2022 - 1806ICT Assignment

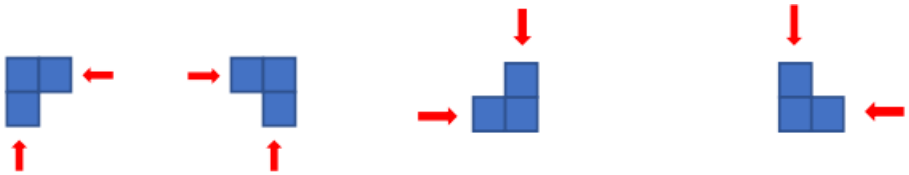
Milestone 2, Part 1

Tiam Lamb (s5259308)

Contents

Problem Statement

The goal is to create a program in C that does the following: A player can input both a board size (length x length) and a symbol amount for a game where the player is able to swap symbols within a grid, to attempt to make one of four “L” shapes shown below. Once the player’s move is successful in creating one of the four shapes, the player will receive 10 points and the symbols will be removed from the board. The gaps in the board will be filled with existing symbols from one of the four directions on the board shown by the arrows below. And new symbols will be generated in any gaps that exist once the symbols have been moved. The game is played with a timer, and the final score is determined by the (number of points / the time spent playing in seconds).



Requirements Section

User Requirements

The user should be able to:

1. Run the program directly from the command line.
2. Start a new game or restart a previously saved game.
3. Input a size for the board.
4. Input the amount of symbols wanted for the game.
5. View the board at the beginning of the game.
6. Input both, to and from coordinates to make a move.
7. View an error if any input is invalid or a move is not possible.
8. View the points gained by each move as well as the total points accumulated.
9. View the board after each move.
10. Quit, save or start a new game at any point.
11. View the final score after the game.

Derived Requirements

1. The program shall be run by compiling milestone2p1 then typing ./a to run.
2. The user shall have the option of entering the parameters for a new game or being able to restart a saved game.
3. The user will be prompted by a print output, and will be asked to input a size for the board using which will be received by scanf.
4. The user will be prompted by a print output, and will be asked to input number of symbols for the board using which will be received by scanf.
5. The board will be printed out at the start after being generated by a function.
6. The user will be prompted by a print output to enter both to and from coordinates one at a time which will be received by scanf.

7. After each input the program will check the input within a while loop. If it's a valid input no error message will be printed out and the loop will end.
8. After each move the total score will be printed out as well as the score for the move.
9. After each move the board will be printed out again using a function.
10. After each input the input will be checked to see if the user would like to save, load or create a new game, if so the relevant functions will be called to excute that part of the program.
11. After the game, the final score will be calculated and printed out.

Software Design

Program I/O

Input size when prompted at start of new game
- Determines size of board

Input symbol amount when prompted at start of new game
- Determines how many symbols will be used in the game

Input from and to coordinates when prompted while playing
- Determines the symbol coordinates that you want to move to and from

Input s (filename) at any time
- Saves the current game with specified file name

Input n at any time
- Creates a new game and overwrites the current one

Input l (filename) at any time
- Loads the previous game from the file specified and overwrites the current one

Data Section

inputFile
- File
- Used to read the data of previous game
- Used in loadGame

outputFile
- File
- Used to store the data of the current game
- Used in saveGame

board
- 2D array
- Used to store the symbols in each row and column
- Most functions use it (functions shown below)

size
- Integer.
- Stores the size of the board.

- Most functions use it (functions shown below).

points

- Integer.
- Stores amount of points the player has.
- saveGame, loadGame and shapeFound use it.

symbols

- predefined array of symbols.
- used to store symbols that will be used to make up the board.
- Most functions use it (functions shown below).

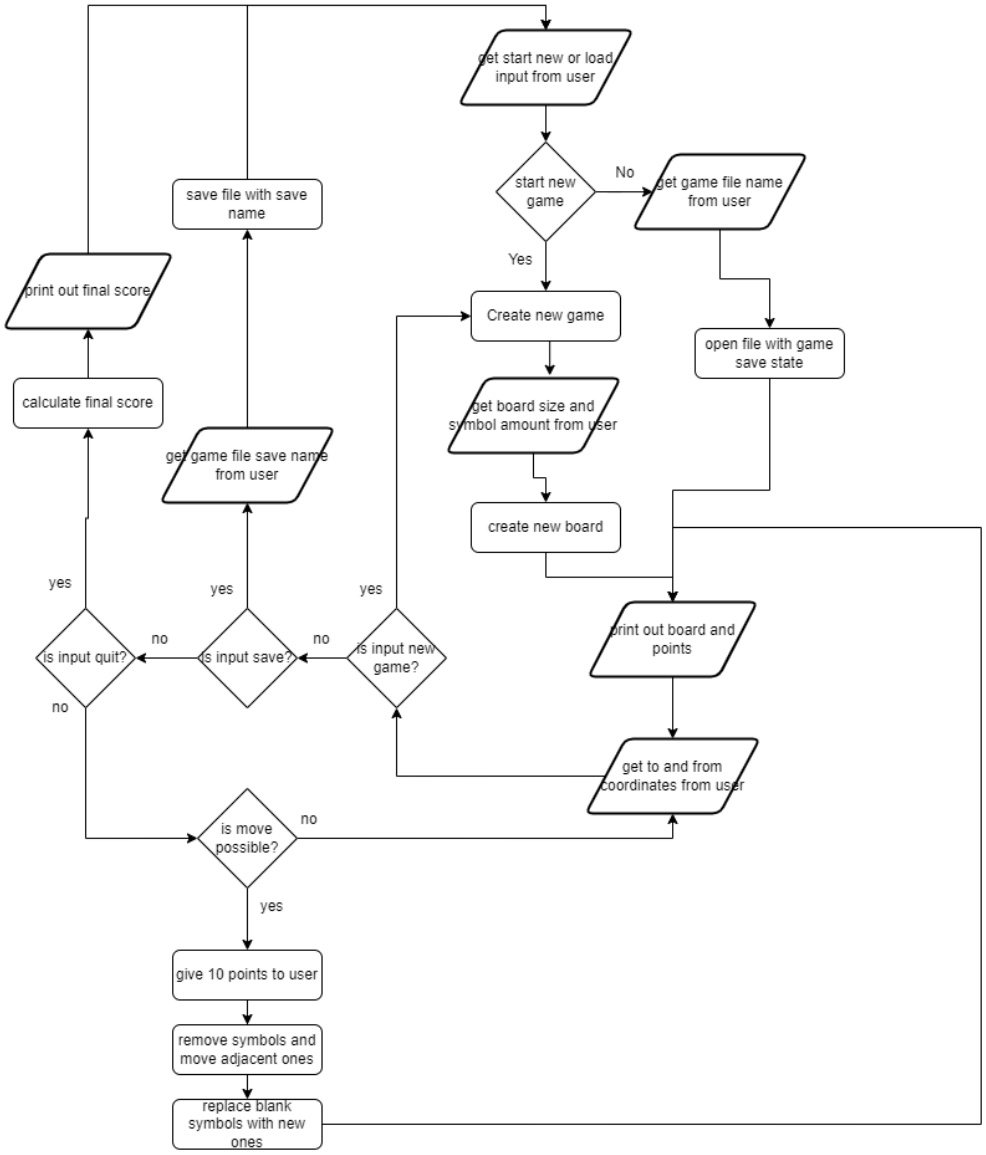
running

- Boolean.
- Set to true when game is being played and set to false when game is required to end.

fromCoordR, fromCoordC, toCoordR, toCoordC

- Integers.
- Used to store the row and column locations inputted by the user to make a move.
- Used in getCoords, swapBoard and checkIfShape.

High Level Design



Program Functions

List of all functions in the software.

checkIfShape()

- checks if the input location is part of a "L" shape by checking the symbol above or below the location and then checking the adjacent symbols to see if it matches the symbol in the location specified.
- input parameters: char board (game board), int row (row position), int col (column position), int size (size of board)
- doesn't change any global or member variables
- returns true is there is a shape at specified location if not it returns false (boolean function)

getCoords()

- asks user to input to and from coords to move symbols on the board. Checks if coords are valid inputs if not gets user to reinput them.

- input parameters: int fromCoordR (from row coordinate), int fromCoordC (from column coordinate), int toCoordR (to row coordinated), int toCoordC (to column coordinated), int sizeOfBoard (size of the board)
- changes all coordinate variables to inputs determined by user. Only when returned valid.
- No return value, void function.

swapBoard()

- Swaps symbols from the two coordinates specified by user, then checks if there is a shape when swapped by running the checkIfShape() function. If there is no shape then it will swap them out.
- input parameters: char board (game board), int fromCoordR (from row coordinate), int fromCoordC (from column coordinate), int toCoordR (to row coordinated), int toCoordC (to column coordinated), int sizeOfBoard (size of the board)
- changes the board 2D array if successful otherwise it stays the same
- returns true if swap is successful and false if not. (bool function)

shapeFound()

- replaces shape with surrounding symbols by checking the top, bottom and adjacent symbols to see if they match the symbol in the specified location. If they do, the symbols above, below or adjcent to the shape will be moved to the place of the existing shape. Then a new random symbol will be generated in the location where the symbol was moved from.
- input parameters: char board (game board, int row (row position), int col (column position), int size (size of board), int points (amount of points user has), char *symbolsPtr (pointer for symbols array), int symAmount (amount of symbols specified in game by use).
- Increases the points variable by 10 and changes the game board variable
- Returns the points (integer function)

printBoard()

- Literates through each item in the board 2D array and prints the symbol stored.
- input parameters: char board (game board)
- doesn't change any global or member variables
- doesn't return anything (void function)

createColumn()

- generates a random number from 1 to the length of symbols specified then gets the symbol from the symbol array in that number location and assigns it to the current index within the column. Then itterates this until the full column is filled with random symbols.
- input parameters: char board (game board), int row (row location), int size (size of board), char *symbolsPtr (symbol array pointer), int symAmount (amount of symbols specified).
- Changes game board 2D array
- Doesn't return anything (void function)

saveGame()

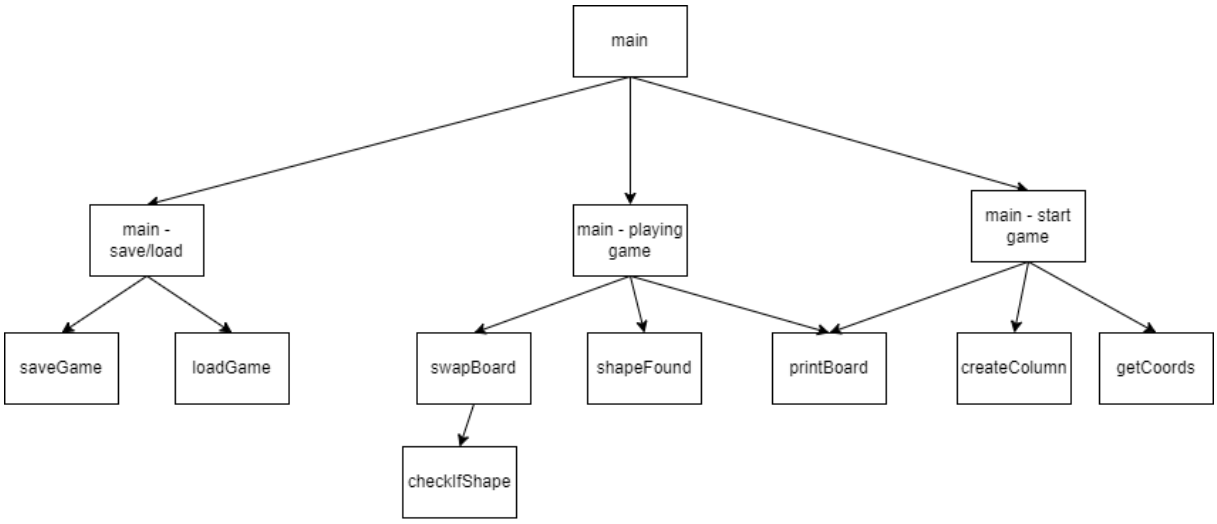
- Saves the game to a txt file by itterating through the game board and writing it into a text file (gets name of txt file from user). Also writes the size of board and the symAmount.
- input parameters: char board (game board), int size (size of board), int symAmount (amount of symbols specified)
- doesn't change any global or member variables
- Doesn't return anything (void function)

loadGame()

- loads the game from a txt file by itterating through the file and adding the symbols to the board 2D array (gets name of txt file from user). Also retrives the size of board and the symAmount at the end of the file.

- input parameters: char board (game board), int size (size of board), int symAmount (amount of symbols specified)
- changes game board, size and symAmount
- Doesn't return anything (void function)

Structure Chart



Algorithms

```
swapBoard(board, r1, c1, r2, c2, size)
    SET Sym TO board[r2][c2]
    SET shapeDetected TO false
    SET board[r2][c2] TO board[r1][c1]
    SET board[r1][c1] TO Sym
    SET shapeDetected TO output of checkIfShape(board, r2, c2, size)
    IF shapeDetected IS false THEN:
        SET board[r1][c1] TO board[r2][c2]
        SET board[r2][c2] TO Sym
        OUTPUT true
    ELSE
        OUTPUT false

createColumn(board, row, size, symbolsPtr, symAmount)
    SET index TO 0
    REPEAT size times
        SET num TO random number between 0 and symAmount
        SET board[row][i] TO symbolsPtr[num]
        ADD 1 TO index

printBoard(board, size)
    PRINT new line
    SET i TO 0
    REPEAT size times
```

```
        SET e TO 0
        REPEAT size times
            PRINT board[i][e]
            ADD 1 TO e

        PRINT new line
        ADD 1 TO i


checkIfShape(board, row, col, size)
    SET shape TO board[row][col]
    SET adjacent TO 0
    IF row is greater then 0 THEN:
        IF shape equals board[row-1][col] THEN:
            IF col greater than 0 THEN:
                IF shape equals board[row][col-1] THEN:
                    ADD 1 TO adjacent
            IF col is less than size THEN:
                IF shape equals board[row][col+1] THEN:
                    ADD 1 TO adjacent
    IF row is less then size THEN:
        IF shape equals board[row+1][col] THEN:
            IF col is greater than 0 THEN:
                IF shape equals board[row][col-1] THEN:
                    ADD 1 TO adjacent
            IF col is less then size THEN:
                IF shape equals board[row][col+1] THEN:
                    ADD 1 TO adjacent
    IF adjacent is greater then 0 THEN:
        OUTPUT true
    ELSE
        OUTPUT false


shapeFound(board, row, col, size, points, symbolsPtr, symAmount)
    ADD 10 TO points
    SET shape TO board[row][col]
    SET replacedNo TO 0
    SET num TO random number between 0 and symAmount
    SET board[row][col] TO symbolsPtr[num]
    IF row is greater then 0 THEN:
        IF shape equals board[row-1][col] THEN:
            IF replacedNo is greater than 1 THEN:
                SET num TO random number between 0 and symAmount
                SET board[row-1][col] TO symbolsPtr[num]
                ADD 1 TO replacedNo
    IF row is less than size THEN:
        IF shape equals board[row+1][col] THEN:
            IF replacedNo is less than 2 THEN:
                SET num TO random number between 0 and symAmount
                SET board[row+1][col] TO symbolsPtr[num]
```

```
            ADD 1 TO replacedNo
    IF col is greater than 0 THEN:
        IF shape equals board[row][col-1] THEN:
            IF replacedNo is less then 2 THEN:
                SET num TO random number between 0 and symAmount
                SET board[row][col-1] TO symbolsPtr[num]
                ADD 1 TO replacedNo
    IF col is less than size THEN:
        IF shape equals board[row][col+1] THEN:
            IF replacedNo is less than 2 THEN:
                SET num TO random number between 0 and symAmount
                SET board[row][col+1] TO symbolsPtr[num]
                ADD 1 TO replacedNo
    OUPUT points;


saveGame(board, size, symAmount)
    INPUT file name
    OPEN file in write mode
    SET i TO 0
    REPEAT size times
        SET e TO 0
        REPEAT size times
            WRITE board[i][e] in next fileline
            ADD 1 TO e
        ADD 1 TO i
    WRITE size in next fileline
    WRITE symAmount in next fileline

loadGame(board, size, symAmount)
    INPUT file name
    OPEN file in read mode
    READ next fileline
    SET symAmount TO fileline
    READ next fileline
    SET size TO fileline
    SET i TO size
    REPEAT size times
        SET e TO size
        REPEAT size times
            READ next fileline
            SET fileline TO board[i][e]
            TAKE 1 FROM e
         TAKE 1 FROM i
```

Test Data

Requirement Acceptance Tests

Requirement No	Test	Implemented (Full /Partial/ None)	Test Results (Pass/ Fail)	Comments (for partial implementation or failed test results)
UR1 + DR1	Run program without errors	Full	Pass	Program successfully ran without errors
UR2 + DR2	Start a new game sucessfully	Full	Pass	Lets you start a new game after each turn
UR3 + DR3	Prints out prompt and lets you input size for board	Full	Pass	Lets you input size and uses size within program to generate board and perform other tasks
UR4 + DR4	Prints out prompt and lets you input amount of symbols for game board	Full	Pass	Lets you input amount of symbols and only uses the amount of symbols specified to generate board
UR5 + DR5	Prints out game board at start of game	Full	Pass	Prints out game board sucessfully
UR6 + DR6	Prints out prompt for coordinates to and from and lets you input the variables	Full	Pass	Successfully lets you input to and from coordinates and uses them to excute the program
UR7 + DR7	Checks inputs to ensure all inputs are valid and prints out an error message if not.	Partial	Pass	Most inputs check if a valid input is entered, and will print an error message if not while repeating the prompt
UR8 + DR8	Prints out points for each move as well as total points	Full	Pass	Sucessfully prints out score for each move as well as total points for player
UR9 + DR9	Prints game board after every move	Full	Pass	After every move the game board is printed out

Requirement No	Test	Implemented (Full /Partial/ None)	Test Results (Pass/ Fail)	Comments (for partial implementation or failed test results)
				sucessfully
UR10 + DR10	Able to quit, save and start a new game at any point	Full	Pass	The game will let you quit, save or start a new game after each turn.
UR11 + DR11	The final score is printed out at the end of the game	Full	Pass	The final score is successfully calculated and printed out at the end of the game.

Detailed Software Testing

No	Test	Expected Results	Actual Results
1.0	**Invalid input**		
1.1	User enters too large or too small a size for game board	User is warned and is prompted to try again	Worked as expected
1.2	User enters too large or too small of an amount of symbols	User is warned and is prompted to try again	Worked as expected
1.3	User enters coordinates that don't exist	User is warned and is prompted to try again	Worked as expected
1.4	User enters characters for integer inputs	User is warned and is prompted to try again	User is only prompted to reinput values without error message
1.5	User enters a file name that doesn't exist.	User is warned and is prompted to try again	Worked as expected
1.6	User saves file with spaces	File is still saved without spaces	Worked as expected
2.0	**Functionality**		
2.1	Symbols are swapped only if shape is made	User is warned of invalid move and is prompted to input new coordinates	Worked as expected

User Instructions

1. **Open milestone1p1.c in IDE (Integrated development environment) of choice.**
2. **Create a new terminal (top left, in “Terminal tab” in Visual Studio Code).**
3. **Type “gcc milestone2p1.c“ in terminal to compile program**
4. **Type ./a to run.**
5. **Any further inputs can be typed in terminal when prompted by the program.**