C++ 과제 #3

구현한 리스트

1. 맵 동적 할당
2. 지뢰 랜덤 배치 및 중복 배제
3. Class Cell의 9개 함수와 윈도우API기반 함수3개  
   사용자 정의 함수 1개 이용 구현
4. 게임 승리와 패배 판단
5. 기타 추가 기능 리스트
   1. 윈도우 API이용
      1. 입력 UI : 방향키 이용 입력
      2. 출력 UI : 커서 안보이게 현재 셀 위치 색 다르게 표기
   2. 도움말 목록
   3. 주변 폭탄 개수
   4. 주변폭탄 개수가 0개일경우 자동 펼침
   5. ‘G’체크 기능: 윈도우 지뢰 찾기에서의 깃발

201810754 조윤직

제출일: 2019. 10. 18

지뢰찾기 : 지뢰찾기게임 프로그램 구현

**문제정의 :** 사용자로부터 맵의 크기와 폭탄의 개수를 입력 받아 폭탄의 위치가 중복 되지 않는 해당하는 맵을 생성하고 사용자에게 맵을 보여주고 사용자에게 지속적으로 위치 정보를 입력 받아 결국 지뢰를 제외한 모든 곳을 다 열어(찾아)내면 승리, 폭탄을 열거나 찾아 내면 패배 , 승리 또는 패배 조건을 만족 하면 결과 출력 후 프로그램 종료

**문제 분석 및 해결책 :**

기본 pch.h

#ifndef PCH\_H

#define PCH\_H

#include <iostream> //기본 출력함수

#include <cstdio>//c언어 라이브러리함수 c언어의 printf문등을 위해

#include <conio.h> //getch, 등의 함수를 위해 포함

#include <string> //string 문자열 을 사용하기위해

#include <cstdlib> //rand함수 사용을 위해

#include <time.h> //rand함수의 시드를 시간에서 받아오기위해

#include "console\_cont.h" //console컨트롤 윈도우api가져와서 함수 정의부

#include "Cell.h" //cell클래스

using namespace std;

//방향키 방향키는 실제로 키입력이 두번됨 224+방향

#define ARROWKEY 224

#define LEFT 75

#define RIGHT 77

#define UP 72

#define DOWN 80

//엔터키

#define ENTER 13

#endif //PCH\_H

-Cell class:

Cell이라는 클래스를 만들어 멤버변수로 셀마다 셀이 폭탄인지, 셀이 숨겨져 있는지, 셀이 체크되어 있는지,주변의 폭탄갯수 등을 셀마다 저장하게 한다.또한 멤버 함수로 printcell로 셀마다 셀의 정보를 출력하는 함수 정의

map은 Cell의 이차원 배열 형태로 선언, 함수인자 전달 방식을 위한 과제 이므로 Cell의 정적 멤버로 정의하지 않고, playGame()함수에서 선언, 다른 함수에서 동적 할당.

**소스코드 :**Cell.h

#pragma once

class Cell

{

bool isboom, ishide, ischeck; //

int nearboom\_count; //이셀 주변 폭탄 갯수 담을 함수

static int show\_count;

public:

Cell();

~Cell();

//맵의 가로세로 변수를 참조변수로 받아와, 사용자에게 가로,세로 크기를 입력 받고 해당하는 크기로 맵생성(동적 할당)

static Cell \*\*makeMap(int&row, int&col);

//매개로 Map과 row,col, boomcount를 참조 변수로 받아와 폭탄갯수를 사용자에게 입력 받아 중복없이 map에 폭탄을 무작위로 설치

static void hideBoom(Cell \*\*(&map), const int&row, const int&col, int& boomcount);

//map과 row,col을 참조변수로 받아와 사용자에게 보여줄 맵 상태를 출력

static void showMap(Cell \*\*(&map), const int &row, const int &col);

//map,row,col을 참조 변수로 받아와 동적할당한 맵을 할당의 역순으로 할당해제하는 함수

static void deleteMap(Cell \*\*(&map), const int &row, const int &col);

//playGame 게임의 실행함수, 맵 생성부터 게임종료까지

static void playGame();

//셀을 이동하여 셀을 열거나 체크할수 있게 하는 함수, 참조변수로 map,row,col을 받으며

//지금사용자가 있는 셀위치 x,y도 참조 변수로 받음 셀을 열때 그셀이 폭탄이면 false리턴, 아니면 true 리턴

static bool select\_cell(Cell \*\*(&map), const int&row, const int &col,int& x, int& y);

//cell을 열어주는 함수, cell을 열때 그 셀의 주변의 폭탄수가 0이라면 자동으로 주변셀을 열어주는 함수

static void cell\_hideoff(Cell \*\*(&map), const int&row, const int &col,int x, int y);

//셀의 정보를 (x,y)위치에 출력하는 함수

void printcell(int x, int y) const;

//게임의 도움말을 출력하는 함수

static void print\_HELP(const int& row, const int& col,const int& boomcount);

};

Cell.cpp

#include "pch.h"

#include "Cell.h"

Cell::Cell()

{

//생성 될때 변수 초기화

ischeck = isboom = false;

ishide = true;

nearboom\_count = 0;

}

//셀클래스의 정적 멤버 변수 show\_count선언

int Cell::show\_count = 0;

//cell의 소멸자

Cell::~Cell()

{

}

Cell \*\* Cell::makeMap(int & row, int & col)

{

cout << "맵의 가로와 세로 크기 입력 : ";

col = get\_num();

row = get\_num();

Cell\*\* map = new Cell\*[row]; //

for (int i = 0; i < row; i++)

map[i] = new Cell[col]; //맵의 칸마다 cell객채생성

return map;

}

void Cell::hideBoom(Cell \*\*(&map), const int & row, const int & col, int& boomcount)

{

cout << "맵안에 폭탄 갯수 입력 : ";

boomcount = get\_num();

//입력한 폭탄이 맵의 칸수보다 작아야함!

while (boomcount >= (row\*col)) { //그럴때까지 재입력

cout << "폭탄 갯수가 맵의 칸수보다 많거나 같습니다 . 다시 입력해 주세요";

boomcount = get\_num();

}

//폭탄 숨기기

for (int i = 0; i < boomcount; i++) {

int x = rand() % row;//랜덤한 row

int y = rand() % col; //랜덤한 col

if (map[x][y].isboom == true) {

i--; //폭탄일 경우 다시 뽑기

continue;

}

else {

map[x][y].isboom = true;

//주변 cell에 nearboom\_count더해주기

if (x > 0) {

map[x - 1][y].nearboom\_count++;

if (y > 0) map[x - 1][y - 1].nearboom\_count++;

if (y < col - 1) map[x - 1][y + 1].nearboom\_count++;

}

if (y > 0) map[x][y - 1].nearboom\_count++;

if (y < col - 1) map[x][y + 1].nearboom\_count++;

if (x < row - 1) {

map[x + 1][y].nearboom\_count++;

if (y > 0) map[x +1][y - 1].nearboom\_count++;

if (y < col - 1) map[x + 1][y + 1].nearboom\_count++;

}

}

}

}

void Cell::playGame()

{

Cell\*\* map;

int row, col, boomcount;

show\_count = 0;

int x = 0, y = 0; //사용자가 위치한 셀

map = makeMap(row, col); //맵 할당

hideBoom(map, row, col, boomcount);//폭탄 숨기기

setCursorView(false); //이제 커서 안보이게

while (show\_count + boomcount < row\*col) { //폭탄을 제외한 셀을 다 열 때까지 반복

system("cls");

print\_HELP(row, col, boomcount); //도움말 출력

gotoxy(0, 0); cout << "-----------지뢰 찾기-----------";

showMap(map, row, col); //맵보여주기

if (!select\_cell(map, row, col, x, y)) { //select\_cell함수 실행 -리턴false(폭탄을 열음)

gotoxy(0, 0); cout << "--------GAME OVER!---------" << endl; //게임오버 표시

showMap(map, row, col); //맵보여주기

Sleep(2000); //2초 있다가

system("cls");

cout << "게임에서 지셨네요..." << endl;//게임패배 문구 출력

return; //playgame 종료

}

}

//while문 나왔는데 종료 x? -> 지뢰찾기 성공

gotoxy(0, 0); cout << "--------지뢰 찾기 성공!!---------" << endl;

showMap(map, row, col); //맵보여주기

print\_HELP(row, col, boomcount); //열은 셀 횟수 다시 보여주기

Sleep(2000); //2초쉬고

system("cls");

cout << "게임에서 승리하셨습니다.\n"; //게임승리 문구 출력

deleteMap(map, row, col);

}

bool Cell::select\_cell(Cell \*\*(&map), const int & row, const int & col, int& x, int& y)

{

//지금 커서가있는 셀 색칠

setColor(6, 0);

map[y][x].printcell(x \* 2, 2 + y);

setColor(7, 0);

int old\_x= x,old\_y= y; //원래 셀 저장해두고

while (true) {

int key = 0;

if (\_kbhit()) {//키 입력시에만 실행

key = \_getch();

if (key == ARROWKEY || key == 0) {//방향키이면

key = \_getch();

switch (key)

{

case UP:

y--;

break;

case DOWN:

y++;

break;

case LEFT:

x--;

break;

case RIGHT:

x++;

break;

}

if (y < 0)

y = row-1;

if (y >= row)

y = 0;

if (x < 0)

x = col - 1;

if (x >= col)

x = 0;

setColor(7,0);

map[old\_y][old\_x].printcell(old\_x\*2, 2+ old\_y);//이전셀 원래 색으로 색칠

old\_x = x, old\_y = y;

setColor(6, 0);

map[y][x].printcell(x\*2,2 + y); //커서가 이동한 셀 색칠

setColor(7, 0);

}

else if (key == ENTER) {//엔터 입력시

if (map[y][x].isboom)//폭탄이면 false 리턴

return false;

else {

cell\_hideoff(map, row, col, y, x);//폭탄아니면 true리턴

return true;

}

}

else if (key == 'G'||key =='g') {//체크버튼 G or g입력

if(map[y][x].ishide) //숨겨져있는 셀이라면

map[y][x].ischeck = !map[y][x].ischeck;// 체크되있다고바꾸기

return true; //

}

}

}

}

void Cell::cell\_hideoff(Cell \*\*(&map), const int & row, const int & col, int x, int y)

{

if (map[x][y].ishide) {

if (map[x][y].nearboom\_count == 0) { //이 셀의 주변 폭탄 갯수가 0이라면

show\_count++; //열리니까 show\_count 1더하기

map[x][y].ishide = false; //이 셀 보이게

if (x > 0) {//맨 위쪽 셀이 아니라면

cell\_hideoff(map, row, col, x - 1, y); //바로 위에칸

if (y > 0) cell\_hideoff(map, row, col, x - 1, y - 1); //위 왼쪽칸

if (y < col - 1) cell\_hideoff(map, row, col, x - 1, y + 1); //위 오른쪽칸

}

if (y > 0) cell\_hideoff(map, row, col, x, y - 1); //왼쪽칸

if (y < col - 1) cell\_hideoff(map, row, col, x, y + 1); //오른쪽칸

if (x < row - 1) {

cell\_hideoff(map, row, col, x + 1, y);; //아래칸

if (y > 0) cell\_hideoff(map, row, col, x + 1, y - 1);

if (y < col - 1) cell\_hideoff(map, row, col, x + 1, y +1);

}

return;

}

show\_count++;//열리니까 show\_count 1더하기

map[x][y].ishide = false;//이 셀 보이게

}

else {

return;

}

}

void Cell::printcell(int x, int y) const

{

string aa[] = {"□","①","②","③","④","⑤","⑥","⑦","⑧" };//셀에 출력할 문자열

gotoxy(x, y);

if (ishide) {

if (ischeck)

cout << "★";

else

cout << "■";

}

else {

cout << aa[nearboom\_count];

}

}

void Cell::print\_HELP(const int& row, const int& col, const int& boomcount)

{

int posx = col \* 2+4;

int posy = 2;

gotoxy(posx, posy++); cout << "======도움말======";

gotoxy(posx, posy++); cout << "숫자 : 그 셀을 둘러싼 폭탄의 갯수";

gotoxy(posx, posy++); cout << "□ : 주위에 폭탄 없음. ";

gotoxy(posx, posy++); cout << "연한 초록으로 현재 셀 표기";

gotoxy(posx, posy++); cout << "셀을 이동 : 방향키 ";

gotoxy(posx, posy++); cout << "셀 열기 : 'ENTER' ";

gotoxy(posx, posy++); cout << "셀 ★로 체크해 두기 : 'G'";

gotoxy(posx, posy++); printf("셀 설정 %d X %d ", row, col);

gotoxy(posx, posy++); cout << "폭탄 수 : " << boomcount;

gotoxy(posx, posy++); cout << "지금 까지 연 셀 : "<<show\_count;

}

void Cell::showMap(Cell \*\*(&map), const int & row, const int & col)

{

//맵 보여주기

for (int y = 0; y < row; y++) {

for (int x = 0; x < col; x++) {

map[y][x].printcell(x\*2,2+ y);

}

}

}

void Cell::deleteMap(Cell \*\*(&map), const int & row, const int & col)

{

//동적할당 해제

for (int i = 0; i < row; i++) {

delete[] map[i];

}

delete[] map;

}

- console\_cont.h ,console\_cpp:

윈도우 API 을 가져와 함수로 사용 + get\_num() 정수만을 입력받는 함수

**소스코드:**

#include "pch.h"

#include "console\_cont.h"

//커서를 보이게, 안보이게 해주는 함수

void setCursorView(bool visible)

{

CONSOLE\_CURSOR\_INFO cursor = { 1, visible };

SetConsoleCursorInfo(GetStdHandle(STD\_OUTPUT\_HANDLE), &cursor);//콘솔 핸들

}

//지금 커서의 위치를 x,y로 이동해주는 함수

void gotoxy(int x, int y)

{

COORD Pos = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), Pos); //커서 위치 변경

}

//앞으로 콘솔에서 출력할 텍스트의 색,배경색을 바꿔줌

void setColor(int color, int bgcolor)

{

color &= 0xf; //비트 연산 f 남기고 다 0으로

bgcolor &= 0xf; //비트 연산

static HANDLE std\_output\_handle = GetStdHandle(STD\_OUTPUT\_HANDLE); //콘솔 핸들 가져오기

SetConsoleTextAttribute(std\_output\_handle, bgcolor << 4 | color); //윈도우핸들, bg를 왼쪽4비트 쉬프트+ color 하여 한 int변수 안에 배경,글자색정보 담기

}

int get\_num() { //정수만을 받기 위한 함수 정수가 아닌 값을 받으면 다시 입력 받게

int num;

cin >> num;

if (cin.fail()) {

cout << "다시 입력하세요. : ";

cin.clear();

cin.ignore(256, '\n');

return get\_num();

}

return num;

}

- main.cpp:

#include "pch.h"

int main()

{

bool regame= true;

srand((unsigned)time(0));

while (regame)

{

Cell::playGame();

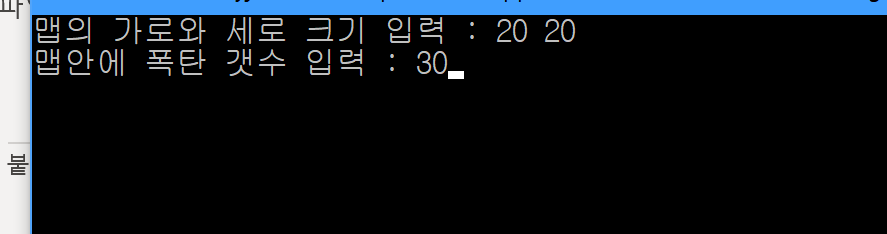
setCursorView(true);

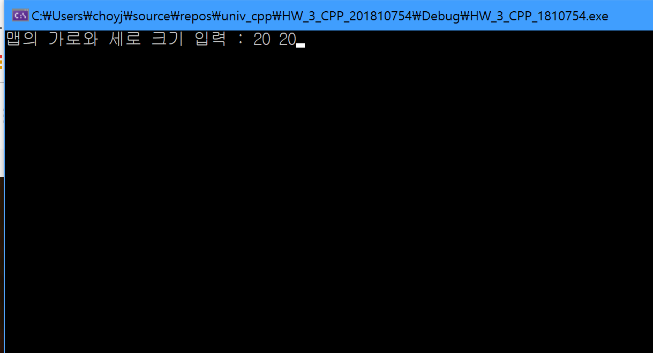
cout << "\n종료하시려면 0을 입력하세요(숫자만 입력, 0이 아닌 수 재시작)\n";

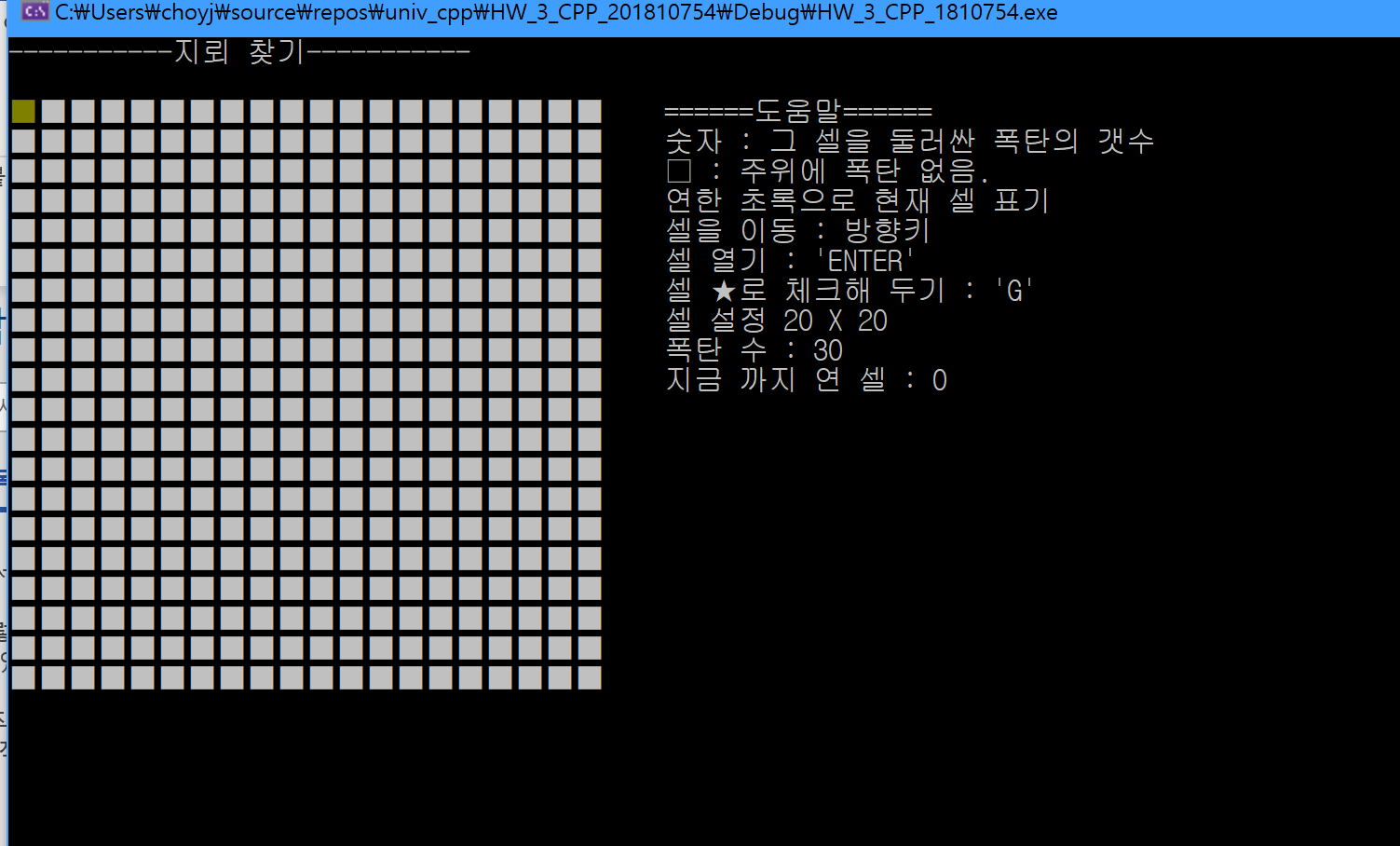
regame = get\_num();

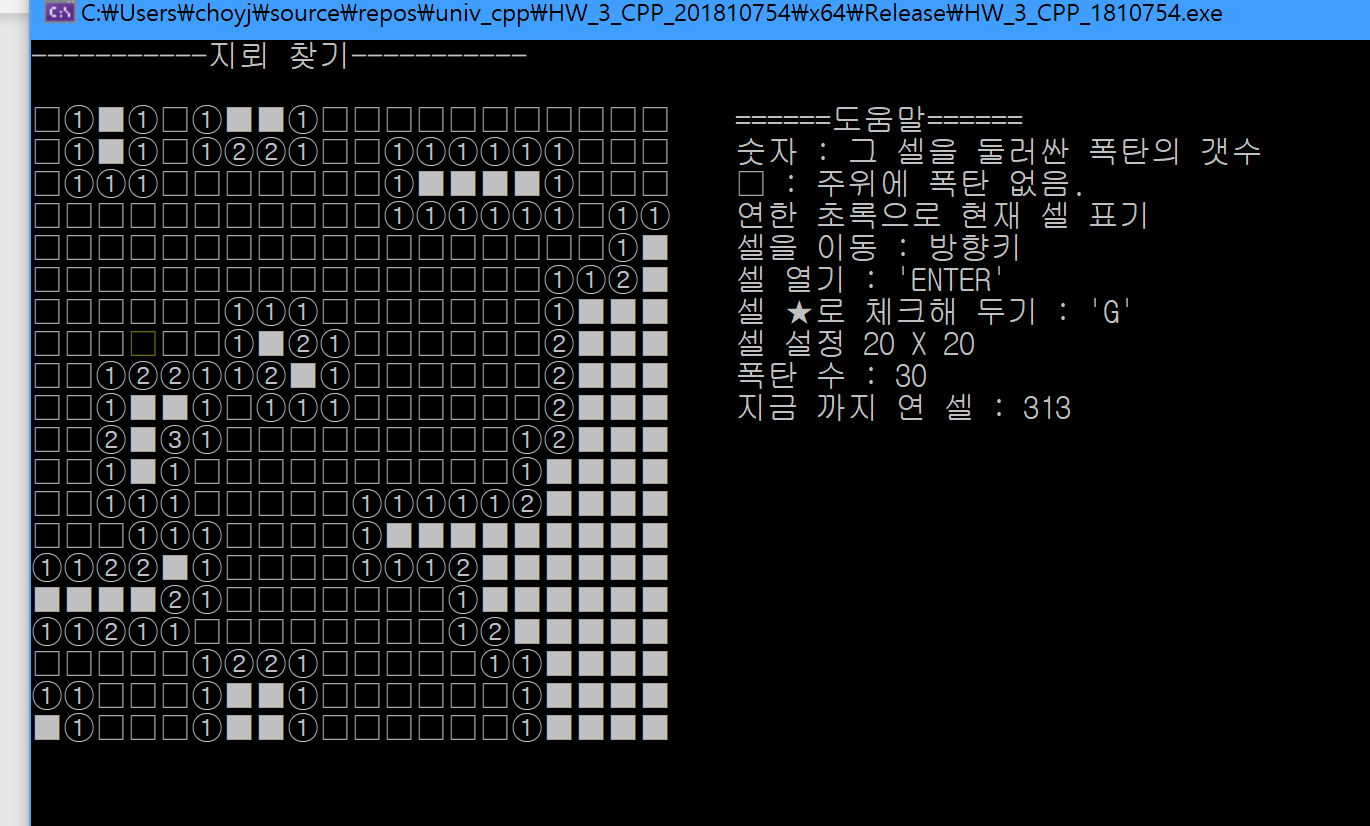
}

}

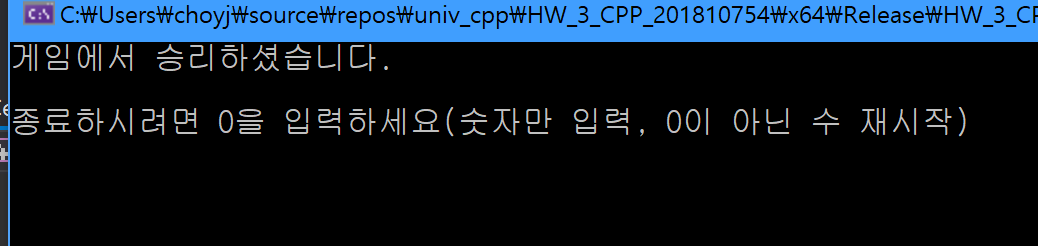
**-실행 결과**











**-느낀 점 및 토의 사항:**

윈도우API 함수: windows.h 사용하여야 입력가능

\_getch(): 출력없이 키 입력 받기

\_kbhit() : 키보드가 입력 되면 true 반환

문자 {"□","①","②","③","④","⑤","⑥","⑦","⑧" }는 2칸 출력 그래서 출력 위치의 x값은 셀 마다 2씩 증가+ char 변수 하나에 담을 수 없다 ,유니코드

-주변 셀을 검색할 때 한 칸 한 칸 다 따져가며 수행, 더 간단한 방법은 없었을까,