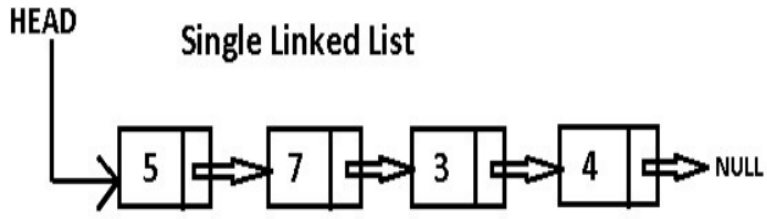
COURSEWORK REPORT

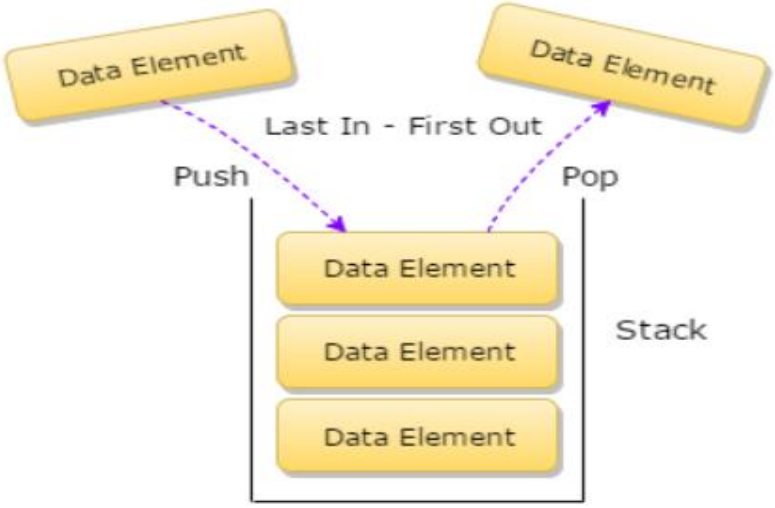
Introduction

The project given is to develop simple noughts and crosses terminal game, implemented with the C programming language using various data structures. Besides the base concept of the game our programme should include enhanced functionality and the ability to undo or redo a move that has been made, with additional option to replay a game. The aim of the following report is to clarify decision making behind the game implementation and justify the use of appropriate data structures.

Design

My initial idea was to use object-oriented design to implement the game, however due to C having poor support for object-oriented programming I encountered many problems and difficulties I could not resolve in time and with the deadline approaching I had to restart and settle with something simpler. For the board I made use of single dimensional array, due to game's fixed size of 9 places available and the ease of accessing each element directly by index. Once the core game was working, next step was the extra functionality of undo and redo. My original plan was to implement stack using linked list, "pushing" the board state and the current player to save it and simply "popping" the last element from the stack and revert it to the previous state it was. The advantage of using linked list being unrestricted size, with the small disadvantage of taking up more memory upfront to save the pointer to the next node in the list. Since object orientation did not work, I ended up saving only the individual moves made, as for the redo functionality I needed another stack to "push" the undone moves.





For the final feature the replay, I serialized the data into a file to make it persistent and utilized the "'queue'' data structure, since it makes perfect sense the first element in, is the first element going out. For the same reason as with the stack I implemented it using linked list to be able to dynamically allocate memory for the size.

Enhancements

With little effort, simple non-intelligent AI can be added making random moves on the board, but after researching AI and the ‘’Minimax algorithm’’ in particular. It has perfect use in simple noughts and crosses where if there is a wining move it is se to the maximum or the minimum denoting the possibility of loss.

Critical evaluation

I believe, I was able to satisfy the minimum requirements to complete the project and correctly identify and use the appropriate data structures to complete the features. In my opinion stack and queue implementation using linked list works perfectly, because it is dynamically allocating memory and unlike the array implementation it does not have a fixed size which can limit the user or pre-allocate too much memory for the task. On the other side, what does not work so great is validation of the user input and serialization, since C dos not support serialization and with using fgets or scanf there are various problems which may put you into infinite loop, example being the newline coming with fgets.

Personal evaluation

It was interesting exploring various data structures more in depth, one of the best “C” features is working directly with the computer’s memory, linked lists consisting of pointers is perfect practice for that. I already knew most of the basic concepts of the structures used, but it did not cross my mind that queue can be implemented using circular linked list, or what can be gained or lost using array or list implementation. One area that “C” lacks is object-oriented support, which is unfortunate having a look at the “command pattern’’.

References

Minmax - https://en.wikipedia.org/wiki/Minimax

Data structures - https://www.geeksforgeeks.org