Final Minesweeper

**Christopher Ramirez-Alba**

**CIS 5**

**Summer 2017**

**45549**

**TABLE OF CONTENTS**

**Introduction 3**

**How the Game Works 3**

**Object of the Game 3**

**Rules of the Game 3**

**Converting the Game 3**

**Rendering Game through Code 3**

**Minesweeper == Code** (how the Game is similar) **4**

**Minesweeper != Code** (how the Game differs) **4**

**Where the Code Flows 4**

**Flowchart 4**

**Pseudo Code 4**

**Checklist 5**

**Proof of Working Program 6**

**References 7**

**Program 7**

**Introduction**

Aside from solitaire and pinball, Minesweeper was one of the games that came with your desktop back in the days and even now.

Minesweeper being one of the more mind-boggling game out of the three was always challenging. Trying to avoid all the mines while uncovering all the empty spaces in the game had everyone upset at least once. It’s about time this game made a comeback to annoy you one more time!

**How the Game Works**

**Object of the Game**

Clear the field without setting off any bombs.

**Game Rules**

This a 1 player game.

* Choose any cell in the field to uncover what is hidden under it.
* If it is a Bomb “Game Over”
* If it is a number pay close attention.
  + 1 number means there is one bomb touching that uncovered cell.
  + 2 means there are 2 bombs touching the uncovered cell.
  + 3 means 3 bombs and 4 means avoid all sides!
* Continue uncovering by process of elimination until all non-explosive cells are uncovered then it’s “Game Won.”

**Converting the Game**

**Rendering Game through Code**

Creating a code for this game was more challenging than expected, plenty of days were spent researching, studying and questioning. Major setbacks were:

* Finding a way to end the program?
  + First few attempts resulted in an infinite code.
* Researching how to uncover the number of “explosives” near the cell.

When near the point of giving up and shortening the code, I ran into a few accidental attempts to solving the problems. It wasn’t until one of my managers at work who works with code hours to no end mentioned looking into standard namespaces. Unfortunately I had not recalled learning that in class but was easy to research online.

**Minesweeper == Code Game** (how the Game is similar)

Coding a minesweeper game was nothing as the game I remember, but it managed to have similarities when printed.

* Amount of Bombs in game is displayed.
* “Explosives” near uncovered cells.
* Score at the end

**Minesweeper != Code Game** (how the Game differs)

The differences from the game I remember.

1. When printed it did not have the same feel as the actual game.
2. Unable to mark where possible mines are.
3. Cells had to be a variable instead of blank.

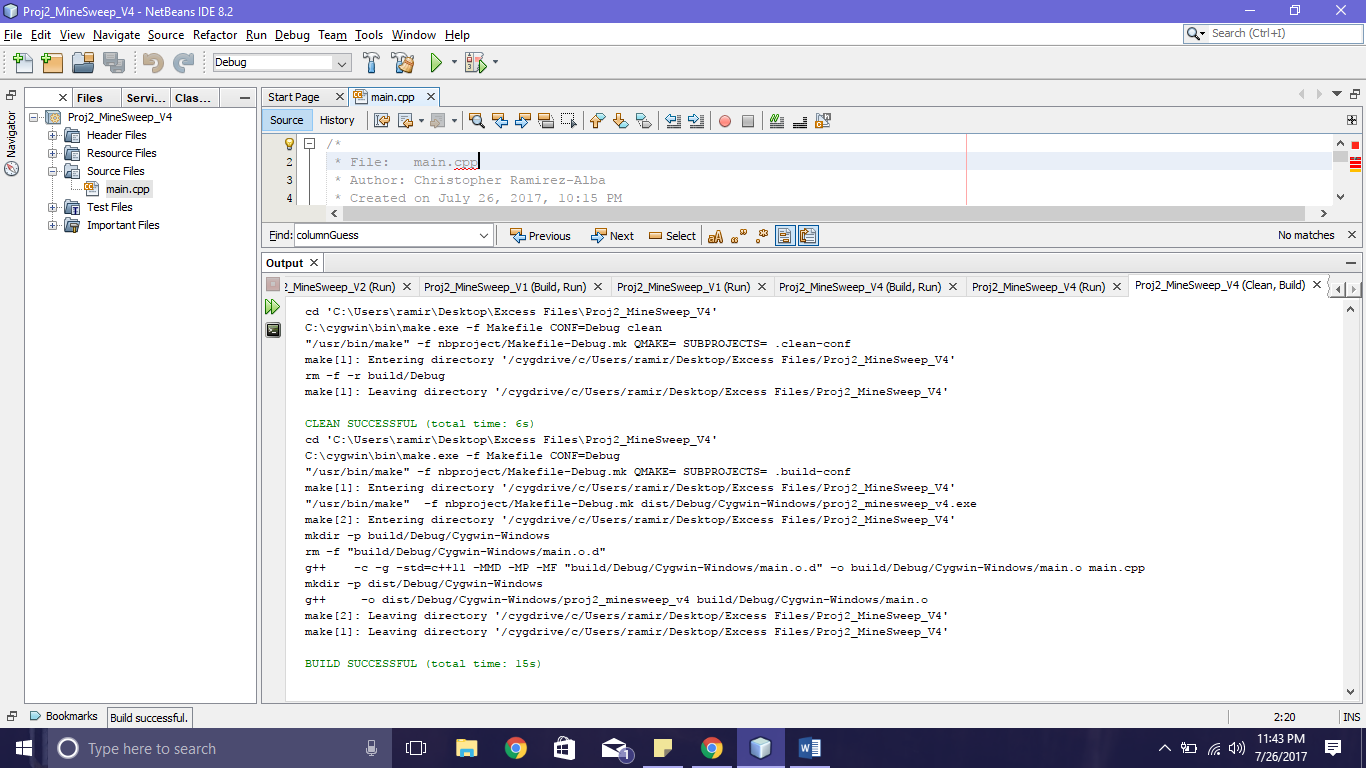
**Flowchart**

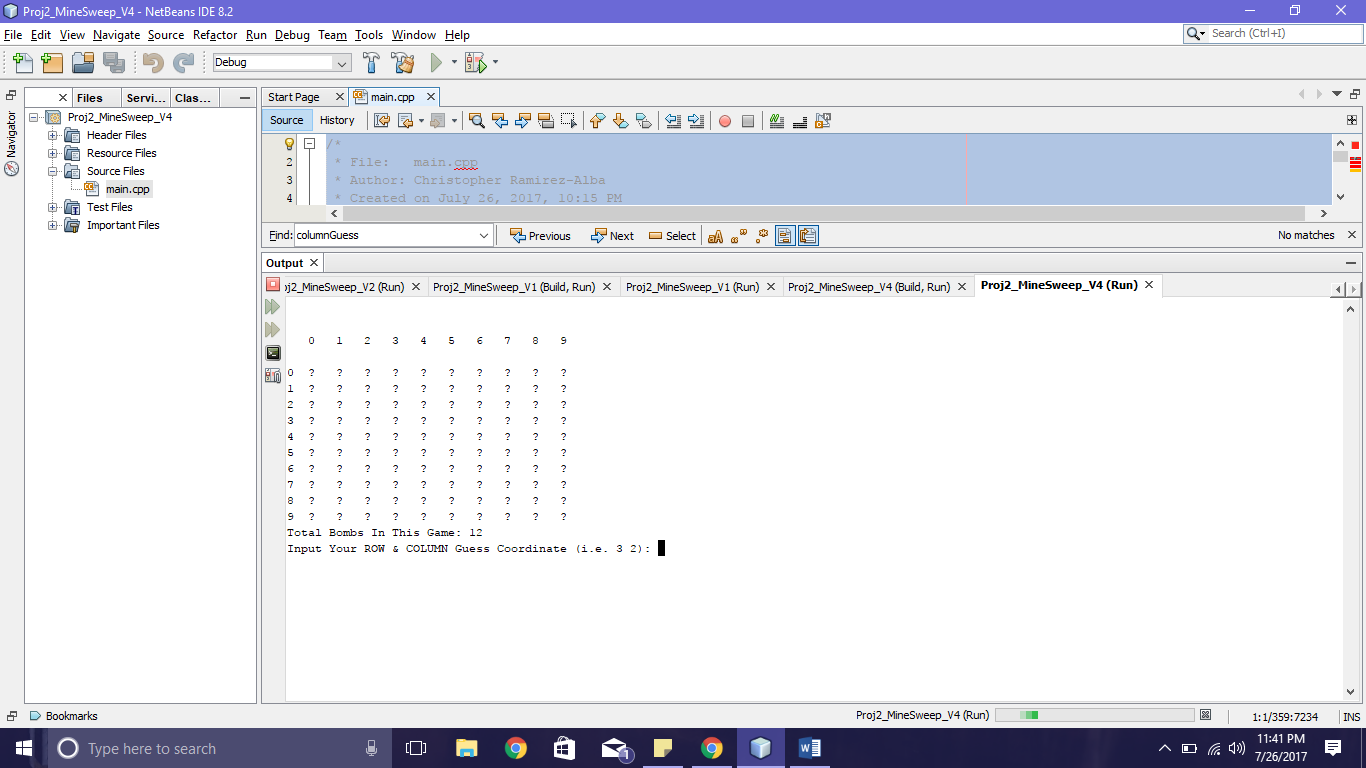
**Pseudo Code**

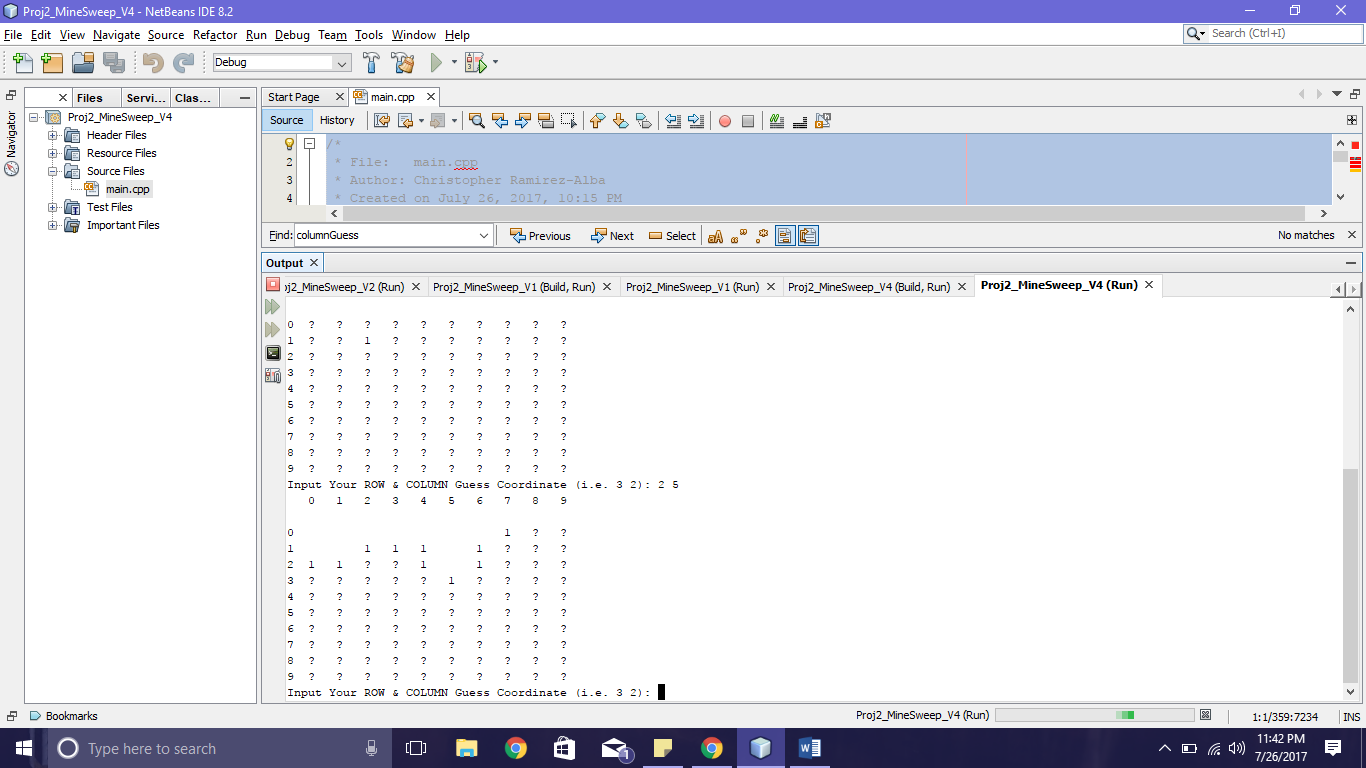
**Checklist**

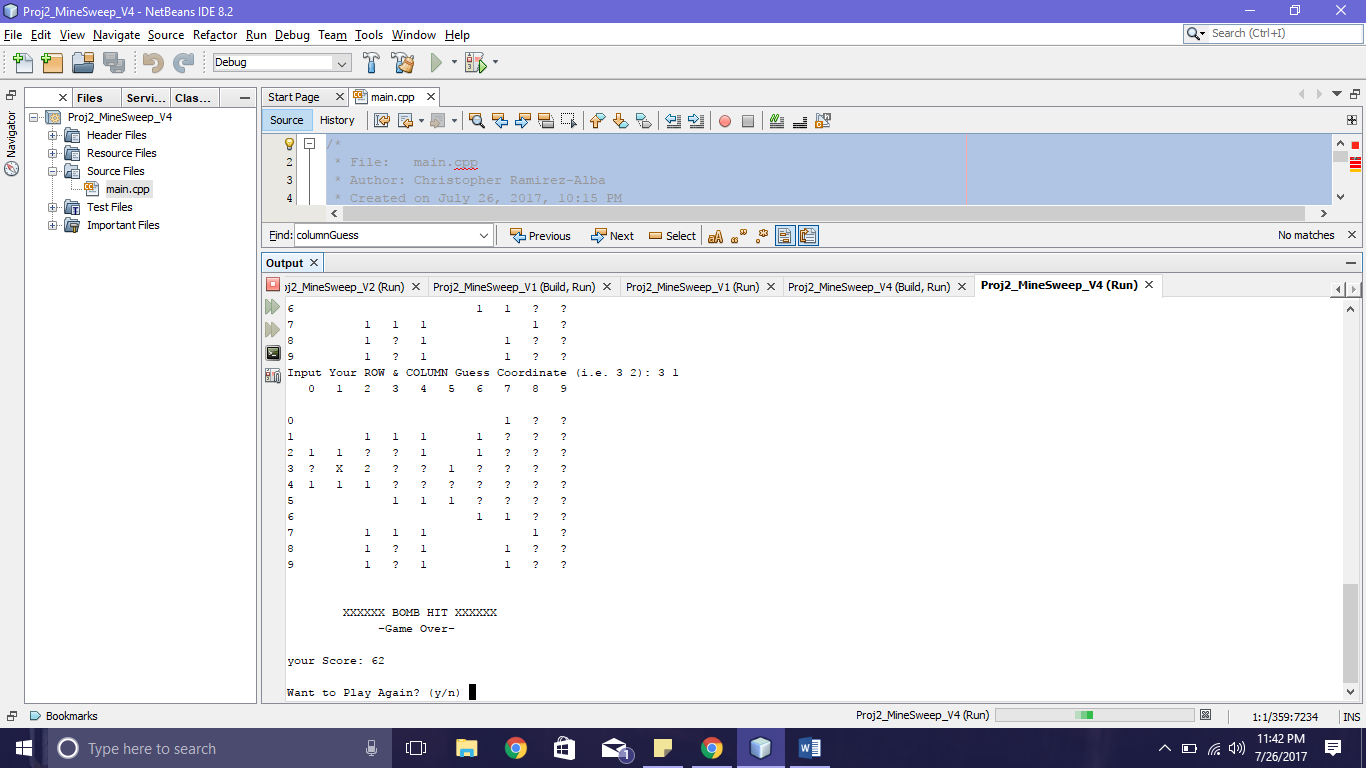
**Proof of a Working Program**

In the event, that my program does not work once it reaches Dr. Lehr, I have provided some screenshots that prove that the program did work at one time on the next few pages









**References**

1. Dr.Lehr’s Lectures
2. [www.cplusplus.com](http://www.cplusplus.com/) when researching standard namespaces
3. Gaddis 8th Edition C++ From Control Structures through Objects

**Program**

/\*

\* File: main.cpp

\* Author: Christopher Ramirez-Alba

\* Created on July 26, 2017, 10:15 PM

\*Purpose: Project 2 Minesweep

\*/

//System Librariez

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

#include <string>

// use all of the things

using std::rand;

using std::srand;

using std::time;

using std::cout;

using std::cin;

using std::endl;

using std::right;

using std::left;

using std::setw;

using std::string;

// BOARD.h

class BOARD

{

public:

BOARD(int rows, int column, float per);

BOARD(int rows, int column);

BOARD();

virtual ~BOARD();

static const int ROW\_AMOUNT, COLUMN\_AMOUNT;

static const float CHANCE\_DEFAULT;

static const unsigned char MINE\_MASK, UNCOVERED\_MASK;

static void random();

void rndBord();

bool Mine(int r, int c);

bool Uncvrd(int r, int c);

int USpace(int r, int c);

int AdjMcnt(int r, int c);

bool onBoard(int r, int c);

char GetSqur(int r, int c);

int GetRows();

int Getcolumn();

float Getprcnt();

int GetMinCnt();

int GetSfeCnt();

private:

int rowcount, colcount;

int minecount, safecount;

float percentage;

char \* board;

bool safedummy;

void Init(int rows, int column, float per);

};

// MineSwp.h

class MineSwp : public BOARD

{

public:

MineSwp();

MineSwp(int rows, int column, float difficulty);

MineSwp(int rows, int column, float difficulty, char mchar, char bchar, char uchar);

static const char MINE\_DEFAULT, BLANK\_DEFAULT, UNKNOWN\_DEFAULT;

void PrintBoard();

char GetCharForSpace(int r, int c);

void Play();

bool WonGame();

private:

char minechar, blankchar, unknownchar;

int cleared;

void Init(char mchar, char bchar, char uchar);

};

// BOARD.cpp

const int BOARD::ROW\_AMOUNT=10, BOARD::COLUMN\_AMOUNT=10;

const float BOARD::CHANCE\_DEFAULT=0.85;

const unsigned char BOARD::MINE\_MASK=0x1, BOARD::UNCOVERED\_MASK=0x2;

void BOARD::random()

{

srand(time(NULL));

}

int BOARD::GetRows()

{

return rowcount;

}

int BOARD::Getcolumn()

{

return colcount;

}

float BOARD::Getprcnt()

{

return percentage;

}

int BOARD::GetMinCnt()

{

return minecount;

}

int BOARD::GetSfeCnt()

{

return safecount;

}

BOARD::BOARD()

{

Init(ROW\_AMOUNT, COLUMN\_AMOUNT, CHANCE\_DEFAULT);

}

BOARD::BOARD(int rows, int column)

{

Init(rows, column, CHANCE\_DEFAULT);

}

BOARD::BOARD(int rows, int column, float per)

{

Init(rows, column, per);

}

BOARD::~BOARD()

{

delete[] board;

}

void BOARD::Init(int rows, int column, float per)

{

minecount=0;

safecount=rows \* column;

percentage=per;

rowcount=rows;

colcount=column;

board=new char [rows \* column];

rndBord();

}

char BOARD::GetSqur(int r, int c)

{

return board[r \* colcount + c];

}

void BOARD::rndBord()

{

for (int i=0, j=rowcount \* colcount; i != j; ++i)

{

float r=(((float) rand()) / ((float) RAND\_MAX));

board[i]=(percentage < r);

if (board[i]) ++minecount;

}

safecount -= minecount;

}

bool BOARD::onBoard(int r, int c)

{

return (

(r >= 0 && r < rowcount) &&

(c >= 0 && c < colcount)

);

}

bool BOARD::Mine(int r, int c)

{

return (

(onBoard(r, c)) &&

(GetSqur(r, c) & MINE\_MASK)

);

}

bool BOARD::Uncvrd(int r, int c)

{

return (

(onBoard(r, c)) &&

(GetSqur(r, c) & UNCOVERED\_MASK)

);

}

int BOARD::USpace(int r, int c)

{

int uncovered=0;

while (onBoard(r, c) && !Uncvrd(r, c))

{

board[r \* colcount + c] |= UNCOVERED\_MASK;

if (!(GetSqur(r, c) & MINE\_MASK)) ++uncovered;

else break;

if (AdjMcnt(r, c) == 0)

{

uncovered += USpace(r + 0, c + 1);

uncovered += USpace(r + 0, c - 1);

uncovered += USpace(r + 1, c + 0);

uncovered += USpace(r - 1, c + 0);

}

break;

}

return uncovered;

}

int BOARD::AdjMcnt(int r, int c)

{

return Mine(r + 0, c + 1) +Mine(r + 0, c - 1)+

Mine(r + 1, c + 0) +Mine(r - 1, c + 0)+

Mine(r + 1, c + 1) +Mine(r - 1, c - 1)+

Mine(r + 1, c - 1) +Mine(r - 1, c + 1);

}

// MineSwp.cpp

const char MineSwp::MINE\_DEFAULT='X', MineSwp::BLANK\_DEFAULT=' ', MineSwp::UNKNOWN\_DEFAULT='?';

MineSwp::MineSwp() : BOARD()

{

Init(MINE\_DEFAULT, BLANK\_DEFAULT, UNKNOWN\_DEFAULT);

}

MineSwp::MineSwp(int rows, int column, float difficulty) : BOARD(rows, column, difficulty)

{

Init(MINE\_DEFAULT, BLANK\_DEFAULT, UNKNOWN\_DEFAULT);

}

MineSwp::MineSwp(int rows, int column, float difficulty, char mchar, char bchar, char uchar) : BOARD(rows, column, difficulty)

{

Init(mchar, bchar, uchar);

}

void MineSwp::Init(char mchar, char bchar, char uchar)

{

minechar=mchar;

blankchar=bchar;

unknownchar=uchar;

}

void MineSwp::PrintBoard()

{

for (int i=0; i < Getcolumn(); ++i) cout<<setw(4)<<right<<i;

cout<<left<<endl<<endl;

for (int r=0; r < Getcolumn(); ++r)

{

cout<<setw(3)<<r;

for (int c=0; c < GetRows(); ++c)

{

cout<<setw(4)<<GetCharForSpace(r, c);

}

cout<<endl;

}

}

char MineSwp::GetCharForSpace(int r, int c)

{

if (Uncvrd(r, c))

{

if (Mine(r, c))

return minechar;

int count=AdjMcnt(r, c);

if (count == 0)

return blankchar;

else

return '0' + count;

}

else

return unknownchar;

}

void MineSwp::Play()

{

int score=0;

PrintBoard();

cout<<"Total Bombs In This Game: "<<GetMinCnt()<<endl;

while (true)

{

string dummy;

int inputrow=-1, inputcol=-1;

cout<<"Input Your ROW & COLUMN Guess Coordinate (i.e. 3 2): ";

cin>>inputrow>>inputcol;

if (!cin || Uncvrd(inputrow, inputcol) || !onBoard(inputrow, inputcol))

{

cout<<"Invalid Selection! ";

if (!cin)

{

cin.clear();

cin>>dummy;

}

continue;

}

int uncovered=USpace(inputrow, inputcol);

PrintBoard();

if (Mine(inputrow, inputcol))

{

cout<<endl<<endl<<" XXXXXX BOMB HIT XXXXXX"

<<endl<<" -Game Over-"<<endl<<endl;

break;

}

else

{

score += uncovered;

cleared += uncovered;

if (WonGame())

{

cout<<endl<<endl<<

" ------ ALL BOMBS CLEARED ------"<<

endl<<" You Win!"<<endl<<endl;

break;

}

}

}

cout<<"your Score: "<<score<<endl;

}

bool MineSwp::WonGame()

{

return (cleared == GetSfeCnt());

}

// main.cpp

int main(int argc, char \* argv[])

{

BOARD::random();

cout<<endl<<endl;

while(true)

{

char again='n';

MineSwp m;

m.Play();

cout<<endl<<"Want to Play Again? (y/n) ";

cin>>again;

if (again == 'y')

{

cout<<endl<<endl;

continue;

}

else break;

}

return 0;

}