Practical

Simple User I/O and Text

Up until this point we have hard coded the values of the variables used with in the programs.

This practical will introduce the **Console.ReadLine** method that enables users to enter and store values entered from the keyboard.

Example 1.

Consider the following example.

```
using System;
namespace parsing
{
   class Program
   {
      static void Main(string[] args)
      {
         //using parsing
         int number;
         float weight;
         float total_weight;
         Console.Write("Enter any number : ");
         number = int.Parse(Console.ReadLine());
         Console.Write("Enter your weight : ");
         weight = float.Parse(Console.ReadLine());

         total_weight = number * weight;

         Console.WriteLine("You have entered : " + number);
         Console.WriteLine("The weight is : " + weight);
         Console.WriteLine("The total weight is : " + total_weight);
         Console.ReadLine();
      }
   }
}
```

When the code is executed the user is presented with a prompt.

Enter any number :

The Console.ReadLine() method stops the program so that the user can enter a value followed by the **Enter** key.

The Console.ReadLine() then reads the data entered by the keyboard excluding the **Enter** key and returns the value represented as a string of characters.

To use the data in an equation we then must convert the characters to the data type we are going to store the value in.

```
number = int.Parse(Console.ReadLine());
```

In this instance we need to convert it to an integer using the **int.Parse** method. **int.Parse** tries to convert the characters to an integer and if successful, assigns the integer to the variable **number** .

```
Console.Write("Enter your weight : ");
weight = float.Parse(Console.ReadLine());
```

We then repeat the process to get the **weight** but this time we use the float.Parse because we need to convert the characters to a float as weight is defined as a float data type.

Now that we have the two variables, **number** and **weight,** with valid values stored in them, we can determine the **total_weight** and output the results.

To validate this process, complete the following:

1. Create a new project prac4_ex1 and enter the code defined above and compile and run the example.
2. Enter a valid number followed by the Enter key
3. Enter a valid weight followed by the Enter key
4. Note the output
5. Now Rerun the program but this time enter 'dog' instead of a valid integer for the number and see what happens.
6. The program should fail. This is because the program could not interpret and convert the characters 'dog' to a valid integer and thus fails to continue. This type of error is common and future practicals will develop code to ensure the data is entered correctly and the program does not fail, but for now always enter valid values.

Question 1. Using code developed in practical 3 (below), convert it to do the following:

- Prompt for the length
- Get the length and convert and store the value
- Prompt for the width
- Get the width and convert and store the value
- Compile, run and verify the program is correct.

```
// Compute the area of a given length and width.
using System;
namespace prac_1
{
 class Area
 {
   static void Main()
   {
     float length;
     float width; float
     area;
     length = 100.51F;
     width = 50.5F;
     area = length * width;
     Console.WriteLine("The area is " + area.ToString("0.###"));
   }
 }
}
```

Example 2.

Consider the following example :

This program is very similar to example 1 but this time we are adjusting the method we gather the information from the user and the way we convert the data from characters to the required data types.

```
using System;
namespace convert_conversion
{
   class Program
   {
      static void Main(string[] args)
      {
         int number;
         decimal weight;
         decimal total_weight;
         string tempVal = "";

         Console.Write("Enter any number : ");
         tempVal =Console.ReadLine();
         number = Convert.ToInt32(tempVal);

         Console.Write("Enter your weight : ");
         tempVal =Console.ReadLine();
         weight= Convert.ToDecimal(tempVal);
         total_weight = number * weight;
         Console.WriteLine("You have entered : " + number);
         Console.WriteLine("The weight is : " + weight);
         Console.WriteLine("The total weight is : " + total_weight);
         Console.ReadLine();
      }
   }
}
```

This time we will store the value read from the user into a temporary string variable and then convert the string into the required data type.

A string is a collection of characters. Any keyboard key is a character and as we enter the key strokes we build a string of characters. When we press the **Enter** key the **Console.ReadLine** returns the string of characters entered and stores it in the tempVal variable we defined as a string data type.

```
string tempVal = "";
```

Now we can use a new process to convert the characters to the required data type.

```
number = Convert.ToInt32(tempVal);
```

`Convert.ToInt32(tempVal)` reads and converts the `tempVal` variable data and attempts to convert the characters to a valid integer. If it is successful the value is then stored in the `number` variable.

We repeat the process for the `weight variable` but this time we use `Convert.ToDecimal(tempVal);`

Note : Convert does not have a conversion method for floats and if required you can use Convert.ToSingle but to avoid confusion I have changed the weight variable to a decimal for this example**.**

To validate this process, complete the following:

1. Create a new project prac_ex2 and enter the code defined above and compile and run the example.
2. Enter a valid number followed by the Enter key
3. Enter a valid weight followed by the Enter key
4. Note the output
5. Now re-run the program but this time enter 'dog' instead of a valid integer for the number and see what happens.

The program should fail. This is because the program could not interpret and convert the characters 'dog' to a valid integer and thus fails to continue. This type of error is common and future practicals will develop code to ensure the data is entered correctly and the program does not fail, but for now always enter valid values.

Question 2.

Write a program that performs the following tasks:

- Declare a string to store a person's name
- Declare a variable to store a person's age
- Prompt the user to enter their name
- Store the name entered
- Prompt the user to enter their age

- Store the age entered (remember you will need to convert the age to the data type you have used to declare the variable age)
- Calculate the number of days the user has been alive by multiplying the age * 365 days
- Output the person's name and number of days they have been alive

Example :

Enter your Name > Barry

Enter your Age > 45

Barry you have been alive 16425 days.

Additional Tasks:

Assume a person will live to an average age of 80.

- Work out and display how many days left there are until the person reaches 80.
- Format the output so that the thousand separator are displayed.
    - EG: Barry you have been alive 16,425 days.