Algorithms and Data Structures Coursework 1 Report

Jonathan Binns

40311703

Introduction

Describing the problem and giving an overview of features

The main objective of the task is to create a functioning naughts and crosses game, while displaying our skills in implementing different data structures and algorithms to act on those data structures. In order to create naughts and crosses I need to represent the different aspects of the game, these are the game board, the players, the pieces and the positions the pieces are played at. These can be done with different data structures and how I will represent these will be discussed in the design section.There will also be some extra features implemented in this game, the first is the ability to display instructions so the user knows how to play the game. The second is that I will decide which player goes first by ‘flipping a coin’. The third feature is adding the ability to undo and redo moves made so players can fix any mistakes they make. Finally the last feature will be the ability to record games so they can be replayed at a future time.

Design

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

What I need to represent/how I represent them

My design approach centers around what needs to be represented in this game and how I could go about representing them.

Board/1d array

The first, and most important, thing I need to represent is the game board itself. I will use a one dimensional array of nine spaces to do this. The main reason for this is that it’s the simplest approach which means I can build my functions around it easily as I can calculate which positions in the array equate to each position in a board. It also helps to minimize the amount of loops I will need when dealing with it, as there is only one ‘row’ I only need one loop to go through each element.

Pieces/chars

The next thing I need to represent is each piece played by the users. For this I will be using a char, either an ‘X’ or an ‘O’. This means I can make the array that represents the board an array of chars. My reason for representing moves like this is that a naughts and crosses game is typically played by writing an ‘X’ or an ‘O’ so by representing each move with those characters it allows the game to look as normal as possible.

Positions/ints

Each piece also needs a place on the board to go, to make sure they go in the intended place I will represent the board positions as ints. These ints will correspond to a position in the array meaning that an insert function can take in a char and an int positioning the char, representing the move, in the place the int dictates in the board array. This is why I chose a one dimentional array to represent the board as it makes adding moves as simple as possible as each position in the array is only accessed by one number, allowing moves to be inserted quickly as there is no need for a loop.

Players/chars

Each player will be represented as a char, just like their moves. This can be kept track of easily with a for loop that changes which player’s turn it is to enter a move.

Undo/redo feature/stack

My undo and redo move features will be represented by a stack. My plan is to have a stack of boards and add a board after a move has been made. This means that when calling the undo or redo feature the top board on the stack just has to be made equal to the board that is currently in play. This means that there is no need to represent moves as individual structs and it takes out any errors in terms of people wanting to undo a move that changes a space on the board that is not null.

Replaying games/stack/.txt

Finally to allow the user to replay games at a future date I will add the stack of all boards to a .txt document at the end of each game. This is the best way to represent this as I already have to keep a stack for undoing moves so printing this to a .txt document is not a hard extension of that.

Enhancements

Describing the features that you would add or improve if you had more time

Loops instead of ifs for checkWinner(), leads to scaling the board up to connect 4 or whatever

Critical evaluation

Explaining the features that you feel work well, or work poorly, and why you think this. You should support your evaluation with experimental results.

Personal Evaluation

Reflecting on what you have learned, the challenges you faced, the methods you used to overcome challenges and how you feel you have performed.

References