Code Documentation

Minesweeper in C

By: Talha Khan

SS2024 ELE PROGRAMMING

Table of Contents

Introduction

Simple C version of the Minesweeper game, where the goal is to uncover all non-mine cells on a grid while avoiding mines. The game offers different difficulty levels. Users can navigate and open cells, and flag potential mine locations using keyboard input.

Libraries

- stdio.h
- stdlib.h
- time.h

- conio.h

Constants/Macros

Constant	Value	Description
`MAX_BOARD_SIZE`	16	Maximum size of the game board
`MINE_VALUE`	-1	Value representing a mine cell
`FLAGGED_CELL`	'F'	Character representing a flagged cell

Game State Variables

Variable(s)	Type	Description
`i, j`	`int`	Index counters to be used by all for() loops
`board_Size`	`int`	Size of the game board (determined by difficulty level)
`num_Mines`	`int`	Number of mines on the board (determined by difficulty level)
`game_Board`	`int[MAX_BOARD_SIZE][MAX_BOARD_SIZE]`	2D array representing the game board
`visible_Cells`	`int[MAX_BOARD_SIZE][MAX_BOARD_SIZE]`	2D array tracking visible cells on the board
`cursor_X`	`int`	X-coordinate of the current cursor position
`cursor_Y`	`int`	Y-coordinate of the current cursor position
`remaining_Cells`	`int`	Number of remaining cells to reveal

Function Documentation

display_Intro()

```
void display_Intro();
```

Displays the game introduction screen. The puts() function prints the ASCII art title screen and calls the `choose_Difficulty_Level()` function.

choose_Difficulty_Level()

```
void choose_Difficulty_Level();
```

Prompts the user to choose the game difficulty level and sets the game parameters. Asks the user to enter '1' for Easy, '2' for Medium, or '3' for Hard. Setting the board to Easy will generate an 8x8 grid with 15 mines. Medium will set the board to a 12x12 grid with 30 mines and Hard will set it to a 16x16 grid with 45 mines. Uses the `getch()` function to read the user's input and sets the `board_Size` and `num_Mines` variables accordingly.

initialize_Board()

```
void initialize_Board();
```

Initializes the game board with mines and calculates the number of adjacent mines. Resets the `game_Board` and `visible_Cells` arrays. Using the `srand()` function it plants mines pseudo-randomly on the board based on the `num_Mines` value and calculates the number of adjacent mines for each non-mine cell, storing the values in the `game_Board` array.

reveal_Adjacent_Cells()

```
void reveal_Adjacent_Cells(int x, int y);
```

Parameters:

- `x`: The x-coordinate of the cell to reveal.
- `y`: The y-coordinate of the cell to reveal.

Recursively reveals adjacent cells starting from the given cell. Reveals the cell at the specified coordinates (`x`, `y`) and recursively reveals all adjacent cells if the initial cell has no adjacent mines. If a cell is out of bounds, already visible, or a mine, the function returns without performing any action.

flag_Cell()

```
void flag_Cell();
```

Flags or unflags the cell at the current cursor position. This function toggles the flag state of the cell at the current cursor position (`cursor_X`, `cursor_Y`). If the cell is not visible, it is marked as flagged (`FLAGGED_CELL`). If the cell is already flagged, it is unmarked.

draw_Board()

```
void draw_Board();
```

Draws the game board on the console. Clears the console screen after each input and prints the game board with the current state of visible cells, mines, and the cursor position. Uses ASCII characters to represent different cell states (visible, flagged, mine).

move_Cursor()

```
void move_Cursor(char direction);
```

Parameters:

- `direction`: The character representing the direction to move the cursor ('w' for up, 's' for down, 'a' for left, 'd' for right).

Moves the cursor in the specified direction. Updates the `cursor_X` and `cursor_Y` variables based on the user's input direction, ensuring that the cursor does not move out of the grid's boundaries.

reveal_Cells()

```
void reveal_Cells();
```

Reveals the values of all cells upon the user winning or losing the game and prints the final game board.

int_main()

```
int main();
```

Main function that runs the game loop. Displays the intro screen, allows the user to choose the difficulty level, and then runs the game loop until the user wins or loses. It also handles the "Play again?" prompt and allows the user to start a new game and change difficulty, or quit.

Additional Notes:

- Used a combination of Visual Studio Code and Dev-C++.
- Attempted to create clearly commented functions for every game mechanic to make the code more understandable.
- Avoided pointers due to a lack of foundational knowledge and to increase readability.
- Used AI to provide more structure and to increase readability.