Edinburgh Napier UNIVERSITY

Coursework Report

Viktor Kanev
40282142@live.napier.ac.uk
Edinburgh Napier University - Algorithms and Data Structures (SET08122)

1 Implementation of TicTacToe game in C programming language

2 Introduction

The aim of this report is to evaluate and justify the use of appropriate data structures in the implementation of the famous TicTacToe game in C programming language. The main features of the game that had to be included are:

- Playable game
- Undo moves up to the begining of the game
- Redo moves up to the last entered move
- Replay previous games

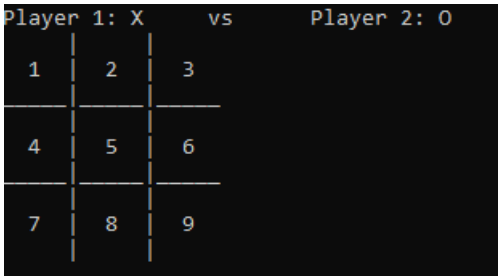
The game I have created can only be played with 2 players. It also has the ability to replay previous games, which are saved to a text file at the end of each played game.

3 Design

In this section I will explain which data structures I have chosen to use for the implementation of the game and why.

3.1 Board

For the board I have chosen to use one dimensional array as opposed to 2D array, because it has better time complexity when looping trough the array. It is also more intuitive when playing as the players would only need to enter the number of the board on which they want to place an 'x' or an 'o', rather than having to enter coordinates for each entry. This is achieved by using the index of an element of the array and accessing it directly. For example: **TicTacToe[0]** is the first square of the grid, **TicTacToe[1]** is number 2 and so on.

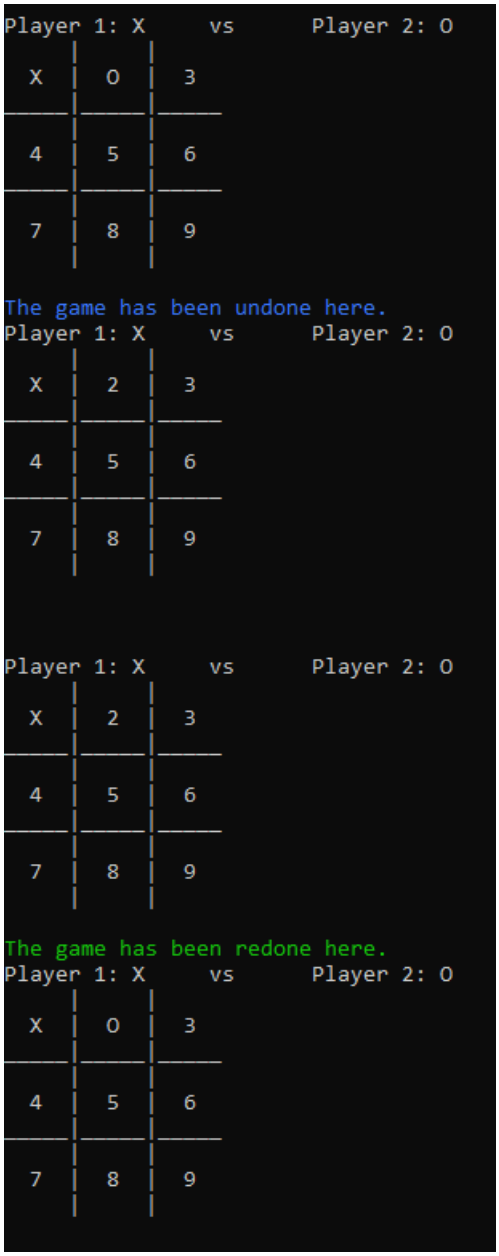


3.2 Undo & Redo

Using an array for the board meant that I would need to create stack structure for the undo and redo functions so I can traverse forward and backward in the list where the player's choices are being stored. Furthermore items are added or deleted in Last in First out order, so it only make sense to use this structure if you want to access or delete the last entered item. For this reason I have included four stacks each of which stores the undone choices, undone marks, redone choices and redone marks. Each move of the players is stored in **undoneCh** stack as well as it's mark(**undoneM**), whilst the redone choices and marks are being deleted from the list as they are no longer relevant. When a player undoes a move the choice and the mark are poped from the undo stacks and then pushed on to the redo stacks. The redo process is vice versa except that it doesn't delete the undone choices, as it it keeps popping elements from the list as the player undoes moves.

3.3 Replay

For the replay function I have used **replayTTT()** function which is basically the same as the play(**TicTacToeGame()**) function except that it gets the input from an array rather than from the console input. When a game is played each choice made by a player is saved to a txt file in the main directory of the program. Each player's choice is then included inside with comma(,) used as delimiter. Each text file is then appended with the current time in the form of an integer, so that every text file has a unique name and it doesn't overwrite them. When a user tries to replay a game from a file all the entries in the file are extracted from it and then placed into an array. The game loops trough all of the indexes of the array then uses the number as an input for the current move. If a player undid or redid a move, those are also recorded in the text file and then on the replay it is indicated if a player undid or redid a move. For the replay function I have also added a **delay()** function which delays each entry in the loop with 2 seconds, so it can be easily followed when watching a replay.



4 Enhancements

There are plenty of features that can be added in order to enhance my current program. A really nice feature would be to give the ability for the user to select the size of the board as now it can only be played on a 3x3 board, this would mean that you would also need to specify new winning conditions, as the board may be uneven(3x6).
Another nice addition to the game would be to make it playable against the computer, this can be easily achieved just by selecting the computer move to be a random location on the grid, however in order for it to be truly efficient a minimax algorithm may be used, so that the computer makes "intelligent" choices.

5 Critical evaluation

The game functionality works well as it stores all the moves correctly, also by using a one dimensional array the time complexity is BigO(I) meaning that every element has equal access time, so the game is robust, however by using an one dimensional array for the board I had to use another structure such as Stack in order to implement the undo and redo functions, so that I don't have to iterate over the array multiple times and copy the new array each time. The undo and redo functions work as intended and are quite robust as well. They are implemented using stacks, so it is easy to work with the last item on the stack, which is all you need for the undo and redo functions. The replay functions works as intended as well, however there is some room for improvements, in terms of speed and how the moves are displayed.

6 Personal evaluation

I found this assignment quite difficult as I have very little experience programming in C, so I had to learn the basic data structures used in this language as well as how to read and write files, using structs and using stacks. The biggest challenge I have face was trying to implement the undo and redo features, as the for the game board I have used an array and it was quite complex to implement as it was, so I have created stacks for the undo and redo, which I have found quite easy to work with. I have watched numerous videos on C programming and read numerous articles and stackoverflow topics in order to overcome the difficulties I have faced when trying to implement a certain feature. I feel I have performed reasonably well, however there is always room for improvement and I am sure the code can be significantly reduced if I had more experience in coding in C.

References

https://www.youtube.com/watch?v=KJgsSFOSQv0 - **C programming full course(basics)**
https://www.youtube.com/watch?v=hiG5G2caZ38 - **Reading integers from file to an array**
https://www.geeksforgeeks.org/stack-data-structure-introduction-program/ - **How to use stacks**
https://stackoverflow.com/questions/8675119/conio-h-doesnt-contain-textcolor - **Changing the color in the console**
https://www.geeksforgeeks.org/arrays-in-c-cpp/ - **Arrays in C/C++**
Simon Wells - **workbook.pdf**