Student Name: Fengxi Zhang & Chengchu Wu

Student ID: 20012100022 & 20012100032

H00365040 & H00365059

Programme: BEng in Telecommunications Engineering

University: Xidian University & Heriot-Watt University

Year: Year 1

Submission Date:17/1/2021

Mine Sweeper game implemented by C language

1. Introduction

The game named Mine Sweeper is definitely one of the most famous and classical games in the world. It is a public puzzle game of all ages. In this game, player should manage to find all the grids which have no mines as soon as possible according to the numbers in the grids which player have chosen yet. There will be a GAME OVER if player chooses the grid which has a mine.

Using C language, we can write a program to simply implement the Mine Sweeper game. In this program, we will use a lot of knowledge including multidimensional array, object-like macro, conditional compilation and so on.

2. Aims and objectives

The aim of our experiment is to use C language to simply implement the Mine Sweeper game. Besides, we had better add more contents to our program which can optimize our program and let the game to implement more functions by studying about more C language's knowledge on our own.

And the objective of our experiment is to deepen our impressions on what we have learnt, enhancing our ability to code programs by using C language.

3. Theory

1) Multidimensional array

A multi-dimensional array is an array with more than one level or dimension. For example, a 2D array, or two-dimensional array, is an array of arrays, meaning it is a matrix of rows and columns (just think of a table). A 3D array adds another dimension, turning it into an array of arrays of arrays.

2) Object-like macro

An object-like macro is a simple identifier which will be replaced by a code fragment. It is called object-like because it looks like a data object in code that uses it. They are most commonly used to give symbolic names to numeric constants.

3) Conditional compilation

Conditional compilation is compilation implementing methods which allow the compiler to produce differences in the executable program produced and controlled by parameters that are provided during compilation.

4. Experimental method

4.1 Modeling Procedure

1) Data structure: Adopt two - dimensional array data structure.

To write a Mine Sweeper, we should first create a checkerboard, which is actually a matric. Thus, we adopt two-dimensional array data structure, creating two two-dimensional char arrays. One of them named show_mine is created to show the checkerboard for player to play. The other one named real_mine is created to set up mines and showing the answer to mines' positions for designer to check the program or failed player.

2) Algorithm: Using multiple functions with clear structures and clear effects.

1>main function

We created main function to call game function to execute the game as well as give player a chance to continue the game or not after a game is over.

2>menu function

We created menu function to print the game beginning menu.

3>game function

We created game function to call other functions to execute main process of a game.

4>init_mine function

We created init_mine function to initialize two two-dimensional char arrays called show_mine and real_mine. Assign all of show_mine's elements to '*'. Assign all of real_mine's elements to '0'.

5>set_mine function

We created set_mine function to set up mines in real_mine char array.

6>print_mine function

We created print_mine function to print checkerboard containing real_mine char array for player to see the answer.

7>print_player function

We created print_player function to print checkerboard containing show_mine char array for player as their reference to their next action to continue the game.

8>safe_mine function

We created safe_mine function to prevent player from being blown up for the first time they input.

9>sweep_mine function

We created sweep_mine function to detect if the player's input is on a mine.

10>count_show_mine function

We created count_show_mine function to count the number of '*' left on the player's checkerboard.

4.2 Programming Procedure

Step 1: Create a new project named MineSweeper

We first created a new project named MineSweeper. And we create a header file named game.h and two resource files named game.c and main.c.

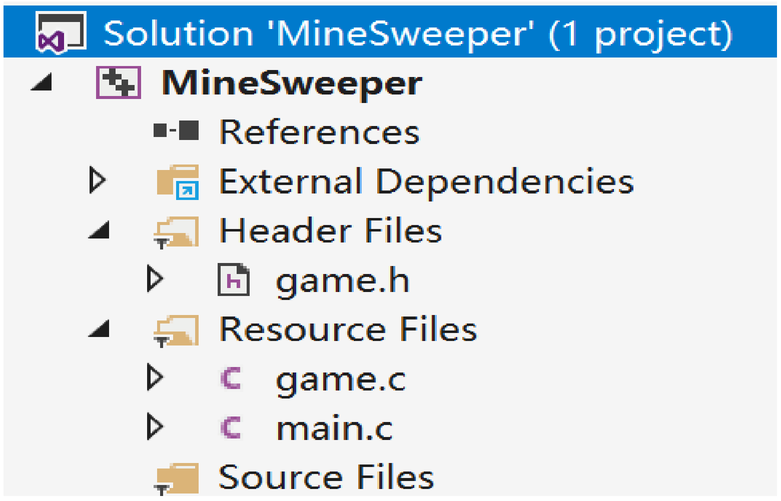


Figure 4.2.1: Project contents

Step 2: Code header file and resource files

1) Code header file

Include header files.

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

Define object-like macros.

The value of ROW is the number of checkerboard's rows.

The value of COL is the number of checkerboard's columns.

The value of COUNT is the number of mines we set.

```
#define ROW 10
#define COL 10
#define COUNT 10
```

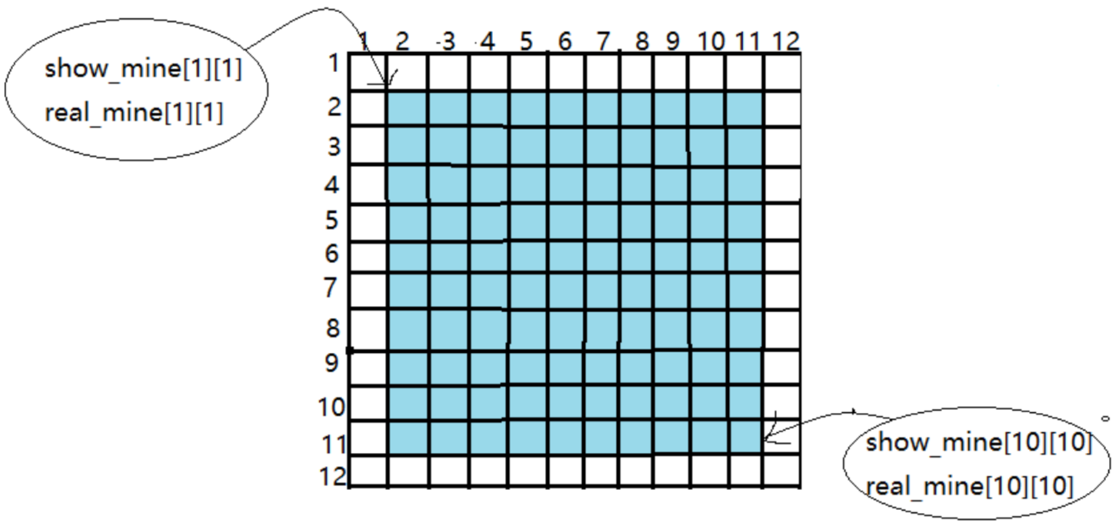


Figure 4.2.2: A visual model of the show_mine and real_mine array.

The value of ROWS is the number of arrays' rows.

The value of COLS is the number of arrays' columns.

```
#define ROWS (ROW+2)
```

```
#define COLS (COL+2)
```
State show_mine and real_mine char array.
```
extern char show_mine[ROWS][COLS];
extern char real_mine[ROWS][COLS];
```
State functions.
```
void menu(void);
void init_mine(void);
void set_mine(void);
int  count_mine(int x, int y);
void print_player(void);
void print_mine(void);
int  sweep_mine(void);
void safe_mine(void);
void open_mine(int x, int y);
int count_show_mine(void);
#endif
```
2)Code resource files

1>Code main.c

Include header file game.h.
```
#include "game.h"
```
Define game function.
```
void game(void)
{
```
Declarations.
```
    double  start, finish;
    int ret = 0;
```
Call init_mine function to initialize two char arrays named show_mine and real_mine.
```
    init_mine();
```
Set up mines for real_time array.
```
    set_mine();
    printf("\n");
```
Call print_player function to print checkerboard of show_mine array for player.
```
    print_player();
```
Use clock function in time.h to get start time.
```
    start = clock();
```
Call safe_mine function to prevent player from being blown up for the first time.
```
    safe_mine();
```
Code the situation that one step to win.
```
    if (count_show_mine() == COUNT)
    {
        print_mine();
        printf("YOU WIN! \n\n");
        return;
    }
```

Print checkerboard of show_mine array for player.

```
    print_player();
```

Use while(1) to Loop sweeping.

```
    while (1)
    {
```

Call the sweep_mine function to sweep the mine. Return 1 if player's input is on a mine; return 0 if player's is not on a mine.

```
        int ret = sweep_mine();
```

If the number of '*' on the player's checkerboard is the number of mines.YOU WIN!

```
        if (count_show_mine() == COUNT)
        {
            print_mine();
            printf("YOU WIN！ \n\n");
```

Use clock function again to get finish time.

```
            finish = clock();
            printf("You have spent %d seconds\n", (int)(finish - start) /
CLOCKS_PER_SEC);
            break;
        }
```

Judge if player's input is on a mine.

```
        if (ret)
        {
            printf("GAME OVER! YOU WERE BLOWN UP!\n");
            finish = clock();
```

Calculate the time the player spent and print.

```
            printf("You have spent %d seconds\n", (int)(finish - start) /
CLOCKS_PER_SEC);
```

Print checkerboard contains real_mine array for player to view the distribution of mines.

```
            print_mine();
            break;
        }
        print_player();
    }
}
int main(void)
{
```

Use srand function in stdlib.h to generate random number seeds.

```
    srand((unsigned int)time(NULL));
    int input = 0;
```

Call menu function to print the beginning menu.

```
    menu();
```

Create a loop (do….while) through which player can replay the game after a game over.

```
    do
    {
```
Assign player's input to the variable "input" .
```
        scanf_s("%d", &input);
        switch (input)
        {
        case 1:
```
Execute the game function to begin the game .
```
            game();
            break;
```
Quit the game if player input 0.
```
        case 0:
            exit(0);
            break;
        default:
            printf("Your input is ILLEGAL,please RETYPE!\n");
            break;
        }
        menu();
        printf("contiue?\n");
    }
    while (input);
    return 0;
}
```
2>Code game.c

Include header file game.h.
```
#include "game.h"
```
Initialize arrays.
```
char show_mine[ROWS][COLS] = { 0 };
char real_mine[ROWS][COLS] = { 0 };
```
Define menu function.
```
void menu(void)
{
    printf("***************************************\n");
    printf("*****   Enter 1 or 0 to continue! *****\n");
    printf("*********1.play      0.exit***********\n");
    printf("***************************************\n");
}
```
Define init_mine function. Use for loop nesting to initialize elements in the two arrays.
```
void init_mine(void)
{
    int i = 0;
    int j = 0;
    for (int i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLS; j++)
        {
            show_mine[i][j] = '*';
            real_mine[i][j] = '0';
        }
    }
}
```
Define print_player function. Use for loop nesting to print checkerboard.
```
void print_player(void)
{
    int i = 0;
    int j = 0;
    printf("0  ");
    for (i = 1; i < ROWS - 1; i++)
    {
```
Print the horizontal coordinates (0—10).
```
        printf("%d ", i);
    }
    printf("\n");
```
Print the vertical coordinates (1—10).
```
    for (i = 1; i < ROWS - 2; i++)
    {
        printf("%d  ", i);
        for (j = 1; j < COLS - 1; j++)
        {
            printf("%c ", show_mine[i][j]);
        }
        printf("\n");
    }
```
Print the last line.
```
    printf("10 ");

    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%c ", show_mine[10][i]);
    }
    printf("\n");
}
```
Define print_mine function. Use for loop nesting to print checkerboard. It has a same principle as print_player function.
```
void print_mine(void)
{
    int i = 0;
    int j = 0;
    printf("0  ");
    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%d ", i);
    }
    printf("\n");
    for (i = 1; i < ROWS - 2; i++)
    {
        printf("%d  ", i);
        for (j = 1; j < COLS - 1; j++)
        {
            printf("%c ", real_mine[i][j]);
        }
        printf("\n");
    }
    printf("10 ");
    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%c ", real_mine[10][i]);
    }
    printf("\n");
}
```

Define set_mine function. Use rand function in stdlib.h to generate random coordinate x and y and set mines (i.e change elements in real_mine from ‘0’ to ‘1’. ‘1’ represent mine.)

```
void set_mine(void)
{
    int x = 0;
    int y = 0;
    int count = COUNT;
```

Break the loop when mines are set up.

```
    while (count)
    {
```

Generates a random number from 1 to 10,setting up a mine at a horizontal coordinate from 1 to 10.

```
        int x = rand() % 10 + 1;
        int y = rand() % 10 + 1;
```

Find a place where is no mine to set up a mine.

```
        if (real_mine[x][y] == '0')
        {
            real_mine[x][y] = '1';
            count--;
        }
    }
}
```

Define count_mine function. Use repetitive ifs to count the mines around a coordinate and return the value by assigning the value to a variable count.

```
int count_mine(int x, int y)
{
    int count = 0;
    if (real_mine[x - 1][y - 1] == '1')
        count++;
    if (real_mine[x - 1][y] == '1')
        count++;
    if (real_mine[x - 1][y + 1] == '1')
        count++;
    if (real_mine[x][y - 1] == '1')
        count++;
    if (real_mine[x][y + 1] == '1')
        count++;
    if (real_mine[x + 1][y - 1] == '1')
        count++;
    if (real_mine[x + 1][y] == '1')
        count++;
    if (real_mine[x + 1][y + 1] == '1')
        count++;
    return count;
}
```

Define safe_mine function. Add more ‘if’ conditions to sweep_mine function to create a function which can reset a mine’s coordinate when the player input this coordinate for the first time to prevent the player blown up for the first time they input, improving the gameplay experience of the player.

```
void safe_mine(void)
{
    int x = 0;
    int y = 0;
    char ch = 0;
    int count = 0;
    int ret = 1;
    printf("Enter coordinate:x(column) y(row) to sweep mine.\n");
    printf("Beware: There is a SPACE between x and y that you input!\n");
    while (1)
    {
        scanf_s("%d %d", &y, &x);
```

Determine if the input coordinate is form 1 to 10, enter again if the input is wrong.

```
        if ((x >= 1 && x <= 10) && (y >= 1 && y <= 10))
```

```c
        {
```
Remedy if the first input is on a mine.
```c
            if (real_mine[x][y] == '1')
            {
                real_mine[x][y] = '0';
                char ch = count_mine(x, y);
```
The difference between the ASCII value of a digit and the ASCII value of a numeric character is 48, which is the ASCII value of '0'
```c
                show_mine[x][y] = ch + '0';
                open_mine(x, y);
```
Set up a mine in the remaining free space.
```c
                while (ret)
                {
```
Generates a random number from 1 to 10,setting up a mine at a horizontal coordinate from 1 to 10 .
```c
                    int x = rand() % 10 + 1;
                    int y = rand() % 10 + 1;
```
Find a place where is no mine to set up a mine.
```c
                    if (real_mine[x][y] == '0')
                    {
                        real_mine[x][y] = '1';
                        ret--;
                        break;
                    }
                }
                break;
```
Break the function.
```c
            }
            if (real_mine[x][y] == '0')
            {
                char ch = count_mine(x, y);
                show_mine[x][y] = ch + '0';
                open_mine(x, y);
                break;
            }
        }
        else
```
When the coordinate is wrong.
```c
        {
            printf("Your input is wrong!Please RETYPE!\n");
        }
    }
}
```
Define sweep_mine function. Use if nesting to return 0 if player's input is not on a

mine while returning 1 if player's input is on a mine.

```
int sweep_mine(void)
{
    int x = 0;
    int y = 0;
    int count = 0;
    printf("Enter coordinate:x(column) y(row) to sweep mine.\n");
    printf("Beware: There is a SPACE between x and y that you input!\n");
    scanf_s("%d %d", &y, &x);
```

Determine if the input coordinate is form 1 to 10, enter again if the input is wrong.

```
    if ((x >= 1 && x <= 10) && (y >= 1 && y <= 10))
    {
```

Player's input is not on a mine.

```
        if (real_mine[x][y] == '0')
        {
            char ch = count_mine(x, y);
```

The difference between the ASCII value of a digit and the ASCII value of a numeric character is 48, which is the ASCII value of '0'.

```
            show_mine[x][y] = ch + '0';
            open_mine(x, y);
```

Determine the number of unknown areas left. If the number is the number of mines, then the player win.

```
            if (count_show_mine() == COUNT)
            {
                print_mine();
                printf("You WIN! \n\n");
                return 0;
            }
        }
        else if (real_mine[x][y] == '1')
        {
            return 1;
        }
    }
    else
    {
        printf("Your input is wrong!Please RETYPE!\n");
    }
    return 0;
}
```

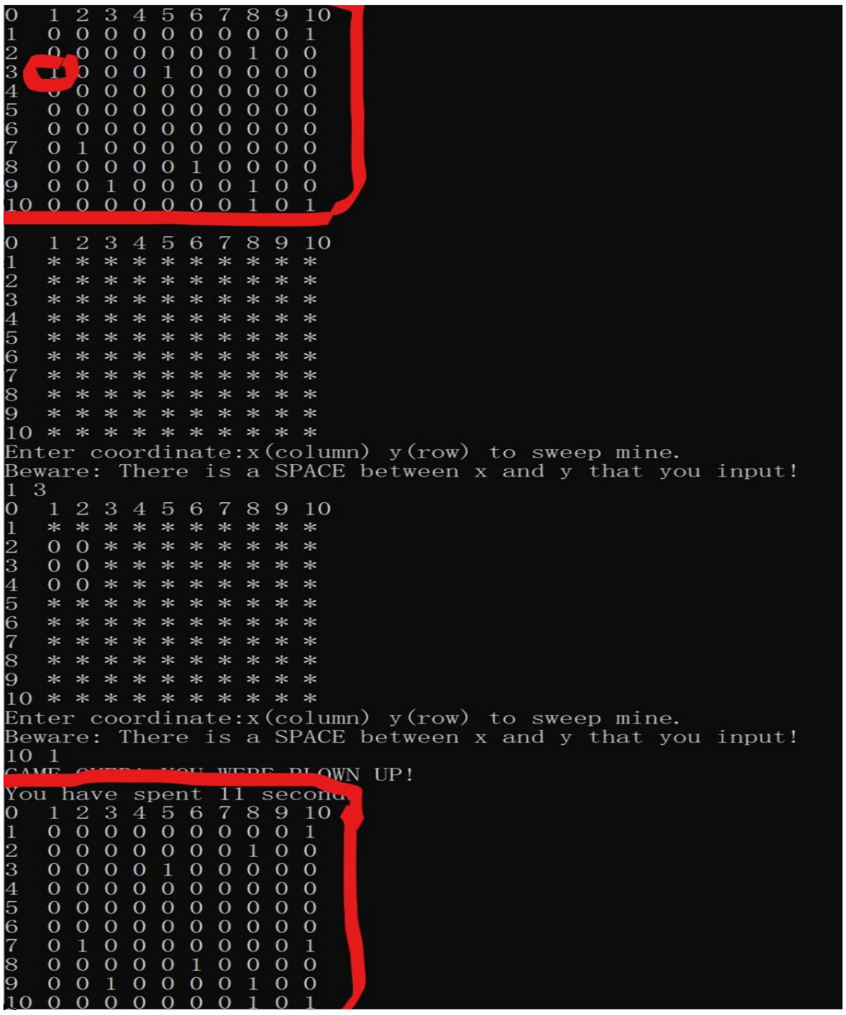
Define open_mine function. Open the grids which is not on the mine around the coordinate the player input and print to display the number of mines around these grids.

```
void open_mine(int x, int y)
```

```
{
    if (real_mine[x - 1][y - 1] == '0')
    {
```
Print to display the number of mines around this coordinate.
```
        show_mine[x - 1][y - 1] = count_mine(x - 1, y - 1) + '0';
    }
    if (real_mine[x - 1][y] == '0')
    {
        show_mine[x - 1][y] = count_mine(x - 1, y) + '0';
    }
    if (real_mine[x - 1][y + 1] == '0')
    {
        show_mine[x - 1][y + 1] = count_mine(x - 1, y + 1) + '0';
    }
    if (real_mine[x][y - 1] == '0')
    {
        show_mine[x][y - 1] = count_mine(x, y - 1) + '0';
    }
    if (real_mine[x][y + 1] == '0')
    {
        show_mine[x][y + 1] = count_mine(x, y + 1) + '0';
    }
    if (real_mine[x + 1][y - 1] == '0')
    {
        show_mine[x + 1][y - 1] = count_mine(x + 1, y - 1) + '0';
    }
    if (real_mine[x + 1][y] == '0')
    {
        show_mine[x + 1][y] = count_mine(x + 1, y) + '0';
    }
    if (real_mine[x + 1][y + 1] == '0')
    {
        show_mine[x + 1][y + 1] = count_mine(x + 1, y + 1) + '0';
    }
}
```
Define count_show_mine function. Use loop nesting to count the number of ‘*’ existing in checkerboard and assign the value to variable count and return it.
```
int count_show_mine()
{
    int count = 0;
    int i = 0;
    int j = 0;
    for (i = 1; i <= ROWS - 2; i++)
    {
        for (j = 1; j <= COLS - 2; j++)
        {
            if (show_mine[i][j] == '*')
            {
                count++;
            }
        }
    }
    return count;
}
```

5. Results

1) safe_mine function demo

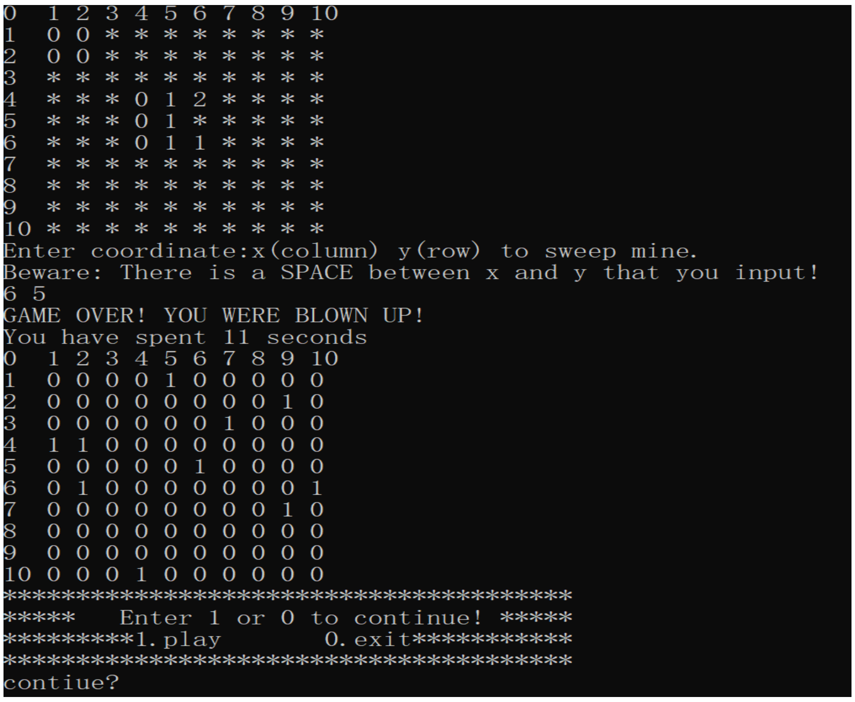


```
0   1 2 3 4 5 6 7 8 9 10
1   0 0 0 0 0 0 0 0 0 1
2   0 0 0 0 0 0 0 1 0 0
3   1 0 0 0 1 0 0 0 0 0
4   0 0 0 0 0 0 0 0 0 0
5   0 0 0 0 0 0 0 0 0 0
6   0 0 0 0 0 0 0 0 0 0
7   0 1 0 0 0 0 0 0 0 0
8   0 0 0 0 0 1 0 0 0 0
9   0 0 1 0 0 0 0 1 0 0
10  0 0 0 0 0 0 0 1 0 1
0   1 2 3 4 5 6 7 8 9 10
1   * * * * * * * * * *
2   * * * * * * * * * *
3   * * * * * * * * * *
4   * * * * * * * * * *
5   * * * * * * * * * *
6   * * * * * * * * * *
7   * * * * * * * * * *
8   * * * * * * * * * *
9   * * * * * * * * * *
10  * * * * * * * * * *
Enter coordinate:x(column) y(row) to sweep mine.
Beware: There is a SPACE between x and y that you input!
1 3
0   1 2 3 4 5 6 7 8 9 10
1   * * * * * * * * * *
2   0 0 * * * * * * * *
3   0 0 * * * * * * * *
4   0 0 * * * * * * * *
5   * * * * * * * * * *
6   * * * * * * * * * *
7   * * * * * * * * * *
8   * * * * * * * * * *
9   * * * * * * * * * *
10  * * * * * * * * * *
Enter coordinate:x(column) y(row) to sweep mine.
Beware: There is a SPACE between x and y that you input!
10 1
GAME OVER! YOU WERE BLOWN UP!
You have spent 11 seconds
0   1 2 3 4 5 6 7 8 9 10
1   0 0 0 0 0 0 0 0 0 1
2   0 0 0 0 0 0 0 1 0 0
3   0 0 0 0 1 0 0 0 0 0
4   0 0 0 0 0 0 0 0 0 0
5   0 0 0 0 0 0 0 0 0 0
6   0 0 0 0 0 0 0 0 0 0
7   0 1 0 0 0 0 0 0 0 1
8   0 0 0 0 0 1 0 0 0 0
9   0 0 1 0 0 0 0 1 0 0
10  0 0 0 0 0 0 0 1 0 1
```

Figure 5.1 safe_mine function demo

As you can see, we input 1 3(which has a mine originally according to the answer (I.e the first checkerboard shown in the figure.)). We didn't die. And according to the last checkerboard as an answer to mine locations we can see that the originally mine in 1 3 had been reset to another location, which prevented us to be blown up for the first time we input a coordinate. It improves player's gameplay experience.

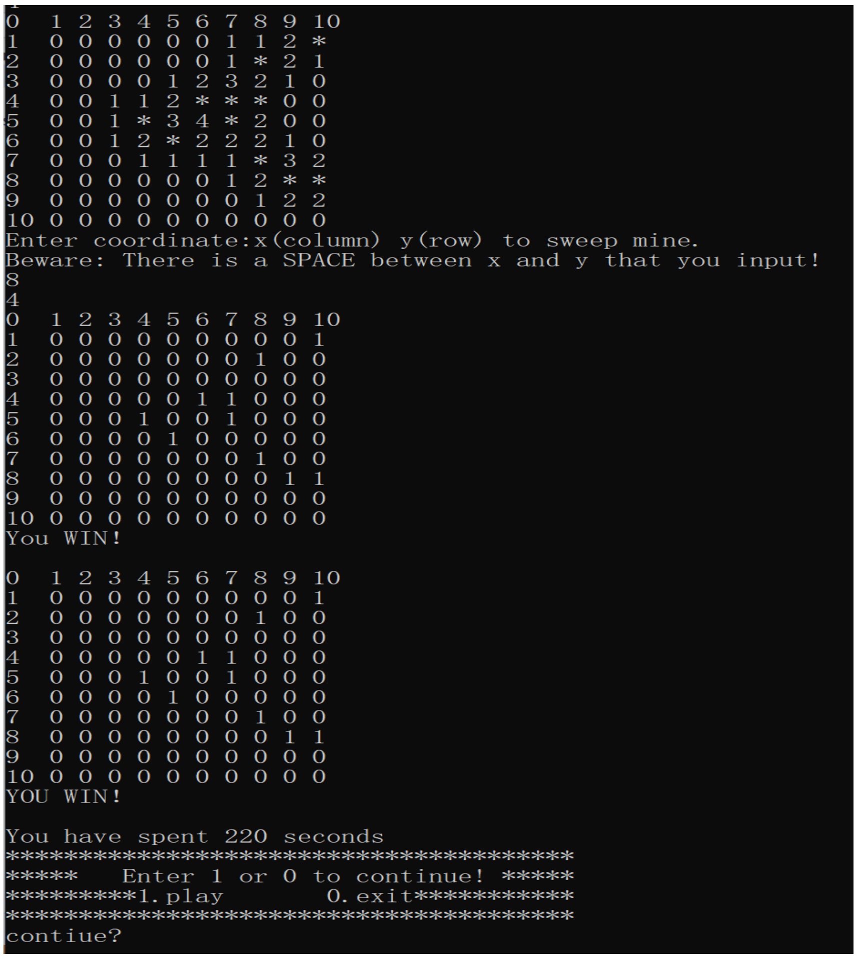
2) The situation when player lose



```
0  1 2 3 4 5 6 7 8 9 10
1  0 0 * * * * * * * * *
2  0 0 * * * * * * * * *
3  * * * * * * * * * * *
4  * * * 0 1 2 * * * * *
5  * * * 0 1 * * * * * *
6  * * * 0 1 1 * * * * *
7  * * * * * * * * * * *
8  * * * * * * * * * * *
9  * * * * * * * * * * *
10 * * * * * * * * * * *
Enter coordinate:x(column) y(row) to sweep mine.
Beware: There is a SPACE between x and y that you input!
6 5
GAME OVER! YOU WERE BLOWN UP!
You have spent 11 seconds
0  1 2 3 4 5 6 7 8 9 10
1  0 0 0 0 1 0 0 0 0 0
2  0 0 0 0 0 0 0 0 1 0
3  0 0 0 0 0 0 1 0 0 0
4  1 1 0 0 0 0 0 0 0 0
5  0 0 0 0 0 1 0 0 0 0
6  0 1 0 0 0 0 0 0 0 1
7  0 0 0 0 0 0 0 0 1 0
8  0 0 0 0 0 0 0 0 0 0
9  0 0 0 0 0 0 0 0 0 0
10 0 0 0 1 0 0 0 0 0 0
***************************************
*****   Enter 1 or 0 to continue! *****
**********1.play       0.exit************
***************************************
contiue?
```

Figure 5.2: The situation when player lose

3) The situation when player win



```
0  1 2 3 4 5 6 7 8 9 10
1  0 0 0 0 0 0 1 1 2 *
2  0 0 0 0 0 0 1 * 2 1
3  0 0 0 0 1 2 3 2 1 0
4  0 0 1 1 2 * * * 0 0
5  0 0 1 * 3 4 * 2 0 0
6  0 0 1 2 * 2 2 2 1 0
7  0 0 0 1 1 1 1 * 3 2
8  0 0 0 0 0 0 1 2 * *
9  0 0 0 0 0 0 0 1 2 2
10 0 0 0 0 0 0 0 0 0 0
Enter coordinate:x(column) y(row) to sweep mine.
Beware: There is a SPACE between x and y that you input!
8
4
0  1 2 3 4 5 6 7 8 9 10
1  0 0 0 0 0 0 0 0 0 1
2  0 0 0 0 0 0 0 1 0 0
3  0 0 0 0 0 0 0 0 0 0
4  0 0 0 0 0 1 1 0 0 0
5  0 0 0 1 0 0 1 0 0 0
6  0 0 0 0 1 0 0 0 0 0
7  0 0 0 0 0 0 0 1 0 0
8  0 0 0 0 0 0 0 0 1 1
9  0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0
You WIN!

0  1 2 3 4 5 6 7 8 9 10
1  0 0 0 0 0 0 0 0 0 1
2  0 0 0 0 0 0 0 1 0 0
3  0 0 0 0 0 0 0 0 0 0
4  0 0 0 0 0 1 1 0 0 0
5  0 0 0 1 0 0 1 0 0 0
6  0 0 0 0 1 0 0 0 0 0
7  0 0 0 0 0 0 0 1 0 0
8  0 0 0 0 0 0 0 0 1 1
9  0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0
YOU WIN!

You have spent 220 seconds
***************************************
*****   Enter 1 or 0 to continue! *****
**********1.play       0.exit************
***************************************
contiue?
```

Figure 5.3: The situation when player win

6. Discussions

1)What is the function of the first print_mine function in game function?

```
    /* Print checkerboard of designer(CAN be delete.Just for testing the
program)

    print_mine();

    */
```

The print_mine() here is written to convenient us designers to check the program written by us. And it should be deleted when the program is offered to player.

2)Why we write a header file named game.h?

To avoid code redundancy and enhance the readability of our codes, we use header file for function statements, including header files, object-like macro definitions and so on.

3)Why we write two resource files named game.c and main.c instead of one? And why we write multiple functions instead of gathering all the codes in one main function?

We define multiple functions instead of gathering all the codes in one main function. In addition, we also define functions except for main and game functions in game.c What's more, we focus on definitions of main and game function in main.c. These measures extremely enhance the readability of our codes, making our codes be of modularization.

7. Conclusions

1)Readability and modularity are very important for a program written in C Language. Writing multiple files can help program gain readability and modularity.

2)Comments is of vital importance for a program written in C Language. A good C Language program needs good comments.

8. Appendices

1) Header File's Codes

1> game.h

```
/*
* File name: game.h
* Author: Fengxi Zhang & Chengchu Wu
* HW ID: H00365040 & H00365059
* IDE: Visual Studio 2017 Community
* Description: Include header files. Define object-like macros.
*             State arrays. State functions.
* Date: 15/1/2021
*/
#ifndef _GAME_H_
#define _GAME_H_
/* Include header files */
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
```

```c
/* Define object-like macros */
#define ROW 10 /* The numbers of arrays' rows */
#define COL 10 /* The numbers of arrays' columns */
#define COUNT 10   /* The numbers of mines */
#define ROWS (ROW+2)  /* The number of checkerboard's rows */
#define COLS (COL+2)  /* The number of checkerboard's columns */
/* State arrays */
extern char show_mine[ROWS][COLS];  /* State an array for show */
extern char real_mine[ROWS][COLS];  /* State an array for setting mines */
/* State functions */
void menu(void);
void init_mine(void);
void set_mine(void);
int  count_mine(int x, int y);
void print_player(void);
void print_mine(void);
int  sweep_mine(void);
void safe_mine(void);
void open_mine(int x, int y);
int count_show_mine(void);
#endif  /* _GAME_H_ */
```

2) Resource Files' Codes

1> game.c

```c
/*
* File name: game.c
* Author: Fengxi Zhang & Chengchu Wu
* HW ID: H00365040 & H00365059
* IDE: Visual Studio 2017 Community
* Description: Define menu, init_mine, print_player, print_mine, set_mine,
count_mine, safe_mine,
*               sweep_mine, open_mine, count_show_mine functions.
* Date: 14/1/2021
*/
/* Include file game.h */
#include "game.h"
/* Initialize arrays */
char show_mine[ROWS][COLS] = { 0 };
char real_mine[ROWS][COLS] = { 0 };
/*
* Function name: menu
* Description: Print game beginning menu
*/
void menu(void)
{
    printf("***************************************\n");
    printf("*****   Enter 1 or 0 to continue! *****\n");
    printf("*********1.play      0.exit***********\n");
    printf("***************************************\n");
}
/*
* Function name: init_mine
* Description: Initialize two arrays.
*/
void init_mine(void)
{
    int i = 0;
    int j = 0;
    for (int i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLS; j++)
        {
            show_mine[i][j] = '*';
            real_mine[i][j] = '0';
        }
    }
}
/*
* Function name: print_player
* Description: Print checkerboard contains show_mine array for player.
*/
void print_player(void)
{
    int i = 0;
    int j = 0;
    printf("0  ");
    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%d ", i); /* Print the horizontal coordinates （0--10） */
    }
    printf("\n");
    for (i = 1; i < ROWS - 2; i++) /* Print the vertical coordinates  （1--10）
*/
    {
        printf("%d  ", i);
        for (j = 1; j < COLS - 1; j++)
        {
            printf("%c ", show_mine[i][j]);
        }
        printf("\n");
    }
    printf("10 "); /* Print the last line */

    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%c ", show_mine[10][i]);
    }
    printf("\n");
}
/*
* Function name: print_mine
* Description: Print the checkerboard contains real_mine array for player to
look at the answer.
*/
void print_mine(void)
{
    int i = 0;
    int j = 0;
    printf("0  ");
    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%d ", i); /* Print the horizontal coordinates (0--10) */
    }
    printf("\n");
    for (i = 1; i < ROWS - 2; i++) /* Print the vertical coordinates (1--10) */
    {
        printf("%d  ", i);
        for (j = 1; j < COLS - 1; j++)
        {
            printf("%c ", real_mine[i][j]);
        }
        printf("\n");
    }
    printf("10 "); /* Print the last line */
    for (i = 1; i < ROWS - 1; i++)
    {
        printf("%c ", real_mine[10][i]);
    }
    printf("\n");
}
/*
* Function name: set_mine
* Description: Set mines for real_mine array. 1 represents there is a mine. 0
```

```c
represents there is no mine.
*/
void set_mine(void)
{
    int x = 0;
    int y = 0;
    int count = COUNT; /* The number of mines */
    while (count) /* Break the loop when mines are set up */
    {
        int x = rand() % 10 + 1; /* Generates a random number from 1 to 
10,setting up a mine at a horizontal coordinate from 1 to 10 */
        int y = rand() % 10 + 1;
        if (real_mine[x][y] == '0') /* Find a place where is no mine to set up 
a mine. */
        {
            real_mine[x][y] = '1';
            count--;
        }
    }
}
/*
* Function name: count_mine
* Description: Detect the number of mines in the surrounding 8 areas of the 
coordinate player input.
*/
int count_mine(int x, int y)
{
    int count = 0;
    if (real_mine[x - 1][y - 1] == '1')
        count++;
    if (real_mine[x - 1][y] == '1')
        count++;
    if (real_mine[x - 1][y + 1] == '1')
        count++;
    if (real_mine[x][y - 1] == '1')
        count++;
    if (real_mine[x][y + 1] == '1')
        count++;
    if (real_mine[x + 1][y - 1] == '1')
        count++;
    if (real_mine[x + 1][y] == '1')
        count++;
    if (real_mine[x + 1][y + 1] == '1')
        count++;
    return count;
}
/*
* Function name: safe_mine
* Description: Prevent player from being blown up for the first time he/she
input.
*/
void safe_mine(void)
{
    int x = 0;
    int y = 0;
    char ch = 0;
    int count = 0;
    int ret = 1;
    printf("Enter coordinate:x(column) y(row) to sweep mine.\n");
    printf("Beware: There is a SPACE between x and y that you input!\n");
    while (1)
    {
        scanf_s("%d %d", &y, &x);
        if ((x >= 1 && x <= 10) && (y >= 1 && y <= 10))/* Determine if the
input coordinate is form 1 to 10, enter again if the input is wrong */
        {
            if (real_mine[x][y] == '1') /* Remedy if the first input is on a
mine. */
            {
                real_mine[x][y] = '0';
                char ch = count_mine(x, y);
                show_mine[x][y] = ch + '0'; /* The difference between the ASCII
value of a digit and the ASCII value of a numeric character is 48, which is the
ASCII value of '0' */
                open_mine(x, y);
                while (ret) /* Set up a mine in the remaining free space */
                {
                    int x = rand() % 10 + 1; /* Generates a random number from
1 to 10,setting up a mine at a horizontal coordinate from 1 to 10 */
                    int y = rand() % 10 + 1;
                    if (real_mine[x][y] == '0') /* Find a place where is no
mine to set up a mine. */
                    {
                        real_mine[x][y] = '1';
                        ret--;
                        break;
                    }
                }
                break; /* Break the function */
            }
            if (real_mine[x][y] == '0')
            {
                char ch = count_mine(x, y);
                show_mine[x][y] = ch + '0'; /* The difference between the ASCII
value of a digit and the ASCII value of a numeric character is 48, which is the
ASCII value of '0' */
                open_mine(x, y);
                break;
            }
        }
        else /* The coordinate is wrong */
        {
            printf("Your input is wrong!Please RETYPE!\n");
        }
    }
}
/*
* Function name: sweep_mine
* Description: Sweep the mine. Return 1 if player's input is on a mine, 0 if
player's input isn't on a mine.
*/
int sweep_mine(void)
{
    int x = 0;
    int y = 0;
    int count = 0;
    printf("Enter coordinate:x(column) y(row) to sweep mine.\n");
    printf("Beware: There is a SPACE between x and y that you input!\n");
    scanf_s("%d %d", &y, &x);
    if ((x >= 1 && x <= 10) && (y >= 1 && y <= 10))/* Determine if the input
coordinate is form 1 to 10, enter again if the input is wrong */
    {
        if (real_mine[x][y] == '0') /* Player's input is not on a mine */
        {
            char ch = count_mine(x, y);
            show_mine[x][y] = ch + '0'; /* The difference between the ASCII
value of a digit and the ASCII value of a numeric character is 48, which is the
ASCII value of '0' */
            open_mine(x, y);
            if (count_show_mine() == COUNT) /* Determine the number of unknown
areas left. If the number is the number of mines, then the player win. */
            {
```

```c
                print_mine();
                printf("You WIN! \n\n");
                return 0;
            }
        }
        else if (real_mine[x][y] == '1') /* Met the mine */
        {
            return 1;
        }
    }
    else
    {
        printf("Your input is wrong!Please RETYPE!\n");
    }
    return 0; /* Player's input is not on a mine */
}
/*
* Function name: open_mine
* Description: Expand coordinate's surroundings.
*/
void open_mine(int x, int y)
{
    if (real_mine[x - 1][y - 1] == '0')
    {
        show_mine[x - 1][y - 1] = count_mine(x - 1, y - 1) + '0';/* Displays
the number of mines around this coordinate */
    }
    if (real_mine[x - 1][y] == '0')
    {
        show_mine[x - 1][y] = count_mine(x - 1, y) + '0';
    }
    if (real_mine[x - 1][y + 1] == '0')
    {
        show_mine[x - 1][y + 1] = count_mine(x - 1, y + 1) + '0';
    }
    if (real_mine[x][y - 1] == '0')
    {
        show_mine[x][y - 1] = count_mine(x, y - 1) + '0';
    }
    if (real_mine[x][y + 1] == '0')
    {
        show_mine[x][y + 1] = count_mine(x, y + 1) + '0';
    }
    if (real_mine[x + 1][y - 1] == '0')
    {
        show_mine[x + 1][y - 1] = count_mine(x + 1, y - 1) + '0';
    }
    if (real_mine[x + 1][y] == '0')
    {
        show_mine[x + 1][y] = count_mine(x + 1, y) + '0';
    }
    if (real_mine[x + 1][y + 1] == '0')
    {
        show_mine[x + 1][y + 1] = count_mine(x + 1, y + 1) + '0';
    }
}
/*
* Function name: count_show_mine function
* Description: Count the number of '*' left on the checkerboard contains
show_mine.
*/
int count_show_mine()
{
    int count = 0;
    int i = 0;
    int j = 0;
    for (i = 1; i <= ROWS - 2; i++)
    {
        for (j = 1; j <= COLS - 2; j++)
        {
            if (show_mine[i][j] == '*')
            {
                count++;
            }
        }
    }
    return count;
}
```

2> main.c

```
/*
* File name: main.c
* Author: Fengxi Zhang & Chengchu Wu
* HW ID: H00365040 & H00365059
* IDE: Visual Studio 2017 Community
* Description: Define main function and game function.
* Date: 15/1/2021
*/
/* Include file game.h */
```

```c
#include "game.h"
/*
* Function name: game function
* Description:  Include other functions to execute the game program.
*/
void game(void)
{
    double  start, finish;
    int ret = 0;
    init_mine();/* Initialize two char arrays named show_mine and real_mine */
    set_mine(); /* Set up mines for real_time array */
    /* Print checkerboard contains real_time array for designer(CAN be 
delete.Just for testing the program)
    print_mine();
    */
    printf("\n");
    print_player(); /* Print checkerboard of show_mine array for player */
    start = clock(); /* Get start time */
    safe_mine(); /* Prevent player from being blown up for the first time */
    if (count_show_mine() == COUNT) /* The situation that one step to win */
    {
        print_mine();
        printf("YOU WIN! \n\n");
        return;
    }
    print_player();    /* Print checkerboard of show_mine array for player */
    while (1)  /* Loop sweeping */
    {
        int ret = sweep_mine();/* Sweep the mine. Return 1 if player's input is 
on a mine;return 0 if player's is not on a mine. */
        if (count_show_mine() == COUNT)/* If the number of '*' on the player's 
checkerboard is the number of mines.YOU WIN! */
        {
            print_mine();  /*Print checkerboard contains real_mine array for 
player to view the distribution of mines */
            printf("YOU WIN! \n\n");
            finish = clock(); /* Get finish time */
            printf("You have spent %d seconds\n", (int)(finish - start) / 
CLOCKS_PER_SEC);
            break;
        }
        if (ret) /* Judge if player's input is on a mine */
        {
            printf("GAME OVER! YOU WERE BLOWN UP!\n");
            finish = clock(); /* Get finish time */
            printf("You have spent %d seconds\n", (int)(finish - start) / 
CLOCKS_PER_SEC);
            print_mine(); /*Print checkerboard contains real_mine array for 
player to view the distribution of mines */
            break;
        }
        print_player(); /* Print checkerboard contains show_mine array for 
player */
    }
}
/*
* Function name: main function
* Description:  Input 1 to begin the game. Input 0 to quit the game.
*              Do not allow to input others.
*/
int main(void)
{
    srand((unsigned int)time(NULL));  /* Generate random number seeds */
    int input = 0;
    menu(); /* Print the beginning menu */
    do
    {
        scanf_s("%d", &input); /* assign player's input to the variable "input" 
*/
        switch (input)
        {
        case 1:
            game(); /* execute the game function to begin the game */
            break;
        case 0:
            exit(0); /* quit the game */
            break;
        default:
            printf("Your input is ILLEGAL,please RETYPE!\n");
            break;
        }
        menu();
        printf("contiue?\n");
    }
    while (input); /* Create a loop through which player can replay the game 
after a game over */
    return 0;
}
```

References

[1] CSDN: https://www.csdn.net/

[2] Brian W. Kernighan & Dennis M. Ritchie. *The C Programming Language (2nd Edition)*. Prentice Hall.