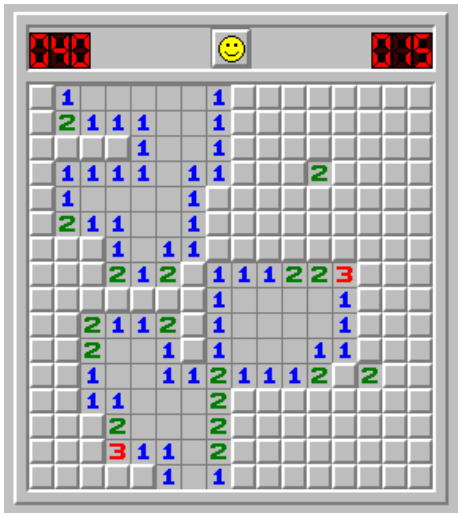
Final Project
Minesweeper



1 Game Description

Minesweeper is a single-player puzzle video game. It has its origins in the earliest mainframe games of the 1960s and 1970s.The objective of the game is to clear a rectangular board containing hidden mines without detonating any of them, with help from clues about the number of neighboring mines in each field. The game originates from the 1960s, and has been written for many computing platforms in use today. In this project you will make your own version of the game. You can read more about Minesweeper on wikipedia.

2 Overview

The goal of Minesweeper is to mark locations of mines with flags and to uncover all other locations that don't contais mines. Score is calculated based on time, number of operations performed and size of map.
The map is a two grid where each location of the grid has either a

- Mine
- Number of adjacent cells that contains mines.
- If no adjacent cell has mine, then this cell is empty

The first cell the player opens shouldn't contain a mine. Also, player is free to start in any cell.

When an empty cell (cell with no adjacent mines) is opened, then all empty cells that are reacheable from this cell are opened.

3 Game Actions

In Minesweeper the is five actions that the player can perform. These actions are about the state of a cell. Here are the actions that you should support.

1. Open : make the cell open whether it has a mine or not.
2. Flag : mark the cell with a flag to indicate that it contains a mine.
3. Question : mark the cell with a question mark. This can be useful when the player is not sure whether this cell contains a mine or not. This can prevent the player from opening it later.
4. Unmark : remove the flag or questin mark from a cell.
5. Opening an open cell : if a cell is opened and it has a number say n and there is n adjacent cells makred with flags. Then all adjacent cells should be opened except those marked with flags whether they realy contains mines or not. This way the player can save time opening cells. If number of adjacent cells that are makred with flag is is not equale to n, then this operation is not performed. If you open an empty cell then you should open all empty cells reachable from this empty cell.

4 End Game

Minesweeper ends in one of two states win or lose.

- Lose : when player opens a cell with a mine the player loses.
- Win : when all cells that don't contains mines are opened that the player wins. This meens that it is not necessary to mark other cells with flags. Flags are used like question marks, to make the game easier.

5 Grid Display

Grid display depends on the game state. Playing, win or lose. Each cell will be displayed a single character: During play :

- Closed cell : 'X'
- Open cell that is empty : ' '
- Open cell with a number from 1 to 8 : the number in this cell
- Cell with a flag : 'F'

- Cell with question mark : '?'
- Win : All cells that contains mines should be printed as flag 'F'.
- Lose :
  - Mine : '*'
  - The missed mine that made the player lose the game : '!'
  - Incorrect Flag should be replaced with : '-'
  - Missed mine should be replaced with : 'M'

Each row and each column should start with a number to make it easy for the player to choose a cell in the gird.

6 User Interface

During the game play you should print extra information beside the grid.

- Number of moves
- Number of cells marked with flags
- Number of cells marked with question mark
- Time passed in minutes and seconds. It should be updated each time the player makes a move

7 User's Rank

When a user successfully finishes a game, You should take his name and calculate his score and rank. If it's the first time for this user, then add him to the users list. Else, add the current score to his old score. User name should be case insensitive. If the player loses then you shouldn't take his score.

8 Formulas

number of mines $= 1 + (numberOfRows * numberOfColumns)/10$
player score $= (numberOfRows^4 * numberOfColumns^4)/(timeInSeconds * numberOfMoves)$

9 Grid Initialization

You should take grid dimenstions as input from the user and randomly chose positions for the mines. number of minses is specified by the given formula.

10 Save and Load

- At any time, the game could be saved to a file
- A saved game could be loaded and continued
- Players names and scores should be saved to a file

11 Empty Cells

When the use opens an empty cell, all empty cells reachable from this cell should be opened till you reach non-empty cells which should be opened. To implement this feature you should use recursion. This can be solve with depth first search (DFS).
The Algorithm :
While doing a DFS, we maintain a set of visited cells. Initially this set is empty. When DFS is called on any cell (say u), first that cell is marked as visited and then for every adjacent cell v, such that v is unvisited, we call DFS on v. Finally, we return when we have exhausted all the adjacent cells.

12 Debuging

You should print the opened grid to a file to help you debug the program and help us test it.

13 Main Menu

The main menu of your program should contain the following options :

- start game
- load game
- show players ordered with their score

You can add other options

14 Bonus

- Split your project on different files and use header files.
- Put your project on GitHub.
- When the user is idle for a minute, update the time.

15 Implementation Notes

- You are required to implement the game using the C programming language.
- Top Priority Your program MUST NOT crash under any circumstances, even against malicious users!
- All users inputs should be validated. Any missed validation will reduce your marks.
- Take care of the following:
  - Naming conventions.
  - Code organization and style.
  - Code comments
- You are welcome to add extra features to your game. Good extra work will be graded as bonus.

16 Deliverables

- CD containing:
  - Complete project and source files
  - Executable version of your project (the *.exe file)
- Report containing the following:
  - Description of your application, and the features it provides.
  - Overview of your design.
  - Any desing/implementation assumptions made, and details you find necessary to be clarified.
  - Description of all used data structures.
  - Description of the important functions/modules in your implementation.

- Flow chart and pseudo code for the main algorithms (e.g. the game loop)
- User Manual that describes how to deal with your application.
- Sample runs.
- References: Any copied material from anywhere should be referenced. Any discovered unreferenced copied material will result in sever reduction in your grade.

17 Notes

- You are required to work in groups of two
- Take your time to design the project and discuss your design
- No time will be allowed in the lab for any modifications, your program should be executed from a copy of your CD during your 15 min discussion.
- All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to all participating students.
- Start in the project early and come forward with your questions.
- Late submissions are not accepted.
- Prior to discussion, you should be prepared to illustrate your work and answer any questions about it. Failing to do so is a strong indication to copying / cheating.

Good Luck