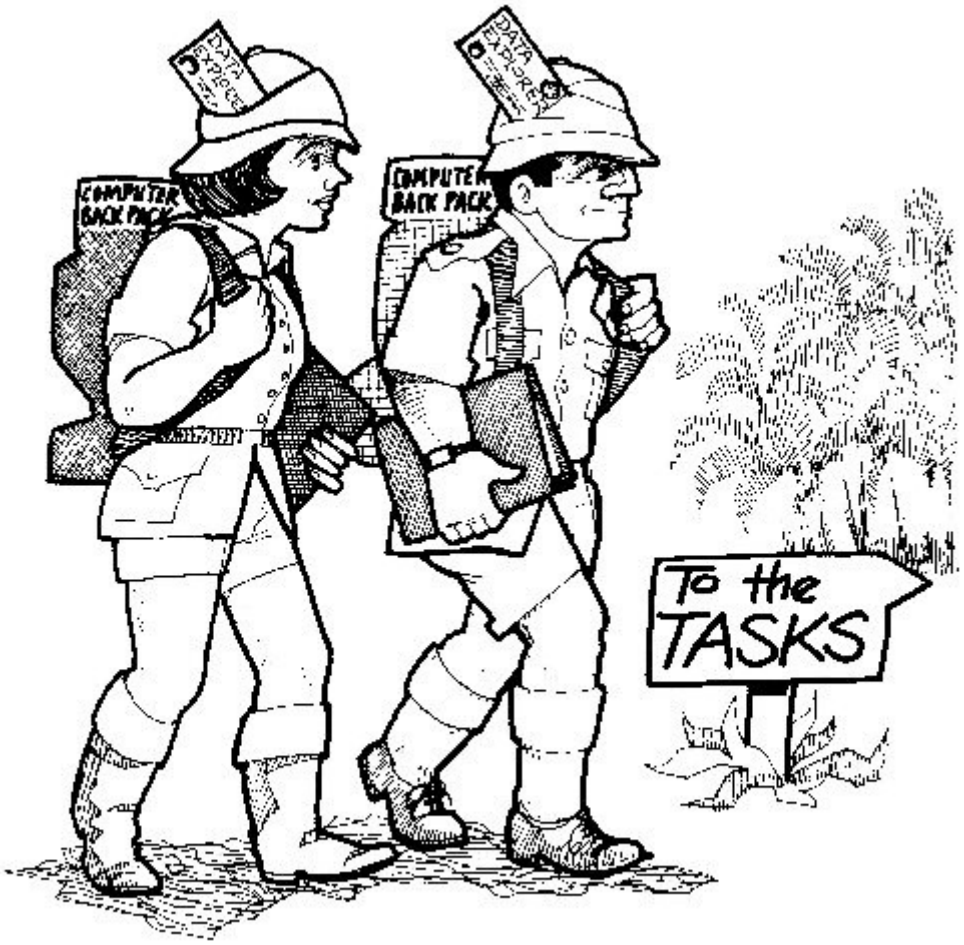
# Programming Concepts - C# Intro Directed Study




## Part C  Weeks 10-12  /  Units 1-3

Original work by Brian Ward

## Contents

# Learning Outcomes

## for the Programming Concepts module: CO452

**On successful completion of the module the student will be able to:**

- Analyse a simple requirement in a structured manner in order to establish a strategy to solve the current problem
- Design, document, implement and test reliable, maintainable programs as solutions to simple problems
- Use structured techniques of design and implementation and good documentation practice

Make effective use of software development tools when implementing fit-for-purpose solutions

# General Introduction

### Welcome to Programming Concepts

- The CO452 module plan and method of assessment for this module is detailed at the back of this booklet, and you will also find the assessment criteria there.
  We want you to enjoy this module and achieve a good result. Therefore it is important that you read the module plan and assessment criteria at your leisure.

- You will need an electronic A4 logbook to record your work. Please get this up and running as soon as possible.
  Classwork will be checked each week, and should be recorded in your logbook.

# Classwork

# Your Log Book

You should try to organise your log book clearly and logically.

- Put **Unit Headings** and **Task Headings**.
- Give a brief description of the task
- Stick in your commented source code solution
- You may sometimes need other documentation such as algorithms or test plans.
- Add brief comments as to your success or otherwise and any problems that occurred

This will become more important in later chapters.

# Using MS Visual Studio C# (Console)
# A beginner's guide

## Starting C#

- First get into Windows
- Double-click the **Visual Studio** icon on the desktop (or use Start/programs)
- The first time it is run, you will need to choose the **Visual C# Development Settings**

## Starting a NEW Project

- Select **File** from the menubar, followed by **New then Project**
- Select a **Visual C#** on the right hand side and then select template: **Console Application**
- Click **Browse** to find suitable **Location** for storing your project
- Enter a **Name** for your new project, then click **OK**
- Now you can type your program code into the new window

## Opening an EXISTING Project

- Select **File** from the menubar, followed by **Open** followed by **Project**
- In the **Open Project** dialog window, select the correct folder and **.csproj** file
- Click **Open** and the project should be loaded together with any source code **.cs** files
- To see the code, you may have to click the **.cs** file in the Solution Explorer window

## Compiling a Program

- Make sure the project and program are open
- Select **Build** from the menubar, followed by **Build Solution** (or use **F6** key)
    - Edit any syntax errors that show up in the bottom of the editor window

## Running (executing) a Program

- Make sure the project and program are open
- Select **Debug** from the menubar, followed by **Start Without Debugging** (or use **Ctrl + F5**)
- If the program doesn't work as expected, you have logical errors and have to correct these.

## Saving a Program

- Choose **File** from the menubar, followed by **Save All**
    - This saves to the location previously chosen

## Copying a Project

- If you want to take a copy of a project so that you can make changes while keeping the original, go to where the project has been saved and copy the whole project folder
    - This folder holds the many files required by the project .. paste it to a new location

## Closing down Visual C#

- Select **File** from the menubar, followed by **Exit**

## Running a Program without Visual Studio

- Open the project folder
- Open its **bin** folder
- Open its **debug** folder
    - Find the **exe** file for the project .. you can copy and paste this to a different area and double-click this to run your project without using Visual Studio.

# Unit 1: C# Input and Output

## Your Log Book

- Write the Unit Number and Task number as a heading
- Add suitable comments to your source code
- Stick in the source code for each solution
- Add a sample screen shot of the program running
- Write a brief statement on your success or any problems you had with your solution

## Classwork (5 Tasks)

Complete as many of the following tasks as you can in class. Ask your class tutor to check them as they are completed.

Before you start this work, copy some programs that have already been written for you. Your lecturer will tell you where to find them. At Bucks this is the **C#_BSc_Progs** folder on the **L: drive**. Copy the programs either to a memory stick or to your own network area

## 1.1 Town Check

Open the **Task1_1** folder, then **Task1_1.csproj**
(then compile and run the program using **Ctrl** + **F5**)

```
using System;

namespace Task1_1
{
   // author B N Ward
   // date : 24 Oct 2010
   class Program
   {
      // the main entry point for the application
      static void Main()
      {
            string town;

                  Console.Clear();              // clear the screen
                  Console.WriteLine("Hello There!");      // output to the screen

                  Console.Write("What town do you live in ? ");
            town = Console.ReadLine();      // input from the keyboard

            Console.WriteLine(town + " is a real dump!");
      }
    }
}
```

**Now PTO for your task:**

# Task 1.1

1. Change the program so that it starts by asking the user to enter their **name** and then inputs this **name** from the keyboard into a suitable variable.
2. Change the output so it looks like the following (using your own inputs of course):
   **Please enter your name :** *<Brian>*
   **What town do you live in,** *<Brian>* **?** *<High Wycombe>*
   *<Brian>* **is a lovely name**
   **But** *<High Wycombe>* **is a real dump!**

- Test your finished program with different inputs to see that it works correctly

---

## 1.2 Taxing Program

**Task1_2.csproj** (load this file into the editor window, then compile and run it)

```
using System;

namespace Task1_2
{
   class Program
   {
        // author B N Ward
        // date : 24 Oct 2010
        static void Main()
        {
           string input;        // string for input
           double price;        // item price
           double vat;          // vat on the item

           Console.Clear();
         Console.WriteLine("Hello There!");
           Console.Write("What is the item price ? ");
           input = Console.ReadLine();
           price = Convert.ToDouble(input);        // convert string input to a number

           vat = (17.5 * price) / 100;             // work out vat

           Console.WriteLine("\nThe vat on an item costing £" + price  + " is £" + vat);
           Console.WriteLine();           // print a blank line
       }
   }
}
```

# Task 1.2

1. Improve the program so that it will work for any possible vat **rate**. You can do this by asking the user to enter the vat **rate** from the keyboard.
2. Also get the program to enter the user's **name**. This name should be output when asking for the item price and vat rate. e.g.
   **What is the item price,** *<Brian>* **?**
   **What is the vat rate,** *<Brian>* **?**
3. Finally output both the vat and the total price for the item (on separate lines)

# 1.3 Course Marks

Project: **Task1_3.csproj**
(load into the editor window, then compile and run it)

```
using System;

namespace Task1_3
{
   class Program
   {
      // author :  B N Ward
      // date :  24 Oct 2010
      static void Main()
      {
         string input;                    // string for input
         int classMark, isMark, total;    // define 3 integer variables
         double average;                  // define 1 double variable

       Console.Clear();              // clear the screen
         Console.WriteLine("A Marks Calculation Program\n");
         Console.Write("Enter your Classwork mark : ");
         input = Console.ReadLine();
         classMark = Convert.ToInt32(input);       // convert to an integer number

         Console.Write("Now your Independent Study mark : ");
         input = Console.ReadLine();
         isMark = Convert.ToInt32(input);          // convert to an integer number

         total = classMark + isMark;

         average = total / 2.0;

         Console.WriteLine("\nThe total of " + classMark + " and " + isMark + " is " + total);
         Console.WriteLine("and the average mark is " + average);
         Console.WriteLine();                  // print a blank line
      }
   }
}
```

## Task 1.3

1. Modify the program so that it also enters a **project mark**

2. As well as the average, a **final mark** is calculated based on 60% classwork, 25% independent study and 15% project. Make all the necessary changes and addition

3. The final output should show the **total** mark, **average** mark and **final** mark.

4. Get the program to force the calculations to always be displayed using **2 decimal places**. (Hint: See Lecture notes and Appendix A at end of this pack)

# 1.4 Salaries

- Now you must create a NEW project (**see page 4 for how**) to do the following.
- Start the program by using the following code to change the console colours used:

```
Console.BackgroundColor = ConsoleColor.Red;
Console.ForegroundColor = ConsoleColor.Yellow;
Console.Clear();       // clear the screen using these new colours
```

- The program then asks the user to enter their weekly **salary** (stored as a double)
- The monthly and annual salaries are then calculated, assuming 4 weeks in a month and 52 weeks in a year
- The weekly, monthly and annual salaries are then displayed - using suitable output statements for each
- Test your program to see that it works properly
- All money should be displayed using 2 decimal places.

# 1.5 Rock Concert Tickets

The Student Union takes bookings for university concerts ..write a simple ticket program for them.

- Get the program to ask the user to enter the name of the group playing at the concert (a string), the concert date (as a string), the number of tickets required (an int) and the cost of each ticket (a double).
- The program then calculates the total cost (a double) for the group ticket
- The screen then clears using some colours of your choice
- all 5 details for the group ticket are displayed with spacing something like this:

```
Bucks New University                                  February 4th

                        Rolling Stones Concert

Number of Tickets: 5                          Cost Per Ticket:
£4.50

                            Total Cost :  £22.50
```

**Some Hints on displaying the ticket details**

- Use **Console.Clear();** to give you a clear screen
- You can then use **\n** and **\t** to position items on the screen or you can use **Console.SetCursorPosition(.. , ..);**.

Ask your lecturer or consult the 'Some Extra Useful C# Stuff' page in **Appendix B** of this pack At the end you can pause the program with **Console.ReadKey();** to avoid having 'Press any key to continue' at the bottom of your screen.

# Unit 1: Independent Study (4 Tasks)

The following are to be done individually and independently, in your own time. They will be assessed at a future date (see the module plan)

## 1.6 Rocket Game

A game starts by filling up a rocket with fuel. Your program should work out how much fuel is needed to fill the fuel tank, the cost of this fuel and how long this rocket fuel will last in the game.

- First input the following 4 items:
  - The length of the fuel tank in metres
  - The width of the fuel tank in metres
  - The height of the fuel tank in metres
  - The price of the rocket fuel (how many credits per cubic metre)
- The program should then calculate and display the volume of fuel needed to fill the tank.
- It should also calculate and display the total cost to fill the tank (in credits)
- Then it should calculate and display the time in seconds that the rocket fuel will last.

**Helpful Hints**

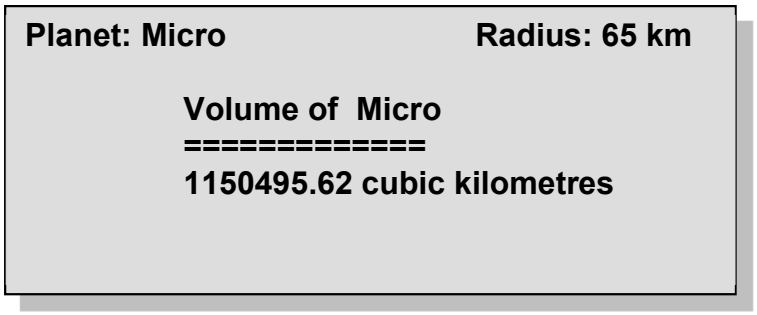
- the volume of any rectangular object is length x width x height.
- assume that rocket fuel burns at a rate of 6 cubic metres per second and define this in the program as a constant.

Test your program with a suitable set of values. Put the source code into your logbook.

## 1.7 Planetary Sums

Create a new program to enter the **name** of a planet and its **radius** in kilometres.

- The program should then work out the **volume** of the planet using a suitable formula (see below) and display all the information as shown below
- Select some appropriate Console colours for your display




**Hints**

- **Volume of a sphere = 4.0/3 x Pi x (radius)$^3$**
- There is a **Math.Pow()** function that should be used to work out the cube of the radius.
  - See the extra programming notes in Appendix A at the end of this booklet
- **Pi** should be given the constant value of 3.142.
- The volume should be output to 2 decimal places.

**Note: Try using 4 and 4.0 in the formula .. describe what happens and explain why**

## 2.1 Game Tournament

You are to **design** and **build** a program for the following:

*"A Games Tournament is being run with teams of players. Write a program that inputs the total number of players and the size of a team. The program calculates the number of teams and how many players (if any) are leftover without a team."*

**Program Hints**

- You should use **int** variables throughout
- Use the modulo operator (**%**) to find the **remainder** (see lecture notes and/or lecturer)

## 2.2 Weapons Store

You are to **design** and **build** a program for the following scenario:

*During the course of a game, a player can pick up Red, Green or Blue tokens which can then be used to purchase weapons at a Weapons Store.*

- *Red tokens are worth 5 credits*
- *Green tokens are worth 10 credits*
- *Blue tokens are worth 20 credits.*

*The program asks you to enter your **name** and then how many **Red** tokens you have, how many **Green** and then how many **Blue**.*
*The program calculates the total **credit value** of all the tokens and how many **arrows** can then be purchased (arrows are 50 credits each).*
*The program then displays the player's **name**, the number of **arrows** purchased and the number of **credits left** after the purchase*