

CS 101 PROJECT

MINESWEEPER

Submitted To: -

Submitted By:-

Dr. D. B. Phatak

Laksh Agarwal (140020082)

Dr. Supratik Chakraborty

Amritesh Aryan (140020087)

Kumar Spandan Sardar (140020107)

Group No. 06

INDEX

1. Status Of Completion
2. Project Overview
3. Rules & Basics
4. Features
5. Functions Used
6. Screenshots

STATUS OF COMPLETION

The Project (Level 1 & 2) has been completed successfully. All members from our team participated well. Minesweeper is working perfectly without graphics.

We have completed the Minesweeper including 3 levels.

Project Overview

Our Team (Lab Batch 06) was working all together. We all attended all the meets. We all had done the work as per assigned by our group leader (Laksh Agarwal).

The work assigned to the Team Members is:-

Laksh Agarwal :- Decided all the meets, decided what work should be given to whom, fabricated the whole project, writes the major part of the program (Knows small amount of programming earlier).

Kumar Spandan Sardar :- Attended all the meets, typed the user manual, helped in writing of the program, Best Typer among us (Beginner at programming).

Amritesh Aryan :- Attended all the meets, typed the SRS file, helped in writing of program, Googling for understanding the SRS file (Beginner at programming).

RULES & BASICS

The objective is to find the empty squares in the minefield while avoiding the mines. When you find all the squares without detonating a single mine, you win the game. The faster you clear the board, the better your score.

The Minesweeper Board

Minesweeper has four standard boards to choose from, each progressively more difficult.

Beginner	Level	:	06*06	Tiles,	06	Mines
Amateur	Level	:	10*10	Tiles,	10	Mines
Professional	Level	:	15*15	Tiles,	15	Mines

How To Play

The rule in minesweeper is simple:

Uncover a mine, and game ends.

Uncover a empty square, and u keep playing.

Uncover a number, and it tells you how many mines lay hidden in the eight surrounding squares-Information you use to deduce which nearby squares are safe to click.

You win if you correctly flag all the boxes which contain mines.

Hints & Tips

Mark the mines. If you suspect a square conceals a mine, click on it when flag mode is on. This puts a flag on the square.

Study the patterns. If three square in a row display 2-3-2, then you know three mines are probably lined up beside that row. If a square says 8, every surrounding square is mined.

Explore the unexplored. Not sure where to click next?

Try clearing some unexplored territory. You're better off clicking in the middle of unmarked squares than in an area you suspect is mined.

FEATURES

Some of the features that is available to the user:

Option to choose **THREE DIFFICULTY LEVELS**.

The user will get randomly mined maps every time.

In case of losing, Dialog box will show TWO options:
New Game, Exit Minesweeper.

We are showing the Minesweeper Board at last, when the User get lost. (Board is showing the place of mines.)

FUNCTIONS USED

The following are the various code intercepts written by our Team Members for various input & output operations. This includes only the main functions that are used. Minor functions have been avoided & details about those functions have been given as comment lines in the program.

Different Libraries Used

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
```

We have used Global Variables to make our project less complicated.

Different Functions Made

```
void build_board_b();  
void build_board_a();  
void build_board_p();
```

```
void build_gboard_b();  
void build_gboard_a();  
void build_gboard_p();
```

```
void create_mines_b();  
void create_mines_a();  
void create_mines_p();
```

```
void print_board_b();  
void print_board_a();  
void print_board_p();
```

```
void print_fullboard_b();  
void print_fullboard_a();  
void print_fullboard_p();
```

\

```
void start_b();  
void start_a();  
void start_p();
```

```
int play_game_b();  
int play_game_a();  
int play_game_p();
```

```
void play_again_b();  
void play_again_a();  
void play_again_p();
```

```
int check_win_game_b();  
int check_win_game_a();  
int check_win_game_p();
```

```
void check_for_mine_b(int, int);  
void check_for_mine_a(int, int);  
void check_for_mine_p(int, int);
```

```
int check_for_nearby_mines_b(int, int);  
int check_for_nearby_mines_a(int, int);  
int check_for_nearby_mines_p(int, int);
```

Explanations Of Different Functions Used

```
void build_board_b();  
void build_board_a();  
void build_board_p();
```

This function is used to build board which is containing the mines & the no. associated to other locations.

This no. contains how many mines are there around that place.

```
void build_gboard_b();  
void build_gboard_a();  
void build_gboard_p();
```

This function is used to display the board which is seen by the user on screen.

```
void create_mines_b();  
void create_mines_a();  
void create_mines_p();
```

This functions is used to create mines on random locations using Randon function.

```
void print_board_b();  
void print_board_a();  
void print_board_p();
```

This function is used to print the board on the screen.

```
void print_fullboard_b();  
void print_fullboard_a();  
void print_fullboard_p();
```

This function is used to print the original board when user either loses or wins the game.

```
void start_b();  
void start_a();  
void start_p();
```

This function is called by main function. This function then calls all the functions sequentially to execute.

```
int play_game_b();  
int play_game_a();  
int play_game_p();
```

This function is the biggest & most complex of all functions. This functions handles all the instructions used in playing Minesweeper.

```
void play_again_b();  
void play_again_a();  
void play_again_p();
```

This function executes when user either lose or wins the game. The role of the function is ask user whether he want to play again or not.

```
int check_win_game_b();  
int check_win_game_a();  
int check_win_game_p();
```

This function checks whether the player has won or not.

```
void check_for_mine_b(int, int);  
void check_for_mine_a(int, int);  
void check_for_mine_p(int, int);
```

This function checks that the place given by user has mine or not.

```
int check_for_nearby_mines_b(int, int);  
int check_for_nearby_mines_a(int, int);  
int check_for_nearby_mines_p(int, int);
```

This functions works in coordination with play_game. This function is the back bone of play_game function. It tells the play_game function what no. is placed on the location asked by the calling function.

