Report Coursework C

1. Introduction

The aim of the coursework is to demonstrate the understanding of both theory and practises in relation to the content of the Algorithms and Data Structures module. The task is to implement a text-based Tic-Tac-Toe game using the C programming language. In this coursework we are going to implement a game board to play, a recording history in order to replay the game, features such as undo/redo moves on the board.

2. Design

Explaining how you designed & architected your software paying particular attention to the algorithms and data structures used.

First of all we need to represent the game state, I decided to implement an array one dimensions of 10 elements 0 to 9. Basically the player chose a number on the board (0-9) and he/she will see a symbol appear in the number. In fact I preferred 1D array to 2D because it's easier to use in order to create the board and making the position choice for the player. When a player make a move the position is going to be quick and efficient. We also know that an array of one dimension take less memory than a 2D array. Usually a disadvantages of arrays is that we need to know how many elements are to be stored, here we know that a board of Tic-tac-toe is 3x3 so for a 1D array 9 elements are required. I especially paid attention to the fixed size of arrays in order to not waste memory neither create problem by allocating less memory. Since the elements of arrays are stored in consecutive memory locations insertions and deletions are time consuming so I decided to implement stack in order to allow the game to support undo and redo function.

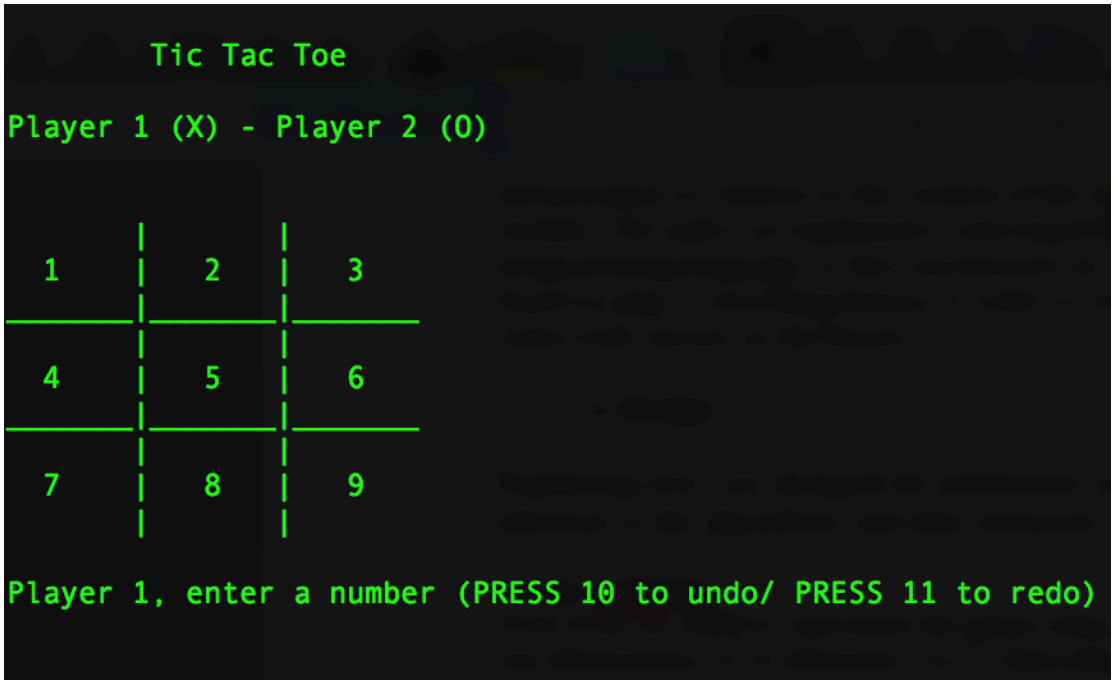

Figure 1: Undo/redo option for players

As we can see in the previous figure both players can undo their moves by pressing 10 and redo their move by pressing 11. I implemented a function for each, the undo consist of using the stack in order to remove the marks and choices made by the player during the game. Basically we use our pop from stack to remove the appropriate choice made eg. If the choice was one, the stack is going to look for the move one of the board and cancel it.
The standard is to keep the objects in a stack to support multi level undo. In order to support redo, a second stack has been implemented to keeps all the commands I have undone. So when I pop the undo stack to undo a command, you push the same command you popped into the redo stack. You do the same thing in reverse when you redo a command. You pop the redo stack and push the popped command back into the undo stack. This is because the stack is implemented with Last in first out order which is equivalent to redo-undo feature.