

# Final Project.

(Minesweeper)

**Name:** Zeyad Osama Abd-El-Monem

**ID:** 4555

**Group:** ( 2 )

**Section:** ( 1 )

**Name:** Marwan Amr Farouk El-Safty

**ID:** 4690

**Group:** ( 4 )

**Section:** ( 1 )

---

## Content:

- Overview and Description of the Application.
- User Manual and Sample Runs.
- Pseudocode, Flowchart and Main Algorithms Used.
- Description of the Important Implemented Functions and Modules.
- References.

## 1. Overview and Description of the Application.

The application starts with a welcome message, and then the user is asked if he wants to start game, see the user manual, display the score board or exit.

### **Start the game:**

The user is asked to enter his name for the saving operation, and then he must enter the number of rows and columns to make the board of the game.

The 'x' refers to an unopened cell, each row and columns have its indexes written next to it.

The user enters the number of the cell he wants to open by referring to its row and column and then he choose wether to open the cell or to mark it as a flag.

When the user tries to open a mined cell he loses the game, and he wins it by putting flags on all the mined cells and opens the rest of the cells, then his time is printed on the screen and his name and score is saved in a file to be displayed when the user asks to.

### **User manual:**

The user manual option displays the 'How to play' manual.

### **Score board:**

The score board displays the names and scores of all the recent players.

## 2. Sample Runs.

The starting menu:

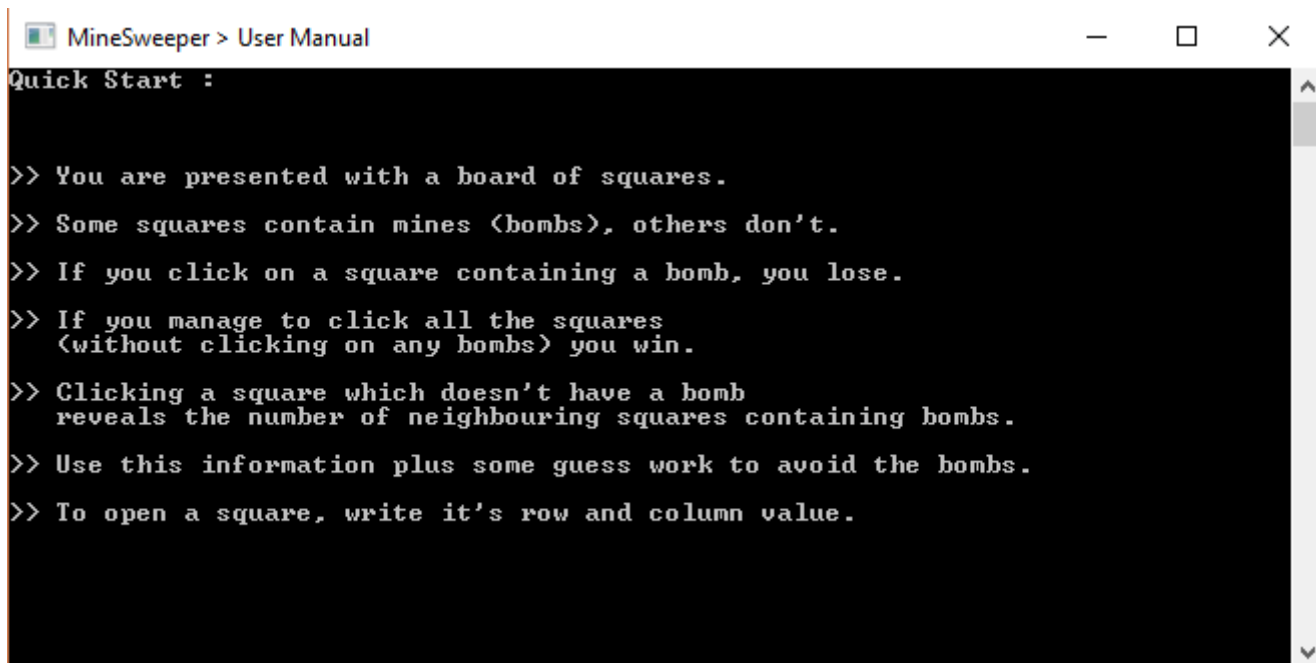
A screenshot of a terminal window titled "MineSweeper". The window has standard window controls (minimize, maximize, close) in the top right. The terminal content shows a welcome message "Welcome to Minesweeper" followed by a menu with four options: "Start Game <S>", "User Manual <M>", "Score Board <B>", and "Exit <E>". Below the menu, it prompts "Your choice :".

```
MineSweeper

----- Welcome to Minesweeper -----
Start Game      <S>
User Manual     <M>
Score Board     <B>
Exit            <E>

Your choice :
```

The user manual menu:

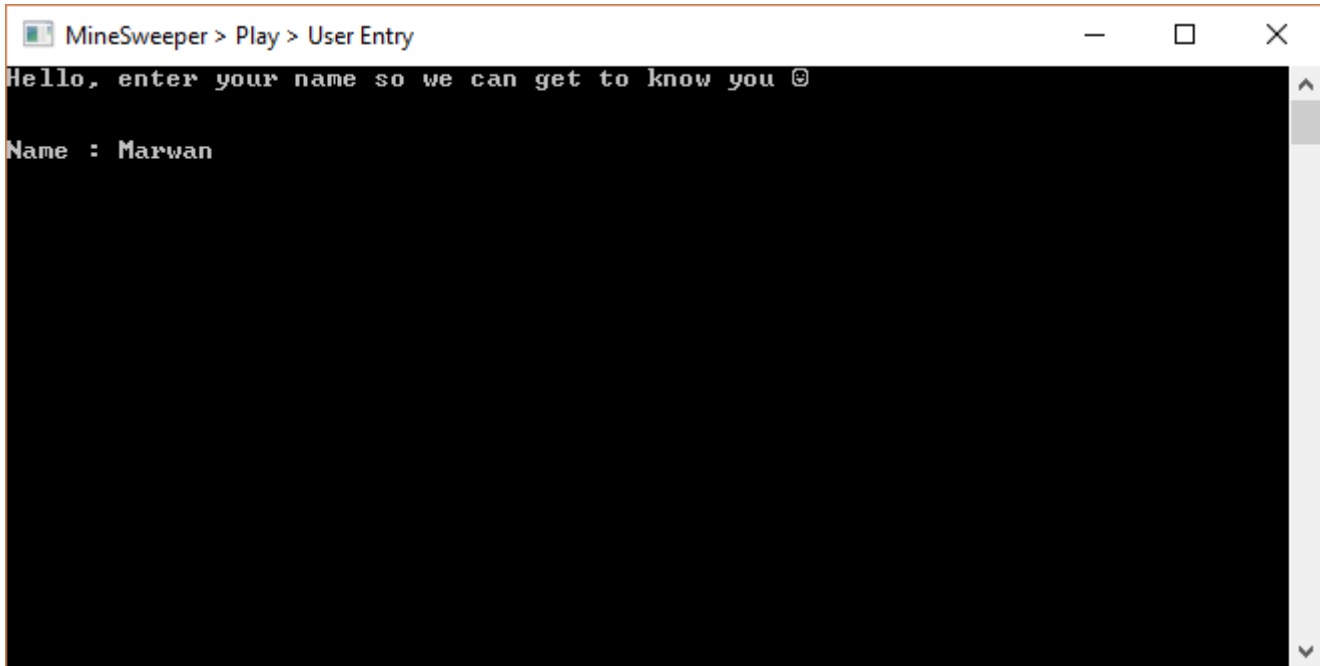
A screenshot of a terminal window titled "MineSweeper > User Manual". The window has standard window controls. The terminal content shows a "Quick Start" section with seven lines of instructions, each preceded by ">>".

```
MineSweeper > User Manual

Quick Start :

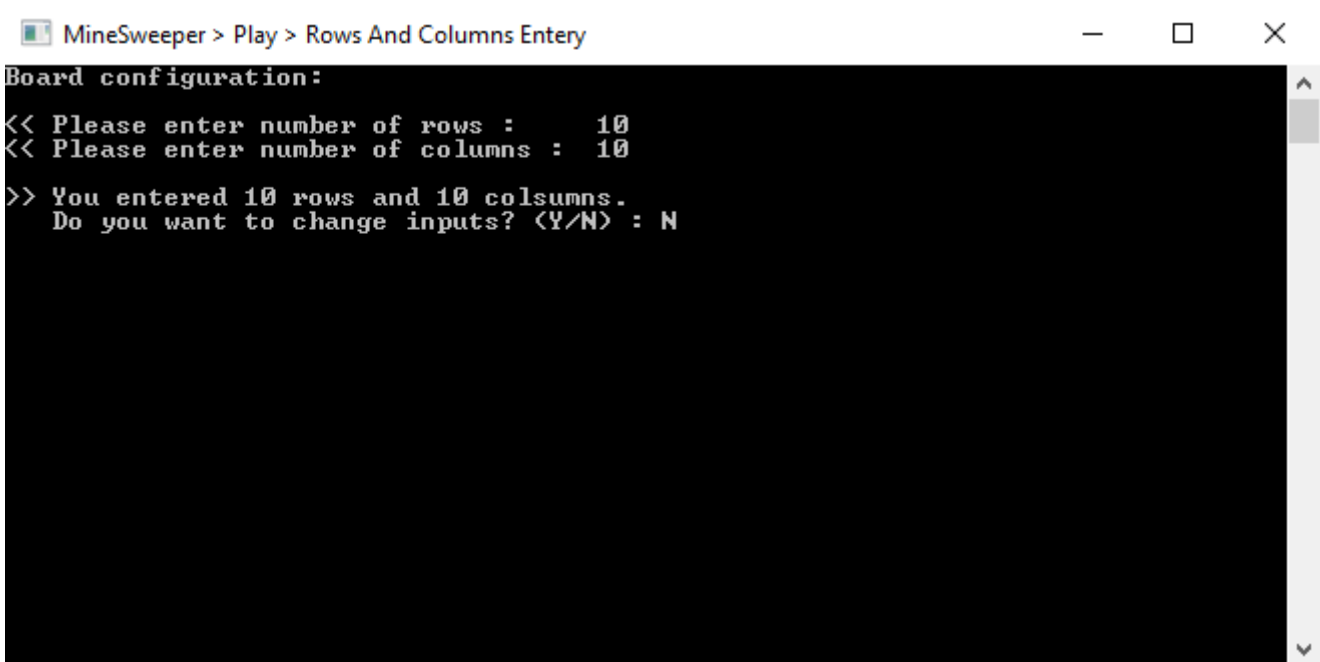
>> You are presented with a board of squares.
>> Some squares contain mines (bombs), others don't.
>> If you click on a square containing a bomb, you lose.
>> If you manage to click all the squares
    (without clicking on any bombs) you win.
>> Clicking a square which doesn't have a bomb
    reveals the number of neighbouring squares containing bombs.
>> Use this information plus some guess work to avoid the bombs.
>> To open a square, write it's row and column value.
```

### The start game menu:



```
MineSweeper > Play > User Entry
Hello, enter your name so we can get to know you 😊
Name : Marwan
```

### Choosing dimensions of the game board:



```
MineSweeper > Play > Rows And Columns Entry
Board configuration:
<< Please enter number of rows : 10
<< Please enter number of columns : 10
>> You entered 10 rows and 10 columns.
Do you want to change inputs? (Y/N) : N
```

Opening cell [1][1]:

```
MineSweeper > Play

      1  2  3  4  5  6  7  8  9 10
1  x  x  x  x  x  x  x  x  x  x
2  x  x  x  x  x  x  x  x  x  x
3  x  x  x  x  x  x  x  x  x  x
4  x  x  x  x  x  x  x  x  x  x
5  x  x  x  x  x  x  x  x  x  x
6  x  x  x  x  x  x  x  x  x  x
7  x  x  x  x  x  x  x  x  x  x
8  x  x  x  x  x  x  x  x  x  x
9  x  x  x  x  x  x  x  x  x  x
10 x  x  x  x  x  x  x  x  x  x

<< Enter row : 1
<< Enter col : 1
<< Hit [O] to open cell or Hit [F] to mark it as Flag : 0
```

Opening other cells:

```
MineSweeper > Play

      1  2  3  4  5  6  7  8  9 10
1      1  x  x  x  x  x  x  x  x
2      1  x  x  x  x  x  x  x  x
3      2  x  x  x  x  x  x  x  x
4      1  x  x  x  x  x  x  x  x
5  1  2  x  x  x  x  x  x  x  x
6  x  x  x  x  x  x  x  x  x  x
7  x  x  x  x  x  x  x  x  x  x
8  x  x  x  x  x  x  x  x  x  x
9  x  x  x  x  x  x  x  x  x  x
10 x  x  x  x  x  x  x  x  x  x

<< Enter row : 10
<< Enter col : 10
<< Hit [O] to open cell or Hit [F] to mark it as Flag : 0_
```

```
MineSweeper > Play

      1  2  3  4  5  6  7  8  9 10
1      1  2  x  x  x  x  x  x  x  1
2      1  x  x  x  x  x  x  x  x  x
3      2  x  x  x  x  x  x  x  x  x
4      1  x  x  x  x  x  x  x  x  x
5  1  2  x  x  x  x  x  x  x  x
6  x  x  x  x  x  x  x  x  x  x
7  x  x  x  x  x  x  x  x  x  x
8  x  x  x  x  x  x  x  x  x  x
9  x  x  x  x  x  x  x  x  x  x
10 x  x  x  x  x  x  x  x  x  1

<< Enter row : 6
<< Enter col : 3
<< Hit [0] to open cell or Hit [F] to mark it as Flag : 0
```

Mark cell[6][5] as Flag:

```
MineSweeper > Play

      1  2  3  4  5  6  7  8  9 10
1      1  2  x  x  1      1  x  1
2      1  x  3  2  1      1  1  1
3      2  2  2
4      1  F  1
5  1  2  1  2  1  1      1  1
6  F  1      1  x  1      1  x
7  1  1  1  2  2  1      1  x
8  x  x  x  x  1      1  x
9  x  x  x  x  2  1      1  x
10 x  x  x  x  x  1      1  1

<< Enter row : 6
<< Enter col : 5
<< Hit [0] to open cell or Hit [F] to mark it as Flag : f
```

Opening cell [8][4]:

```
MineSweeper > Play

      1  2  3  4  5  6  7  8  9 10
1      1  2  F  F  1      1  F  1
2      1  F  3  2  1      1  1  1
3      2  2  2
4      1  F  1
5  1  2  1  2  1  1      1  1
6  F  1      1  F  1      1  F
7  1  1  1  2  2  1      1  1
8      1  x  1      1  1
9      1  2  2  1      1  F
10     0  F  1      1  1

<< Enter row : 8
<< Enter col : 4
<< Hit [O] to open cell or Hit [F] to mark it as Flag : o
```

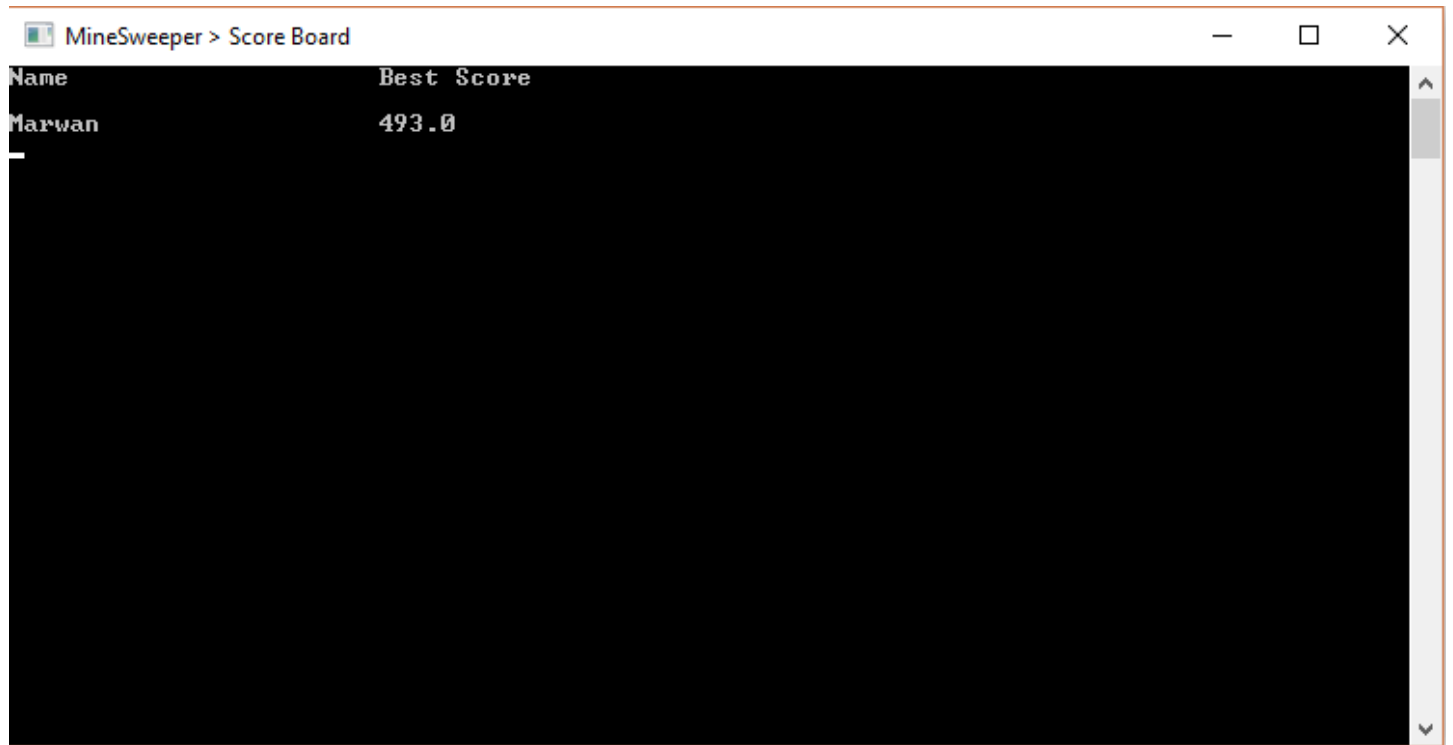
Mark the last cell as a flag to win the game:

```
MineSweeper > Play

      1  2  3  4  5  6  7  8  9 10
1      1  2  F  F  1      1  F  1
2      1  F  3  2  1      1  1  1
3      2  2  2
4      1  F  1
5  1  2  1  2  1  1      1  1
6  F  1      1  F  1      1  F
7  1  1  1  2  2  1      1  1
8      1  F  1      1  1
9      1  2  2  1      1  F
10     0  F  1      1  1

CONGRATULATIONS. YOU WON !
Marwan, your time is : 493.0 seconds.
```

## The score board:



Name	Best Score
Marwan	493.0



### 3. Pseudocode, Flowchart and Main Algorithms Used.

The game mainly consists of two boards; the original board and the game board.

**The original board** - referred to in the code by "board[i][j]" - is the board that contains the solution of the game, it consists of numbers from 0 to 9, a cell that has the number 0 means an empty cell, the number 9 means a mined cell and a cell that has any other number than refers to the numbers of mines located near that cell.

It is constructed using an array with the same dimensions as entered by the user, each element of the array represents a cell from the board, the mined cells are located randomly.

After locating the mines we loop through every cell of the board and count the numbers of mines near this cell and assign this number to the cell.

**The game board** – referred to as 'gameBoard[i][j]' in the code - is the board that appears for the user during the gameplay, it is represented in the code by an array of character, at the beginning of the game all element of this array is an 'x', every time a user chooses to put a flag on a cell the element representing this cell is switched to 'F', but if he chooses to open the cell, the app compares the same element in the original board's array with 3 value 9,  $0 < x < 9$  and 0.

If it is a 9, this means the user has hit a mine which means that he has lost the game.

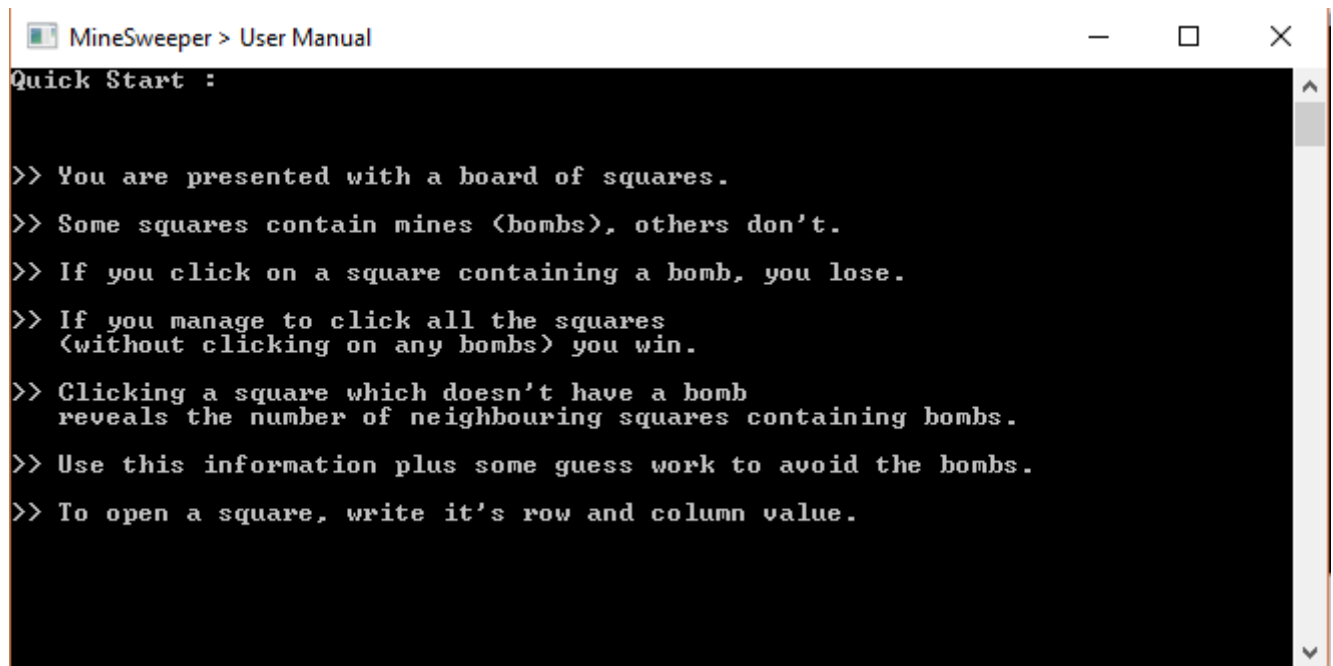
If it is number from 1 to 8 the cell is automatically assigned with that number.

If it is a 0, this means the cell is empty, so all the cells around this cell are automatically opened using the recursion.

#### The application step by step:

- The user chooses whether to start game, see the user manual, see the score board or exit.
- If the user chooses to **start the game**, he chooses the dimensions of the board (rows i and columns j) .
- when the user confirms the board configuration time is taken at that moment and two arrays are immediately created, the board[i][j] which contains the solved board, it is created by randomly putting a specific number of mines in the (i\*j) elements of the array and then looping through every element to count the number of mines located around this element and assign this count to this element. If there is no mine around this cell it is assigned with the number 0. And the game board[i][j] which consists of character that is displayed to the user during the gameplay.
- The user starts playing by choosing the row, column of the cell and whether to open it or mark it as flag.
- If he chooses to open the cell, the element with the same row and column indexes in the board[i][j] array is checked, if it is a 9 the user have hit a mines and he loses the game, if it is a number between 1 and 8 the game board cell he chose changes to that number, and if it is a 0 that means this is an empty cell and all of its surrounding empty cells are opened by a recursive way (discussed in the functions section)
- If he chooses to mark the cell as flag, the 'x' closing the cell is removed and replaced by an 'F'.
- The user continues to play until he hit a mined cell or when he marks all mined cells with a flag and opens all the non-mined cells
- If the user wins, another time is taken that moment and the time between starting the game and winning is displayed to the user.

- If the user chooses the **user manual**, a simple explanation on how to play is displayed:



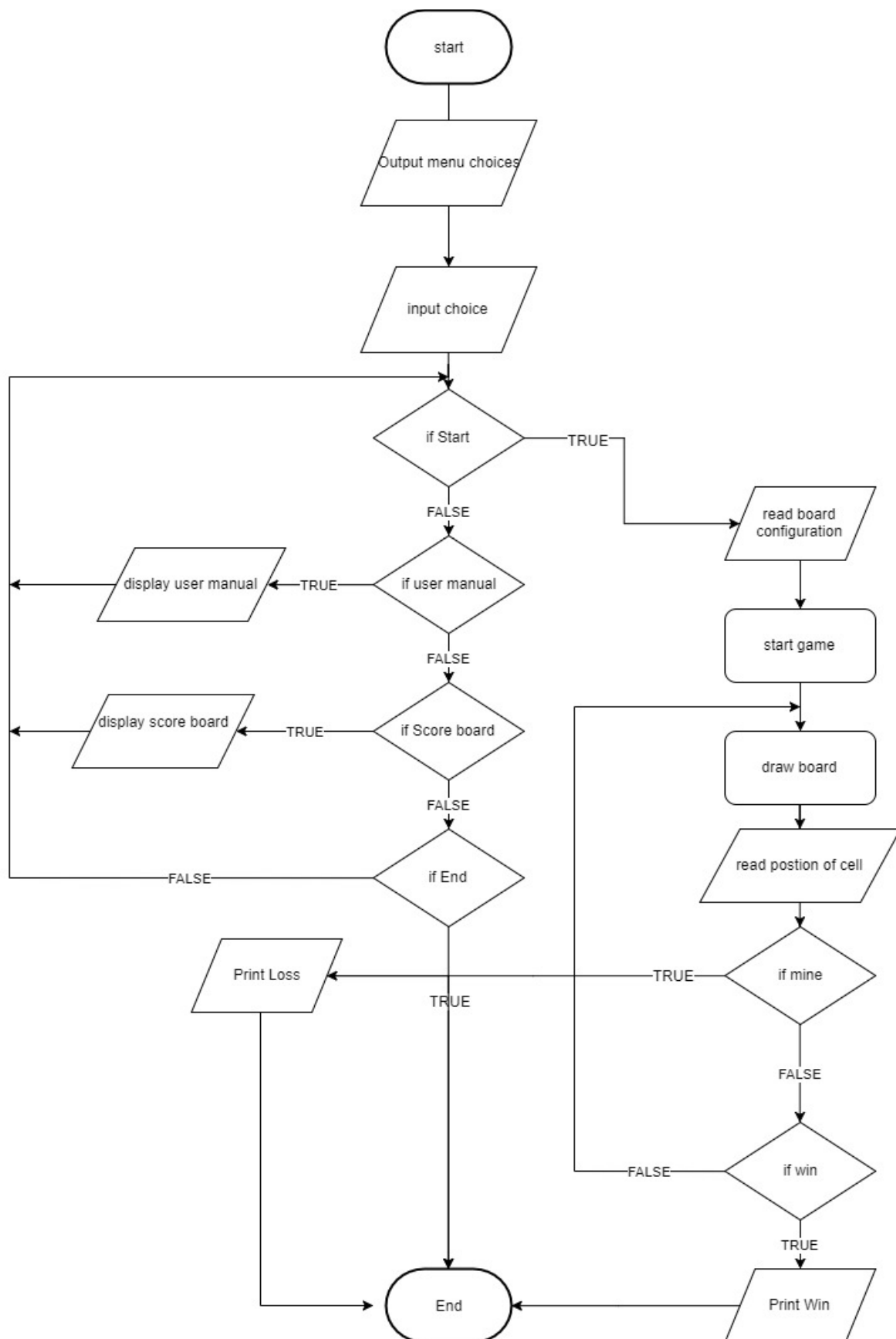
The screenshot shows a window titled "MineSweeper > User Manual" with standard Windows window controls (minimize, maximize, close). The content area has a black background with white text. It begins with "Quick Start :" followed by seven lines of instructions, each preceded by ">>".

```
MineSweeper > User Manual
Quick Start :

>> You are presented with a board of squares.
>> Some squares contain mines (bombs), others don't.
>> If you click on a square containing a bomb, you lose.
>> If you manage to click all the squares
    (without clicking on any bombs) you win.
>> Clicking a square which doesn't have a bomb
    reveals the number of neighbouring squares containing bombs.
>> Use this information plus some guess work to avoid the bombs.
>> To open a square, write it's row and column value.
```

- If the user chooses the **score board**, a list with all players, their time and their score is displayed.

## FlowChart:



## 4. Description of Important Implemented Functions and Modules.

### 4.1. initMinesLocation() :

This function distributes the mines throughout the array of board[i][j] randomly.

```
1.  int i,j;
2.      int n=0;
3.      int minesNum = 1 + (int)ceil((double)(rows * cols)/10);
4.
5.      srand(time(NULL));
6.
7.      while(n<minesNum)
8.      {
9.          i=rand()%rows;
10.         j=rand()%cols;
11.
12.         if(board[i][j]!=9)
13.         {
14.             board[i][j]=9;
15.             n++;
16.         }
17.     }
18.
```

### 4.2. initCellRank():

This function assign a number from 0 to 9 for every element in the board[i][j] array.

```
1.  int i,j;
2.
3.      for(i=0; i<rows; i++)
4.          for(j=0; j<cols; j++)
5.          {
6.              if(board[i][j]<9)
7.              {
8.
9.                  if(i-1>=0 && j-1>=0 && i-1<rows && j-1<cols)
10.                     if(board[i-1][j-1]==9)
11.                         board[i][j]++;
12.
13.                  if(i-1>=0 && j>=0 && i-1<rows && j<cols)
14.                     if(board[i-1][j]==9)
15.                         board[i][j]++;
16.
17.                  if(i-1>=0 && j+1>=0 && i-1<rows && j+1<cols)
18.                     if(board[i-1][j+1]==9)
19.                         board[i][j]++;
20.
21.                  if(i>=0 && j-1>=0 && i<rows && j-1<cols)
22.                     if(board[i][j-1]==9)
23.                         board[i][j]++;
24.
25.                  if(i>=0 && j+1>=0 && i<rows && j+1<cols)
26.                     if(board[i][j+1]==9)
27.                         board[i][j]++;
28.
29.                  if(i+1>=0 && j-1>=0 && i+1<rows && j-1<cols)
```

```

30.             if(board[i+1][j-1]==9)
31.                 board[i][j]++;
32.
33.             if(i+1>=0 && j>=0 && i+1<rows && j<cols)
34.                 if(board[i+1][j]==9)
35.                     board[i][j]++;
36.
37.             if(i+1>=0 && j+1>=0 && i+1<rows && j+1<cols)
38.                 if(board[i+1][j+1]==9)
39.                     board[i][j]++;
40.         }
41.     }

```

### 4.3. blankBlock(int i,int j):

This function is responsible for opening ever empty cell around the selected cell if it is empty.

It uses recursion so we can open every neighboring cells, it uses the flood fill algorithm.

```

1. gameBoard[i][j] = ' ' ;
2.
3.     if(i-1>=0 && j-1>=0 && i-1<rows && j-1<cols)
4.     {
5.
6.         if(gameBoard[i-1][j-1]==' ');
7.
8.         else if(board[i-1][j-1]==0)
9.             blankBlock(i-1,j-1);
10.
11.        else
12.            gameBoard[i-1][j-1]= board[i-1][j-1]+'0' ;
13.    }
14.
15.
16.    if(i-1>=0 && j>=0 && i-1<rows && j<cols)
17.    {
18.
19.        if(gameBoard[i-1][j]==' ');
20.
21.        else if(board[i-1][j]==0)
22.            blankBlock(i-1,j);
23.
24.        else
25.            gameBoard[i-1][j]= board[i-1][j]+'0';
26.    }
27.
28.
29.    if(i-1>=0 && j+1>=0 && i-1<rows && j+1<cols)
30.    {
31.
32.        if(gameBoard[i-1][j+1]==' ');
33.
34.        else if(board[i-1][j+1]==0)
35.            blankBlock(i-1,j+1);
36.
37.        else
38.            gameBoard[i-1][j+1]= board[i-1][j+1] + '0';
39.    }
40.
41.
42.    if(i>=0 && j-1>=0 && i<rows && j-1<cols)
43.    {
44.
45.        if(gameBoard[i][j-1]==' ');
46.

```

```

47.         else if(board[i][j-1]==0)
48.             blankBlock(i,j-1);
49.
50.         else
51.             gameBoard[i][j-1]=board[i][j-1]+'0';
52.     }
53.
54.
55.     if(i>=0 && j+1>=0 && i<rows && j+1<cols)
56.     {
57.
58.         if(gameBoard[i][j+1]==' ');
59.
60.         else if(board[i][j+1]==0)
61.             blankBlock(i,j+1);
62.
63.         else
64.             gameBoard[i][j+1]= board[i][j+1]+'0';
65.     }
66.
67.
68.     if(i+1>=0 && j-1>=0 && i+1<rows && j-1<cols)
69.     {
70.         if(gameBoard[i+1][j-1]==' ');
71.
72.         else if(board[i+1][j-1]==0)
73.             blankBlock(i+1,j-1);
74.
75.         else
76.             gameBoard[i+1][j-1]= board[i+1][j-1]+'0';
77.     }
78.
79.
80.     if(i+1>=0 && j>=0 && i+1<rows && j<cols)
81.     {
82.
83.         if(gameBoard[i+1][j]==' ');
84.
85.         else if(board[i+1][j]==0)
86.             blankBlock(i+1,j);
87.
88.         else
89.             gameBoard[i+1][j]=board[i+1][j]+'0';
90.     }
91.
92.
93.     if(i+1>=0 && j+1>=0 && i+1<rows && j+1<cols)
94.     {
95.
96.         if(gameBoard[i+1][j+1]==' ');
97.
98.         else if(board[i+1][j+1]==0)
99.             blankBlock(i+1,j+1);
100.
101.         else
102.             gameBoard[i+1][j+1]= board[i+1][j+1] + '0';
103.     }
104.
105.     return;

```

## 6. References.

- <http://www.cplusplus.com/reference/>
  - <https://reference.cs50.net/>
  - <https://stackoverflow.com/questions/>
  - <https://www.quora.com/>
  - [http://wiki.codeblocks.org/index.php/Main\\_Page/](http://wiki.codeblocks.org/index.php/Main_Page/)
-

**Notes.**