MineSweeper

By: Byeongju Jung

Introduction:

Minesweeper is a classic game that everyone in my generation (90s) has played or at least heard of. Even though it was originally designed to assist those who were not familiar with the concept of mouse on a computer, to me and everyone else who grew up during the technological boom of 90s and early 2000s, it was a great, competitive game to play when the internet was down.

Rules of the Game:

Despite the fact that many people have heard of this game, not many knows how to play it. The rule is very simple: you click on one of the tiles on your screen to find a mine and “flag” it. When you have clicked near a mine, the game will tell you by displaying a number on the tile you have selected. The number indicates the number of mines next to the tile you have selected and the player must use his/her reasoning skill to find all of the mines and “flag” them.

Once the player finds all of the mines, the game ends and the player wins. If, however, the player clicks on a mine, the player will lose. “Flagging” a tile that is not a mine won’t make the player lose, but it will not let the player win either.

In my version, instead of clicking with a mouse, I let the players input the row and column and their move by using keyboard inputs.

Development:

When I was in CIS-5 class, I tried to create minesweeper using the skills and knowledge I had learn. Although I was able to finish the game, the project left few unsatisfactory and showed many inexperienced coding.

It has not been a long time since I created that project, but I like to believe that I grew somewhat better as a programmer. So, I decided to revisit the coding that I created and tried to apply new skills that I have learned from CIS-17A class and chapter 9~12 of Gaddis book.

My first approach was to read the coding that I wrote few months back. It was little difficult because I had forgotten reasons for writing certain coding as it is. My writing style for coding did not help either because it was all over the place and it was difficult to read. So, I sat down and read through the coding multiple times and tried to find any mistakes and inappropriate coding. Although it was little tiresome, it was fun experience because I realized that I was moving forward little by little as a programmer.

After understanding the code, I started to think of ways to apply the structure and the pointer to it. Obvious approach was to apply structure and 2D pointer to my function that filled and created the board, killing 2 birds with 1 stone. After that, I erased the name list I had created before (from what I remembered, it was last minute work and was pretty sloppy) and tried to create a scoreboard using coding from chapter 10 (c-string chapter) and chapter 12 (data input/output chapter).

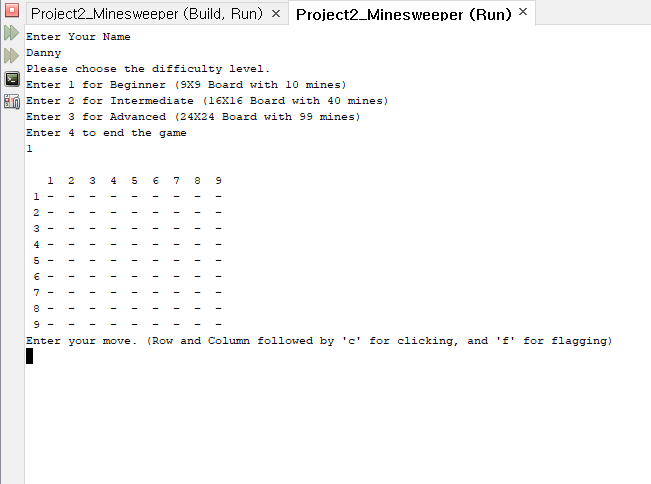
I managed to create a way to write player name, player’s initial, and score to text file.

One difficult thing was trying to fix a problem where choosing certain row made the program fail. I tried to ask the lab aids but even they could not find a way to solve it. After hours of thinking, it turned out that my function for checking the board boundaries had a problem and I managed to fix it.

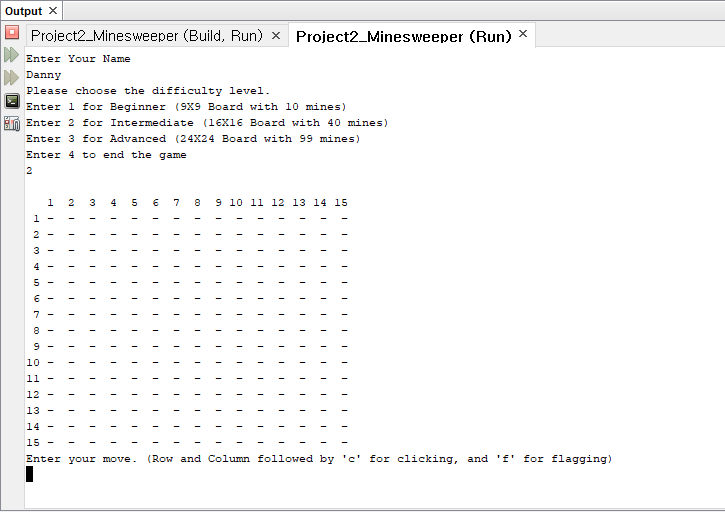
I like the way the game turned out, but I feel little bothered by the fact that I could not satisfy many of the project’s requirements.

Game Play

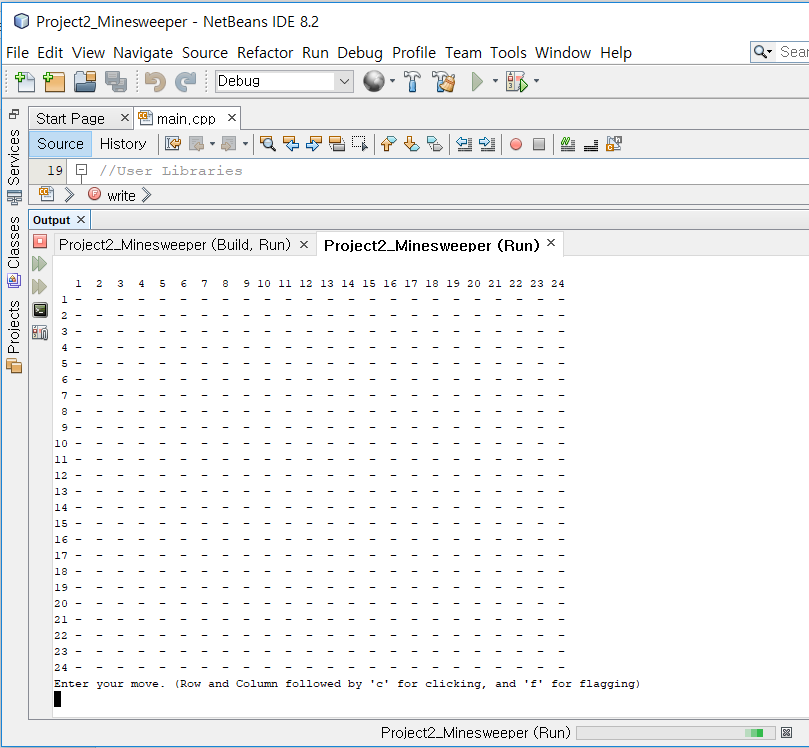
Beginner Level



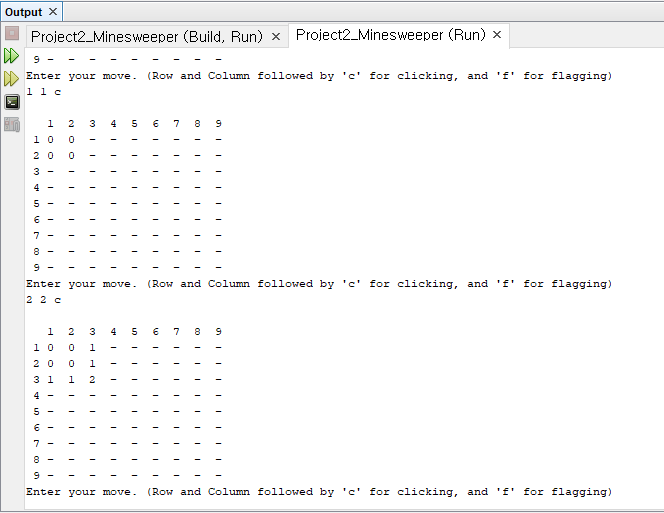
Intermediate Level

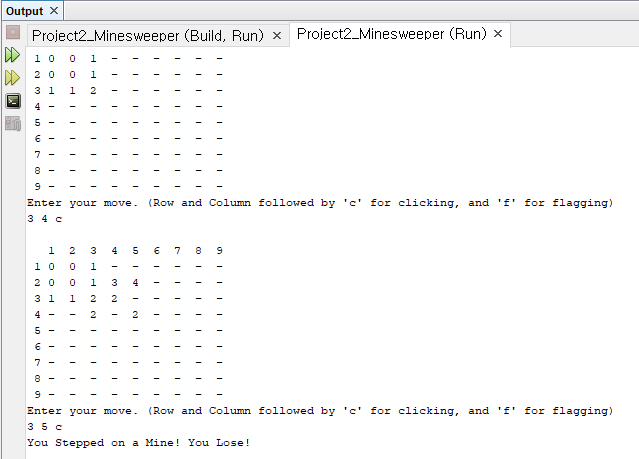


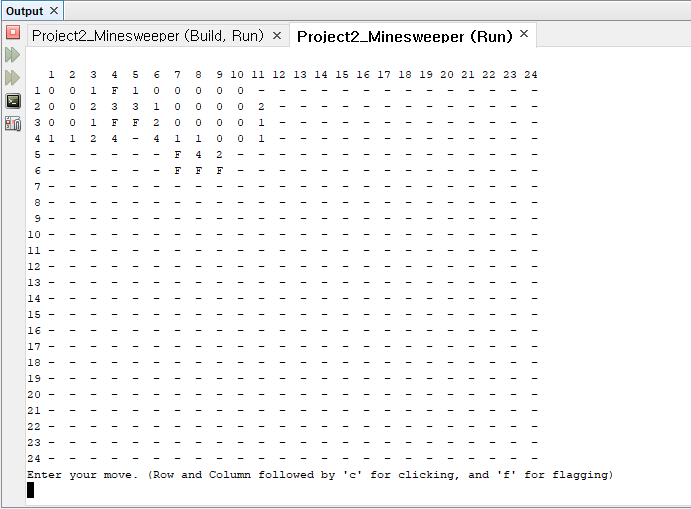
Advanced Level

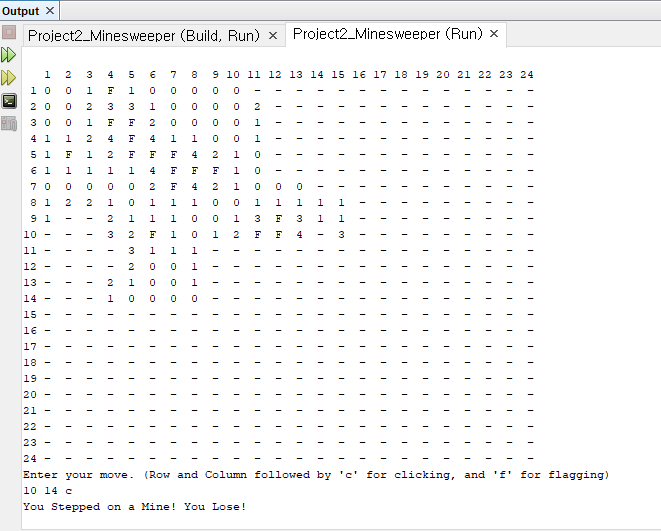


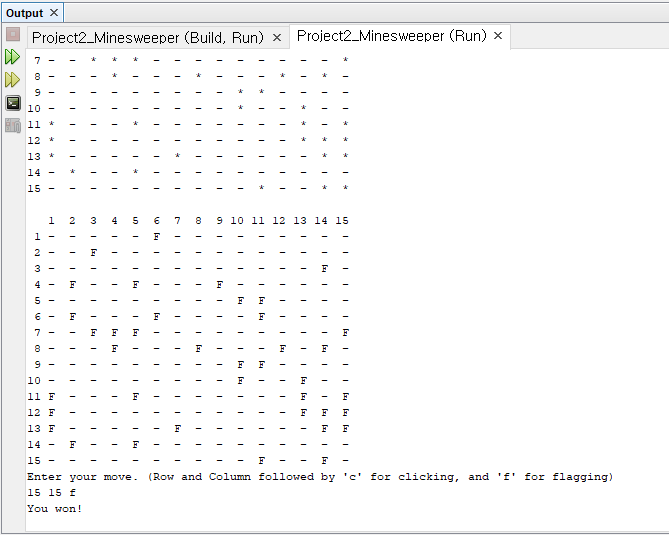
Game Play











(Cheat enabled, when all the mines are flagged)

Pseudo-code

/\*

\* File: main.cpp

\* Author: Byeongju Jung

\* Created on April 11, 2018, 12:13 PM

\* Purpose: Minesweeper Version 3

\*/

//System Libraries

//I/O Library

//Random number Library

//Time Library

//Formatting Library

//read/write Library

//character testing Library

//c-string Library

#include "Array.h"

using namespace std;

//User Libraries

//Global Constants - Math/Physics Constants, conversions,

// 2-D Array Dimensions

//Function Prototypes

//function for choosing level of the game

//function for filling the board up

//function for creating the board

//Function that will check if the given tile is valid or not

//Function that will replace a mine to a normal tile when it is the user's

//first move

//Function that will check if the tile that the user choose has a mine

//Function that will check the surrounding tile and give the number of mines

//Function that will "pop" the tile surrounding the chosen tile until it

//reaches a point where the tile is next to the mine

//Function that will check if all mines are flagged

//function for playing the game

//Function to write player info

//Function to change c-string info

//function for deallocating memory

//Execution Begins Here

//Seed the random number function

//Declare Variables//Choose level of the game

//Fill the array for the board

//Process/Map inputs to outputs

//Deallocate memory

//Exit stage right!

//Option to show the player list

//Option to end the game

// If the file was open successfully, continue

//Read from the file

//Close the file

//Is the size parameter valid

//Allocate memory

//testing fix if prob array->

//Filling Board

//testing fix if prob array->

//testing fix if prob array->

//Add mines.

//To check if the tile already has a mine assigned

//'-' for tiles not mine

//'\*' for tiles that are mine

//For displaying numbered columns

//For displaying numbered row

//Check to see if the given tile is inside the

//game boundaries

//Created problems when not checked

//Replacing mine

//if stepped on first move

//Top-Left from tile chosen

//Top from tile chosen

//Top-Right from tile chosen

//Left from tile chosen

//Right from tile chosen

//Down-Left from tile chosen

//Down from tile chosen

//Down-Right from tile chose

//checking and popping the 8 neighboring tiles

//Top-Left from tile chosen

//Top from tile chosen

//Top-Right from tile chosen

//Left from tile chosen

//Right from tile chosen

//Down-Left from tile chosen

//Down from tile chosen

//Down-Right from tile chosen

//Check to see if all mines are flagged

//Number of moves taken set to 0

//score

//Main Game-play Loop

//Display the board

//To see the mines, uncomment it

//Get input from user.

//Input row and columns for the user

//Input user's move

//Checking/Counting the neighboring tiles

//Input Validation

//Check if the tile that the user clicked is mine or not.

//To ensure that the first tile chosen is not a mine

//Checking if the chosen tile is mine or not

//Player lose when stepped on mine

//Checking neighbor tiles for mines

//Displaying the number of mines near

//Popping tiles that does not have mine next to it

//When the move is flag, display "F" on the tile

//Checking for a mine

//when all of the mines are flagged

//The player will win

//Ending the loop for the game play

//get name from terminal

//store in file

//Close the file