# Algorithms and Data Structures: Assignment #1

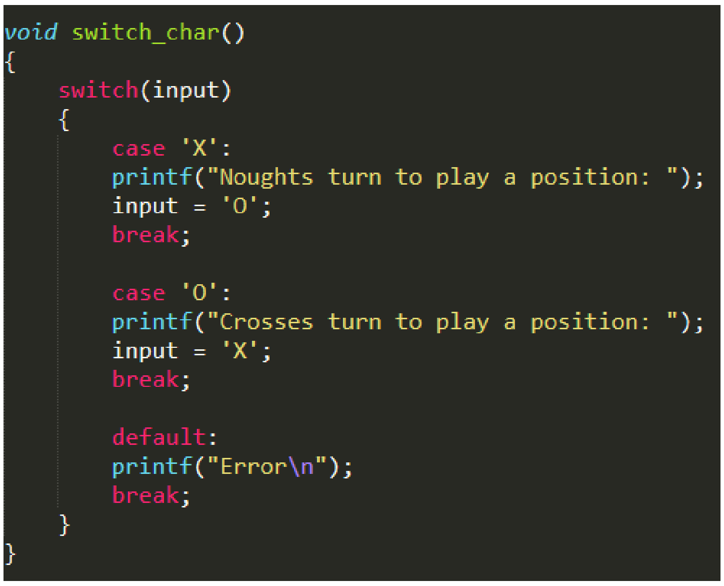
## Introduction

In this assignment we were tasked with creating a noughts and crosses game using algorithms and data structures. I have created a simple noughts and crosses game using array's and array manipulation using functions in the source code, such as an initialise board, insert, check winner and swap player functions which allows the game to be played between two players.

## Design

During the design phase of my assignment I had came up with the idea of using a series of different functions to manipulate a singular dimensioned array. To represent the board, I had used a single dimensional array as I felt this was easier to work with an insert method in order to insert a character such as 'X' or 'O'. To represent the players, I had used a single character that was switched using a function with a switch case between each player move. I felt that using an array for the board was a simple choice to make as I knew I could use the indexing of the array for the controls. I felt the simplest controls were to use the numpad as both the board and numpad are placed in a 3x3 grid.

After each move, the player's character is swapped over. To achieve this I used the following function as shown below.

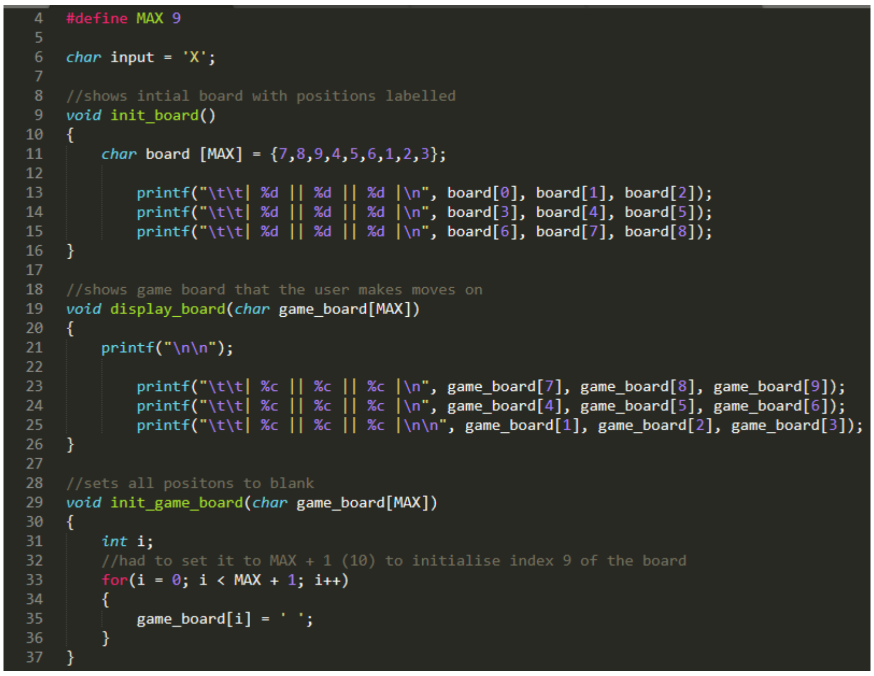


```
void switch_char()
{
    switch(input)
    {
        case 'X':
        printf("Noughts turn to play a position: ");
        input = 'O';
        break;

        case 'O':
        printf("Crosses turn to play a position: ");
        input = 'X';
        break;

        default:
        printf("Error\n");
        break;
    }
}
```

This switch case is then called in the main function inside a while loop, which means that after each turn the player is switched around.

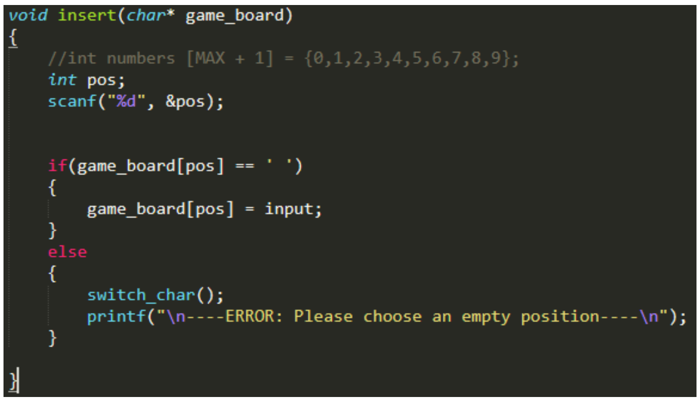
To start off with my project I had created a board which shows the positions with the controls labelled, this is displayed at the start of the game to inform the user on how to play the game.

I then had created a “game_board” array in the main function, which is then initiated in a separate function setting each position equal a blank space, and then another function so that it can be displayed for the user.



```c
#define MAX 9

char input = 'X';

//shows intial board with positions labelled
void init_board()
{
    char board [MAX] = {7,8,9,4,5,6,1,2,3};

        printf("\t\t| %d || %d || %d |\n", board[0], board[1], board[2]);
        printf("\t\t| %d || %d || %d |\n", board[3], board[4], board[5]);
        printf("\t\t| %d || %d || %d |\n", board[6], board[7], board[8]);
}

//shows game board that the user makes moves on
void display_board(char game_board[MAX])
{
    printf("\n\n");

        printf("\t\t| %c || %c || %c |\n", game_board[7], game_board[8], game_board[9]);
        printf("\t\t| %c || %c || %c |\n", game_board[4], game_board[5], game_board[6]);
        printf("\t\t| %c || %c || %c |\n\n", game_board[1], game_board[2], game_board[3]);
}

//sets all positons to blank
void init_game_board(char game_board[MAX])
{
    int i;
    //had to set it to MAX + 1 (10) to initialise index 9 of the board
    for(i = 0; i < MAX + 1; i++)
    {
        game_board[i] = ' ';
    }
}
```

To complete the insert function, I used an if statement so that if the player has chosen a position that has already been played then it calls the switch player function and displays an error message.



```c
void insert(char* game_board)
{
    //int numbers [MAX + 1] = {0,1,2,3,4,5,6,7,8,9};
    int pos;
    scanf("%d", &pos);


    if(game_board[pos] == ' ')
    {
        game_board[pos] = input;
    }
    else
    {
        switch_char();
        printf("\n----ERROR: Please choose an empty position----\n");
    }

}
```

The insert function uses a scanf statement in order to give the position, which is the index of the array. This is then inserted into the array.

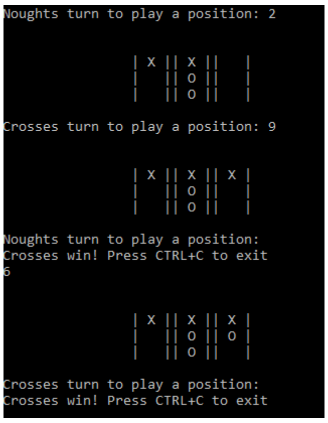
## Enhancements

Features that I would include if given more time would have been a stack in order to record and replay previous games. I would have also implemented a method where the board would update instead of displaying an updated board each move. I would also implement a feature for each player where they could input their names at the start and then use that to declare the winner at the end of the game. Another feature I would implement would be the option to play another game, instead of just exiting the program.

A feature that I would improve in the project currently would be the check win function. In the function I have created a bundle of if statements which I feel looks messy. Initially I had created a 2D array containing each of the winning moves that can be played but could not figure out how to create a loop for the game board and check if a player has won.

## Critical Evaluation

I feel that my check win function can be greatly improved due to the fact it contains 8 if statements which could have been avoided using a for loop to loop through an array of winning combos in order to check the winner. This would greatly increase the readability of my code and reduce the lines that the function takes up.  Another feature that can be greatly improved would be the check win function as it is unable to break out of the while loop. This is shown below



```
Noughts turn to play a position: 2


                | X || X ||   |
                |   || O ||   |
                |   || O ||   |

Crosses turn to play a position: 9


                | X || X || X |
                |   || O ||   |
                |   || O ||   |

Noughts turn to play a position:
Crosses win! Press CTRL+C to exit
6


                | X || X || X |
                |   || O || O |
                |   || O ||   |

Crosses turn to play a position:
Crosses win! Press CTRL+C to exit
```

I feel that a strength of my program would be the controls in which the user interacts with the game. The game board can be physically represented on the keyboard with the use of the numpad, this allows for a quick game and less confusion. I also feel that the game board's look nicely displayed and easy to read from due to using tab characters to centre the board to the middle of the command line.

## Personal Evaluation

Through this assignment I have learned a lot about arrays and array manipulation, however I feel that I could have performed better by extending my knowledge on certain things and had spent more time on implementing a stack for the additional features to bring my mark up. One challenge id faced was implementing the function for checking the winner and feel I could have performed better in creating a more efficient algorithm to scan through the board's indexes by doing more background reading and spending more time on it.

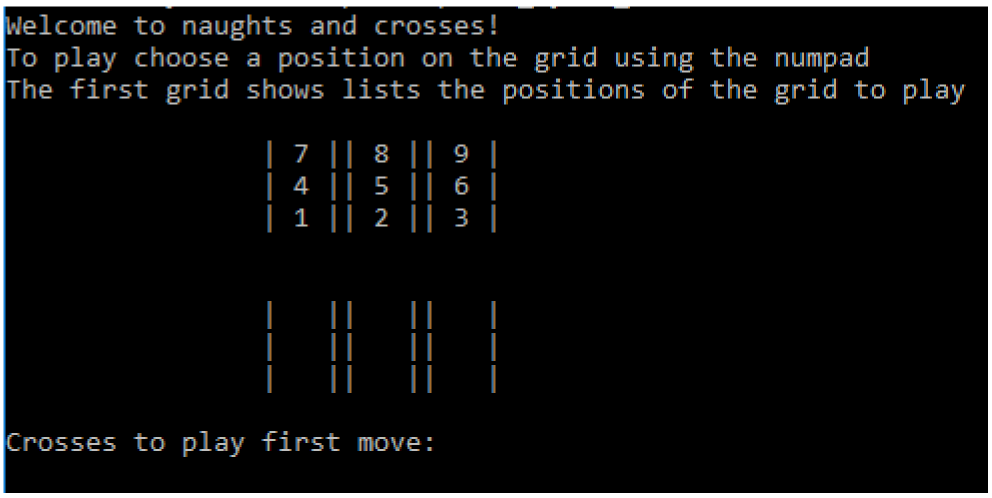
Throughout the coursework I have applied knowledge that I have gained through the practical labs we had completed in class as the API for lab three was like what I had to do in terms of an insert method. I was required to do some research on how to initialise an array of characters as I had initially set each index equal to '0' but this caused problems later and then had to change it to a blank space, allowing the user to input a character into a specific position.

Something I think I did well would be the keeping the code tidy for the most part in the main function by using separate functions. This allows the code to be easily read and maintained in the future and if I was interested in extending the program I could do so easily.

I feel this coursework has made me understand different functions and how to use data structures for achieving a goal in a different way than you had previously thought you could. This coursework has also made me realise the importance of different data structures as it is a lot more efficient and can be used alongside other known algorithms to fix common programming problems.

I feel I have performed well in this coursework with room for improvement over certain areas which can be solved by more background reading or spending more time in general on the assigned tasks.

Screenshot of the start of the game below



```
Welcome to naughts and crosses!
To play choose a position on the grid using the numpad
The first grid shows lists the positions of the grid to play

              | 7 || 8 || 9 |
              | 4 || 5 || 6 |
              | 1 || 2 || 3 |


              |   ||   ||   |
              |   ||   ||   |
              |   ||   ||   |

Crosses to play first move:
```