Chang Soo Joe Son (105053217)

Rashedul Khan (108921821)

Rezoan Hasib (108991264)

**Minesweeper**

**Project description overview and interface:**

| Menu |

| 1. Start Game |

| 2. View Status |

| 3. Quit Program |

The overall functionality of the minesweeper program is to allow for the player to input one of the three choices in the main Menu to a) Start Game, b) View stats or c) Quit Program to begin the implementation of the program. The player is asked to pick the type of row and column board to initiate the game and also the number of mines to be placed in the game. Then the game begins where the player inputs points within the board as either ‘r’ to mark or ‘l’ to reveal the points within the board. If a reveal hits a mine, the game is finished and the remaining mine(s) within the board is revealed. If it does not hit a mine, the areas around the reveal point is either shown as having approximate mines or clearing areas around the mark as having no approximate mines. The player wins when all the tiles are revealed except the ones with the mines.

**The Architecture of the Program:**

Initialization >**>** Program Menu >> Choose Choice 1 >>Start Game >> Game Function >> Print Board >> Load File(Y/N) >> Y >> Select File >> Reveal Location >> endGame >> Game Over(Y/N) >>Y>> Play Again >> Main Menu

Initiation >**>** Program Menu >> Start Game >> Game Function >> Print Board >> Load File(Y/N) >> N >> First Turn (Y/N) >> Y >> initialMineField() >> RevealLocation() >> endGame() >>Game Over(Y/N) >>N >> Print Board(back in the loop) >> Load File(Y/N) >> N >> First Turn (Y/N) >> Y >> initialMineField() >> RevealLocation() >> endGame() >>Game Over(Y/N) >>Y >> Calculate/Save statistics >> Play Again (Y/N) >> Y >> Game Function Begins(again)

Initiation >> Program Menu >> Choose Choice 2 >>View Stats >> No stats implemented >> Program Menu

Initiation >> Program Menu >> Choose Choice 3 >> Quit Program >> Exit out

**Class hierarchies:**

|  |  |
| --- | --- |
|  | **Public** members of the class minesweeper are accessible for all and everyone.  **Protected** members of class minesweeper are not accessible outside of minesweeper's code, but is accessible from the code of any class derived from minesweeper. |

**The minesweeper class has declaration of protected for:**

1. Int colNum and rowNum are designated as ‘int’ since the number of designated column(s) and row(s) can be accessed within the code and should not be modified from outside the program. Also, the mineNum can be accessed to place mines within the program but once designated, the number of the mine(s) should not be accessed outside the program.
2. The vector<vector<int> > mineField, vector<vector<int> > bitField, and vector<vector<int> > markField must also be protected since they must not be modified after initial and sequential inputs.
3. The void resizeVector(), calculateSurrounding(int row, int col), onlyMines(), int randomPick(int num), and void unmask(introw, int col) are Class implementation as protected.

**The minesweeper class has declaration of public for:**

1. Destructor ~minesweeper();
2. Minesweeper(); as the default constructor
3. Int getColNum(), getRowNum(), getMinesNum(), and endGame() are required as int function so that the values can be returned to the main program.
4. Void intialMineField(int fpX, int fpY), initialMineField(string path), revealLocation(int x, int y) and void markLocation(int x, int y)
5. Bool isRevealed function is to check if the tile has been revealed as true or false

The main function of the program is to call upon the a) void playTime(minesweeper& play, bonus& score); b) void viewStatistics(); int userMenu(); c) void printBoard(minesweeper& play, and d) vector<vector<bool>>& markVec); within the main functionality of the program when these proto-program is called within the main function.

In the main function, the program uses overall while functionality to allow for the program to loop when the input fails to allow for the game to continue. Int choice is given as an option between numbers 1 to 3 where the input 1 allows for the game to start, 2 allows for the view stats and 3 for quitting the game. If the input is any random alphanumerical of anything less than 100 inputs, this input will be ignored and the program will ask for another input.

When playtime is invoked within the main function, the player is allowed to enter the values for ‘l for left (Reveal) r for right (Mark)’ and an example of how to enter the values is given as an output. When the correct input has been placed, the r or R of the [row][col] of the first input is given as a mark on the board and the mark is placed within the printBoard. In order to input the first turn, the playtime program is called within the main function, the inputting for the error checking of the input allows for the input to be any alphanumerical input within 100 character to be considered where any such input that deviates from the example input will allow for the program to ask for another input.

The class initialMineField and revealLocation class implementation is then demonstrated within the main program. InitialMineField is implemented with int fpX and fpY as the parameters and the for loop of the random pick fo the colNum and rowNum is simulated for conditions of (int x=0; x<minesNum; x++), do function of randcol and randrow. When the randcol equals fpX and randrow equals fpX or minefield vector array equals -1, the minefield array of randrow and randcol is -1 and then implement the calculateSurrounding of parameters randrow and randcol. CalculateSurrounding(int row,int col) updates the surrounding tiles to check for the adjacent mines by: checking the row and columns and continue when the argument is a<0 or a>rowNum-1 or b<0 or b>colNum-1 or (a==row && b==col) and if minefield[a][b] is not -1 then minefield[a][b]++. Finally, for the initialMineField class implementation, after the class calculate Surrounds(randrow,randcol) has been implemented, the for loop for the (int a=0;a<rowNu, a++) and (int b=0; b<colNum; b++) allows for the minefield[a][b] is outputted and the printBoard is revealed with the corresponding mark or reveal on the board.

After the first input, the user input is compared using the ignore function which is invoked within the program where the error checking of any alphanumerical value is compared with the correct example input and when it does not correspond to the correct type, then the program asks for another input. Marking the location based on the user input, the input r invokes the markLocation(row,col) implementation where the display for the selected coordinate is marked when it is not marked and unmarked if it is marked by functionality of: if(markField[x][y]){ markField[x][y]=false; else markField[x][y]=true and the main function has: if(!play.isRevealed(row,col)), if markVec[row][col] is false else markVec[row][col] is true. Else, if !markVec[row][col] then play the revealLocation(row,col). RevealLocation class implementation is to update the board of the revealed locations and to check the minefield for mines by recursively calling helper function ‘unmask’ for surrounding blank areas. So if minefield[x][y] == -1, then end=-1. The unmask function is to reveal surrounding areas if the revealed coordinate on the MineField has a value of 0. This is implemented by: if (row<0 || row>=rowNum || col<0 || col>=colNum || !bitField[row][col] || markFieldrow][col], the return. Else, unmask the eight adjacent tiles around the selected tile. This is implemented by: unmask(row-1,col); unmask(row,col-1); unmask(row+1,col); unmask(row,col+1); unmask(row+1,col+1); unmask(row-1,col+1); unmask(row-1,col-1); unmask(row+1,col-1).

When all the mines are marked and all the non-mine coordinates are revealed, the endGame is implemented where this method is to update the winning conditions for the game. The endGame function call the onlyMines function when the end!==-1 where the winning conditions are bool win=true for(int a=0;a<rowNum;a++) {for(int b=0;b<colNum;b++) {if(mineField[a][b]>=0 && bitField[a][b]) {win=false. In the main program, the cout<<”You Win!!!” outputs. Else, the condition is to output “You Loss” and still reveal the game Board.

Based on the void printBoard proto-program called within the main program, int nCol and nRow of the program is obtained from the getColNum() and getRowNum() that are called from the class implementation of these class functions. The actual board is combination of nRow index and nCol index from the given input of the Row and Col values. If (marked[i][j]), then the tile is marked with ‘M,’ else the isRevealed function is invoked. If valueOf(i,j)>0 where the valueOf(i,j) is to return the minefield[x][y] and tile = ‘ ‘ and if valueOf(i,j)<0, the tile = ‘\*’ or else, time =’?.’ This means that the board with tile as character cannot output as integer so, the conversion of the char to integer is done through valueOf(i,j) + 48.

**Reason for protected data:**

If the minesweeper class had it's variables (colNum, rowNum and minesNum) and the vectors (mineField, bitField and markField) defined as private instead of protected, then those data could not be accessed by any other class even if that was derived from the minesweeper class. For example because of defining them as protected, the bonus class which inherits from the minesweeper class can also access those data. If they were private then only the minesweeper objects could access the data, nothing else. This way of definition would be helpful in order to perform the bonus by taking inheritance into account. The private data could not also be public because no class other than minesweeper and bonus should have access to the data in the program. In the beginning when the user is asked for the row, column and the mine numbers, the set() and the get() functions are used to access those data because of not being public.

**New in the project:**

Among all the home works given in this course, none of them had a class defined and implemented totally for that but in this project two new classes (minesweeper and bonus) were defined and implemented solely for the project. A minesweeper object was created to access through the minesweeper class. If there were a bonus object created, it would have accessed both the minesweeper and bonus classes through inheritation. This project was an example of defining and implementing classes. How to take private and public data into account and implement those data in the main program was a new experience in this project. So far, this is the biggest program that has been dealt with in the class hence was a good experience dealing with object oriented programing with bigger code.