**C++프로그래밍PROJECT**

박지선

이효준

최광희

**[1단계]**

**< Game.h 파일 >**

class Game

{

private:

int currentStage; // 현재 선택된 스테이지를 나타내는 int 변수

vector<vector<vector<int>>> maps; // 맵을 담는 3차원 벡터값 maps.

public:

void gameStart();

int getCurrentStage() { return currentStage; } // 현재 선택된 스테이지를 리턴한다.

vector<vector<vector<int>>> getMaps() { return maps; } //3차원 벡터 maps를 리턴한다.

void setMaps();

};

**< Game.cpp 파일 >**

void Game::gameStart()

{

setMaps();

}

void Game::setMaps() //3차원 벡터에 maps.txt를 가져와 데이터를 저장하는 함수

{

int numOfRows, numOfCols; // 맵의 행과 열을 저장하는 int 변수이다.

ifstream is("maps.txt"); // 각 맵의 정보가 저장되어 있는 maps.txt 파일을 불러온다.

is >> numOfStages; // 첫 줄에 있는 총 스테이지 수를 읽어온다.

this->maps.assign(numOfStages, vector<vector<int>>()); // 2차원 벡터를 numOfStages만큼 maps에 할당한다.

for (int i = 0; i < numOfStages; i++) // 총 스테이지 수 만큼 반복문을 실행하며 각 map에 대한 정보를 저장.

{

is >> numOfRows >> numOfCols; // 행과 열을 읽어와 변수에 저장한다.

this->maps[i].assign(numOfRows, vector<int>(numOfCols, -1));

// numOfCols \* numOfRows 의 크기를 갖는 이차원 벡터를 -1로 초기화한다.

for (int j = 0; j < numOfRows; j++)

for (int k = 0; k < numOfCols; k++)

is >> this->maps[i][j][k]; // 불러온 값으로 i번째 맵을 초기화 한다.

}

}

**< main.cpp파일 >**

int main()

{

initscr();

// initscr()함수: Curses 모드 시작, 기본 크기의 윈도우 생성

resize\_term(40, 100);

// resize\_term()함수: terminal크기 조정, window까지 동시 조정, 40줄, 100칸

keypad(stdscr, TRUE);

// keypad()함수: 입력 시 키보드 특수 키 입력을 가능하게 설정,

stdscr은 default window포인터, TRUE는 사용 가능

curs\_set(0);

// curs\_set()함수: 화면에 보이는 커서 설정, 0은 커서 사라짐

noecho();

// noecho()함수: 문자 입력 시 입력한 값을 화면에 보이지 않게 함

start\_color();

// start\_color()함수: Color 사용 선언, 성공시OK(0) 반환, 에러시ERR(-1) 반환

init\_pair(1, COLOR\_YELLOW, COLOR\_BLACK);

// init\_pair()함수: 색 attribute설정, 1번 팔레트, 폰트 색 yellow, 폰트 배경색 black

bkgd(COLOR\_PAIR(1));

// bkgd()함수: 한 attribute로 윈도우 전체 적용

attron(COLOR\_PAIR(1));

// attorn()함수: Attribute적용, 1번 팔레트 사용

Game game; //Game 클래스 객체인 game을 생성한다.

game.gameStart();

border('\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*');

// border()함수: \*로 윈도우 경계선 설정

//맵 그리기

for (int j = 0; j < game.getMaps()[game.getCurrentStage()].size(); j++)

{ //현재 스테이지의 Row값 만큼 실행되는 for문

for (int k = 0; k < game.getMaps()[game.getCurrentStage()][j].size(); k++)

{ // 현재 스테이지의 Col값 만큼 실행되는 for문

if (game.getMaps()[game.getCurrentStage()][j][k] == 5)

mvprintw(20 + j, 20 + k, "#"); // 만약 값이 5(플레이어)라면 구별을 위해 ‘#’으로

출력한다.

else if (game.getMaps()[game.getCurrentStage()][j][k] == 4)

mvprintw(20 + j, 20 + k, " "); // 만약 값이 4(빈칸)라면 공백으로 출력한다.

else

myprintw(20 + j, 20 + k, “%d”, game.getMaps())game.getCurrentStage()[j][k];

//나머지는 원래 값대로 출력한다.

}

}

refresh(); // refresh()함수: 디폴트 윈도우 내용을 실제 스크린에 출력

attroff(COLOR\_PAIR(1)); // attroff()함수: Attribute 해제, 1번 팔레트 사용 해제

endwin(); // endwin()함수: Curses 모드 종료

}

**[2단계]**

**< Game.h 파일 >**

class Game

{

private:

int currentStage;

vector<vector<vector<int>>> maps;

vector<pair<int, int>> playerPositions;

public:

void gameStart();

void setMaps();

void setPlayer();

void move(const int direction);

void walk(const int y, const int x, const int nextY, const int nextX);

void push(const int y, const int x, const int nextY, const int nextX, const int afterY, const int afterX);

bool checkGoal(const int y, const int x) const;

int getCurrentStage() { return currentStage; }

vector<vector<vector<int>>> getMaps() { return maps; } ();

};

**< Game.cpp 파일 >**

void Game::gameStart()

{

setMaps();

setPlayer();

}

// 대부분 1단계와 같고 목적지 위치 저장 기능이 추가되었다.

void Game::setMaps()

{

int numOfRows, numOfCols;

ifstream is("maps.txt");

is >> numOfStages;

this->maps.assign(numOfStages, vector<vector<int>>());

this->goalPositions.assign(numOfStages, vector<pair<int, int>>());

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

{

is >> numOfRows >> numOfCols;

this->maps[i].assign(numOfRows, vector<int>(numOfCols, -1));

for (int j = 0; j < numOfRows; j++)

{

for (int k = 0; k < numOfCols; k++)

{

is >> this->maps[i][j][k];

if (maps[i][j][k] == 3) //목적지 위치를 goalPosition에 push\_back한다.

goalPositions[i].push\_back(make\_pair(j, k));

}

}

}

}

//맵 위에 플레이어를 위치시킨다.

void Game::setPlayer()

{

int y, x; //y좌표와 x좌표를 나타내는 변수

ifstream is("PlayerPositions.txt"); //player의 시작 위치는 나타내는 txt 파일

is >> numOfStages;

for (int i = 0; i < numOfStages; i++) //stage 만큼 for문을 실행하며 y값과 x값을

{ playerPositions에 push\_back한다.

is >> y >> x;

playerPositions.push\_back(make\_pair(y, x));

}

////////////////////// 이부분 업데이트 해주세요!

for (int i = 0; i < numOfStages; i++) // playerPositions에 저장된 y,x를 이용해 각 맵의 해당

{ 위치에 저장된 값을 5로 바꾼다.

maps[i][playerPositions[i].first][playerPositions[i].second] = 5;

}

////////////////////// 이부분 업데이트 해주세요!

}

void Game::move(const int direction) //direction은 getch()으로, 키보드로부터 입력되는 값이다.

{

int y = playerPositions[currentStage].first; //현재 스테이지 player의 y위치

int x = playerPositions[currentStage].second; //현재 스테이지 player의 x위치

int nextY, nextX, afterY, afterX; //next는 다음 위치, after는 그 다음 위치를 나타내는 변수

if (direction == KEY\_UP) //입력이 방향키 up이라면

{

nextY = y - 1; //다음 위치를 y-1, x로 한다.

nextX = x;

afterY = y - 2; //그 다음 위치를 y-2, x로 한다.

afterX = x;

}

else if (direction == KEY\_DOWN)

{

nextY = y + 1; //다음 위치를 y+1, x로 한다.

nextX = x;

afterY = y + 2; //다음 위치를 y+2, x로 한다.

afterX = x;

}

else if (direction == KEY\_LEFT)

{

nextY = y; //다음 위치를 y, x -1 로 한다.

nextX = x - 1;

afterY = y; //다음 위치를 y, x-2 로 한다.

afterX = x - 2;

}

else if (direction == KEY\_RIGHT)

{

nextY = y; //다음 위치를 y, x + 1로 한다.

nextX = x + 1;

afterY = y; //다음 위치를 y, x + 2로 한다.

afterX = x + 2;

}

//다음칸이 벽이 아닐 때

if (maps[currentStage][nextY][nextX] != 1)

{

//다음칸이 비어있을 때 그냥 앞으로 전진

if (maps[currentStage][nextY][nextX] == 0 || maps[currentStage][nextY][nextX] == 3)

walk(y, x, nextY, nextX);

//다음칸이 상자일 때

else

//다음다음칸이 빈칸일 경우 PUSH

if (maps[currentStage][afterY][afterX] == 0 || maps[currentStage][afterY][afterX] == 3)

push(y, x, nextY, nextX, afterY, afterX);

}

}

//빈칸으로 한 칸 앞으로 가는 함수

void Game::walk(const int y, const int x, const int nextY, const int nextX)

{

maps[currentStage][nextY][nextX] = 5; //플레이어를 다음칸으로 위치시킨다.

maps[currentStage][y][x] = 0; //기존 플레이어 위치는 0으로 한다.

playerPositions[currentStage].first = nextY; //플레이어의 y값을 업데이트한다.

playerPositions[currentStage].second = nextX; //플레이어의 x을 업데이트한다.

if (checkGoal(y, x)) // 원래 있던 자리가 목적지였다면 다시 3으로 변경한다.

maps[currentStage][y][x] = 3;

}

//앞에 상자가 있을 경우 밀면서 한 칸 앞으로 가는 함수

void Game::push(const int y, const int x, const int nextY, const int nextX, const int afterY, const int afterX)

{

maps[currentStage][afterY][afterX] = 2; //다음다음칸을 2로 바꾼다. 즉 상자를 민다

maps[currentStage][nextY][nextX] = 5; //player를 다음칸으로 위치시킨다.

maps[currentStage][y][x] = 0; //기존 플레이어 위치는 0으로 한다.

playerPositions[currentStage].first = nextY; //플레이어의 y값을 업데이트한다.

playerPositions[currentStage].second = nextX; //플레이어의 x을 업데이트한다.

if (checkGoal(y, x)) // 원래 있던 자리가 목적지였다면 다시 3으로 변경한다.

maps[currentStage][y][x] = 3;

}

//해당 좌표 위치가 목적지인지 체크하는 함수

bool Game::checkGoal(const int y, const int x) const

{

for (int i = 0; i < goalPositions[currentStage].size(); i++)

if (goalPositions[currentStage][i].first == y && goalPositions[currentStage][i].second == x)

return true;

return false;

}

**< main.cpp파일 >**

int main()

{

initscr();

resize\_term(40, 100);

keypad(stdscr, TRUE);

curs\_set(0);

noecho();

start\_color();

init\_pair(1, COLOR\_YELLOW, COLOR\_BLACK);

bkgd(COLOR\_PAIR(1));

attron(COLOR\_PAIR(1));

Game game;

game.gameStart();

//게임이 완전히 끝날때까지 반복

while (!game.checkAllSuccess())

{

//맵 그리기

for (int j = 0; j < game.getMaps()[game.getCurrentStage()].size(); j++)

{

for (int k = 0; k < game.getMaps()[game.getCurrentStage()][j].size(); k++)

{

if (game.getMaps()[game.getCurrentStage()][j][k] == 5)

mvprintw(20 + j, 20 + k, "#");

else if (game.getMaps()[game.getCurrentStage()][j][k] == 4)

mvprintw(20 + j, 20 + k, " ");

else

mvprintw(20 + j, 20 + k, "%d", game.getMaps()[game.getCurrentStage()][j][k]);

}

}

refresh();

//입력받아 움직이기

int input = getch();

game.move(input);

//화면 지우기

clear();

}

attroff(COLOR\_PAIR(1));

endwin();

}

3단계에 다시 추가시켜야 할 것들.

move에서 키보드 입력이 F2일 경우 리셋하는 함수 삭제

Move에서 마지막 줄에 있던거 삭제

//목적지에 모두 상자가 차있는지 체크

if (checkSuccess())

nextStage();

walk의 stepcount push의 stepcount, pushcount 삭제