Push Box Game 보고서

C++ 프로그래밍 (6분반) 최진우 교수님

조장 20181685 임호준

20181699 최승준

목차

1. 전역변수/함수

```
#include <ncurses.h>
#include <iostream>
#include <locale.h>
#include <string>
#include <clocale>
#include <algorithm>
static int map3copy[11][9] = {
    {4,4,4,4,4,4,4,4,4},
    {1,1,1,1,1,1,1,1,4},
    {1,1,0,1,0,1,1,1,4},
    {1,0,0,0,0,0,1,1,4},
    {1,0,2,1,0,0,0,1,1},
    {1,0,0,2,0,0,2,0,1},
    {1,1,1,0,0,0,0,0,1},
    {1,1,1,0,0,0,0,1,1},
    {4,1,1,1,1,1,1,1,1},
    {4,1,1,1,1,1,1,1,1},
    {4,4,4,4,4,4,4,4,4}
};
static int map2copy[11][9] = {
    {4,4,4,4,4,4,4,4,4},
    {1,1,1,1,1,4,4,4,4},
    {1,1,1,1,1,1,1,1,1},
    {1,1,1,1,0,0,1,1,1},
    {4,1,1,1,0,0,0,0,1},
    {1,1,0,2,0,0,2,0,1},
    {1,1,0,0,0,0,2,0,1},
    {1,1,0,0,0,0,0,1,1},
    {1,1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,1,1,1},
    {4,4,4,4,4,4,4,4,4}
};

static int map1copy[11][9] = {
    {4, 4, 4, 4, 4, 4, 4, 4, 4},
```

```
    {4, 1, 1, 1, 1, 1, 1, 4, 4},
    {4, 1, 0, 0, 0, 0, 1, 4, 4},
    {4, 1, 0, 0, 0, 0, 1, 4, 4},
    {4, 1, 0, 0, 2, 0, 1, 4, 4},
    {4, 1, 1, 2, 0, 0, 1, 1, 4},
    {4, 4, 1, 0, 2, 0, 0, 1, 4},
    {4, 4, 1, 0, 0, 0, 0, 1, 4},
    {4, 4, 1, 0, 0, 1, 1, 1, 4},
    {4, 4, 1, 1, 1, 1, 4, 4, 4},
    {4, 4, 4, 4, 4, 4, 4, 4, 4},
};
static int map1[11][9] = {
    {4, 4, 4, 4, 4, 4, 4, 4, 4},
    {4, 1, 1, 1, 1, 1, 1, 4, 4},
    {4, 1, 0, 0, 0, 0, 1, 4, 4},
    {4, 1, 0, 0, 0, 0, 1, 4, 4},
    {4, 1, 0, 0, 2, 0, 1, 4, 4},
    {4, 1, 1, 2, 0, 0, 1, 1, 4},
    {4, 4, 1, 0, 2, 0, 0, 1, 4},
    {4, 4, 1, 0, 0, 0, 0, 1, 4},
    {4, 4, 1, 0, 0, 1, 1, 1, 4},
    {4, 4, 1, 1, 1, 1, 4, 4, 4},
    {4, 4, 4, 4, 4, 4, 4, 4, 4},
};
int pr = 3, pc = 2;
static int map1solve[3][2] = { {2,2}, {3,2}, {4,2} };
static int map2solve[3][2] = { {3,4}, {3,5}, {4,7} };
static int map3solve[3][2] = { {4,5}, {4,6}, {7,6} };

void clear1(int maps);
int check1();
void moveon(int s);
void print(WINDOW * win1);

static int stage = 1;
static int step;
static int push;
```

각 스테이지에서 사용할 맵을 전역변수로 표시한다. 이때, map1은 게임판의 역할을 하고 이 게임판을 채워줄 맵들을 map1copy, map2copy, map3copy라는 전역 배열 변수로 선언하였다.

```
int pr = 3, pc = 2;
static int map1solve[3][2] = { {2,2}, {3,2}, {4,2} };
static int map2solve[3][2] = { {3,4}, {3,5}, {4,7} };
static int map3solve[3][2] = { {4,5}, {4,6}, {7,6} };
```

게임판 위에서 사용자의 위치를 pc,pr이라는 변수에 저장한다. Pc는 player column의 줄임말이고, pr 은 player row의 줄임말이다. 격자 위에서 플레이어의 위치를 나타내는 것이다.

map1solve[][], map2solve[][], map3solve[][] 는 각 맵에 목적지들의 행과 열이다.

```
static int stage = 1;
static int step;
static int push;
```

Stage는 현재 단계를 의미한다.

Step은 사용자가 움직인 횟수이며

Push는 사용자가 상자를 민 횟수이다.

구현할 메소드는 다음과 같다.

```
void clear1(int maps);
int check1();
void moveon(int s);
void print(WINDOW * win1);
```

Clear1()은 사용자가 맵을 클리어했거나, R키를 눌러 리스타트를 했을 때 호출되는 맵 초기화 메서드이다.

Check1()은 준비된 박스가 모두 목적지에 도달했는지를 검사한다.

Moveon()은 맵 위에서 플레이어가 박스를 밀거나, 빈 공간으로 플레이어가 이동하는 행위를 모두 담당하는 메서드이다.

Print는 win1이라는 Ncurses WINDOW 객체 위에서 게임판을 표시해준다.

2. 메소드 설명

```
void moveon(int s) {

    int r, c;
    if (s == 259) { r = -1; c = 0; }
    else if (s == 258) { r = 1; c = 0; }
    else if (s == 260) { r = 0; c = -1; }
    else if (s == 261) { r = 0; c = 1; }
    else if (s == 114) {
        clear1(stage);

        return;
    }

    if (map1[pr + r][pc + c] == 1) {

        return;
    }
    else if (map1[pr + r][pc + c] == 0) {
        pr += r; pc += c;
        step++;
        return;
    }
    else if (map1[pr + r][pc + c] == 2 && map1[pr + r * 2][pc + c * 2] == 2) {

        return;
    }
    else if (map1[pr + r][pc + c] == 2 && map1[pr + r * 2][pc + c * 2] == 0) {
        pr += r; pc += c;
        map1[pr][pc] = 0;
        map1[pr + r][pc + c] = 2;
        step++;
        push++;

    }
}
```

moveon() - 사용자의 입력에 따른 맵 위 요소들의 이동을 구현해주는 메소드이다. 키 입력을 키 코드로 변환해서, 주어진 입력에 따라 r, c를 다르게 설정한다. 움직일 곳이 벽인지, 아니면 빈 공간인지 판단하는 조건문이 들어있다. 아래 더 자세한 설명이 있다.

```
int r, c;
if (s == 259) { r = -1; c = 0; }
else if (s == 258) { r = 1; c = 0; }
else if (s == 260) { r = 0; c = -1; }
else if (s == 261) { r = 0; c = 1; }
else if (s == 114) {
    clear1(stage);

    return;
}
```

사용자의 키보드 입력 s는 w/s/a/d/r 에 따라 다른 조건을 수행한다.

W와s는 사용자의 위아래값, a와d는 사용자의 왼쪽오른쪽 값을 변화시킨다.

R을 입력받으면 스테이지를 리셋시킨다.

```
1 if (map1[pr + r][pc + c] == 1) {

    return;
}
2 else if (map1[pr + r][pc + c] == 0) {
    pr += r; pc += c;
    step++;
    return;
}
3 else if (map1[pr + r][pc + c] == 2 && map1[pr + r * 2][pc + c * 2] == 2) {

    return;
}
4 else if (map1[pr + r][pc + c] == 2 && map1[pr + r * 2][pc + c * 2] == 0) {
    pr += r; pc += c;
    map1[pr][pc] = 0;
    map1[pr + r][pc + c] = 2;
    step++;
    push++;

}
```

사용자의 위치 변화가 일어날 때 상황에 맞게 맵을 변화시켜주는 조건문.

1) 사용자 이동 위치에 벽이 있으면 변화없다.
2) 사용자 이동 위치에 빈 칸이 있으면 이동하고 step에 1더해준다.
3) 사용자 이동 위치에 상자가 있고 가려는 방향 한칸 뒤에도 상자가 있으면 변화없다.
4) 사용자 이동 위치에 상자가 있고 가려는 방향 한칸 뒤에 빈 칸이 있으면 상자를 옮겨주고 이동하며 step,push에 1씩 더해준다.

clear1() – 스테이지에 맞는 맵 초기화 또는 리셋키가 눌릴 때.

```
void clear1(int maps) {
    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 9; j++) {
            if (maps == 1)
                map1[i][j] = map1copy[i][j];
            else if (maps == 2)
                map1[i][j] = map2copy[i][j];
            else if (maps == 3)
                map1[i][j] = map3copy[i][j];
        }
    }
    if (maps == 2) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 2; j++) {
                map1solve[i][j] = map2solve[i][j];
            }
        }
    }
    else if (maps == 3) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 2; j++) {
                map1solve[i][j] = map3solve[i][j];
            }
        }
    }
    if (maps == 1) {
        pr = 3; pc = 3;
    }
    else if (maps == 2) {
        pr = 7; pc = 3;
    }
    else if (maps == 3) {
        pr = 3; pc = 2;
    }

    step = 0;
}
```

각 스테이지에 맞게 맵과 사용자의 위치를 초기화시켜서

사용자가 r을 누르면 새롭게 게임을 할 수 있다.

이때, map1solve에 상자가 도착해야할 목적지를 스테이지마다 넣어줘서, 게임을 클리어하는지를 check()로 판단한다.

Check1();

```
int check1() {
    int k = 0;
    if (map1[map1solve[0][0]][map1solve[0][1]] == 2 && map1[map1solve[1][0]][map1solve[1][1]] == 2 && map1[map1solve[2][0]][map1solve[2][1]] == 2)
    {
        return 3;
    }
    if (map1[map1solve[0][0]][map1solve[0][1]] == 2) {
        k++;
    }
    if (map1[map1solve[1][0]][map1solve[1][1]] == 2) { k++; }
    if (map1[map1solve[2][0]][map1solve[2][1]] == 2) { k++; }
    return k;
}
```

박스가 목적지에 도달할때마다 k를 한 개씩 더해주고

모든 목적지에 도달하면 3을 리턴해 클리어를 알려준다.

Print()

```
void print(WINDOW * win1) {

    map1[pr][pc] = 7;

    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 9; j++) {
            if (map1[i][j] == 1) {
                wattron(win1, COLOR_PAIR(1));
                mvwprintw(win1, i, j, "%s", "0");
                wattroff(win1, COLOR_PAIR(1));
            }
            else if (map1[i][j] == 2) {
                wattron(win1, COLOR_PAIR(2));
                mvwprintw(win1, i, j, "%s", "0");
                wattroff(win1, COLOR_PAIR(2));
            }
            else if (map1[i][j] == 7) {
                wattron(win1, COLOR_PAIR(3));
                mvwprintw(win1, i, j, "%s", "*");
                wattroff(win1, COLOR_PAIR(3));
            }
            else if (map1[i][j] == 4) {
                wattron(win1, COLOR_PAIR(4));
                mvwprintw(win1, i, j, "%s", "0");
                wattroff(win1, COLOR_PAIR(4));
            }
            else {
                mvwprintw(win1, i, j, "%s", " ");
                if (i == map1solve[0][0] && j == map1solve[0][1]) {
                    wattron(win1, COLOR_PAIR(5));
                    mvwprintw(win1, i, j, "%s", " ");
                    wattroff(win1, COLOR_PAIR(5));
                }
                else if (i == map1solve[1][0] && j == map1solve[1][1]) {
                    wattron(win1, COLOR_PAIR(5));
                    mvwprintw(win1, i, j, "%s", " ");
                    wattroff(win1, COLOR_PAIR(5));
                }
                ,
                else if (i == map1solve[2][0] && j == map1solve[2][1]) {
                    wattron(win1, COLOR_PAIR(5));
                    mvwprintw(win1, i, j, "%s", " ");
                    wattroff(win1, COLOR_PAIR(5));
                }

            }

        }

    }

    refresh();
    map1[pr][pc] = 0;

}
```

맵을 출력해주는 메소드.

미리 설정해둔 팔레트들을 적용하고

맵 위 숫자에 따라 보여주는 아이콘을 설정.

```
if (map1[i][j] == 1) {
    wattron(win1, COLOR_PAIR(1));
    mvwprintw(win1, i, j, "%s₩n", "O");
    wattroff(win1, COLOR_PAIR(1));
}
else if (map1[i][j] == 2) {
    wattron(win1, COLOR_PAIR(2));
    mvwprintw(win1, i, j, "%s₩n", "O");
    wattroff(win1, COLOR_PAIR(2));
}
else if (map1[i][j] == 7) {
    wattron(win1, COLOR_PAIR(3));
    mvwprintw(win1, i, j, "%s₩n", "*");
    wattroff(win1, COLOR_PAIR(3));
}
else if (map1[i][j] == 4) {
    wattron(win1, COLOR_PAIR(4));
    mvwprintw(win1, i, j, "%s₩n", "O");
    wattroff(win1, COLOR_PAIR(4));
}
```

숫자1->벽 숫자2->빈칸 숫자3->목적지 ->숫자4->장외 설정.

```
        else {
            mvwprintw(win1, i, j, "%s₩n", " ");
            if (i == map1solve[0][0] && j == map1solve[0][1]) {
                wattron(win1, COLOR_PAIR(5));
                mvwprintw(win1, i, j, "%s₩n", " ");
                wattroff(win1, COLOR_PAIR(5));
            }
            else if (i == map1solve[1][0] && j == map1solve[1][1]) {
                wattron(win1, COLOR_PAIR(5));
                mvwprintw(win1, i, j, "%s₩n", " ");
                wattroff(win1, COLOR_PAIR(5));
            }
            else if (i == map1solve[2][0] && j == map1solve[2][1]) {
                wattron(win1, COLOR_PAIR(5));
                mvwprintw(win1, i, j, "%s₩n", " ");
                wattroff(win1, COLOR_PAIR(5));
            }

        }
```

클리어된 목적지는 색깔을 따로 설정한다.

3. Main 함수

```
int main() {
    initscr(); // ncurses 초기화/ 가장 먼저 수행
    setlocale(LC_ALL, ""); // 유니코드를 사용하기 위해 하는 행위.
    start_color(); // 색깔을 사용하기 위해 하는 행위임
    keypad(stdscr, TRUE); // 키 입력을 받기 위해 하는 행위임.
    curs_set(0); // 커서를 감춤
    noecho(); // 콘솔 출력을 보이지 않게 함.

    int key; // key를 받기 위한 셋임

    init_pair(1, COLOR_BLUE, COLOR_BLACK);
    init_pair(2, COLOR_RED, COLOR_BLACK);
    init_pair(3, COLOR_YELLOW, COLOR_BLACK);
    init_pair(4, COLOR_CYAN, COLOR_BLACK);
    init_pair(5, COLOR_MAGENTA, COLOR_GREEN);

    resize_term(999, 999); // 스크린 크기 조절
    mvprintw(10, 0, "This is Push-Box1"); // 스크린에 보일 내용임.
    mvprintw(11, 0, "All right is on 20181685");

    key = getch();
    clear();
    printw("%d₩n", key);
    refresh();
    getch();
    clear();
    WINDOW *win1;
    WINDOW *win2;
```

Ncurses로 게임 화면을 만들어주는 부분이다. 강의 자료를 참고해서 만들었고, win1(게임창) win2(스테이터스창)의 두 WINDOW객체가 있다.

```
    while (true) {
        if (key == 27) { break; }
        if (check1() == 3) {
            clear();
            if (stage == 3) { break; }
            printw("clear");
            refresh();
            clear1(++stage);
            getch();
        }
        clear();
        win1 = newwin(11, 9, 2, 3);
        print(win1);
        wrefresh(win1);

        win2 = newwin(9, 15, 5, 20);
        mvwprintw(win2, 0, 1, "%s₩n    ", "Stage");
        mvwprintw(win2, 1, 1, "%d₩n    ", stage);
        mvwprintw(win2, 3, 1, "%s₩n    ", "Step");
        mvwprintw(win2, 4, 1, "%d₩n", step);
        mvwprintw(win2, 5, 1, "%s₩n    ", "Push");
        mvwprintw(win2, 6, 1, "%d₩n    ", push);
        mvwprintw(win2, 8, 1, "Restart : R");
        //s 258 // 259 w // 261 d // 260 a // 114 r//
        wrefresh(win2);
        refresh();
        key = getch();


        moveon(key);
    }
    printw("ALL STAGE CLEAR!!");
    refresh();

    getch();
    return 0;
}
```

게임이 실행되면 모든 스테이지가 클리어 될 때까지 게임을 실행한다. 키 입력을 받아 수행하며, 3스테이지를 모두 클리어하면 화면에 ALL STAGE CLEAR!! 이라는 텍스트와 함께 게임을 종료시킨다. WIN2의 스테이터스 내용을 지속적으로 업데이트하는 내용이 포함되어있다.

4. 실행 화면

게임을 켰을 때 화면.

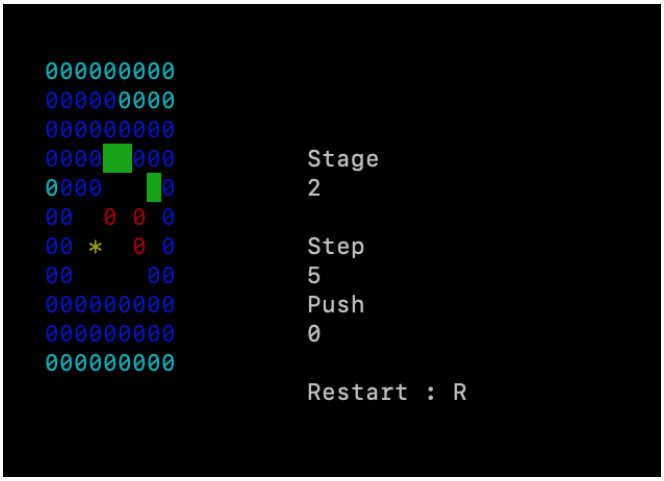
왼쪽에는 사용자가 게임하는 화면이다.

하늘색은 하드블럭, 파란색은 벽, 빨간색은 상자이다.

플레이어는 노란색 별표이고, 목적지는 초록 배경색이다.

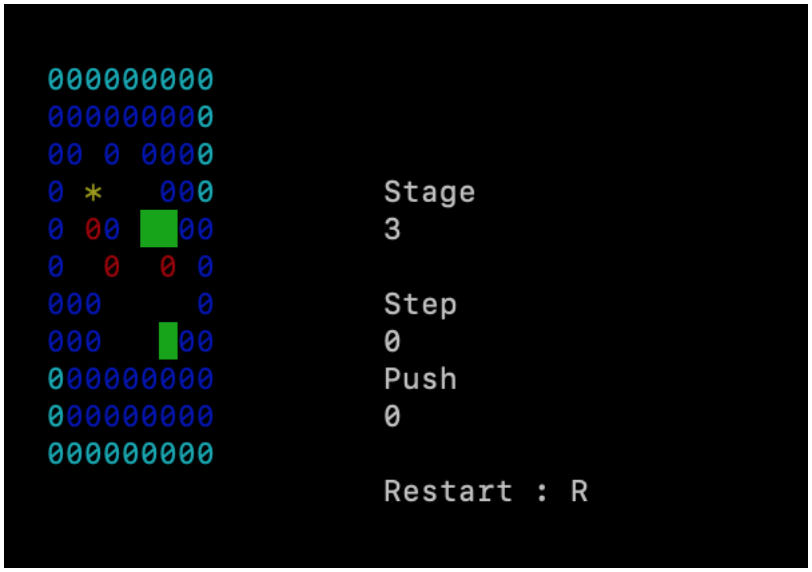
오른쪽에는 현재 스테이지,Step횟수,Push횟수, restart 버튼을 알려주는 라벨이 있다.

현재 스테이지를 깨면 클리어가 뜬다.



Clear가 뜨면 스테이지2가 실행된다

스테이지 마다 다른 맵이 적용된다.



스테이지 2를 깨면 이어서 스테이지3 까지 실행된다.

모든 스테이지를 깨면 ALL STAGE CLEAR!! 가 실행되고 게임이 클리어된다.