C++ Week 2 Functions

Introduction

During this week's practical we shall develop a Noughts and Crosses game. This will involve creating a number of functions and manipulating data contained within an array.

Create a new project and define the following **global** variables and constants.

```
const char PLAYER_1 = 'O';
const char PLAYER_2 = 'X';
char board[3][3];
```

The two constants represent the symbols used within the game and the two-dimensional array the board on which the game is played.

Exercise 1

Above the main function create two functions

1. void Initialise() That sets all the elements within board to '-' to indicate that the square is free. (see presentation).
2. void Display() That displays the board on the screen.

From within main invoke Initialise and then Display. The output of Display() should be formatted to resemble that below. Hint use the tab escape character \t

```
-	-	-
-	-	-
-	-	-
```

Note: When using **cout** you will need to include <iostream> and indicate you are using the std namespace.

Exercise 2

Now create two more functions

1. void SetValue(int row, int col, char symbol) That sets the cell indicated by the parameters to the value that is contained within the symbol parameter.
2. Bool IsFree(int row, int col) That returns true if the cell identified by the parameters contains '-'.

C++ Week 2 Functions

Exercise 3

Change the code located within main to implement the following algorithm, choosing appropriate types for the variables..

```
Seed random number
Set gameOver to false;
Set row to 0
Set col to 0
Set player to PLAYER_1
Set count to 0
Call Initialise
Call Display
While(not gameOver)
        Do
                Set row to random number 0..2
                Set col to random number 0...2
        While (not Call IsFree(row col))
        Call SetValue(row,col,player)
        If player = PLAYER_1
                Player = PLAYER_2
        Otherwise
                Player = PLAYER_1
        End if
        Add one to count
        If count = 9 set gameOver to true
        Call Display
End while
```

Now run the program. The logic within the main function plays the game however, it will not pick up a winning combination and consequently will only finish when all the squares are occupied.

Our next task is to determine if a player has won the game. One approach to this would be to record all the winning 3 cell combinations within an array and then check this against the current state of the board.

C++ Week 2 Functions

Exercise 4

First let's create the array to store the winning combinations. Define it as a global variable just under the constant definitions.

```
int winningStates[8][3][2]=  { {{0,0},{0,1},{0,2}},
                               {{1,0},{1,1},{1,2}},
```

Only two of the eight winning combinations have been included, so you will need to complete the remaining 6. Remember the order is {row, column}
So the first winning combination {{0,0},{0,1},{0,2} is the top row.

Exercise 5

Now lets create the function that will determine if a player has won.

The signature is

bool HasWon(char symbol)

The algorithm is

```
Set row to 0
Set col to 0
Set match to false
For m = 0 to 7
        For s = 0 to 2
          Set row to winningStates[m][s][0]
          Set col to winningStates[m][s][1]
          If board[row][col] = symbol
                Set match to true
           Otherwise
                Set match to false
                Break
           End if
        End loop
      If match = true return true
End loop
return false
```

Now invoke HasWon within an appropriate place within main assigning its return value to gameOver. Test the program a number of times to ensure it is working correctly.

C++ Week 2 Functions

Exercise 6

Rather than making the board array a global variable we could declare it within main. To provide the relevant functions access to it we would have to pass it as an argument to their parameters. Make the appropriate changes.

Exercise 7

A number of the functions are passed information that should not be altered within the function. Make appropriate changes to ensure this rule is enforced.

Exercise 8

Modify the HasWon function to return the coordinates of the winning cell combination as a two-dimensional array parameter. The main function should display a message indicating which player has won and the cells that make up the winning combination.