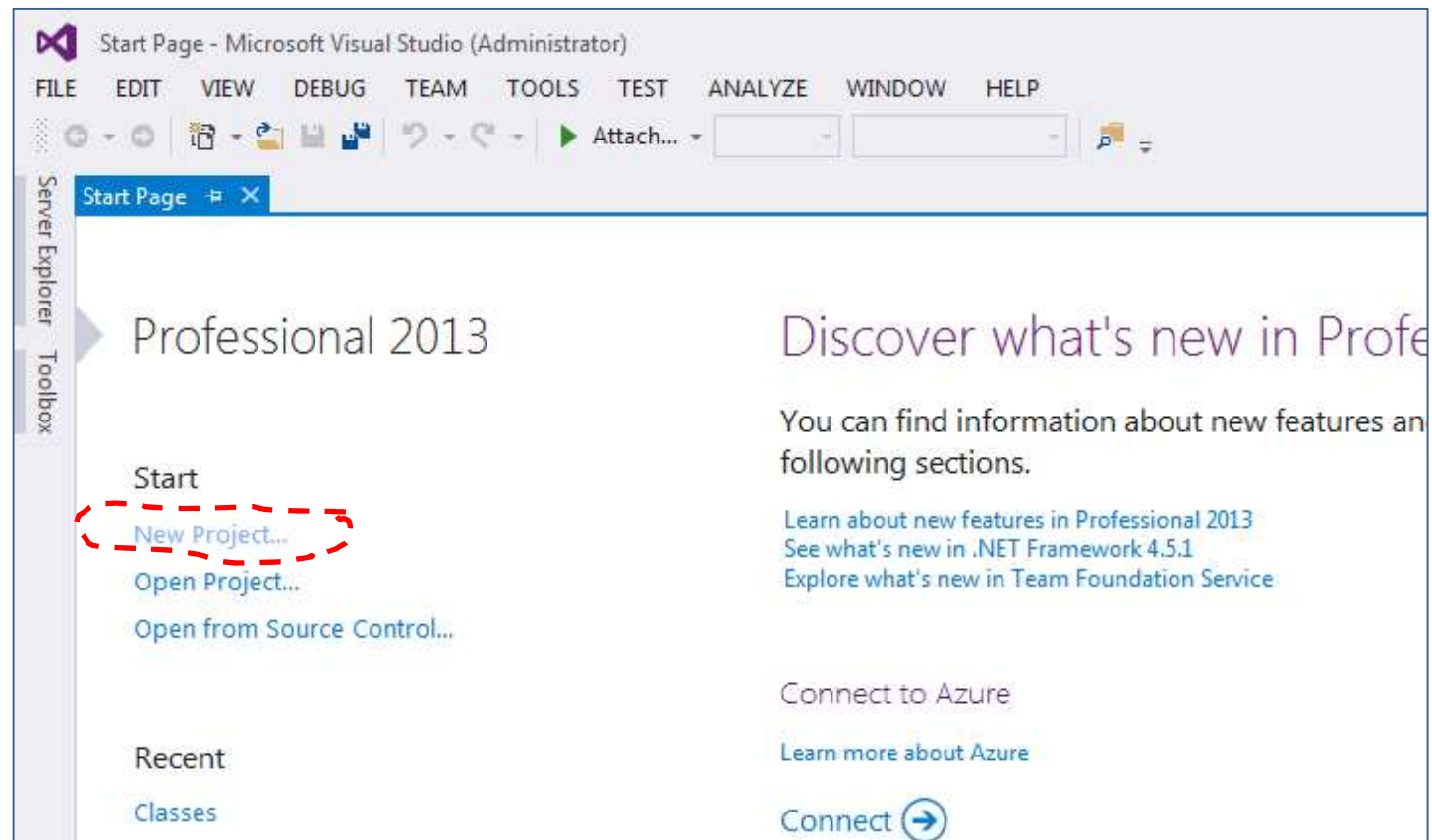
- **Ceebot**

```
input = dialog("How old are you?);
```

```
age = strval(input);
```

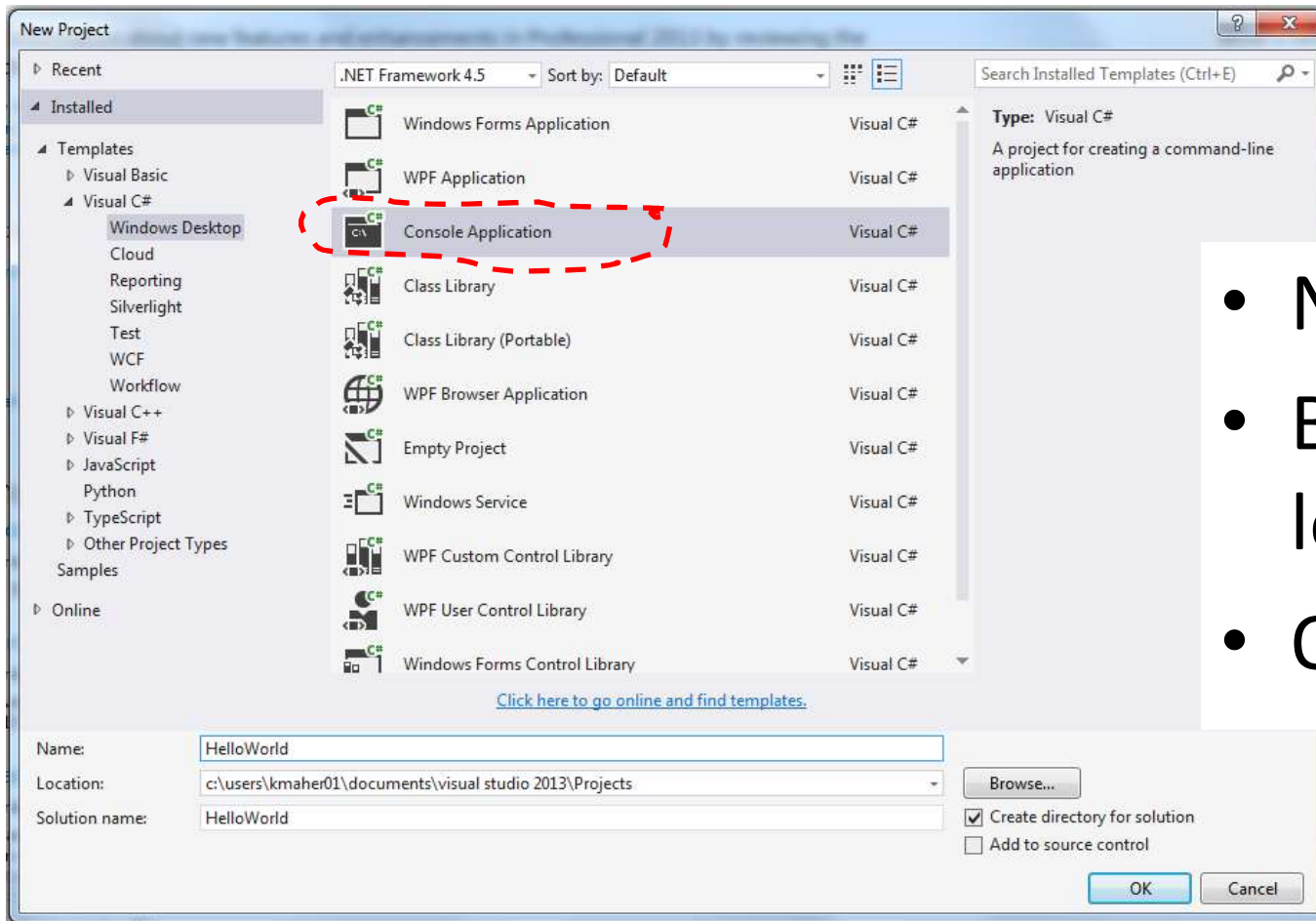
# Getting started with Visual Studio

- Open up MS Visual Studio
- New Project.....



# Getting started with Visual Studio

- Installed Templates
- Visual C# > Windows Desktop > Console Application



- Name
- Browse to save location
- OK

# Let's try Visual Studio

- **Output exercises**

- Slide 13      `Console.Write();`
- Slide 16      `Console.WriteLine();`
- Slide 19      using control characters

- **Input exercise**

- Slide 24

# Activity

## Exercise 1:

Write a program that asks the user to input their first name, second name, age, and town of origin, and then output these in a sentence.



# Note about running programs

Pressing **F5** will debug the program (check it performs correctly) and automatically exit once finished

Pressing **Ctrl + F5** will start the program without debugging, meaning that the program will not automatically exit once the last line of code has been executed.

Instead you'll see this statement at the end of the program:

```
Press any key to continue . . .
```

# Activity

## Exercise 2:

Write a program that calculates the area of a circle by being given the radius.

Use the formula  $A = \pi r^2$

## Exercise 3:

Ask the user to input the price of an item and work out the amount of VAT that would be added on to the item (say VAT is 20%).

# A Calculation Program

**Using Input and Output**

# A C# Program: Simple Addition

```
using System;
namespace Task5
{
    class Program
    {
        static void Main ()
        {
            string input ;
            int number1, number2, total;

            Console.WriteLine( "Please Enter two numbers" );
            Console.Write( "Enter the first number:" );
            input = Console.ReadLine();
            number1 = Convert.ToInt32(input);

            Console.Write( "Enter the next number : " );
            input = Console.ReadLine();
            number2 = Convert.ToInt32(input);

            total = number1 + number2;

            Console.WriteLine( "Sum of the numbers is " + total);
        }
    }
}
```



**convert input  
string to int**

# Integer width comparison (U cannot be negative)

## 16, 32 and 64 bits

Short Name	Class	Type	Width	Range (bits)
int	<a href="#">Int32</a>	Signed integer	32	-2,147,483,648 to 2,147,483,647
uint	<a href="#">UInt32</a>	Unsigned integer	32	0 to 4,294,967,295
short	<a href="#">Int16</a>	Signed integer	16	-32,768 to 32,767
ushort	<a href="#">UInt16</a>	Unsigned integer	16	0 to 65,535
long	<a href="#">Int64</a>	Signed integer	64	-9223372036854775808 to 9223372036854775807
ulong	<a href="#">UInt64</a>	Unsigned integer	64	0 to 18446744073709551615

```
Console.WriteLine("The number of bytes each data type takes up in memory: \n");
Console.WriteLine("Size of byte: \t" + sizeof(byte));
Console.WriteLine("Size of bool: \t" + sizeof(bool));
Console.WriteLine("Size of char: \t" + sizeof(char));
Console.WriteLine("Size of short: \t" + sizeof(short));
Console.WriteLine("Size of int: \t" + sizeof(int));
Console.WriteLine("Size of float: \t" + sizeof(float));
Console.WriteLine("Size of double: " + sizeof(double));
Console.WriteLine("Size of long: \t" + sizeof(long));
```

The number of bytes each data type takes up in memory:

```
Size of byte:      1
Size of bool:     1
Size of char:     2
Size of short:    2
Size of int:      4
Size of float:    4
Size of double:   8
Size of long:     8
Press any key to continue . . .
```

# Bytes and bits

128	64	32	16	8	4	2	1
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

What we type in

How it's stored in binary:

1	0	0	0	0	0	0	0	1
12	0	0	0	0	1	1	0	0
255	1	1	1	1	1	1	1	1

# Running the program

*These are typed in by the user and they are put into the variables **number1** and **number2***

```
Please Enter two numbers
Enter the first number: 12
Enter the next number : 34
Sum of the numbers is 46
_
```

***number1** and **number2** are added and the sum stored in the variable **total**. Then **total** is output here.*



# Alternative Input method

**Instead of :**

```
input = Console.ReadLine();  
number = Convert.ToDouble(input);
```

**We can use one statement :**

```
number = Convert.ToDouble(Console.ReadLine());
```

# Constants and Decimal places

# Defining constants

C# allows us to use the word const to set constant values for a program

For example:

```
const int MAX = 10 ;  
const int SIZE = 50 ;  
const string HEADING = "BNU" ;  
const double RATE = 17.5 ;  
const double PI = 3.142;
```

**Constants are useful in programs when you want to use a name instead of a value for items that will not change throughout a program**

# Constants and Decimal places

```
using System;
namespace Task5
{
    class Program           // program to convert inches to metres
    {
        static void Main ()
        {
            string input;
            double inchHeight, mHeight;
            const double INCHTOMS = 0.0254;
            Console.WriteLine( "Height Calculation" );
            Console.Write( "What is your height in inches : " );

            input = Console.ReadLine();
            inchHeight = Convert.ToDouble(input);

            mHeight = inchHeight * INCHTOMS;

            Console.WriteLine( "Your height is " + mHeight.ToString("0.00")
                               + " metres");
        }
    }
}
```

constants can't be changed

display mHeight to 2 decimal places

# The Last Slide



# Alternative input methods

```
input = Console.ReadLine();
```

then

```
number1 = int.Parse(input);
```

or

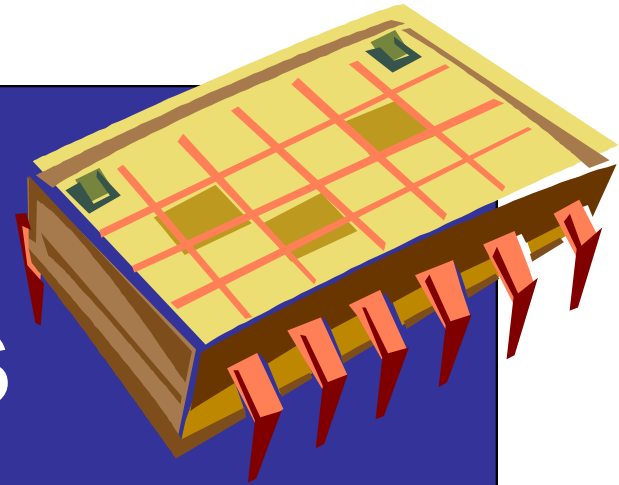
```
number2 = float.Parse(input);
```

or

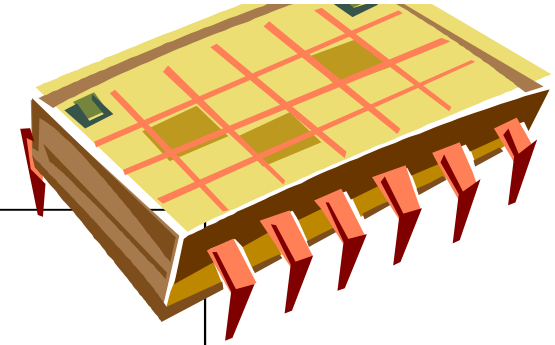
```
number3 = double.Parse(input);
```

# Variables

**How C# stores  
temporary data  
within a program**



# What is a Variable?



1. A storage area in the computer memory
2. Can hold numbers, words, etc.
3. The contents may change as the program runs
4. Variables need to be given unique names
5. A variable name is also known as an Identifier

## Rules for Identifiers (or variable names)

- a. Must start with a letter
- b. No spaces
- c. Can only have letters, digits, underscore
- d. No reserved words (main, cout, etc.)
- e. Name, name and NAME are all different (i.e. C# is case-sensitive)

### Name OK or not?

Tax_Code	✓
my-name	✗
1stname	✗
D2	✓
Number4	✓
%cost	✗
first name	✗



# Data Types for C# Variables

There are 5 main data types for variables

## int

Can store whole numbers e.g.

3 0 -261 46 -7

## double

Can store numbers with decimal places e.g.

10.67 -0.05 13.0 176.4

## char

Can store one character

e.g. 'A' '£' 'y' '+' '5' 'd'

## string

Can store text (strings of characters) e.g.

"High Wycombe" "Brian"

## bool

Can be either **true** or **false**

*Each type needs a different amount of storage space*

## Declaring Variables

```
int age ;  
char sex ;  
double wage ;  
string name ;  
string address ;
```

# Assignment

**How to put values  
into variables**

# Assignments in C#

Information can be stored in a variable using:

the assignment statement  
and assignment operator (=)

e.g: *(assuming variables declared)*

```
age = 25 ;
```

```
wage = 15.50 ;
```

```
choice = "A" ;
```

```
name = "Brian Ward" ;
```

```
title = "Menu List" ;
```

Computer Memory

Variable	Contents
age	25
choice	A
wage	15.50
name	Brian Ward
title	Menu List

# Initialising Variables

Variables can be given an initial value at the same time as they are declared

e.g.

```
int count = 0 ;
```

```
float price = 7.54 ;
```

```
string name = "Joe Smith" ;
```

# Multiple Assignments

Multiple variables of the same type can be assigned the same value

e.g.

```
a = b = c = 8 ;
```

```
price1 = price2 = 7.54 ;
```

```
adult_count = child_count = 0 ;
```

# Why are there 2 Equals?

## Equals (=)

This is the assignment operator.  
It is used to change the value of a variable e.g.

```
total = num1 + num2 ;
```



total  
is  
changed

## Equals (==)

This is a relational operator.  
It is used in conditions.  
Nothing is changed e.g.

```
if (total == 100) ....
```



total  
stays  
the same

# Constants in C#



# Example Program using constants

```
using System;  
namespace Task8
```

```
{  
  class Program
```

```
{  
  static void Main ()
```

```
{  
  const string HEADING = "CD Purchase Program\n" ;
```

```
  const double CD_COST = 9.50 ;
```

```
  string input , name;
```

```
  int num;
```

```
  double total;
```

```
  Console.WriteLine( HEADING );
```

```
  Console.Write( "\tEnter your name : " );
```

```
  name = Console.ReadLine();
```

```
  Console.Write( "\tHow many CDs : ");
```

```
  input = Console.ReadLine();
```

```
  num = Convert.ToDouble(input);
```

```
  total = num * CD_COST;
```

```
  Console.WriteLine( "\n\tMr " + name + " owes £" + total);
```

```
}  
}
```



output a constant



calculate total  
using constant



# Running the Program

## CD Purchase Program

**Enter your name : Brian Ward**

**How many CDs : 5**

**Mr Brian Ward owes £47.5**