milestone1.md

2019 C++ 프로그래밍 프로젝트

ncurses library를 사용한 Push Box Game 구현

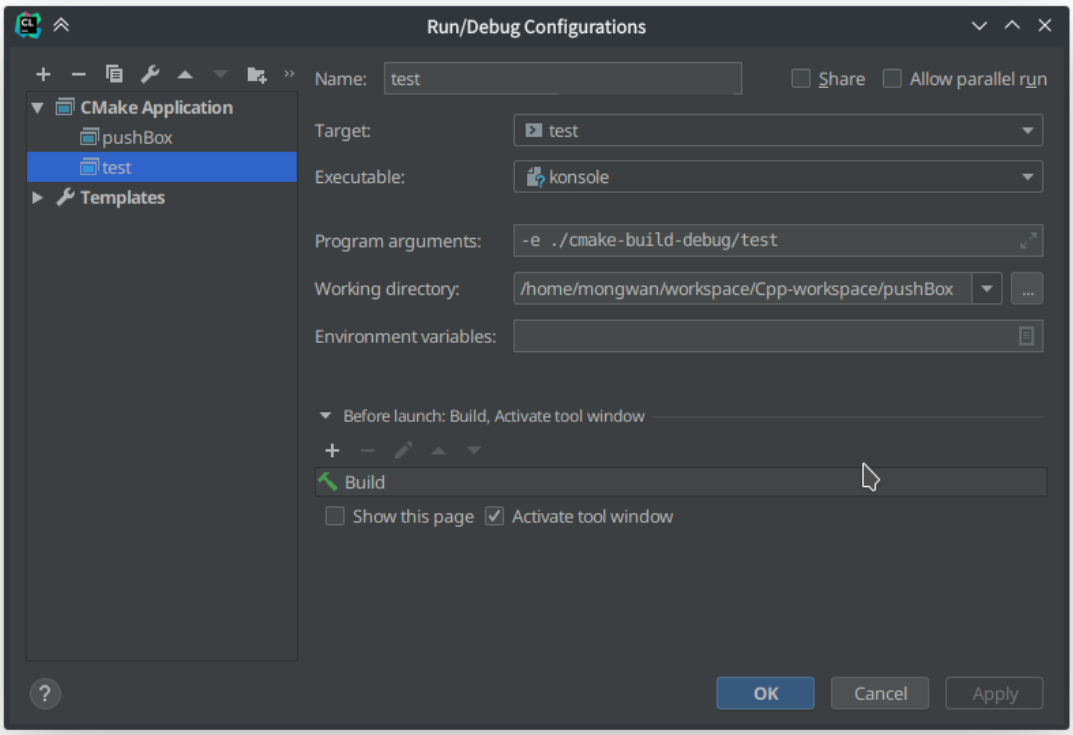
20181649 유현석

1단계

작업 환경 설정

CLion

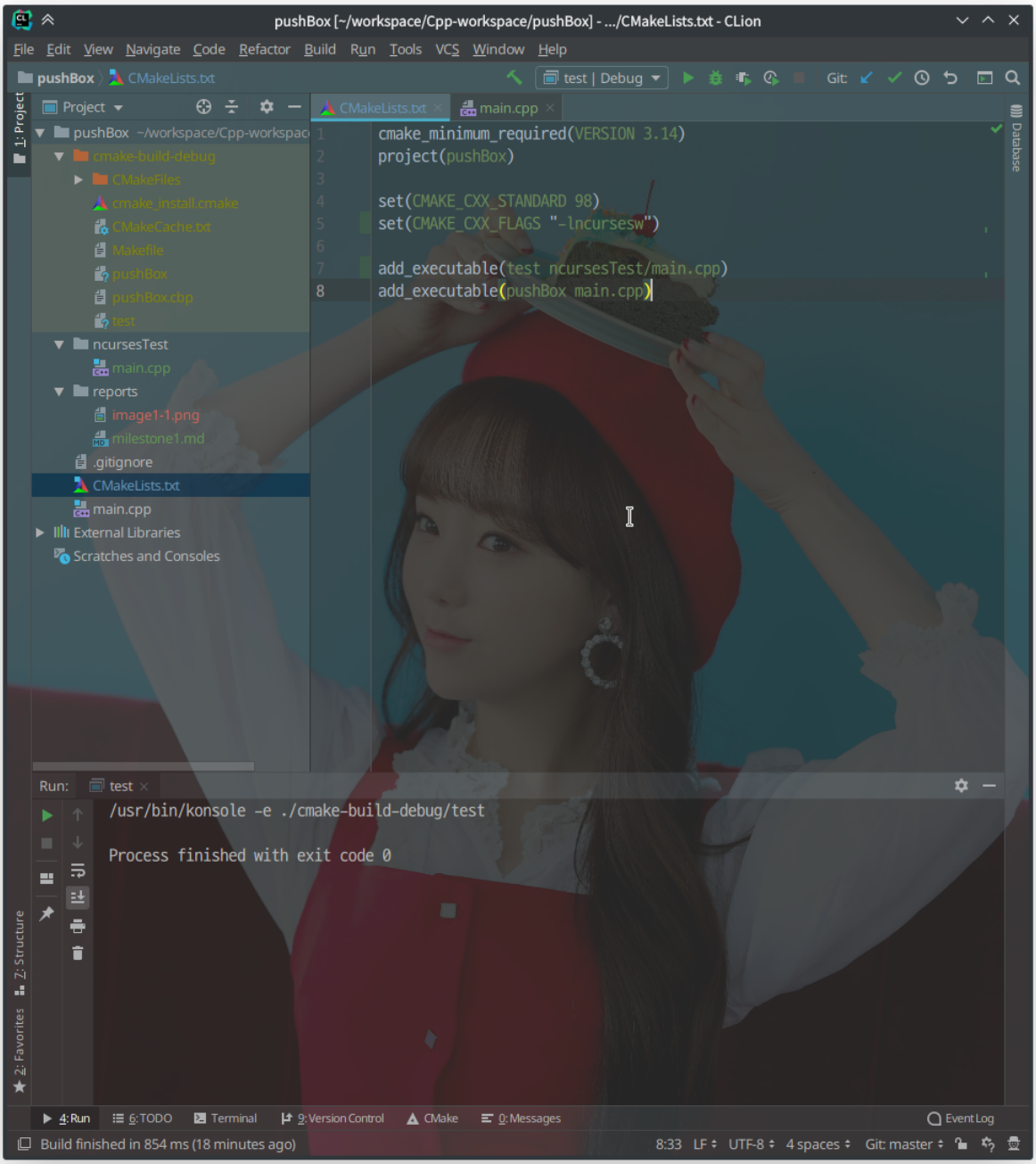
전부터 사용하던 Jetbrain의 CLion을 이번 프로젝트에서도 활용하기 위해 몇가지 설정이 필요했다. CLion 내장 터미널에서는 ncurses가 구동되지 않았기 때문에, 외부 콘솔 창을 여는 것을 설정해야 했다.



현재 데스크탑 환경으로 KDE를 사용하기 때문에, 터미널은 내장되어있는 Konsole을 사용했다. 그림과 같이 설정함으로써 CLion에서 실행 버튼을 누르는 것 만으로 build를 한 뒤에 Konsole을 열어 현재 작성중인 코드를 실행할 수 있게 되었다. 이 링크의 글을 참고하였다.

CMakeLists

CLion에서는 CMake를 이용해 프로그램을 빌드한다. 따라서 CMakeLists.txt를 수정하여 ncurses를 빌드 환경에 추가해주어야한다. 이는 단순히 빌드 옵션에 -lncurses를 추가하는 것으로 가능하다. 이 링크의 글을 참고하였다.

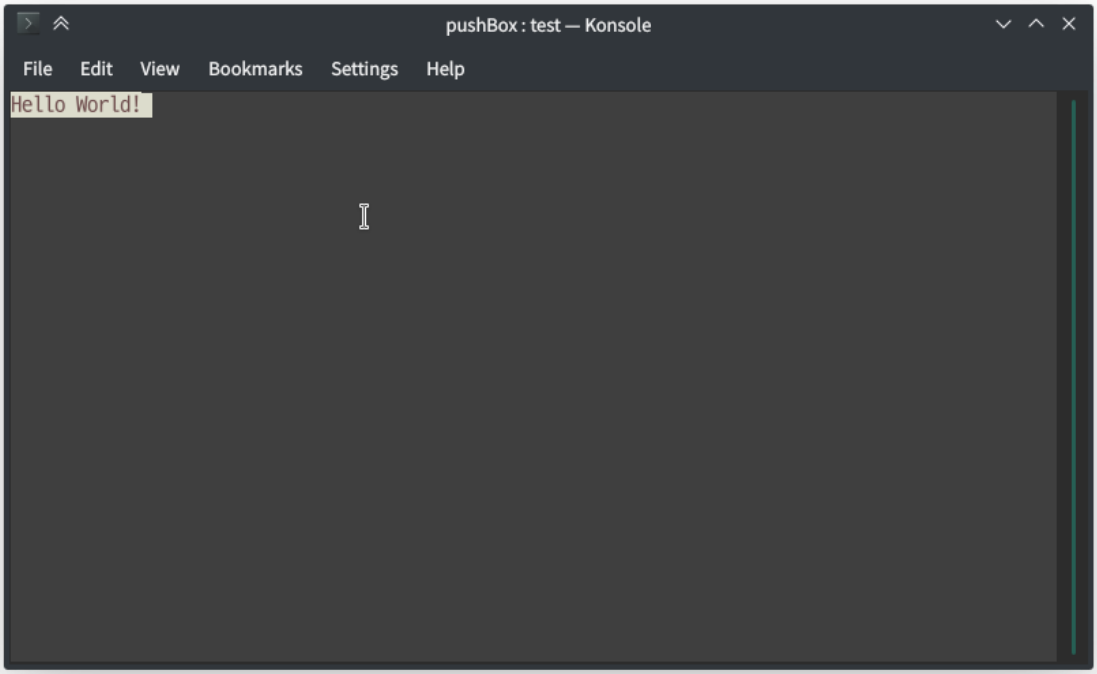


ncurses 기본 알아보기

창 띄우기

주어진 예제를 참고하여 터미널 창에 Hello, World!를 출력했다.

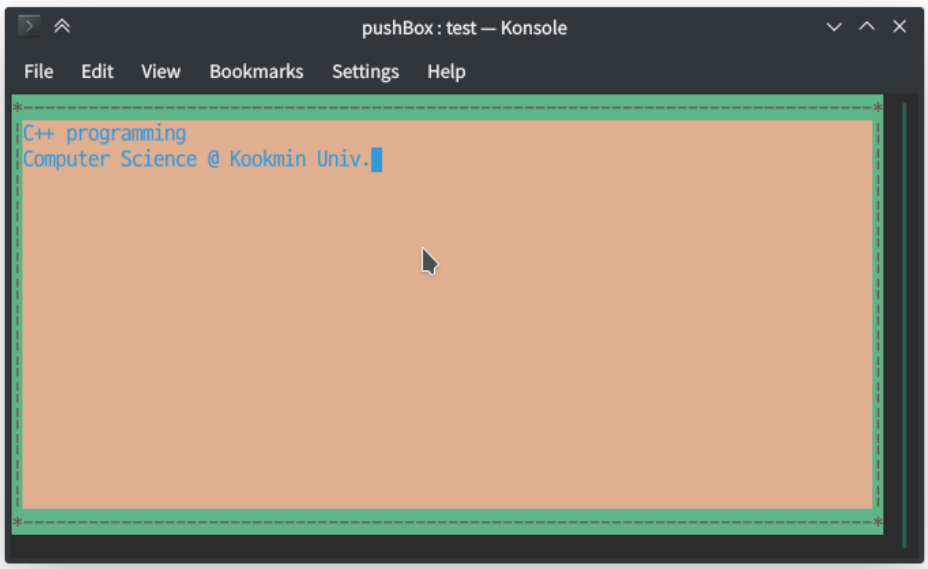
```
#include <ncurses.h>
int main() {
    initscr();
    start_color();
    init_pair(1, COLOR_RED, COLOR_WHITE);

    attron(COLOR_PAIR(1));
    printw("Hello World!");
    attroff(COLOR_PAIR(1));

    refresh();
    getch();
    endwin();

    return 0;
}
```



- 헤더파일은 ncurses.h를 불러온다.
- initscr() -> initiate screen
- printw(const *chr)은 0, 0 위치에 글자를 출력한다(고 선언한다).
- refresh를 통해 위에 선언되었던 명령들을 실제로 실행한다.
- getch() -> get Character. 명령 없이 창이 닫히지 않도록 실행
- endwin() -> end Window

유니코드 및 색상 설정

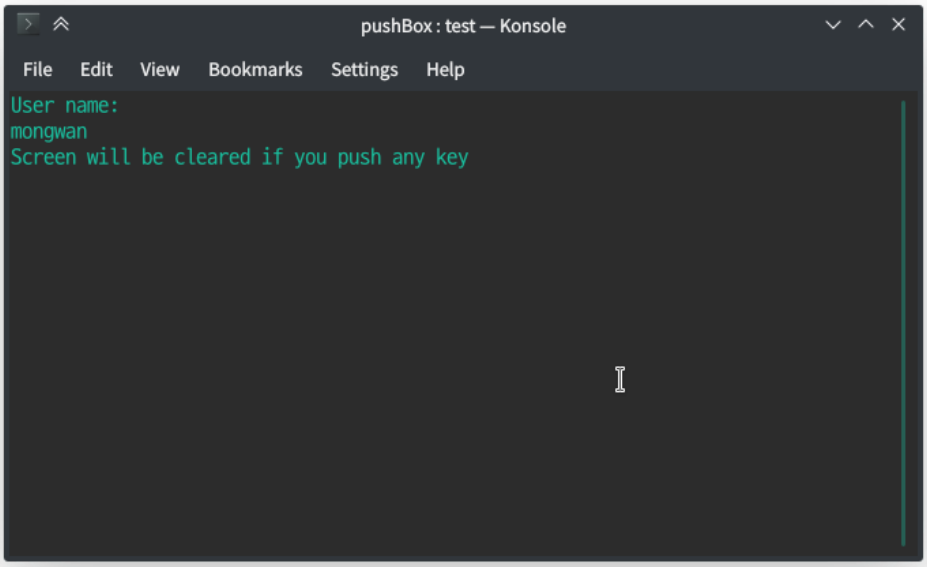
```
#include <ncurses.h>
#include <clocale>
int main() {
    setlocale(LC_ALL, "");

    initscr();
    start_color();
    init_pair(1, COLOR_BLUE, COLOR_YELLOW);
    init_pair(2, COLOR_RED, COLOR_GREEN);

    bkgd(COLOR_PAIR(1));
    attron(COLOR_PAIR(1));
    mvprintw(1, 1, "C++ programming");
    mvprintw(2, 1, "Computer Science @ Kookmin Univ.");
    attroff(COLOR_PAIR(1));

    attron(COLOR_PAIR(2));
    border('|', '|', '-', '-', '*', '*', '*', '*');
    attroff(COLOR_PAIR(2));

    refresh();
    getch();
    endwin();

    return 0;
}
```



- start_color()은 color 설정 전에 실행되어야 한다.
- init_pair(n, font-color, background-color) : 글자 색상과 배경 색상 순으로 n번 색상 Pair을 정의한다.
- bkgd -> background, 인자가 문자일때는 그 문자로 배경 설정
- attron -> attribute on / attroff -> attribute off 두 명령 사이의 명령들에 색상 속성을 적용
- border : 좌우상하 좌상단 우상단 좌하단 우하단 순으로 화면의 테두리 설정

사용자 입력받기

```
#include <ncurses.h>
#include <clocale>

int main() {
    setlocale(LC_ALL, "");
    char userName[8];

    initscr();

    keypad(stdscr, TRUE);
    curs_set(0);
    noecho();

    printw("User name: ");
    scanw("%s", userName);

    printw("%s\n", userName);
    printw("Screen will be cleared if you push any key");
    refresh();

    getch();
    clear();
    refresh();

    getch();
    endwin();
    return 0;
}
```



- keypad(*win, bool) : 키보드 특수 키 입력을 가능하게 설정해준다. 첫 인자로 윈도우 포인터를, 두번째 인자로 사용 가능, 불가능을 bool로 받는다.

- curs_set(int visibility) 커서 설정, 0은 안보임, 1은 작은 커서, 2는 큰 커서이나 나의 환경에서는 1과 2의 차이가 없었다.
- noecho() 입력한 문자를 표시하지 않는다.
- scanw(const char *fmt) scanf와 유사

윈도우 관리

```
#include <ncurses.h>
#include <clocale>

int main() {
    setlocale(LC_ALL, "");

    WINDOW *win1;

    initscr();
    resize_term(25, 25);
    start_color();
    init_pair(1, COLOR_WHITE, COLOR_RED);

    border('*','*','*','*','*','*','*','*');
    mvprintw(1, 1, "A default window");
    refresh();
    getch();

    win1 = newwin(20, 20, 3, 3);
    wbkgd(win1, COLOR_PAIR(1));
    wattron(win1, COLOR_PAIR(1));
    mvwprintw(win1, 1, 1, "A new window");
    wborder(win1, '@', '@', '@', '@', '@', '@', '@', '@');
    wrefresh(win1);

    getch();
    delwin(win1);
    endwin();

    return 0;
}
```

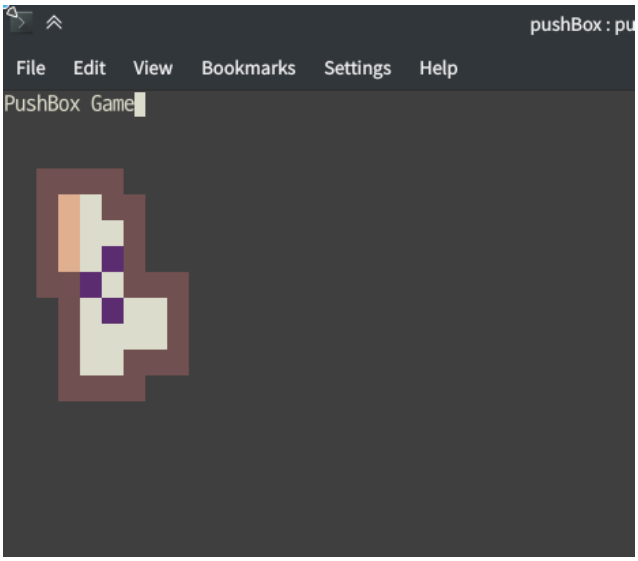
- 함수 내에 'w'가 들어있는 "wbkgd", "wattron" 등은 첫 인자로 윈도우 포인터를 받고 나머지는 bkgd, attron 처럼 w를 제외한 함수와 같다.
- newwin(lines, cols, y, x) : 윈도우 크기와 위치를 지정해서 윈도우를 표시한다.

1단계 개발설계

2차원 배열을 선언하고, 그 배열 크기의 window를 만든다. 배열의 각 값을 통해 ncurses의 내장 함수를 사용해서 화면을 표시한다.

```
#include <ncurses.h>
#include <clocale>

enum {P_DEFAULT = 1, P_WALL, P_BOX, P_GOAL, P_OUTSIDE};
// Enum starts with 1 because We can't assign 0 to COLOR_PAIR palette.

int main() {
    setlocale(LC_ALL, ""); // to use unicode
    keypad(stdscr, TRUE);
    curs_set(0);
    noecho();

    int curr_arr[9][7] = {
            {1, 1, 1, 1, 4, 4, 4},
```

```
            {1, 3, 0, 1, 1, 4, 4},
            {1, 3, 0, 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}
    };
    int arr1_height = 9;
    int arr1_width = 7;

    WINDOW *game_win;

    initscr();
    resize_term(30, 25);

    start_color();

    init_pair(P_DEFAULT, COLOR_WHITE, COLOR_WHITE);
    init_pair(P_WALL, COLOR_RED, COLOR_RED);
    init_pair(P_BOX, COLOR_MAGENTA, COLOR_MAGENTA);
    init_pair(P_GOAL, COLOR_YELLOW, COLOR_YELLOW);
    init_pair(P_OUTSIDE, COLOR_BLACK, COLOR_BLACK);

    mvprintw(0, 0, "PushBox Game");
    refresh();

    game_win = newwin(arr1_height, arr1_width*2, 3, 3);
    wbkgd(game_win, COLOR_PAIR(P_DEFAULT));
    for(int y=0; y < arr1_height; y++) {
        for(int x=0; x < arr1_width*2; x++) {
            int n = curr_arr[y][x];
            wattron(game_win, COLOR_PAIR(n+1));
            char *c = new char;
            sprintf(c, "%d", n);
            mvwprintw(game_win, y, (x*2), c);
            mvwprintw(game_win, y, (x*2)+1, c);
            wattroff(game_win, COLOR_PAIR(n+1));
            delete c;
        }
    }
    wrefresh(game_win);
    refresh();

    getch();
    delwin(game_win);
    endwin();

    return 0;
}
```



위 코드 실행 시 위와 같은 창이 표시된다. 색상을 설정하기 위해 여러가지 시도를 했지만 (init_color()를 통해 다른 색상을 지정하는 등) 현재 나의 터미널 환경에서는 변경이 반영되지 않아 색상 관련 코드는 최소화하였다. 열거형을 통해 COLOR_PAIR간의 구분이 더 잘 가도록 코드를 작성했고, 이 중 for문을 통해 각 칸의 정보를 색으로 표현하였다. 각 칸은 문자 2개로 이루어져 있는데, 나의 터미널 환경의 글꼴이 세로로 긴 고정폭 글꼴이기 때문에 타일 모양을 구현하기 위해서 문자 두개로 한 칸을 표현하였다.

milestone2.md

2단계

2단계 개발설계

사용자가 키보드 방향키를 입력하면, 캐릭터를 기준으로 그 방향의 타일에 무엇이 있는지 확인한다.

1. 벽이라면, 이동할 수 없으므로 무시한다.
2. 상자라면, 그 방향의 다음 타일을 확인한다.
    i. 그 다음 타일이 벽 또는 상자라면 이동할 수 없으므로 무시한다.
    ii. 그 다음 타일이 빈 공간 또는 목표 지점이라면, 상자를 그 빈 공간에 옮기고 직전의 상자 위치로 캐릭터를 옮긴다.
3. 빈 공간 또는 목표 지점이라면, 캐릭터를 그 타일로 옮긴다.

```
#include <ncurses.h>
#include <clocale>
#include <memory.h>

enum Direction {LEFT, RIGHT, UP, DOWN};
enum Tile {DEFAULT, WALL, BOX, GOAL, OUTSIDE, CURR};
enum Pair {P_DEFAULT = 1, P_WALL, P_BOX, P_GOAL, P_OUTSIDE, P_CURR};
// Enum starts with 1 because We can't assign 0 to COLOR_PAIR palette.

struct Pos {
    int y;
    int x;
    Direction heading;
};

Pos chk_pos(Direction dir, Pos curr);
void refr_game(WINDOW* w, Pos curr);

int origin_arr1[9][7] = {
        {1, 1, 1, 1, 4, 4, 4},
        {1, 3, 0, 1, 1, 4, 4},
        {1, 3, 0, 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}

};

int arr1_height = 9;
int arr1_width = 7;
int curr_arr[9][7];
int main() {
    memcpy(curr_arr, origin_arr1, sizeof(origin_arr1));

    setlocale(LC_ALL, ""); // to use unicode

    Pos curr = {2, 2, LEFT}; // y, x, Dir
```

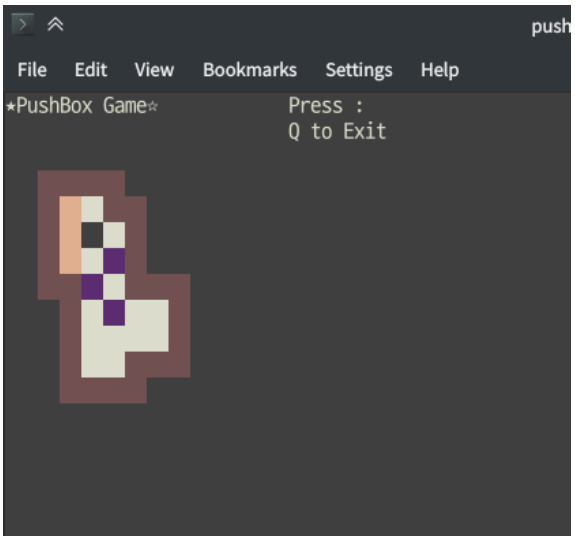
```
    WINDOW *game_win;

    initscr();
    keypad(stdscr, TRUE);
    curs_set(0);
    noecho();
    resize_term(40, 40);

    start_color();
    init_pair(P_DEFAULT, COLOR_WHITE, COLOR_WHITE);
    init_pair(P_WALL, COLOR_RED, COLOR_RED);
    init_pair(P_BOX, COLOR_MAGENTA, COLOR_MAGENTA);
    init_pair(P_GOAL, COLOR_YELLOW, COLOR_YELLOW);
    init_pair(P_OUTSIDE, COLOR_BLACK, COLOR_BLACK);
    init_pair(P_CURR, COLOR_WHITE, COLOR_GREEN);

    mvprintw(0, 0, "★PushBox Game☆"); // length = 14
    mvprintw(0, 26, "Press :"); // length = 7
    mvprintw(1, 26, "Q to Exit"); // length = 9
    refresh();

    game_win = newwin(arr1_height, arr1_width*2, 3, 3);
    wbkgd(game_win, COLOR_PAIR(DEFAULT));

    refr_game(game_win, curr);

    int chr = 0;
    Pos chk = curr;
    while (chr != 'q' && chr != 'Q') {
        chr = getch();

        if (chr == KEY_LEFT) chk = chk_pos(LEFT, curr);
        else if (chr == KEY_RIGHT) chk = chk_pos(RIGHT, curr);
        else if (chr == KEY_UP) chk = chk_pos(UP, curr);
        else if (chr == KEY_DOWN) chk = chk_pos(DOWN, curr);

        int chk_num = curr_arr[chk.y][chk.x];

        if (chk_num == WALL) continue; // heading to wall
        else if (chk_num == DEFAULT || chk_num == GOAL) {
            curr.y = chk.y;
            curr.x = chk.x;
            refr_game(game_win, curr);
        }
        else if (chk_num == BOX) {
            Pos alt_chk = chk_pos(chk.heading, chk);
            int alt_chk_num = curr_arr[alt_chk.y][alt_chk.x];

            if (alt_chk_num == WALL || alt_chk_num == BOX) continue;
            else if (alt_chk_num == DEFAULT || alt_chk_num == GOAL) {
                curr_arr[alt_chk.y][alt_chk.x] = BOX;
                curr_arr[chk.y][chk.x] = DEFAULT;
                curr.y = chk.y;
                curr.x = chk.x;
                refr_game(game_win, curr);
            }
        }
    }

    delwin(game_win);
    endwin();

    return 0;
}
```

```
Pos chk_pos(Direction dir, Pos curr) {
    Pos chk = {0, 0, LEFT};
    switch (dir) {
        case LEFT:
            chk.heading = LEFT;
            chk.y = curr.y;
            chk.x = curr.x-1;
            break;
        case RIGHT:
            chk.heading = RIGHT;
            chk.y = curr.y;
            chk.x = curr.x+1;
            break;
        case UP:
            chk.heading = UP;
            chk.y = curr.y-1;
            chk.x = curr.x;
            break;
        case DOWN:
            chk.heading = DOWN;
            chk.y = curr.y+1;
            chk.x = curr.x;
            break;
    }

    return chk;
}

void refr_game(WINDOW *w, Pos curr) {
    for(int y=0; y < arr1_height; y++) {
        for(int x=0; x < arr1_width*2; x++) {
            int n = curr_arr[y][x];
            wattron(w, COLOR_PAIR(n+1));
            char *c = new char;
            sprintf(c, "%d", n);
            mvwprintw(w, y, (x*2), c);
            mvwprintw(w, y, (x*2)+1, c);
            wattroff(w, COLOR_PAIR(n+1));
            delete c;
        }
    }

    wattron(w, COLOR_PAIR(CURR));
    mvwprintw(w, curr.y, (curr.x*2), "C");
    mvwprintw(w, curr.y, (curr.x*2)+1, "C");
    wattron(w, COLOR_PAIR(CURR));

    wrefresh(w);
    refresh();
}
```

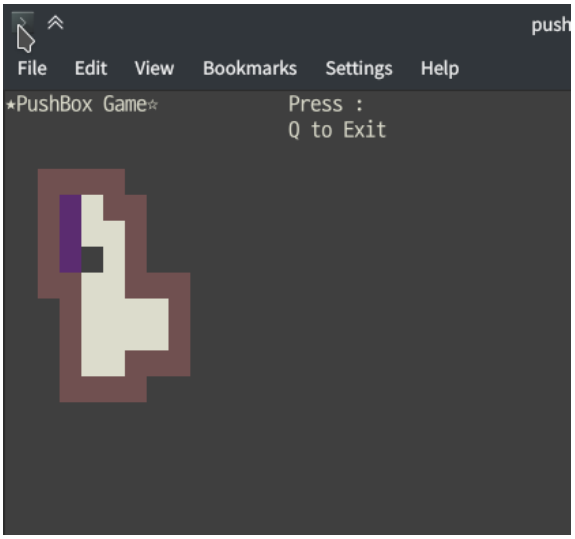
개요

- Direction 열거형은 현재 바라보고 있는 방향을 더 이해하기 쉽도록 만들었다.
- Pair과 Tile을 같은 열거형으로 사용하려고 했으나, 색상 Pair은 index가 1부터 시작해야 했고(COLOR_PAIR가 0을 넣었을때 정상적으로 작동하지 않았다.) Tile은 0부터 시작해야 했기 때문에 구분했다.
- 구조체 Pos를 만들었다. Pos는 (int) y 좌표, (int) x 좌표, 현재 바라보고 있는 방향인 열거형 Direction을 포함한다. 현재 바라보는 방향은 박스를 밀 때 그 다음 칸이 이동 가능한지 확인할 때 쓰인다.
- chk_pos 함수는 이동할 방향과, 현재 위치 좌표를 받아 한칸 이동한 그 좌표를 반환한다. 반복 사용이 많은 코드기 때문에 함수로 구분했다.

- refr_game 함수는 게임 윈도우와 현재 위치를 받아 게임 윈도우를 refresh한다. 게임이 시작될때, 이동 할 때 호출된다.
- 현재 바뀐 상태를 저장하는 배열과 기존 배열을 구분했다. 추후에 다시 시작 기능 등을 구현하기 위해서 필요하다. 배열 복사를 위해 memcpy를 사용했고, 이는 memory.h 헤더에 저장되어있다. 구현의 용이함을 위해 전역변수로 선언했다.
- int chr은 입력받을 문자를 저장하는 변수이다.
- Pos chk는 chk_pos의 반환값을 저장한다. 그 체크중인 타일이 이동 가능한지 파악하는 코드는 main()에 있다.
- if~else if 구문은 위의 개발설계를 그대로 반영했다. 이 이동 과정은 (현재는) Q가 입력되기 전까지 가능하다.



초기 상태



게임을 클리어한 상태

현재 문제점

- 94번 line에서, GOAL 위에 상자가 올라갔다가, 다시 상자가 사라지면 GOAL로 남아있어야 하는데 DEFAULT로 초기화된다. 단순히 DEFAULT를 입력하게끔 코드를 작성했기 때문인데, 이는 3차 과제를 작성하면서 배열의 원형을 불러오는 방법으로 수정할 것이다.
- MAP을 하나만 사용할 수 있게끔 코드가 작성되어 있기 때문에 이를 수정하려면 배열을 참조하는 전체 구조를 조금씩 바꿀 필요가 있다. 이도 3차 과제에서 수정할 예정이다.

- 57번 line에서, 분명 배경을 COLOR_GREEN으로 설정했지만 적용되지 않았는데, 색상 적용이 확실히 되지 않는 것이 현재 나의 터미널 환경의 문제인지 코드의 문제인지 파악하고 고쳐야 한다.

milestone3.md

3단계

3단계 개발설계

현재 한단계만 실행 가능한 구조를 개선하기 위해 자료가 저장된 구조를 변경하고, 각 단계를 이터레이터로 돌릴수 있게끔 수정한다. 모든 GOAL 구역이 BOX로 채워지면 다음 단계로 넘어가는 로직을 추가한다. 현재 상태를 표시하는 WINDOW인 info_win을 만든다. info_win에는 현재 스테이지 번호, 현재 스테이지에서 움직인 횟수, 현재 스테이지에서 박스를 민 횟수를 표시한다. Q를 눌렀을때 터미널이 바로 종료되는 것이 아닌, 종료 한다는 메시지를 띄우고 종료하도록 수정한다.

1차 수정

- info_win 구현

```
#include <ncurses.h>
#include <clocale>
#include <memory.h>
#include "stages.h"

enum Direction {LEFT, RIGHT, UP, DOWN};
enum Tile {DEFAULT, WALL, BOX, GOAL, OUTSIDE, CURR};
enum Pair {P_DEFAULT = 1, P_WALL, P_BOX, P_GOAL, P_OUTSIDE, P_CURR};
// Enum starts with 1 because We can't assign 0 to COLOR_PAIR palette.
struct Pos { int y; int x; Direction heading; };

Pos chk_pos(Direction dir, Pos curr);
void refr_game(WINDOW* w, Pos curr);
void refr_info(WINDOW* w);

int heights[STAGEN] = {7, 9, 6, 7, 8};
int widths[STAGEN] = {6, 7, 8, 7, 10};

int arr1_height = 9;
int arr1_width = 7;

int curr_arr[9][7];

int step = 0;
int push = 0;
int level = 1;
bool cleared = false;
int main() {
    memcpy(curr_arr, stage2, sizeof(stage2));

    setlocale(LC_ALL, ""); // to use unicode

    Pos curr = {2, 2, LEFT}; // y, x, Dir
    WINDOW *game_win;
    WINDOW *info_win;

    initscr();
    keypad(stdscr, TRUE);
    curs_set(0);
    noecho();
    resize_term(40, 40);
```

```
start_color();
init_pair(P_DEFAULT, COLOR_WHITE, COLOR_WHITE);
init_pair(P_WALL, COLOR_RED, COLOR_RED);
init_pair(P_BOX, COLOR_MAGENTA, COLOR_MAGENTA);
init_pair(P_GOAL, COLOR_YELLOW, COLOR_YELLOW);
init_pair(P_OUTSIDE, COLOR_BLACK, COLOR_BLACK);
init_pair(P_CURR, COLOR_WHITE, COLOR_GREEN);

mvprintw(0, 0, "★PushBox Game☆"); // length = 14
mvprintw(0, 26, "Press :"); // length = 7
mvprintw(1, 26, "Q to Exit"); // length = 9
refresh();

game_win = newwin(arr1_height, arr1_width*2, 3, 3);
wbkgd(game_win, COLOR_PAIR(DEFAULT));
refr_game(game_win, curr);

info_win = newwin(20, 15, 6, 26);
wbkgd(game_win, COLOR_PAIR(DEFAULT));
mvwprintw(info_win, 0, 0, "STEP : ");
mvwprintw(info_win, 1, 0, "PUSH : ");
mvwprintw(info_win, 2, 0, "LEVEL");
refr_info(info_win);

refresh();

int chr = 0;
Pos chk = curr;
while (chr != 'q' && chr != 'Q') {
    chr = getch();

    if (chr == KEY_LEFT) chk = chk_pos(LEFT, curr);
    else if (chr == KEY_RIGHT) chk = chk_pos(RIGHT, curr);
    else if (chr == KEY_UP) chk = chk_pos(UP, curr);
    else if (chr == KEY_DOWN) chk = chk_pos(DOWN, curr);
    else continue;

    int chk_num = curr_arr[chk.y][chk.x];

    if (chk_num == WALL) continue; // heading to wall
    else if (chk_num == DEFAULT || chk_num == GOAL) {
        curr.y = chk.y;
        curr.x = chk.x;
        refr_game(game_win, curr);
        step += 1;
        refr_info(info_win);
    }
    else if (chk_num == BOX) {
        Pos alt_chk = chk_pos(chk.heading, chk);
        int alt_chk_num = curr_arr[alt_chk.y][alt_chk.x];

        if (alt_chk_num == WALL || alt_chk_num == BOX) continue;
        else if (alt_chk_num == DEFAULT || alt_chk_num == GOAL) {
            curr_arr[alt_chk.y][alt_chk.x] = BOX;
            curr_arr[chk.y][chk.x] = DEFAULT;
            curr.y = chk.y;
            curr.x = chk.x;
            refr_game(game_win, curr);
            step += 1;
            refr_info(info_win);
        }
    }
}
```

```
    mvprintw(1, 0, cleared? "You win!" : "Quit game");
    mvprintw(2, 0, "Press any key to quit game");

    getch();
    delwin(game_win);
    endwin();

    return 0;
}

Pos chk_pos(Direction dir, Pos curr) {
    Pos chk = {0, 0, LEFT};
    switch (dir) {
        case LEFT:
            chk.heading = LEFT;
            chk.y = curr.y;
            chk.x = curr.x-1;
            break;
        case RIGHT:
            chk.heading = RIGHT;
            chk.y = curr.y;
            chk.x = curr.x+1;
            break;
        case UP:
            chk.heading = UP;
            chk.y = curr.y-1;
            chk.x = curr.x;
            break;
        case DOWN:
            chk.heading = DOWN;
            chk.y = curr.y+1;
            chk.x = curr.x;
            break;
    }

    return chk;
}

void refr_game(WINDOW *w, Pos curr) {
    for(int y=0; y < arr1_height; y++) {
        for(int x=0; x < arr1_width*2; x++) {
            int n = curr_arr[y][x];
            wattron(w, COLOR_PAIR(n+1));
            char *c = new char;
            sprintf(c, "%d", n);
            mvwprintw(w, y, (x*2), c);
            mvwprintw(w, y, (x*2)+1, c);
            wattroff(w, COLOR_PAIR(n+1));
            delete c;
        }
    }

    wattron(w, COLOR_PAIR(CURR));
    mvwprintw(w, curr.y, (curr.x*2), "C");
    mvwprintw(w, curr.y, (curr.x*2)+1, "C");
    wattron(w, COLOR_PAIR(CURR));

    wrefresh(w);
    refresh();
}

void refr_info(WINDOW *w) {
    char *c = new char;
    sprintf(c, "%d", step);
```

```
    mvwprintw(w, 0, 7, c);
    sprintf(c, "%d", push);
    mvwprintw(w, 1, 7, c);
    sprintf(c, "%d", level);
    mvwprintw(w, 2, 6, c);
    delete c;
    wrefresh(w);
}
```

- WINDOW* info_win을 만들어 기초적인 설정 후, step, push 변수를 적절한 상황에서만 증가시킨다.

2차 수정

- 현재 상태 (curr_status)를 std::vector< std::vector >, 즉 2차원 int형 벡터형으로 바꿨다. 이 변경을 통해 다양한 크기의 stage를 현재 상태로써 만들수 있다.
- 1차 수정 이후로 바뀐 부분만 작성했다.

```
#include <vector> // vector 사용을 위해 추가

void refr_game(WINDOW* w, Pos curr, std::vector< std::vector<int> > curr_status);
// 현재 상태 변수를 main()의 지역 변수로 만들기 위해 함수에 인자 추가

std::vector< std::vector<int> > curr_status;
for (int i = 0; i < arr1_height; i++) {
    curr_status.push_back(std::vector<int>());
    for (int j = 0; j < arr1_width; j++)
        curr_status[i].push_back(stage1[i][j]);
} // memcpy 대신, for문으로 직접 복사. (std::copy를 쓰려 했으나, 사용하지 못했다)

// 이하 단순 변수 이름 변경
```

최종 수정

- 스테이지를 정수형 배열로 각각 선언했더니, 스테이지를 넘어가는 것을 구현하기 힘들었다. 일단은, 각 배열의 메모리 주소를 배열로 가지고 있는 int* 형 배열 stagep를 만들었다. 각 배열을 (int*)를 통해 명시적 형변환하여 넣었다. 이 때문에 main 코드에서 2차원적으로 값을 찾던 방식을 사용하지 못하고, [y * width + x] 로 index를 찾는 방식을 변경했다.

in stages.h

```
int* stagep[STAGEN] = {(int*)stage1, (int*)stage2, (int*)stage3, (int*)stage4, (int*)stage5};
```

in main.cpp

```
curr_status[i].push_back(stagep[level][i * widths[level]+j]);
// 등 모든 배열 참조 형식 변경
```

- 각 색상이 어떤 타일을 뜻하는지 알려주는 정보창을 추가했다.

```
attron(COLOR_PAIR(P_BOX));
mvprintw(4, 26, "  ");
attroff(COLOR_PAIR(P_BOX));
mvprintw(4, 28, " is BOX");
attron(COLOR_PAIR(P_WALL));
mvprintw(5, 26, "  ");
attroff(COLOR_PAIR(P_WALL));
```

```
mvprintw(5, 28, " is WALL");
attron(COLOR_PAIR(P_GOAL));
mvprintw(6, 26, "  ");
attroff(COLOR_PAIR(P_GOAL));
mvprintw(6, 28, " is GOAL");
attron(COLOR_PAIR(P_CURR));
mvprintw(7, 26, "  ");
attroff(COLOR_PAIR(P_CURR));
mvprintw(7, 28, " is Character");
```

- 현재 GOAL에 들어오지 못한 상자의 개수를 나타내는 정수형 변수를 만들어, 이 변수가 0일 때 다음 스테이지로 넘어가는 로직을 추가했다.

```
if (not_goaled_box <= 0) { // when stage clear
    curr_status.clear();

    wattron(game_win, COLOR_PAIR(P_OUTSIDE));
    for(int y=0; y < heights[level]+1; y++) {
        for (int x = 0; x < widths[level]+1; x++)
            mvprintw(y + 2, (x * 2) + 2, "    ");
    }
    wattroff(game_win, COLOR_PAIR(P_OUTSIDE));
    refresh();

    level += 1;
    if (level >= STAGEN) {
        cleared = true;
        break;
    }
    step = 0;
    push = 0;

    game_win = newwin(heights[level], widths[level]*2, 3, 3);

    for (int i = 0; i < heights[level]; i++) {
        curr_status.push_back(std::vector<int>());
        for (int j = 0; j < widths[level]; j++)
            curr_status[i].push_back(stagep[level][i * widths[level]+j]);
    }

    not_goaled_box = boxn[level];
    curr.y = init_curr[level][0];
    curr.x = init_curr[level][1];
    curr.heading = LEFT; // y, x, Dir
    refr_game(game_win, curr, curr_status);
    refr_info(info_win);
}
```

- R을 누르면, 현재 스테이지가 0으로 초기화되는 기능을 넣었다.

```
else if (chr == 'r' || chr == 'R') {
    step = 0;
    push = 0;
    not_goaled_box = boxn[level];

    curr_status.clear();
    for (int i = 0; i < heights[level]; i++) {
        curr_status.push_back(std::vector<int>());
        for (int j = 0; j < widths[level]; j++)
            curr_status[i].push_back(stagep[level][i * widths[level]+j]);
    }
```

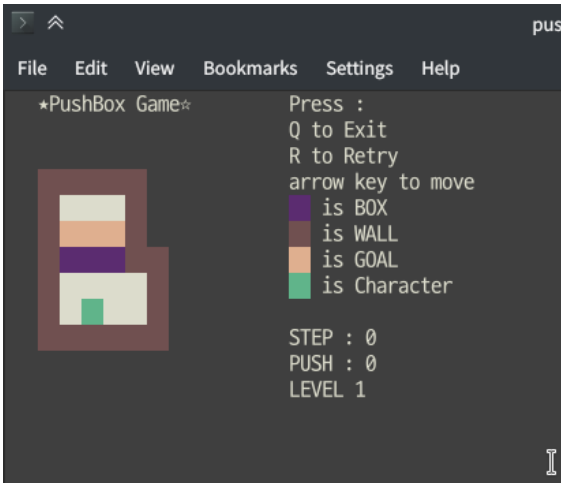
```
            curr.y = init_curr[level][0];
            curr.x = init_curr[level][1];
            curr.heading = LEFT; // y, x, Dir
            refr_game(game_win, curr, curr_status);
            refr_info(info_win);
            continue;
        }
```

- **상자가 지나가면 GOAL이 DEFAULT로 변경되는 오류를 수정했다**. curr_status[chk.y][chk.x] = (stagep[level][chk.y*widths[level]+chk.x] == GOAL) ? stagep[level][chk.y*widths[level]+chk.x] : DEFAULT;
- **COLOR_GREEN이 적용되지 않는 것은 단순히 변수를 잘못 넣어 있던 오류였고, 수정했다**. wattron(w, COLOR_PAIR(P_CURR));

최종 보고서.md

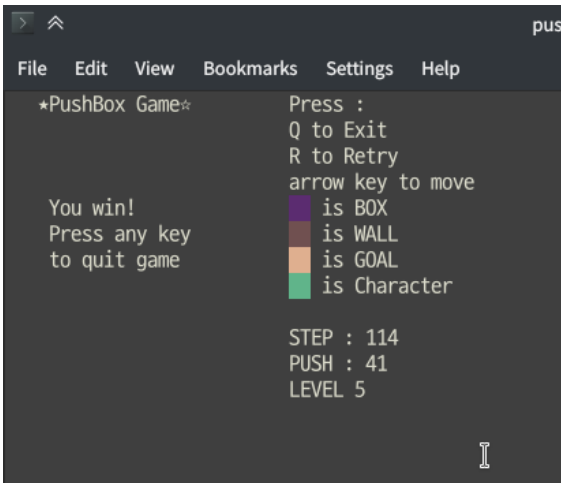
최종

구현 이미지



초기 상태

LEVEL 2를 클리어하기 직전의 상태



모든 스테이지를 클리어한 상태

최종 코드

stages.h

```
#ifndef PUSHBOX_STAGES_H
#define PUSHBOX_STAGES_H

const int STAGEN = 5;
const int heights[STAGEN] = {7, 9, 6, 8, 8};
const int widths[STAGEN] = {6, 7, 8, 7, 10};
const int init_curr[STAGEN][2] = {
        {5, 2},
        {2, 2},
        {2, 6},
        {6, 3},
        {4, 6}
};
const int boxn[STAGEN] = {3, 3, 3, 6, 3};
int stage1[7][6] = {
        {1, 1, 1, 1, 1, 4},
        {1, 0, 0, 0, 1, 4},
        {1, 3, 3, 3, 1, 4},
        {1, 2, 2, 2, 1, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 0, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1}
};

int stage2[9][7] = {
        {1, 1, 1, 1, 4, 4, 4},
        {1, 3, 0, 1, 1, 4, 4},
        {1, 3, 0, 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}
};

int stage3[6][8] = {
```

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

int stage4[8][7] = {
        {1, 1, 1, 1, 1, 1, 1},
        {1, 0, 0, 0, 0, 0, 1},
        {1, 0, 3, 2, 3, 0, 1},
        {1, 0, 2, 3, 2, 0, 1},
        {1, 0, 3, 2, 3, 0, 1},
        {1, 0, 2, 3, 2, 0, 1},
        {1, 0, 0, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1, 1}
};

int stage5[8][10] = {
        {4, 1, 1, 1, 1, 4, 4, 4, 4, 4},
        {4, 1, 0, 0, 1, 1, 1, 1, 4, 4},
        {4, 1, 0, 0, 0, 0, 0, 1, 1, 4},
        {1, 1, 0, 1, 1, 0, 0, 0, 1, 4},
        {1, 3, 0, 3, 1, 0, 0, 2, 1, 1},
        {1, 0, 0, 0, 1, 0, 2, 2, 0, 1},
        {1, 0, 0, 3, 1, 0, 0, 0, 0, 1},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
};

int* stagep[STAGEN] = {(int*)stage1, (int*)stage2, (int*)stage3, (int*)stage4, (int*)stage5};
#endif //PUSHBOX_STAGES_H
```

main.cpp

```
#include <ncurses.h>
#include <clocale>
#include <vector>
#include "stages.h"

enum Direction {LEFT, RIGHT, UP, DOWN};
enum Tile {DEFAULT, WALL, BOX, GOAL, CURR = 5};
enum Pair {P_DEFAULT = 1, P_WALL, P_BOX, P_GOAL, P_OUTSIDE, P_CURR};
// Enum starts with 1 because We can't assign 0 to COLOR_PAIR palette.
struct Pos { int y; int x; Direction heading; };

Pos chk_pos(Direction dir, Pos curr);
void refr_game(WINDOW* w, Pos curr, std::vector< std::vector<int> > curr_status);
void refr_info(WINDOW* w);

int step = 0;
int push = 0;
int level = 0;
bool cleared = false;

int main() {
    setlocale(LC_ALL, ""); // to use unicode

    WINDOW *game_win;
    WINDOW *info_win;

    initscr();
    keypad(stdscr, TRUE);
```

```
curs_set(0);
noecho();
resize_term(40, 45);

start_color();
init_pair(P_DEFAULT, COLOR_WHITE, COLOR_WHITE);
init_pair(P_WALL, COLOR_RED, COLOR_RED);
init_pair(P_BOX, COLOR_MAGENTA, COLOR_MAGENTA);
init_pair(P_GOAL, COLOR_YELLOW, COLOR_YELLOW);
init_pair(P_OUTSIDE, COLOR_BLACK, COLOR_BLACK);
init_pair(P_CURR, COLOR_GREEN, COLOR_GREEN);

mvprintw(0, 0, "   ★PushBox Game☆"); // length = 17
mvprintw(0, 26, "Press :"); // length = 7
mvprintw(1, 26, "Q to Exit"); // length = 9
mvprintw(2, 26, "R to Retry");
mvprintw(3, 26, "arrow key to move");

attron(COLOR_PAIR(P_BOX));
mvprintw(4, 26, "  ");
attroff(COLOR_PAIR(P_BOX));
mvprintw(4, 28, " is BOX");
attron(COLOR_PAIR(P_WALL));
mvprintw(5, 26, "  ");
attroff(COLOR_PAIR(P_WALL));
mvprintw(5, 28, " is WALL");
attron(COLOR_PAIR(P_GOAL));
mvprintw(6, 26, "  ");
attroff(COLOR_PAIR(P_GOAL));
mvprintw(6, 28, " is GOAL");
attron(COLOR_PAIR(P_CURR));
mvprintw(7, 26, "  ");
attroff(COLOR_PAIR(P_CURR));
mvprintw(7, 28, " is Character");

refresh();

std::vector< std::vector<int> > curr_status;
for (int i = 0; i < heights[level]; i++) {
    curr_status.push_back(std::vector<int>());
    for (int j = 0; j < widths[level]; j++)
        curr_status[i].push_back(stagep[level][i * widths[level]+j]);
}
Pos curr = {init_curr[level][0], init_curr[level][1], LEFT}; // y, x, Dir

game_win = newwin(heights[level], widths[level]*2, 3, 3);
wbkgd(game_win, COLOR_PAIR(DEFAULT));
refr_game(game_win, curr, curr_status);

info_win = newwin(20, 15, 9, 26);
wbkgd(game_win, COLOR_PAIR(DEFAULT));
mvwprintw(info_win, 0, 0, "STEP : ");
mvwprintw(info_win, 1, 0, "PUSH : ");
mvwprintw(info_win, 2, 0, "LEVEL");
refr_info(info_win);

refresh();

int chr = 0;
Pos chk;
int not_goaled_box = boxn[level];
while (chr != 'q' && chr != 'Q') {
    chr = getch();
```

```
if (chr == KEY_LEFT) chk = chk_pos(LEFT, curr);
else if (chr == KEY_RIGHT) chk = chk_pos(RIGHT, curr);
else if (chr == KEY_UP) chk = chk_pos(UP, curr);
else if (chr == KEY_DOWN) chk = chk_pos(DOWN, curr);
else if (chr == 'r' || chr == 'R') {
    step = 0;
    push = 0;
    not_goaled_box = boxn[level];

    curr_status.clear();
    for (int i = 0; i < heights[level]; i++) {
        curr_status.push_back(std::vector<int>());
        for (int j = 0; j < widths[level]; j++)
            curr_status[i].push_back(stagep[level][i * widths[level]+j]);
    }

    curr.y = init_curr[level][0];
    curr.x = init_curr[level][1];
    curr.heading = LEFT; // y, x, Dir
    refr_game(game_win, curr, curr_status);
    refr_info(info_win);
    continue;
}
else continue;

int chk_num = curr_status[chk.y][chk.x];

if (chk_num == WALL) continue; // heading to wall
else if (chk_num == DEFAULT || chk_num == GOAL) {
    curr.y = chk.y;
    curr.x = chk.x;
    refr_game(game_win, curr, curr_status);
    step += 1;
    refr_info(info_win);
}
else if (chk_num == BOX) {
    Pos alt_chk = chk_pos(chk.heading, chk);
    int alt_chk_num = curr_status[alt_chk.y][alt_chk.x];

    if (alt_chk_num == WALL || alt_chk_num == BOX) continue;
    else if (alt_chk_num == DEFAULT || alt_chk_num == GOAL) {
        curr_status[alt_chk.y][alt_chk.x] = BOX;
        curr_status[chk.y][chk.x] = (stagep[level][chk.y*widths[level]+chk.x] == GOAL) ?
                                    stagep[level][chk.y*widths[level]+chk.x]
                                    : DEFAULT;
        curr.y = chk.y;
        curr.x = chk.x;
        refr_game(game_win, curr, curr_status);
        step += 1;
        push += 1;
        refr_info(info_win);

        if (stagep[level][chk.y*widths[level] + chk.x] == GOAL)
            not_goaled_box += 1;
        if (alt_chk_num == GOAL)
            not_goaled_box -= 1;
    }
}

if (not_goaled_box <= 0) { // when stage clear
    curr_status.clear();

    wattron(game_win, COLOR_PAIR(P_OUTSIDE));
    for(int y=0; y < heights[level]+1; y++) {
```

```
                for (int x = 0; x < widths[level]+1; x++)
                    mvprintw(y + 2, (x * 2) + 2, "    ");
            }
            wattroff(game_win, COLOR_PAIR(P_OUTSIDE));
            refresh();

            level += 1;
            if (level >= STAGEN) {
                cleared = true;
                break;
            }
            step = 0;
            push = 0;

            game_win = newwin(heights[level], widths[level]*2, 3, 3);

            for (int i = 0; i < heights[level]; i++) {
                curr_status.push_back(std::vector<int>());
                for (int j = 0; j < widths[level]; j++)
                    curr_status[i].push_back(stagep[level][i * widths[level]+j]);
            }

            not_goaled_box = boxn[level];
            curr.y = init_curr[level][0];
            curr.x = init_curr[level][1];
            curr.heading = LEFT; // y, x, Dir
            refr_game(game_win, curr, curr_status);
            refr_info(info_win);
        }
    }

    mvprintw(4, 4, cleared? "You win!" : "Quit game.");
    mvprintw(5, 4, "Press any key");
    mvprintw(6, 4, "to quit game");

    getch();
    delwin(game_win);
    endwin();

    return 0;
}

Pos chk_pos(Direction dir, Pos curr) {
    Pos chk = {0, 0, LEFT};
    switch (dir) {
        case LEFT:
            chk.heading = LEFT;
            chk.y = curr.y;
            chk.x = curr.x-1;
            break;
        case RIGHT:
            chk.heading = RIGHT;
            chk.y = curr.y;
            chk.x = curr.x+1;
            break;
        case UP:
            chk.heading = UP;
            chk.y = curr.y-1;
            chk.x = curr.x;
            break;
        case DOWN:
            chk.heading = DOWN;
            chk.y = curr.y+1;
            chk.x = curr.x;
```

```
            break;
    }

    return chk;
}

void refr_game(WINDOW *w, Pos curr, std::vector< std::vector<int> > curr_status) {
    for(int y=0; y < heights[level]; y++) {
        for(int x=0; x < widths[level]*2; x++) {
            int n = curr_status[y][x];
            wattron(w, COLOR_PAIR(n+1));
            char *c = new char;
            sprintf(c, "%d", n);
            mvwprintw(w, y, (x*2), c);
            mvwprintw(w, y, (x*2)+1, c);
            wattroff(w, COLOR_PAIR(n+1));
            delete c;
        }
    }

    wattron(w, COLOR_PAIR(P_CURR));
    mvwprintw(w, curr.y, (curr.x*2), "C");
    mvwprintw(w, curr.y, (curr.x*2)+1, "C");
    wattron(w, COLOR_PAIR(P_CURR));

    wrefresh(w);
    refresh();
}

void refr_info(WINDOW *w) {
    mvwprintw(w, 0, 7, "       ");
    mvwprintw(w, 1, 7, "       ");
    mvwprintw(w, 2, 6, "       ");
    char *c = new char;
    sprintf(c, "%d", step);
    mvwprintw(w, 0, 7, c);
    sprintf(c, "%d", push);
    mvwprintw(w, 1, 7, c);
    sprintf(c, "%d", level+1);
    mvwprintw(w, 2, 6, c);
    delete c;
    wrefresh(w);
}
```

구현된 사항

- 구현되어야 하는 3단계의 내용은 모두 구현하였다.
- 게임 진행 중 Q를 누르면, 게임을 종료한다는 알림 이후에 창을 종료한다.
- 게임 진행 중 R을 누르면, 그 스테이지의 첫 상태로 게임을 초기화한다.
- 게임의 간단한 설명을 창 오른쪽에 표시했다.

개선 가능한 사항

- 친구의 테스트 결과 어려움 없이 잘 클리어 했으나, 처음에 각 타일이 무엇을 의미하는 것인지 잘 파악하지 못했다. 게임을 시작할 때 게임 방법을 창으로 표시하거나, 좀 더 눈에 잘 보이는 형식으로 바꾸면 좋을 것 같다.
- main 함수 내에 화면 표시와 관련해서 중복된 코드가 다수 있다. 이를 함수로 분리한다면 코드가 더 간결해 질 것이다.