**C++프로그래밍PROJECT**

박지선

이효준

최광희

**[1단계]**

**< Game.h 파일 >**

class Game

{

private:

int currentStage; // 현재 스테이지

vector<vector<vector<int>>> maps; // 맵을 담는 3차원 벡터

public:

int getCurrentStage() { return currentStage; }

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

void gameStart(); // 게임 시작

void setMaps(); // 맵을 세팅

};

**< Game.cpp 파일 >**

// 게임을 시작하여 맵을 세팅

void Game::gameStart()

{

setMaps();

}

void Game::setMaps()

{

int numOfRows, numOfCols; // 맵의 행, 열의 수

ifstream is("maps.txt"); // maps.txt 파일

is >> numOfStages; // 총 스테이지 수

this->maps.assign(numOfStages, vector<vector<int>>()); // 2차원 벡터 초기화

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

총 스테이지 수 만큼 각 map의 행과 열의 수를 입력 받아

벡터를 -1로 초기화한 후, maps.txt 파일로부터 맵 정보를 입력받는다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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];

}

}

**< main.cpp파일 >**

int main()

{

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

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

keypad(stdscr, TRUE); // 입력 시 키보드 특수 키 입력을 가능하게 설정 // stdscr은 default window포인터, TRUE는 사용 가능

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

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

start\_color(); // Color 사용 선언, 성공시OK(0), 에러시ERR(-1)

init\_pair(1, COLOR\_YELLOW, COLOR\_BLACK); // 색 attribute설정

// (팔레트, 폰트 색 , 폰트 배경색)

bkgd(COLOR\_PAIR(1)); // 한 attribute로 윈도우 전체 적용

attron(COLOR\_PAIR(1)); // Attribute적용, 1번 팔레트 사용

Game game; // Game 클래스 객체 생성

game.gameStart(); // 게임 시작

border('\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'); // \*로 윈도우 경계선 설정

// 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++)

// 만약 값이 4이면 공백으로 출력, 아니면 원래 값대로 출력

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

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

else

myprintw(

20 + j, 20 + k, “%d”,

game.getMaps()[game.getCurrentStage()][j][k]

);

}

}

refresh(); // 디폴트 윈도우 내용을 실제 스크린에 출력

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

endwin(); // Curses 모드 종료

}

**[3단계]**

**< Game.h 파일 >**

class Game

{

private:

int numOfStages, currentStage, stepCount, pushCount; // 카운트 변수 추가

vector<bool> finished; // 스테이지 별 종료 체크

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

vector<pair<int, int>> playerPositions;

vector<vector<pair<int, int>>> goalPositions;

public:

int getStepCount() { return stepCount; }

int getPushCount() { return pushCount; }

int getCurrentStage() { return currentStage; }

vector<bool> getFinised() { return finished; }

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

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;

void nextStage(); // 다음 스테이지로 이동

bool checkSuccess() const; // 현재 스테이지 종료 체크

bool checkAllSuccess() const; // 모든 스테이지 종료 체크

};

**< Game.cpp 파일 >**

// 변수 및 벡터 초기

void Game::gameStart()

{

setMaps();

setPlayer();

this->finished.assign(numOfStages, false);

this->currentStage = 0;

this->stepCount = 0;

this->pushCount = 0;

}

void Game::move(const int direction)

{

/\*

생략

\*/

// 목적지에 모두 상자가 차 있는지 체크

if (checkSuccess())

nextStage();

}

}

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

{

/\*

생략

\*/

stepCount++;

}

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

const int afterY, const int afterX)

{

/\*

생략

\*/

stepCount++;

pushCount++;

}

void Game::nextStage()

{

finished[currentStage] = true; // 현 스테이지 클리어

currentStage++;

stepCount = 0; // 스텝 카운트 초기화

pushCount = 0; // 푸시 카운트 초기화

}

// 현재 모든 goal Position에 모두 상자(2)가 있는지 체크 하여 boolean 반환

bool Game::checkSuccess() const

{

int y, x;

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

{

y = goalPositions[currentStage][i].first;

x = goalPositions[currentStage][i].second;

if (maps[currentStage][y][x] != 2)

return false;

}

return true;

}

// 모든 스테이지의 finished 값이 true 인지 체크

bool Game::checkAllSuccess() const

{

for (int i = 0; i < finished.size(); i++)

if (finished[i] == false)

return false;

return true;

}

**< main.cpp파일 >**

int main()

{

/\*

생략

\*/

Game game;

game.gameStart();

// 게임이 완전히 끝날 때까지 반복

while (!game.checkAllSuccess())

{

/\*

생략

\*/

//현재 발자국 수, 상자 민 횟수 그리기

mvprintw(13, 12, "steps: %d", game.getStepCount());

mvprintw(13, 23, "pushes: %d", game.getPushCount());

refresh();

/\*

생략

\*/

// 반복 전에 전 스테이지와 맵이 달라질 경우, 맵이 겹치는 경우를 대비해 화면 지우고 다시 그리기

clear();

}

attroff(COLOR\_PAIR(1));

endwin();

}