Push Box Game 만들기

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Name : 임현진(20181684), 이지선(20181676), 하준혁(20181705)

\* Description :

1 단계 : ncurses library 함수들을 사용하여 아래와 같은 2차원 배열로 표시된 push box map 을 game 화면으로 표시하는 프로그램을 완성합니다. 그림의 세부사항은 각자 정합니다. 아래 보이는 예의 배열에서 1은 벽을, 2는 상자를, 3은 목적지를 , 4는 바깥부분, P는 캐릭터를 나타내므로 각각 다른 색으로 화면에 구분이 되게 표시합니다. 다음과 같은 map 은 오른쪽 그림과 유사한 모양으로 표시됩니다.

|  |  |
| --- | --- |
| 1 1 1 1 4 4 4  1 3 0 1 1 4 4  1 3 P 0 1 4 4  1 3 0 2 1 4 4  1 1 2 0 1 1 1  4 1 0 2 0 0 1  4 1 0 0 0 0 1  4 1 0 0 1 1 1  4 1 1 1 1 4 4 |  |

2 단계 : 1단계의 맵 위에 캐릭터를 표시하고 화살표를 입력받아 캐릭터가 움직이도록 프로그램을 완성합니다. 프로그램은 캐릭터의 이동방향에 벽이 있거나 상자가 두개 이상 연속으로 있으면 캐릭터가 움직이지 않도록 해야하고, 이동 방향에 상자가 한 개 있으면서 그 상자 반대편이 또다른 상자나 벽으로 막혀있지 않으면 상자도 캐릭터와 함께 이동 방향으로 한 칸 이동해야 합니다.

3단계 : 2단계 프로그램에서 step 횟수 (캐릭터가 이동한 횟수) 와 push 횟수 (상자가 움직인 횟수) 를 화면에 보여주고 모든 상자가 목적지에 도달하면 게임을 끝내고 다음 map 으로 넘어가서 다시 게임을 시작하는 프로그램을 완성합니다.

\* Function :

void style() : 스타일들에 번호를 매겨서 나중에 사용하기 편하도록 만든 init\_pair()를 가지

고 있는 함수입니다.

void map() : 게임에서 사용 할 맵을 가지고 있는 함수로 단계별로 모두 맵이 다릅니다.

void levelTest() : 게임을 처음 시작하거나 새로 시작했을 때, 화면상에 벽,길, 상자 등을

대표하는 숫자들이 정해진 위치에 들어가고 그 숫자들에 스타일을 입혀

주도록 해주는 함수입니다.

void playgame() : 게임의 전반적인 진행을 도와주는 함수 입니다. 예를 들어서 캐릭터가

상자를 옮길 수 있는지 없는지를 판단하여 이벤트를 실행해주는 함수입니다.

int main() : 게임 끝나기 버튼을 누르기 전까지, 게임을 계속 실행시켜주는 함수입니다.

\* Code and Code explanation:

#include <ncurses.h>

#include <stdlib.h>

#include <iostream>

using namespace std;

//0 : street

//1 : wall

//2 : box

//3 : destination

//4 : outside

//5 : character

#define N 100

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

struct Pos의 역할 :

다음과 같이 구조체를 선언하여 하나의 구조체 변수가 여러가지 변수를 가질 수 있도록 하여서 변수 하나를 통해서 x의 위치, y의 위치, 화면상에 보여주는 값을 가지는 변수를 가질 수 있도록 해줍니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

struct Pos {

int xpos;

int ypos;

unsigned char z;

chtype z2;

};

int lev = 1; //현재 레벨 변수

int boxes = 0; //box의 개수 변수

int arrive = 0; //목적지의 개수 변수

int moving = 0; //캐릭터가 움직인 횟수 변수

int boxmoving = 0; //박스가 움직인 횟수 변수

struct Pos pos[N] = {};

struct Pos des[N] = {};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void style()함수의 역할 :

init\_pair()라는 함수에 각 각 번호와, 색깔을 달리 하여서 정의해줍니다. 그러면 나중에 화면에 나타나는 숫자나 알파벳의 스타일을 쉽게 변경 할 수 있습니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void style()

{

init\_pair(0, COLOR\_WHITE, COLOR\_BLACK);

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

init\_pair(2, COLOR\_GREEN, COLOR\_BLACK);

init\_pair(3, COLOR\_RED, COLOR\_BLACK);

init\_pair(4, COLOR\_BLUE, COLOR\_BLACK);

init\_pair(5, COLOR\_BLACK, COLOR\_YELLOW);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void map()함수의 역할 :

레벨에 따른, 맵의 모습을 담고 있는 함수입니다.

이 함수는 맵마다 크기,레벨, 값들이 모두 다르기 때문에 높이, 넓이, 레벨 등을 사용하는 변수를 포이터형 변수인자로 값을 받아서 map()함수 안에서 직접적으로 바꿔줍니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void map(int \*height, int \*wide, int y, int x, int \*arr, int level)

{

if (level == 1) //레벨 1인 경우

{

int map1[5][5] = { //map을 나타내는 변수

{ 1, 1, 1, 1, 4 },

{ 1, 3, 2, 0, 1 },

{ 1, 3, 2, 0, 1 },

{ 1, 5, 0, 0, 1 },

{ 1, 1, 1, 1, 1 },

};

\*height = 5; //맵의 높이 값 설정

\*wide = 5; //맵의 넓이 값 설정

\*arr = map1[y][x]; //arr의 2차원 배열을 map으로 초기화

}

else if (level == 2) //레벨 2인 경우

{

int map1[8][8] = { //map을 나타내는 변수

{ 1, 1, 1, 1, 4, 4, 4 },

{ 1, 0, 3, 3, 0, 4, 4 },

{ 1, 1, 5, 0, 0, 4, 4 },

{ 1, 1, 0, 2, 3, 4, 4 },

{ 1, 1, 0, 0, 0, 0, 1 },

{ 4, 1, 2, 2, 0, 0, 1 },

{ 4, 1, 0, 0, 0, 0, 1 },

{ 4, 1, 1, 1, 1, 1, 4 },

};

\*height = 8; //맵의 높이 값 설정

\*wide = 8; //맵의 넓이 값 설정

\*arr = map1[y][x]; //arr의 2차원 배열을 map으로 초기화

}

else if (level == 3) //레벨 3인 경우

{

int map1[9][7] = { //맵을 나타내는 변수

{ 1, 1, 1, 1, 4, 4, 4 },

{ 1, 3, 0, 0, 1, 4, 4 },

{ 1, 3, 5, 0, 0, 4, 4 },

{ 1, 3, 0, 2, 0, 4, 4 },

{ 1, 1, 2, 0, 1, 1, 1 },

{ 4, 1, 0, 2, 0, 0, 1 },

{ 4, 1, 0, 0, 0, 0, 1 },

{ 4, 1, 0, 0, 1, 1, 1 },

{ 4, 1, 1, 1, 1, 4, 4 },

};

\*height = 9; //맵의 높이 값 설정

\*wide = 7; //맵의 넓이 값 설정

\*arr = map1[y][x]; //arr의 2차원 배열을 map으로 초기화

}

else if (level == 4) //레벨 4인 경우

{

int map1[9][7] = { //map을 나타내는 변수

{ 1, 1, 1, 1, 4, 4, 4 },

{ 1, 0, 0, 1, 1, 4, 4 },

{ 1, 3, 5, 0, 1, 4, 4 },

{ 1, 3, 0, 2, 1, 4, 4 },

{ 1, 1, 2, 3, 1, 1, 1 },

{ 4, 1, 0, 2, 0, 0, 1 },

{ 4, 1, 0, 0, 0, 0, 1 },

{ 4, 1, 0, 0, 0, 1, 1 },

{ 4, 1, 1, 1, 1, 4, 4 },

};

\*height = 9; //맵의 높이 값 설정

\*wide = 7; //맵의 넓이 값 설정

\*arr = map1[y][x]; //arr의 2차원 배열을 map으로 초기화

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void levelTest()함수의 역할 :

` 이 함수는 게임을 새로 시작하거나, 다른 맵으로 넘어갈 때 실행되는 함수입니다. 많은 변수들을 모두 초기화해주고,캐릭터를 움직이는 방법, 현재 레벨, 다시 시작 등 다양한 기능들을 화면상에서 실행하기 위한 키를 알려주고 있습니다. 또한 맵을 통해서 값을 읽어와서 각 숫자들이 게임상에서 의미하는 것에 맞게 스타일과 위치를 지정해줍니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void levelTest(int level)

{

clear(); //화면이 깨끗해 지도록 해주는 함수

int i = 0, j = 0, h = 1, w = 1, mp; //map, 높이, 넓이 등의 변수 선언

moving = 0; //캐릭터가 움직인 횟수 초기화

arrive = 0; //도착 지점 개수 초기화

boxes = 0; //박스 개수 초기화

boxmoving = 0; //박스가 움직인 횟수 초기화

/\*다양한 값들을 화면에 출력\*/

mvprintw(1, 1, "Push Game");

mvprintw(2, 1, "Map -> %d", level);

mvprintw(15, 1, "Character Move -> %d", moving);

mvprintw(15, 25, "Box Move -> %d", boxmoving);

mvprintw(3, 1, "Move : W A S D, Restart : R, Previous Map : 1, Next Map : 2, Exit : Q");

for (j = 0; j<h; j++)

{

for (i = 0; i<w; i++)

{

map(&h, &w, j, i, &mp, level); //map함수를 통해서 높이, 넓이, map

등의 값을 받아옵니다.

if (mp == 0) //street case

{

mvaddch(j + 6, i + 6, '0' | COLOR\_PAIR(0));

}

else if (mp == 1) //wall case

{

mvaddch(j + 6, i + 6, '1' | COLOR\_PAIR(1));

}

else if (mp == 2) //box case

{

mvaddch(j + 6, i + 6, '0' | COLOR\_PAIR(0));

boxes++;

pos[boxes].z2 = mvinch(j + 6, i + 6);

pos[boxes].ypos = j + 6;

pos[boxes].xpos = i + 6;

pos[boxes].z = '2';

mvaddch(pos[boxes].ypos, pos[boxes].xpos, pos[boxes].z | COLOR\_PAIR(2));

}

else if (mp == 3)//destination

{

mvaddch(j + 6, i + 6, '3' | COLOR\_PAIR(3));

des[arrive].ypos = j + 6;

des[arrive].xpos = i + 6;

arrive++;

}

else if (mp == 4) //outsice case

{

mvaddch(j + 6, i + 6, '4' | COLOR\_PAIR(4));

}

else if (mp == 5) //character case

{

mvaddch(j + 6, i + 6, '0' | COLOR\_PAIR(0));

pos[0].z2 = mvinch(j + 6, i + 6);

pos[0].ypos = j + 6;

pos[0].xpos = i + 6;

pos[0].z = 'P';

mvaddch(pos[0].ypos, pos[0].xpos, pos[0].z | COLOR\_PAIR(5));

}

}

}

move(pos[0].ypos, pos[0].xpos);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void playgame()의 역할 :

` 이 함수는 게임을 진행할 때 사용되는 함수입니다. 캐릭터가 움직였을 때, 다양한 경우에 맞게 화면에 캐릭터의 위치, 박스의 위치, 목적지의 모양 등을 화면상에 보이게 해주는 함수입니다. 예를 들어서, 캐릭터가 위로 움직이거나 아래로 움직이거나 했을 때, 그 모습이 화면상에도 그대로 반영되게 해줍니다. 또한 박스와 캐릭터가 움직인 횟수를 세어서 화면상에 보여주고 각 버튼이 눌렸을 때, 게임에 반영되도록 해주는 함수입니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void playgame(char num)

{

chtype up, left, down, right, up2, left2, down2, right2, check; // 위, 아래, 왼쪽, 오른쪽 버튼을 눌렀을 때, 각 한번 혹은 두번 눌렸을 때 위치에 위치하는 캐릭터 값 변수

int restart = 0, n = 0; //다시 시작 및 목적지에 박스가 위치한 개수 변수

up = (mvinch(pos[0].ypos - 1, pos[0].xpos) & A\_CHARTEXT); //현재 위치에서 바로 위

캐릭터 값

left = (mvinch(pos[0].ypos, pos[0].xpos - 1) & A\_CHARTEXT //현재 위치에서 바로 왼쪽

캐릭터 값

down = (mvinch(pos[0].ypos + 1, pos[0].xpos) & A\_CHARTEXT); // 현재 위치에서

바로 아래쪽 캐릭터 값

right = (mvinch(pos[0].ypos, pos[0].xpos + 1) & A\_CHARTEXT);//현재 위치에서 바로

오른쪽 캐릭터 값

up2 = (mvinch(pos[0].ypos - 2, pos[0].xpos) & A\_CHARTEXT); //현재 위치에서 두번째

위 캐릭터 값

left2 = (mvinch(pos[0].ypos, pos[0].xpos - 2) & A\_CHARTEXT); //현재 위치에서 두번째

왼쪽 캐릭터 값

down2 = (mvinch(pos[0].ypos + 2, pos[0].xpos) & A\_CHARTEXT); //현재 위치에서 두번째

아래쪽 캐릭터 값

right2 = (mvinch(pos[0].ypos, pos[0].xpos + 2) & A\_CHARTEXT); //현재 위치에서 두번째

오른쪽 캐릭터 값

/\*목적지에 박스가 몇 개나 도착했는지 확인해주는 for문\*/

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

{

check = (mvinch(des[i].ypos, des[i].xpos) & A\_CHARTEXT);

if (check == '2')

{

n++;

}

}

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

{

mvaddch(pos[i].ypos, pos[i].xpos, pos[i].z2);

}

/\*박스가 목적지에 도착한 개수와 목적지의 개수가 같은 경우 – 레벨이 4보다 낮으면 +1

그렇지 않으면 다시 1로 돌아가기\*/

if (n == arrive)

{

restart = 1;

if (lev<4)

{

lev += 1;

}

else

{

lev = 1;

}

levelTest(lev);

}

/\*위로 올라가기 버튼을 눌렀을 경우\*/

else if (num == 'w' || num == 'W')

{

if (up != '1' && up != '4') // 바로 위가 벽과 바탕이 아닌 경우

{

/\*바로 위가 박스이고, 두번째 위가 길 이거나 목적지인 경우 \*/

if (up == '2' && (up2 == '0' || up2 == '3'))

{

moving++;

boxmoving++;

pos[0].ypos -= 1;

for (int j = 1; j <= boxes; j++)

{

if ((pos[0].ypos == pos[j].ypos) && (pos[0].xpos == pos[j].xpos))

{

pos[j].ypos -= 1;

}

}

}

else if (up != '2') //바로 위가 박스가 아닌 경우

{

moving++;

pos[0].ypos -= 1;

}

}

mvprintw(15, 1, "Character Move -> %d", moving); // 캐릭터가 움직인 횟수 출력

mvprintw(15, 25, "Box Move -> %d", boxmoving); // 박스가 움직인 횟수 출력

}

/\*왼쪽으로 가기 버튼이 눌렸을 경우\*/

else if (num == 'a' || num == 'A')

{

if (left != '1' && left != '4') //바로 왼쪽이 벽이거나 바탕이 아닌 경우

{

/\*바로 왼쪽이 박스이고, 두번째 왼쪽이 길이거나 목적지 인 경우 \*/

if (left == '2' && (left2 == '0' || left2 == '3'))

{

moving++;

boxmoving++;

pos[0].xpos -= 1;

for (int j = 1; j <= boxes; j++)

{

if ((pos[0].ypos == pos[j].ypos) && (pos[0].xpos == pos[j].xpos))

{

pos[j].xpos -= 1;

}

}

}

else if (left != '2') /\*바로 왼쪽이 박스가 아닌 경우

{

moving++;

pos[0].xpos -= 1;

}

}

mvprintw(15, 1, "Character Move -> %d", moving); //캐릭터가 움직인 횟수 출력

mvprintw(15, 25, "Box Move -> %d", boxmoving); //박스가 움직인 횟수 출력

}

/\*아래로 가기 버튼을 눌렀을 경우\*/

else if (num == 's' || num == 'S')

{

if (down != '1' && down != '4') //바로 아래가 벽이거나 바탕인 경우

{

/\*바로 아래가 박스 이고, 두번째 아래가 길이거나 목적지 인 경우 \*/

if (down == '2' && (down2 == '0' || down2 == '3'))

{

moving++;

boxmoving++;

pos[0].ypos += 1;

for (int j = 1; j <= boxes; j++)

{

if ((pos[0].ypos == pos[j].ypos) && (pos[0].xpos == pos[j].xpos))

{

pos[j].ypos += 1;

}

}

}

else if (down != '2') // 바로 아래가 박스가 아닌 경우

{

moving++;

pos[0].ypos += 1;

}

}

mvprintw(15, 1, "Character Move -> %d", moving); // 캐릭터가 움직인 횟수 출력

mvprintw(15, 25, "Box Move -> %d", boxmoving); //박스가 움직인 횟수 출력

}

/\*오른쪽으로 가기 버튼일 눌렸을 경우\*/

else if (num == 'd' || num == 'D')

{

if (right != '1' && right != '4') //바로 오른쪽이 벽이거나 바탕인 경우

{

/\*바로 오른쪽이 박스 이고, 두번째 오른쪽이 길이거나 목적지 인 경우 \*/

if (right == '2' && (right2 == '0' || right2 == '3'))

{

moving++;

boxmoving++;

pos[0].xpos += 1;

for (int j = 1; j <= boxes; j++)

{

if ((pos[0].ypos == pos[j].ypos) && (pos[0].xpos == pos[j].xpos))

{

pos[j].xpos += 1;

}

}

}

else if (right != '2') // 바로 오른쪽이 박스가 아닌 경우

{

moving++;

pos[0].xpos += 1;

}

}

mvprintw(15, 1, "Character Move -> %d", moving); // 캐릭터가 움직인 횟수

mvprintw(15, 25, "Box Move -> %d", boxmoving); // 박스가 움직인 횟수

}

/\*이전 레벨로 가기 버튼을 눌렀을 경우\*/

else if (num == '1')

{

restart = 1;

if (lev>1)

{

lev--;

}

levelTest(lev);

}

/\*다음 레벨로 가기 버튼을 눌렀을 경우\*/

else if (num == '2')

{

restart = 1;

if (lev<4)

{

lev += 1;

}

levelTest(lev);

}

/\*다시 시작하기 버튼을 눌렀을 경우\*/

else if (num == 'r' || num == 'R')

{

restart = 1;

levelTest(lev);

}

/\*다시 시작하기 버튼이 활성화 되지 않았다면 \*/

if (!restart)

{

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

{

pos[i].z2 = mvinch(pos[i].ypos, pos[i].xpos);

mvaddch(pos[i].ypos, pos[i].xpos, pos[i].z | ((i == 0) ? COLOR\_PAIR(5) : COLOR\_PAIR(2)));

}

move(pos[0].ypos, pos[0].xpos);

}

/\*다시 시작하기 버튼이 활성화 되었다면\*/

else

{

restart = 0;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()의 역할 :

게임을 시작했을 때, 종료버튼을 누르기 전까지 게임이 화면상에 유지되도록 해주는

함수입니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

char c;

initscr();

start\_color();

style();

levelTest(lev);

while (1)

{

c = getch(); // 게임에서 버튼역할을 하는 함수

if (c == 'q')

{

break;

}

playgame(c);

}

endwin();

return 0;

}

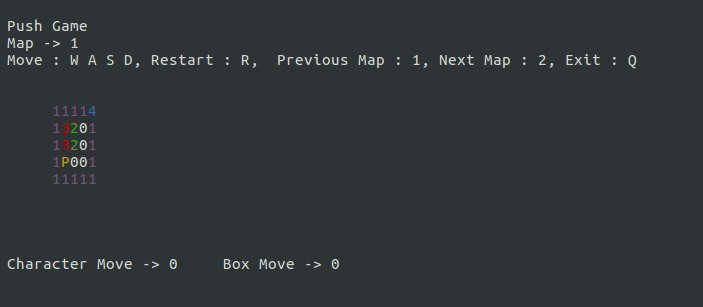
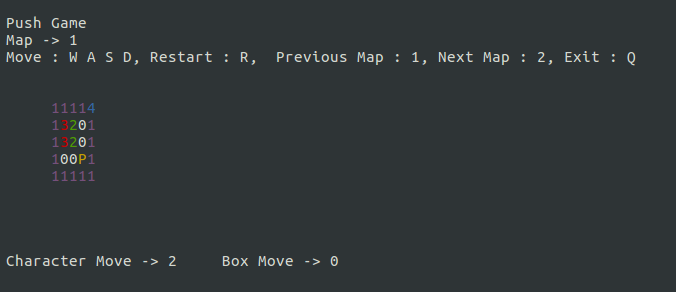
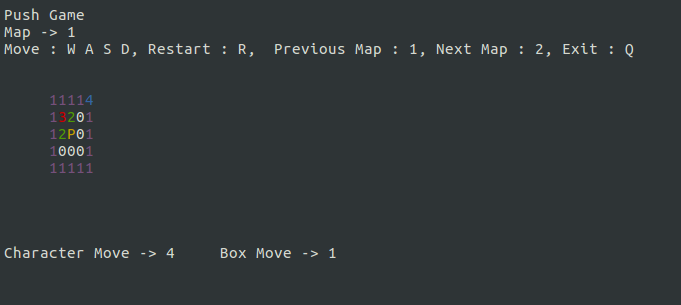
1단계 실행 화면

|  |  |
| --- | --- |
| 1 1 1 1 4  1 3 2 0 1  1 3 2 0 1  1 5 0 0 1  1 1 1 1 1 |  |
| 1 1 1 1 4 4 4  1 0 3 3 0 4 4  1 1 5 0 0 4 4  1 1 0 2 3 4 4  1 1 0 0 0 0 1  4 1 2 2 0 0 1  4 1 0 0 0 0 1  4 1 1 1 1 1 4 |  |
| 1 1 1 1 4 4 4  1 3 0 0 1 4 4  1 3 5 0 0 4 4  1 3 0 2 0 4 4  1 1 2 0 1 1 1  4 1 0 2 0 0 1  4 1 0 0 0 0 1  4 1 0 0 1 1 1  4 1 1 1 1 4 4 |  |

맵별로 코드 상에서 쓰여진 맵과 게임 화면 상에서 보여지는 맵입니다.

게임상에서 보여지는 맵은 벽, 바탕, 캐릭터, 박스, 목적지 등 각 각의 역할에 따라 색깔을 달리하여 구분하기 편하도록 하였습니다.

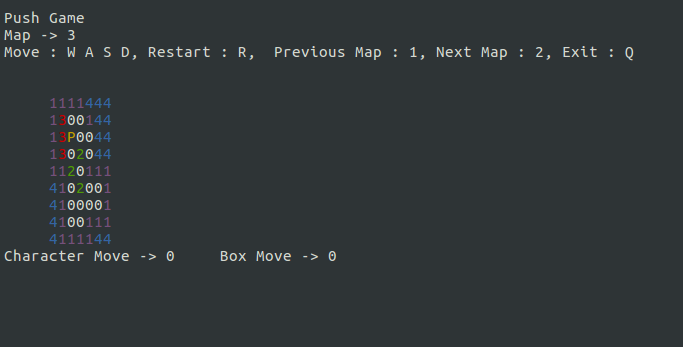
2단계 실행 화면



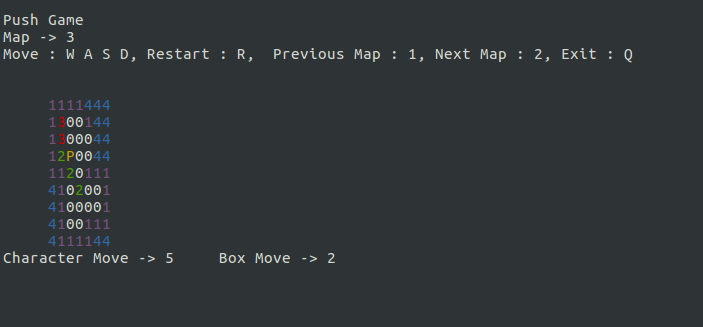
캐릭터가 박스를 움직여서 목적지에 넣은 모습입니다.

오른쪽으로 2번 움직이고, 오른쪽 한번 아래쪽 한번 움직인 화면입니다. 화면을 보면 오른쪽으로 2번 움직인 후에 오른쪽 아래키를 눌렀지만 움직이지 않았다는 것을 확인 할 수 있습니다.

레벨 1의 처음 게임 화면 입니다.

3단계 실행 화면

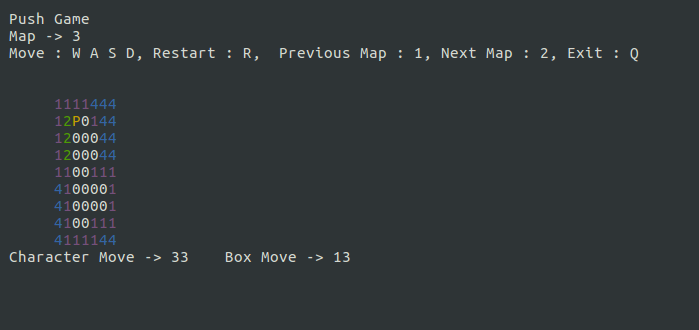
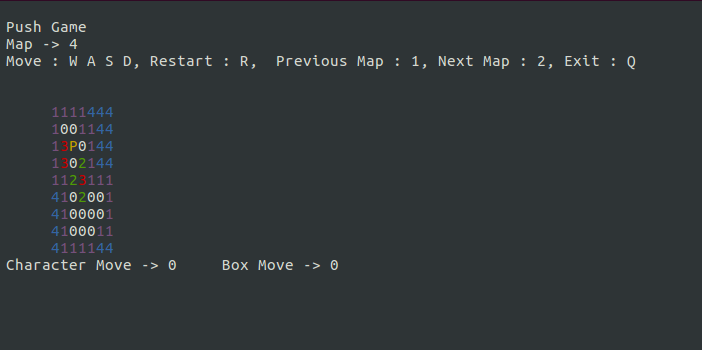
레벨 3 처음 실행 화면입니다.



현재 캐릭터를 움직여서 목적지에 박스를 하나 옮겨 놓았습니다. 보시면 캐릭터가 움직인 횟수와 박스가 움직인 횟수가 처음과 달라진 것을 알 수 있습니다.



현재 캐릭터를 움직여서 목적지에 2번째 박스를 옮겨 놓았습니다. 보시면 첫번째 박스를 옮겼을 때와 2번째 박스를 옮겨 놓았을 때 캐릭터와 박스가 움직인 횟수가 달라진 것을 알 수 있습니다.



이렇게 박스를 목적지에 다 넣은 후에, 아무 키나 누르면 바로 다음 레벨로 넘어간 것을 볼 수 있습니다.

오른쪽 화면은 레벨 3이 끝나서 레벨 4로 넘어 간 화면입니다.

마지막 박스까지 다 옮겨 놓으면 최종적으로

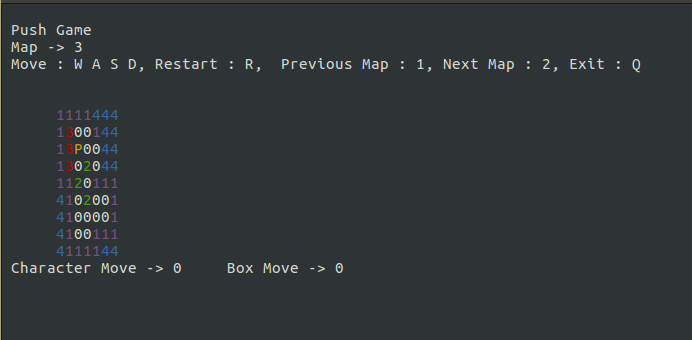
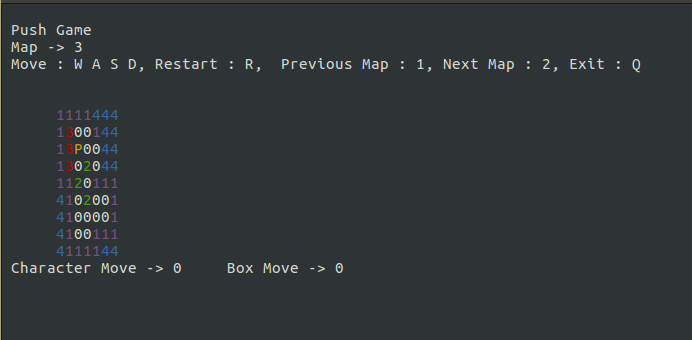
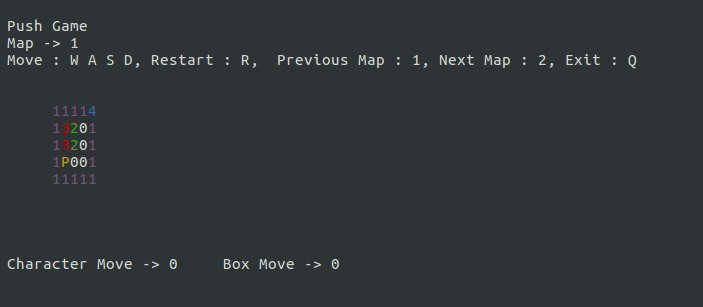
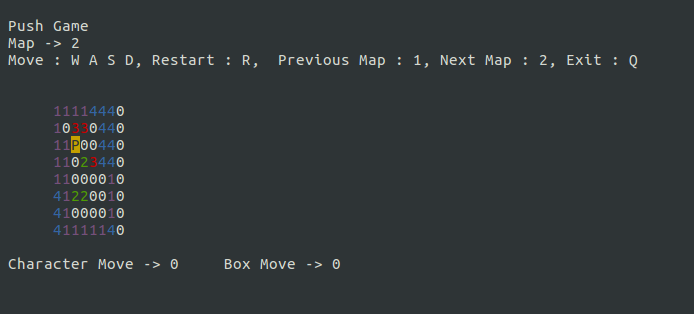
캐릭터가 움직인 횟수 : 33

박스가 움직인 횟수 : 13

이라는 것을 알 수 있습니다.

그 외 기능

이전 맵, 다음 맵 접근 버튼



이전 맵 접근 버튼 : 1

다음 맵 접근 버튼 : 2

다음 사진을 보시면 현재 2레벨에서 이전 맵 버튼을 누르면 레벨 1로 다음 맵 버튼을 누르면 레벨 2로 바뀌는 것을 볼 수 있습니다.

리셋 버튼 : R

리셋 버튼을 누르면 현재 레벨의 맵이 처음 상태로 초기화 되면서, 캐릭터와 박스가 움직인 횟수도 같이 초기화 됩니다. 이를 통해서 처음부터 다시 게임을 할 수 있습니다.

리셋 버튼