

**CSE 102 ASSIGNMENT 7 REPORT**  
**MUHAMMET FATİH ALBAYIN**

## THE CODE

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <strings.h>
4  #include <stdlib.h>
5  #include <time.h>
6
7  #define MAX_SIZE 10
8  #define MAX_MOVES 100
9
10 int move_rows[MAX_MOVES];          /* To store the move's rows    */
11 int move_cols[MAX_MOVES];          /* To store the move's columns */
12 int move_start[MAX_MOVES];         /* For undo functionality     */
13 int move_top = -1;                 /* To keep the top move       */
14 int move_count = 0;                /* To keep the move count     */
15 int total_cells, revealed_cells = 0; /* To check if the game ended */
16
17
18 int randomize_size();
19 int randomize_board(char board[][MAX_SIZE], int size);
20 int mine_amount(int size);
21 void print_board(char board[][MAX_SIZE], int size);
22 int update_board(char board[][MAX_SIZE], int row, int col, int mine_amount);
23 int save_move(int move_row, int move_col);
24 int check_move(int row, int col, int size);
25 int play_game();
26 void check_around(char board[][MAX_SIZE], int row, int col, int size);
27 void push_move(int row, int col);
28 void start_player_move();
29 void undo_move(char board[][MAX_SIZE]);
30 void remove_last_move();
31 void save_total_moves();
32
33
34 int main() {
35     play_game();
36
37     return 0;
38 }
39
40
41 int play_game() {                  /* General function for
managing the whole game */
42     char move[10];                /* The move
*/
43     int row, col, size;           /* Row, column input and size
of the grid */
44     int isFaulty = 1, gameon = 1; /* For the input and game
loops */
45     int mines;                   /* Amount of mines
*/
46
47     size = randomize_size();      /* Randomizes the grid size
*/
48     char board[size][MAX_SIZE];
```

```

49     randomize_board(board, size);                                /* Randomizes the board
*/
50     mines = mine_amount(size);                                  /* Counts the mines in the
board                                     */
51
52     total_cells = (size*size) - mines;                          /* Amount of empty cells
*/
53
54
55     printf("===Welcome to the Minesweeper Game===\n");
56
57     FILE *file = fopen("moves.txt", "w");                        /* Opens the file to store
moves                                     */
58
59     fprintf(file, "---Game Moves---\n");
60
61     while(gameon) {
62         print_board(board, size);                                /* Prints the board before
each move                               */
63
64         while(isFaulty) {                                        /* Checks the input and
directs to the needed function */
65             printf("Enter the row and the column(or type undo): ");
66             fgets(move, sizeof(move), stdin);
67
68             move[strcspn(move, "\n")] = 0;
69
70             if(sscanf(move, "%d %d", &row, &col) == 2 && row ≤ size - 1 && col ≤
size - 1 && row ≥ 0 && col ≥ 0) {
71                 int n = check_move(row, col, size);
72                 if (n == 1) {
73                     printf("BOOM! You hit a mine. Game Over.\n");
74                     gameon = 0;
75                 }
76                 else {
77                     start_player_move();
78                     check_around(board, row, col, size);
79                     save_move(row, col);
80                 }
81
82                 isFaulty = 0;
83             }
84             else if(strcasecmp(move, "undo") == 0) {
85                 undo_move(board);
86                 remove_last_move();
87                 isFaulty = 0;
88             }
89             else {
90                 printf("Please enter a valid input!\n");
91                 isFaulty = 1;
92             }
93             if (revealed_cells == total_cells) {
94                 printf("Congratulations! You cleared the board without hitting a
mine!\n");
95                 gameon = 0;

```

```

96         }
97
98     }
99     isFaulty = 1;
100 }
101 if(!gameon)
102     save_total_moves();
103 }
104
105 int randomize_size() { /* Randomize the board size
106     */
107     int size;
108     srand(time(NULL));
109
110     size = rand() % 8 + 2; /* Between 2 and 10
111     */
112     return size;
113 }
114
115 int randomize_board(char board[][MAX_SIZE], int size){ /* Randomly places mines
116     and prints it to the file */
117     int i, j;
118     srand(time(NULL));
119     FILE *fptr = fopen("map.txt", "w");
120
121     for(i = 0; i < size; i++){
122         for (j = 0; j < size; j++) {
123             if (rand() % 5 == 0){ /* 20% chance of mine
124                 */
125                 board[i][j] = '*';
126                 fprintf(fptr, "%c ", board[i][j]);
127                 if (j == size - 1 && i != size - 1)
128                     fprintf(fptr, "\n");
129             }
130             else {
131                 board[i][j] = '.';
132                 fprintf(fptr, "%c ", board[i][j]);
133                 if (j == size - 1 && i != size - 1)
134                     fprintf(fptr, "\n");
135             }
136             board[i][j] = '#'; /* Covers the board with
137             '#' */
138         }
139     }
140     fclose(fptr);
141 }
142
143 void print_board(char board[][MAX_SIZE], int size){ /* Prints the board to the
144     console */
145     int row, col;
146     printf(" ");
147     for (col = 0; col < size; col++) {

```

```

144     printf("%d ", col);
145 }
146 printf("\n");
147
148 for (row = 0; row < size; row++) {
149     printf("%d ", row);
150     for (col = 0; col < size; col++) {
151         printf("%c ", board[row][col]);
152         if(col == size - 1) {
153             printf("\n");
154         }
155     }
156 }
157 }
158
159 int update_board(char board[][MAX_SIZE], int row, int col, int mine_amount) {
160     board[row][col] = mine_amount + '0';          /* Places the number
161     values to where it is told to */
162 }
163
164 void check_around(char board[][MAX_SIZE], int row, int col, int size) {
165     int mines_around = 0, i, j;
166     int directions[8][2] = { {-1,-1}, {-1,0}, {-1,1}, {0,-1}, {0,1}, {1,-1},
167     {1,0}, {1,1} };
168     char array[MAX_SIZE][MAX_SIZE];
169     int first_call = 1;
170
171     if (first_call) {                                /* Copies the board from
the file at first run */
172         FILE *fptr = fopen("map.txt", "r");
173         for (i = 0; i < size; i++) {
174             for (j = 0; j < size; j++) {
175                 fscanf(fptr, " %c", &array[i][j]);
176             }
177         }
178         fclose(fptr);
179         first_call = 0;
180     }
181
182     if (board[row][col] != '#')                    /* Doesn't repeat the
process if it's already opened */
183         return;
184
185     revealed_cells++;                                /* Increments the revealed
cell count */
186
187     for(i = 0; i < 8; i++) {                        /* Checks the adjacent
coordinates */
188         int adj_row = row + directions[i][0];
189         int adj_col = col + directions[i][1];
190
191         if(adj_row ≥ 0 && adj_row < size && adj_col ≥ 0 && adj_col < size) {

```

```

192         if(array[adj_row][adj_col] == '*')
193             mines_around++;
194     }
195 }
196
197     update_board(board, row, col, mines_around);           /* Updates the board
coordinate with the mine count */
198     push_move(row, col);                                   /* Pushes move to stack
*/
199
200     if(mines_around == 0) {                                /* Recursively calls the
function if there are no mines around */
201         for(i = 0; i < 8; i++) {
202             int adj_row = row + directions[i][0];
203             int adj_col = col + directions[i][1];
204
205             if(adj_row ≥ 0 && adj_row < size && adj_col ≥ 0 && adj_col < size) {
206                 check_around(board, adj_row, adj_col, size);
207             }
208         }
209     }
210 }
211
212
213
214 int check_move(int row, int col, int size) {
215     char array[size][size];
216     int i, j;
217
218     FILE *fptr = fopen("map.txt", "r");                    /* Checks the move from
the file */
219     for (i = 0; i < size; i++) {
220         for(j = 0; j < size; j++) {
221             fscanf(fptr, " %c", &array[i][j]);
222         }
223     }
224     if(array[row][col] == '*')
225         return 1;
226     else
227         return 0;
228 }
229
230
231
232 int save_move(int move_row, int move_col) {                /* Saves the move to the
file */
233     FILE *file = fopen("moves.txt", "r");
234
235     char lines[MAX_MOVES][50];
236     int line_count = 0, i;
237
238     while (fgets(lines[line_count], sizeof(lines[0]), file)) { /* Counts the
lines */
239         if (strncmp(lines[line_count], "Total Moves", 11) ≠ 0) {

```

```

240         line_count++;
241     }
242 }
243 fclose(file);
244
245 file = fopen("moves.txt", "w");
246
247 fprintf(file, "---Game Moves---\n");           /* Prints the
move info to the file */
248 for (i = 1; i < line_count; i++) {
249     fprintf(file, "%s", lines[i]);
250 }
251
252 move_count++;
253 fprintf(file, "Move %d: (Row %d, Col %d)\n", move_count, move_row, move_col);
254
255 fprintf(file, "Total Moves: %d\n", move_count);
256
257 fclose(file);
258 return 0;
259 }
260
261
262 int mine_amount(int size) {                     /* Counts the mines the
board */
263     int i, j, mines = 0;
264     char check;
265     FILE *file = fopen("map.txt", "r");
266
267     for(i = 0; i < size; i++) {
268         for(j = 0; j < size; j++){
269             fscanf(file, " %c", &check);
270             if(check == '*')
271                 ++mines;
272         }
273     }
274     return mines;
275 }
276
277
278 void push_move(int row, int col) {              /* Pushes the move to the
stack with coordinate info */
279     if (move_top < MAX_MOVES - 1) {
280         move_top++;
281         move_rows[move_top] = row;
282         move_cols[move_top] = col;
283     }
284 }
285
286
287 void start_player_move() {                      /* Starts the move saving
process */
288     move_start[move_count] = move_top + 1;
289 }

```

```

290
291
292 void undo_move(char board[][MAX_SIZE]) {
293     if (move_count == 0) {
294         printf("There is no move to undo!\n");           /* Checks the move count
*/
295         return;
296     }
297
298     int start = move_start[move_count - 1];
299
300     while (move_top ≥ start) {                             /* Hides the opened cells
*/
301         int row = move_rows[move_top];
302         int col = move_cols[move_top];
303         board[row][col] = '#';
304         move_top--;
305     }
306
307     move_count--;                                           /* Decrements the move
count                                     */
308 }
309
310
311 void remove_last_move() {                                  /* Removes the last move
from the stack                             */
312     FILE *file = fopen("moves.txt", "r");
313
314     char lines[MAX_MOVES][50];
315     int total_lines = 0, i;
316
317     while (total_lines < MAX_MOVES && fgets(lines[total_lines], sizeof(lines[0]),
file)) {
318         total_lines++;                                     /* Count lines in the file
*/
319     }
320     fclose(file);
321
322     if (total_lines < 3) {
323         return;
324     }
325
326     int moveLineIndex = total_lines - 2;                   /* Copies the file info
excluding last two lines and paste it to the file */
327
328     for (i = moveLineIndex; i < total_lines - 1; i++) {
329         strcpy(lines[i], lines[i + 1]);
330     }
331     total_lines--;
332
333     sprintf(lines[total_lines - 1], "Total Moves: %d\n", move_count);
334
335     file = fopen("moves.txt", "w");
336

```



```

337     for (i = 0; i < total_lines; i++) {
338         fprintf(file, "%s", lines[i]);
339     }
340     fclose(file);
341 }
342
343
344 void save_total_moves() { /* Prints the total moves
to the file */
345     FILE *file = fopen("moves.txt", "a");
346
347     fprintf(file, "\nTotal Moves: %d\n", move_count);
348     fclose(file);
349 }

```

## OUTPUTS OF THE CODE

```

albay@albay-VirtualBox: ~/Desktop
albay@albay-VirtualBox:~/Desktop$ gcc -ansi minesweeper.c -o a
albay@albay-VirtualBox:~/Desktop$ ./a
===Welcome to the Minesweeper Game===
 0 1 2 3 4 5 6
0 # # # # # #
1 # # # # # #
2 # # # # # #
3 # # # # # #
4 # # # # # #
5 # # # # # #
6 # # # # # #
Enter the row and the column(or type undo): 7 0
Please enter a valid input!
Enter the row and the column(or type undo): 6 0
 0 1 2 3 4 5 6
0 # # # # # #
1 # # # # # #
2 # # # # # #
3 # # # # # #
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
Enter the row and the column(or type undo): 0 5
 0 1 2 3 4 5 6
0 # # # # # 1 #
1 # # # # # #
2 # # # # # #
3 # # # # # #
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
Enter the row and the column(or type undo): undo

```

```
albay@albay-VirtualBox: ~/Desktop
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
Enter the row and the column(or type undo): 0 5
0 1 2 3 4 5 6
0 # # # # # 1 #
1 # # # # # # #
2 # # # # # # #
3 # # # # # # #
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
Enter the row and the column(or type undo): undo
0 1 2 3 4 5 6
0 # # # # # # #
1 # # # # # # #
2 # # # # # # #
3 # # # # # # #
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
Enter the row and the column(or type undo): 3 0
0 1 2 3 4 5 6
0 # # # # # # #
1 # # # # # # #
2 # # # # # # #
3 2 # # # # # #
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
Enter the row and the column(or type undo): 1 3
0 1 2 3 4 5 6
0 # # # # # # #
1 # # # 2 # # #
2 # # # # # # #
3 2 # # # # # #
4 # 1 1 3 # 3 1
5 1 1 0 1 1 1 0
6 0 0 0 0 0 0 0
```

## THE FILES' CONTENT

```
moves.txt
~/Desktop
Ln 5, Col 15
1 ---Game Moves---
2 Move 1: (Row 6, Col 0)
3 Move 2: (Row 3, Col 0)
4 Move 3: (Row 1, Col 3)
5 Total Moves: 3
```

```
map.txt
~/Desktop
1 . * * * . . .
2 . . . . . *
3 * . . . . *
4 . . . * * . *
5 * . . . * . .
6 . . . . . .
7 . . . . . .
```