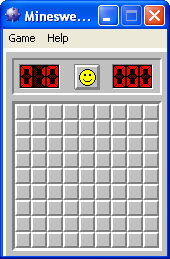
LW: Minesweeper

# Overview

There are many minesweeper-type computer games. But, probably, the most well-known was Microsoft Minesweeper, originally created by Curt Johnson for IBM's OS/2 and ported by Robert Donner to Microsoft Windows. It was released officially as part of the Microsoft entertainment package in 1990 and in Windows 3.1 in 1992.



Initially you have a board with *m* rows and *n* columns, represented as a m x n matrix, as above (9x9 in this case), where all positions are hidden. On the board are hidden some mines (10 in the figure above) and the goal is to uncover all the squares on the board that do not contain [mines](https://en.wikipedia.org/wiki/Land_mine) (or, equivalently, to find the location of all the mines) without being "blown up" by clicking on a square with a mine underneath. When you choose a covered position of the grid there are two possible outcomes:

* You choose a mine position: you lose;
* You choose a free position: it is uncovered to reveal how many neighboring positions have a mine. In the case that the chosen position does not have any neighboring mines, all of its neighbor positions are also uncovered. This uncovering process proceeds recursively: every time a position is uncovered which has no mines as neighbors, all of its neighbors are recursively uncovered.

# Objectives

1. Use structs and 2D arrays generated dynamically;
2. Familiarity with operations over 2D-matrices, like checking neighbor, verifying bounds, etc;
3. Design recursive functions.

# Description of the task

We are going to represent each position of the game matrix by a struct with an integer and a boolean. The boolean indicates if the position is open or not. If the position is not open, the integer is either -1 (indicating a mine) or 0 (not a mine). If the position is open, the value is the number of mines adjacent to the position.

struct minePosition{

int neighborMines;

bool open;

};

**The three functions you have to implement in functions.cpp are**:

* int countNeighborMines(struct minePosition \*\*mineMatrix, int nRows,   
   int nCols,

int row, int col);

This function gets a game matrix with nRows rows and nCols columns and a position (row, col) and returns how many mines (-1) there are in positions adjacent to (row, col) in the matrix. The adjacent positions are the (up to) 8 cells surrounding the cell at (row, col). Be careful on the edges not to access the matrix out of bounds.

* int countClosedPositions (struct minePosition \*\*mineMatrix,

int nRows, int nCols);

This function gets a game matrix with nRows rows and nCols columns and returns how many positions are closed. Closed positions are those cells in the matrix which have not yet been opened by the user.

* int openPosition(struct minePosition \*\* mineMatrix, int nRows, int nCols,

int row, int col);

This **recursive** function gets a game matrix with nRows rows and nCols columns and a position (row, col) and does one of the following actions:

* If the user choose a position with a mine, it opens the position, prints “BOOOOOOOOOOOOOOOOOOOOOOM” and returns MINE (-1);
* Otherwise, it returns 0. Furthermore, it opens the position and counts the number of neighbors which have mines, updating the cell accordingly. If the position has no neighbor mines, it **recursively** opens all neighboring positions.

The main function is given in file minesweeper.cpp, and the header file containing the MINE constant, the definition of struct minePosition and the function declarations is functions.h.

# Requirements

When developing your solution to this problem, ensure that your program conforms to the following requirements and assumptions:

* You will submit 1 file: functions.cpp. This file, along with functions.h and minesweeper.cpp, is provided to you with part of the code already done. You only have to write three functions in functions.cpp.
* The main function in minesweeper.cpp implements the game. So, when you finish the three functions, and compile the program, you will be able to play the game on your computer.