```
Script started on 2024-04-11 16:10:35-05:00 [TERM="xterm-256color" TTY="/dev/pts/0" COLUMNS="290" LINES="65"]
[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ pwd
[?2004l
/home/jovyan/CS2/Projects/Project_6
[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ ls -la
[?2004l
total 64
drwxr-sr-x 3 jovyan users  4096 Apr 11 16:10 [0m[01;34m.[0m
drwxr-sr-x 9 jovyan users  4096 Apr  3 16:06 [01;34m..[0m
-rw-r--r-- 1 jovyan users  9682 Apr 11 16:10 Board.cpp
-rw-r--r-- 1 jovyan users   880 Apr 10 15:07 Board.h
drwxr-sr-x 2 jovyan users  4096 Apr  3 16:06 [01;34m.ipynb_checkpoints[0m
-rw-r--r-- 1 jovyan users   610 Apr 10 15:08 playable.cpp
-rw-r--r-- 1 jovyan users     0 Apr 11 16:10 Sabin_Project_6.log
-rwxr-xr-x 1 jovyan users 22280 Apr 11 16:06 [01;32msweep[0m
-rw-r--r-- 1 jovyan users   233 Apr 10 14:42 Tile.cpp
-rw-r--r-- 1 jovyan users   345 Apr 10 14:40 Tile.h
[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ cat -n Board.h
[?2004l
     1  #ifndef BOARD_H
     2  #define BOARD_H
     3  #include "Tile.h"
     4
     5  class Board{
     6  private:
     7      int m_width{};
     8      int m_height{};
     9      int m_size{};
    10      int mine_count{};
    11      //Row
    12      int m_x{};
    13      //Col
    14      int m_y{};
    15      Tile *tile{nullptr};
    16      Tile *board{nullptr};
    17
    18  public:
    19      //default constructor
    20      Board(int, int, int);
    21      //function called place_mines; will randomly place mines on a board
    22      void place_mines();
    23      // //function called update_counts; will show the amount of mines near the tile
    24      void update_counts();
    25
    26      //Constructor for play board
    27      void construct_playBoard();
    28      //View the Play board
    29      void view_playBoard();
    30
    31      void get_game_input();
    32      void PlayGame();
    33
    34      // //function called revealed_board to display the board in 2-d fasshion
    35      void revealed_board();
    36      //default destructor
    37      ~Board(){
    38          delete[] tile;
    39      }
    40  };
    41
    42
    43  #endif[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ cat -n Board.cpp
[?2004l
     1  #include <iostream>
     2  #include "Board.h"
     3  #include <cstdlib>
     4  #include <ctime>
     5  #include <iomanip>
     6  #include "Tile.h"
     7
     8
     9  //Loop through the array, and set the index to 0
    10  //Initialize mine_count to 10
    11  Board::Board(int width, int height, int mine){
    12      m_width = width;
    13      m_height = height;
    14      m_size = m_width * m_height;
    15      tile = new Tile[m_size];
    16
```

```
    for(int i{0}; i < m_size; i++){
        tile[i].update_tile('0');
    }

    mine_count = mine;
}

void Board::place_mines(){
    //Seed the random number generator
    std::srand(static_cast<unsigned int>(std::time(nullptr)));

    for(int i{0}; i < mine_count; i++){
        //Produce a random index
        int index = std::rand() % m_size;

        //Check to see if the index is already a mine
        if(tile[index].get_tile() != 'M'){
            //If a mine is not present, place the mine
            tile[index].update_tile('M');
        } else{
            //If no mine is placed, subtract one
            //As we did not place a mine
            i--;
        }
    }

}

//Loop through the array, print out the index
void Board::revealed_board(){
    std::cout << '\n';
    for(int i{0}; i < m_width; i++){
        std::cout << "|---";
    }
    std::cout << "|\n";
    for(int i{0}; i < m_size; i++){
        if(i % m_width == 0 && i != 0){
            std::cout << '\n';
            for(int i{0}; i < m_width; i++){
                std::cout << "|---";
            }
            std::cout << "|\n";
        }
        std::cout << '|' << std::setw(2) << tile[i].get_tile() << ' ' ;
        if((i+1) % m_width == 0){
            std::cout << '|';
        }
    }
    std::cout << '\n';
    for(int i{0}; i < m_width; i++){
        std::cout << "|---";
    }
    std::cout << "|\n";
}

void Board::update_counts(){
    int row{-1};
    int col{0};
    int count{0};
    for(int index{0}; index < m_size; index++){
        if(index % m_width == 0){
            row++;
        }
        //index = width * row + column
        if(tile[index].get_tile() != 'M'){
        //if row is 0
            if(row == 0){
                //if col is 0
                if(col == 0){
                        //check index + m_width, index + m_width + 1, index + 1
                    if(tile[index + m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                    }
                //else if col is (m_width - 1)
                else if(col == (m_width -1)){
                    //check index + m_width, index + m_width - 1, index -1
                    if(tile[index + m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
```

```
                //else
                else{
                    //check index +/- 1, index + m_width, index + m_width +/- 1
                    if(tile[index + m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
            }
            //else if row is (m_height - 1)
            else if(row == (m_height - 1)){
                //if col is 0
                if(col == 0){
                    //check index - m_width, index - m_width + 1, index + 1
                    if(tile[index - m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
                //else if col is (m_width - 1)
                else if(col == (m_width - 1)){
                    //check index - m_width, index - m_width - 1, index -1
                    if(tile[index - m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
                //else
                else{
                    //check index +/- 1, index - m_width, index - m_width +/- 1
                    if(tile[index - m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
            }
            //else
            else{
                //if col is 0
                if(col == 0){
                    //check index +/- m_width, index +/- m_width + 1, index + 1
                    if(tile[index + m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
                //else if col is (m_width - 1)
                else if(col == (m_width - 1)){
                    //check index +/- m_width, index +/- m_width - 1, index -1
                    if(tile[index + m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
                //else
                else{
                    //check index +/- m_width, index +/- m_width +/- 1, index +/- 1
                    if(tile[index + m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + m_width + 1].get_tile() == 'M'){count += 1;}
                    if(tile[index - 1].get_tile() == 'M'){count += 1;}
                    if(tile[index + 1].get_tile() == 'M'){count += 1;}
                    tile[index].update_tile(static_cast<char>(count + '0'));
                }
            }
        }
        if(col == (m_width - 1)){
            col = 0;
            count = 0;
        }else{
            col++;
            count = 0;
```

```
   183         }
   184     }
   185 }
   186 
   187 void Board::construct_playBoard(){
   188 
   189     board = new Tile[m_size];
   190 
   191     for(int i{0}; i < m_size; i++){
   192         board[i].update_piece('#');
   193     }
   194 }
   195 
   196 void Board::view_playBoard(){
   197     std::cout << '\n';
   198     for(int i{0}; i < m_width; i++){
   199         std::cout << "|---";
   200     }
   201     std::cout << "|\n";
   202     for(int i{0}; i < m_size; i++){
   203         if(i % m_width == 0 && i != 0){
   204             std::cout << '\n';
   205             for(int i{0}; i < m_width; i++){
   206                 std::cout << "|---";
   207             }
   208             std::cout << "|\n";
   209         }
   210         std::cout << '|' << std::setw(2) << board[i].get_piece() << ' ' ;
   211         if((i+1) % m_width == 0){
   212             std::cout << '|';
   213         }
   214     }
   215     std::cout << '\n';
   216     for(int i{0}; i < m_width; i++){
   217         std::cout << "|---";
   218     }
   219     std::cout << "|\n";
   220 }
   221 
   222 void Board::get_game_input(){
   223     int tempX{};
   224     int tempY{};
   225     std::cout << "Enter a row and column to dig: ";
   226     std::cin >> tempX >> tempY;
   227     m_x = tempX;
   228     m_y = tempY;
   229 }
   230 
   231 void Board::PlayGame(){
   232     view_playBoard();
   233     get_game_input();
   234     //Convert to an index
   235     int index = (m_width * m_x) + m_y;
   236     //Intialize the count to 1 since the user already made a move
   237     int count{1};
   238     //Check to see if it is a mine & the count is less than the size - mine count
   239     while(tile[index].get_tile() != 'M' && count < (m_size - mine_count)){
   240         board[index].update_piece(tile[index].get_tile());
   241         count++;
   242         view_playBoard();
   243         get_game_input();
   244         index = (m_width * m_x) + m_y;
   245     }
   246     //Update the last tile
   247     board[index].update_piece(tile[index].get_tile());
   248     std::cout << '\n';
   249     //Check to see if won or lost
   250     if(count == (m_size - mine_count)){
   251         std::cout << "YOU'VE WON!!!" << '\n';
   252         std::cout << "Final Board:" << '\n';
   253         view_playBoard();
   254         std::cout << '\n';
   255     }else{
   256         std::cout << "YOU'VE LOST!!!" << '\n';
   257         std::cout << "Final Board:" << '\n';
   258         revealed_board();
   259         std::cout << '\n';
   260     }
   261 }[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ cat -n Tile.h
[?2004l
     1  #ifndef TILE_H
     2  #define TILE_H
```

```
     3
     4  class Tile{
     5  private:
     6      //Tile data member
     7      char m_Tile{};
     8      //Play Board data member
     9      char m_PlayBoard{};
    10
    11  public:
    12      //Update the tile data member
    13      void update_tile(char value);
    14      //Get the value of the tile
    15      char get_tile();
    16
    17      void update_piece(char value);
    18      char get_piece();
    19  };
    20
    21  #endif[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ cat -n Tile.cpp
[?2004l
     1  #include "Tile.h"
     2  void Tile::update_tile(char value){
     3      m_Tile = value;
     4  }
     5  char Tile::get_tile(){
     6      return m_Tile;
     7  }
     8  void Tile::update_piece(char value){
     9      m_PlayBoard = value;
    10  }
    11  char Tile::get_piece(){
    12      return m_PlayBoard;
    13  }[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ cat -n Pl[K[Kplayable.cpp
[?2004l
     1  #include <iostream>
     2  #include "Board.h"
     3  #include <cstdlib>
     4  #include <ctime>
     5  #include <iomanip>
     6  #include "Tile.h"
     7
     8  int main(){
     9
    10      int width{};
    11      int height{};
    12      int mines{};
    13
    14      std::cout << "Enter width of the board: ";
    15      std::cin >> width;
    16      std::cout << "Enter the height of the board: ";
    17      std::cin >> height;
    18      std::cout << "Enter the amount of mines: ";
    19      std::cin >> mines;
    20      std::cout << '\n';
    21
    22      Board sampleBoard(width,height,mines);
    23      sampleBoard.place_mines();
    24      sampleBoard.update_counts();
    25      sampleBoard.construct_playBoard();
    26      sampleBoard.PlayGame();
    27
    28      return 0;
    29  }[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ g++ -Wall -Wextra -Werror playable.cpp Board.cpp Tile.cpp -o sweep
[?2004l
[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ ./sweep
[?2004l
Enter width of the board: 4
Enter the height of the board: 4
Enter the amount of mines: 2


|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
```

```
|---|---|---|---|
Enter a row and column to dig: 0 0

|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 0 1

|---|---|---|---|
| 0 | 0 | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 0 2

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 1 0

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 1 1

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 1 2

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 2 0

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 3 0

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
```

```
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 2 1

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | # | # |
|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 3 1

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | # | # |
|---|---|---|---|
| 0 | 0 | # | # |
|---|---|---|---|
Enter a row and column to dig: 2 2

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | # | # |
|---|---|---|---|
Enter a row and column to dig: 3 2

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
Enter a row and column to dig: 3 3

|---|---|---|---|
| 0 | 0 | 1 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
|---|---|---|---|
Enter a row and column to dig: 0 3

YOU'VE WON!!!
Final Board:

|---|---|---|---|
| 0 | 0 | 1 | 1 |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 2 | # |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
|---|---|---|---|

[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ ./sweep
[?2004l
Enter width of the board: 4
Enter the height of the board: 4
Enter the amount of mines: 2
```

```
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 0 0

|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig:  1 0

|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| 1 | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 1 1

|---|---|---|---|
| 0 | # | # | # |
|---|---|---|---|
| 1 | 1 | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
| # | # | # | # |
|---|---|---|---|
Enter a row and column to dig: 2 0

YOU'VE LOST!!!
Final Board:

|---|---|---|---|
| 0 | 0 | 1 | 1 |
|---|---|---|---|
| 1 | 1 | 1 | M |
|---|---|---|---|
| M | 1 | 1 | 1 |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
|---|---|---|---|

[?2004h(base) ]0;jovyan@jupyter-tes4j: ~/CS2/Projects/Project_6[01;32mjovyan@jupyter-
tes4j[00m:[01;34m~/CS2/Projects/Project_6[00m$ exit
[?2004l
exit

Script done on 2024-04-11 16:14:04-05:00 [COMMAND_EXIT_CODE="0"]
```