



## 2019 C++ Term Project - Push Box Game

### Report

팀명	FRIDAY	
팀원	학번	성명
	20181595	김주연
	20181597	김지민
	20171619	맹산하
팀장	맹산하	

# 목차

## 0. Push Box Game

## 1. 코드 설계

### 1.1. 1단계 코드 설계

### 1.2. 2단계 코드 설계

## 2. 코드 구현

### 2.1. 1단계 코드 구현

### 2.2. 2단계 코드 구현

### 2.3. 3단계 코드 구현

## 3. 게임 실행

## 4. 개선점

## 5. 기타

### 5.1. Makefile

### 5.2. README

### 5.3. Github

## 0. Push Box Game 설명

일명 Sokoban이라 불리는 Push Box Game은 퍼즐 비디오 게임의 한 종류로, 맵에 들어있는 플레이어가 박스를 움직이며 목적지까지 박스를 운반하는 것이 목표인 게임이다.

플레이어는 빈 곳으로 움직이거나 박스와 함께 움직일 수 있으며, 진행방향에 벽이 있거나 박스가 2개 이상 겹쳐져 있을 경우 움직일 수 없다.

이 프로젝트에서는 GNU에서 제공하는 text-based user-interface library인 ncurses library를 사용하여 기본적인 Push Box Game을 구현한다.

## 1. 코드 설계

### 1.1. 1단계 코드 설계

1단계에서는 ncurses library의 함수들을 사용하여 Push Box Map을 구현하는데, 클래스를 이용해 Push Box Map이 가져야할 요소들을 구현했다.

소스 파일 중 Map.h는 Push Box Map을 구현하는 데에 필요한 클래스의 인터페이스이다.

```
src > C Map.h > _MSVCR70_DLL > _ALLOCATOR_H
1  #ifndef MAP_H
2  #define MAP_H
3
4  // Interface for Push Box Game Map and its Logics
5
6  /*
7   0 : Null space
8   1 : Wall
9   2 : Box
10  3 : Destination
11  4 : Outside
12  5 : Character
13  6 : Box on destination
14  */
15
```

(본 보고서는 한글로 된 주석이 깨지는 것을 방지하기 위해 Windows 환경에서 작성하여 일부 소스코드에 error toggle이 표시될 수 있습니다.)

Push Box Map에는 빈 공간 (캐릭터가 제약 없이 지나다닐 수 있는 공간), 벽, 박스, 목적지, 맵의 바깥 부분, 캐릭터, 그리고 목적지에 들어간 박스를 표시하도록 각각 0~6의 값을 할당했다.

```
16 class Map{
17     public:
18         // Constructor
19
20         // Default Constructor
21         Map();
22
23         // Copy Constructor
24         Map(int map_of_stage[10][10]);
25
26         // Member Functions
27         Map& setElement(int target_x, int target_y, int input);
28         Map& setCharacter(int target_x, int target_y);
29         Map& move(char arrow);
30         Map& find_character();
31         Map& do_nothing();
32         Map& countDest();
33         Map& countBoxOnDest();
34         Map& increaseNumStep();
35         Map& increaseNumPush();
36
37         // Member Variables
38         int numDest;
39         int numBoxOnDest;
40         int map[10][10];
41         int default_map[10][10];
42         int location_of_character[2];
43         int numStep;
44         int numPush;
45
46 };
47 #endif
```

class Map에는 Push Box Map을 만들기 위한 생성자, (기본 생성자와 복사 생성자를 만들었다.) Push Box Map이 가져야할 멤버 변수들과 멤버 함수들이 선언되어 있다.

멤버 변수는 numDest부터 numPush까지 총 7개인데, 각각의 기능은 다음과 같다.

- int numDest : 단계별 Push Box Map의 목적지의 개수를 저장한다.
- int numBoxOnDest : 게임 진행 도중 목적지에 들어간 박스의 개수를 저장한다.
- int map[10][10] : 10\*10크기의 2차원 정수형 배열로 각 단계별 Push Box Map의 원소들의 정보를 갖고 있다.

- int default\_map[10][10] : Push Box Map을 처음 생성할 때의 정보를 갖고있는 2차원 정수형 배열로 Push Box Map의 처음 상태를 갖고 있다.
- int location\_of\_character[2] : 캐릭터의 현재 위치를 갖고 있는 1차원 정수형 배열로 캐릭터 위치의 행 값(char\_row)과 열 값(char\_col)을 저장한다.
- int numStep : 각 단계별 Push Box Game 진행 도중 발생한 Step 횟수를 저장한다.
- int numPush : 각 단계별 Push Box Game 진행 도중 발생한 Push 횟수를 저장한다.

## 1.2. 2단계 코드 설계

```

16  class Map{
17      public:
18          // Constructor
19
20          // Default Constructor
21          Map();
22
23          // Copy Constructor
24          Map(int map_of_stage[10][10]);
25
26          // Member Functions
27          Map& setElement(int target_x, int target_y, int input);
28          Map& setCharacter(int target_x, int target_y);
29          Map& move(char arrow);
30          Map& find_character();
31          Map& do_nothing();
32          Map& countDest();
33          Map& countBoxOnDest();
34          Map& increaseNumStep();
35          Map& increaseNumPush();
36
37          // Member Variables
38          int numDest;
39          int numBoxOnDest;
40          int map[10][10];
41          int default_map[10][10];
42          int location_of_character[2];
43          int numStep;
44          int numPush;
45
46  };
47  #endif

```

2단계에서는 1단계에서 만든 맵 위에 캐릭터를 표시하고 화살표를 입력받아 캐릭터가 움직이도록 프로그램을 완성한다.

이를 위해 class Map에서는 setElement()부터 increaseNumPush()까지 총 9개의 멤버 함수들을 선언했는데, 각각의 기능은 다음과 같다.

- Map& setElement(int target\_x, int target\_y, int input) : Push Box Map의 (target\_x, target\_y) 위치에 input 값을 삽입한다.
- Map& setCharacter(int target\_x, int target\_y) : Push Box Map의 (target\_x, target\_y) 위치에 캐릭터를 놓는다.
- Map& move(char arrow) : arrow라는 char형 변수(키보드 입력 값)를 입력 받아 Push Box Game을 진행하는 Game Logic이 담겨있는 함수이다. arrow 값과 캐릭터의 위치, 캐릭터 주변의 상황을 파악하여 캐릭터가 어느 방향으로 진행하고 주변의 값들은 어떻게 변화시킬 지 결정한다.
- Map& find\_character() : 캐릭터의 현재 위치를 찾아내서 location\_of\_character에 캐릭터 위치의 행 값(char\_row)과 열 값(char\_col)을 저장한다.
- Map& do\_nothing() : move함수에서 캐릭터가 진행할 수 없다고 판단될 때, 아무런 변화도 일어나지 않도록 한다.
- Map& countDest() : Push Box Map에서 목적지의 개수를 탐색해서 numDest에 저장한다.
- Map& countBoxOnDest() : Push Box Game 도중 목적지에 들어간 박스의 개수를 탐색해서 numBoxOnDest에 저장한다.
- Map& increaseNumStep() : 캐릭터가 진행방향으로 갈 때마다, 즉 Step이 발생할 때마다 numStep의 값을 증가시킨다.
- Map& increaseNumPush() : 캐릭터가 박스를 밀 때마다, 즉 Push가 발생할 때마다 numPush의 값을 증가시킨다.

## 2. 코드 구현

### 2.1. 1단계 코드 구현

```
16      /*
17      0 : Null space
18      1 : Wall
19      2 : Box
20      3 : Destination
21      4 : Outside
22      5 : Character
23      6 : Box on destination
24      */
25
26      int mapOne[10][10] = {
27          {4, 4, 1, 1, 1, 1, 4, 4, 4, 4},
28          {4, 4, 1, 3, 0, 1, 1, 4, 4, 4},
29          {4, 4, 1, 3, 0, 0, 1, 4, 4, 4},
30          {4, 4, 1, 3, 0, 0, 1, 4, 4, 4},
31          {4, 4, 1, 1, 0, 0, 1, 1, 1, 4},
32          {4, 4, 4, 1, 0, 0, 0, 0, 1, 4},
33          {4, 4, 4, 1, 0, 0, 0, 0, 1, 4},
34          {4, 4, 4, 1, 0, 0, 1, 1, 1, 4},
35          {4, 4, 4, 1, 1, 1, 1, 4, 4, 4},
36          {4, 4, 4, 4, 4, 4, 4, 4, 4, 4}
37      };
38
```

실제 맵의 구현은 main함수가 있는 main.cpp에서 이루어진다.

2차원 배열인 mapOne, mapTwo, mapThree는 각각 1단계, 2단계, 3단계 맵을 표현하는 맵 객체 생성자의 인자로, 0~6은 각각 빈 공간, 벽, 박스, 목적지, 바깥 부분, 캐릭터, 그리고 목적지에 들어간 박스를 나타낸다. 이때 배열에는 박스를 가리키는 2와 캐릭터를 가리키는 5를 배열에 포함시키지 않고 Map.cpp에서 그 위치를 가져와 디폴트 배열을 가져올 때마다 캐릭터와 상자의 처음 위치가 고정되어 있는 에러를 해결하도록 했다.

```

39     int mapTwo[10][10] = {
40         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
41         {4, 4, 1, 1, 1, 1, 1, 4, 4, 4},
42         {4, 4, 1, 0, 0, 0, 1, 4, 4, 4},
43         {4, 4, 1, 3, 3, 3, 1, 4, 4, 4},
44         {4, 4, 1, 0, 0, 0, 1, 1, 4, 4},
45         {4, 4, 1, 0, 0, 0, 0, 1, 4, 4},
46         {4, 4, 1, 0, 0, 0, 0, 1, 4, 4},
47         {4, 4, 1, 1, 1, 1, 1, 1, 4, 4},
48         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
49         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4}
50     };

52     int mapThree[10][10] = {
53         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
54         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
55         {4, 1, 1, 1, 1, 1, 1, 1, 1, 4},
56         {4, 1, 3, 0, 0, 0, 0, 0, 1, 4},
57         {4, 1, 0, 3, 0, 0, 0, 0, 1, 4},
58         {4, 1, 3, 0, 0, 0, 0, 0, 1, 4},
59         {4, 1, 1, 1, 1, 1, 0, 0, 1, 4},
60         {4, 4, 4, 4, 4, 1, 1, 1, 1, 4},
61         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4},
62         {4, 4, 4, 4, 4, 4, 4, 4, 4, 4}
63     };

64
65     Map map1(mapOne);
66     Map map2(mapTwo);
67     Map map3(mapThree);

```

이렇게 만든 2차원 배열 mapOne, mapTwo, mapThree를 생성자의 인자로 넣어 각 단계별 Push Box Map에 쓰일 맵 객체 map1, map2, map3를 만든다.



```

WINDOW *win1; // Map1
WINDOW *win2; // Map2
WINDOW *win3; // Map3
WINDOW *win4; // step 횃수
WINDOW *win5; // push 횃수
WINDOW *win6; // end

initscr();
resize_term(40, 50);
start_color();
init_pair(10, COLOR_BLACK, COLOR_WHITE);
init_pair(0, COLOR_BLACK, COLOR_BLUE);
init_pair(1, COLOR_BLACK, COLOR_GREEN);
init_pair(2, COLOR_YELLOW, COLOR_BLACK);
init_pair(3, COLOR_WHITE, COLOR_BLACK);
init_pair(4, COLOR_BLACK, COLOR_WHITE);
init_pair(5, COLOR_RED, COLOR_BLACK);

```

또한 화면에 맵과 맵의 요소들을 표시하기 위해 map1, map2, map3, step횃수, push횃수, 그리고 게임 종료 시 나오는 화면을 위한 WINDOW 객체를 main에서 선언한다. resize\_term()함수를 이용하여 윈도우의 크기를 설정한다. 그리고 init\_pair() 함수를 통해 나중에 사용할 색깔을 속성을 미리 선언한다.

```

1  #include "Map.h"
2
3  // 기본 생성자는 map과 default_map을 Null space로 채운다.
4  Map::Map(){
5      for (int i=0; i<10; ++i){
6          for (int j=0; j<10; ++j){
7              this->map[i][j] = 4;
8              this->default_map[i][j] = 4;
9          }
10     }
11 }
12
13 /*
14 복사 생성자는 main()에서 선언된 2차원 배열 map_of_stage를 map과 default_map에 복사
15 목적지의 갯수를 numDest에 저장
16 Step 횟수와 Push 횟수를 0으로 초기화
17 */
18 Map::Map(int map_of_stage[10][10]){
19     for (int i=0; i<10; ++i){
20         for (int j=0; j<10; ++j){
21             this->map[i][j] = map_of_stage[i][j];
22             this->default_map[i][j] = map_of_stage[i][j];
23         }
24     }
25     countDest();
26     numStep = 0;
27     numPush = 0;
28 }
29

```

또한 생성자에서 인자로 받은 2차원 배열을 맵 객체 내의 맵 정보를 나타내는 default\_map과 map에 저장해서 화면을 띄울 때에 참조할 수 있도록 했다.

```

87     border('|', '|', '-', '-', '+', '+', '+', '+');
88     mvprintw(1, 9, "Welcome To The Push Box Game!");
89     mvprintw(2, 12, "Press Any Key To Start");
90     refresh(); // 화면 업데이트. 화면 실제 출력
91     curs_set(0);
92     getch();
93     mvprintw(1, 9, "                Level 1                ");
94     mvprintw(2, 1, "-----");
95
96     win1 = newwin(12, 12, 16, 19);
97     wbkgd(win1, COLOR_PAIR(10));
98     wattron(win1, COLOR_PAIR(10));
99     wborder(win1, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ');
100
101     wrefresh(win1); // 화면 업데이트. win1 화면 실제 출력
102
103     // 박스랑 캐릭터 초기 위치 설정
104     map1.setElement(3, 5, 2);
105     map1.setElement(4, 4, 2);
106     map1.setElement(5, 5, 2);
107     map1.setCharacter(2, 4);
108

```

디폴트 윈도우의 경계선을 설정하고 그 창에 안내문을 출력한다. 그리고 refresh()함수를 통해 화면에 실제로 출력되도록 한다. getch()함수를 통해 키보드의 입력을 받으면 새로운

윈도우를 업데이트한다. 그리고 win1의 속성을 설정한다.

```
109 // 처음 맵 출력
110 for (int i = 0; i < 10; i++) {
111     for (int j = 0; j < 10; j++) {
112         if (map1.map[i][j] == 0) {
113             attron(COLOR_PAIR(0));
114             mvprintw(17 + i, 20 + j, " ");
115             attroff(COLOR_PAIR(0));
116         }
117         else if (map1.map[i][j] == 1) {
118             attron(COLOR_PAIR(1));
119             mvprintw(17 + i, 20 + j, " ");
120             attroff(COLOR_PAIR(1));
121         }
122         else if (map1.map[i][j] == 2) {
123             attron(COLOR_PAIR(2));
124             mvprintw(17 + i, 20 + j, "\u2752");
125             attroff(COLOR_PAIR(2));
126         }
127         else if (map1.map[i][j] == 3) {
128             attron(COLOR_PAIR(3));
129             mvprintw(17 + i, 20 + j, "\u2B1A");
130             attroff(COLOR_PAIR(3));
131         }
132         else if (map1.map[i][j] == 4) {
133             attron(COLOR_PAIR(4));
134             mvprintw(17 + i, 20 + j, " ");
135             attroff(COLOR_PAIR(4));
136         }
137     }
138 }
```

```
137         else if (map1.map[i][j] == 5) {
138             attron(COLOR_PAIR(4));
139             mvprintw(17 + i, 20 + j, "C");
140             attroff(COLOR_PAIR(4));
141         }
142         else if (map1.map[i][j] == 6) {
143             attron(COLOR_PAIR(5));
144             mvprintw(17 + i, 20 + j, "\u2752");
145             attroff(COLOR_PAIR(5));
146         }
147     }
148 }
```

map배열의 값을 if문으로 나눠서 색깔또는 유니코드로 구별을 짓는다. 그리고 이중 for문을 사용하여 화면에 맵을 출력시킨다.

## 2.2. 2단계 코드 구현

```
30 // map의 (target_x, target_y) 위치의 값을 input으로 변경
31 Map& Map::setElement(int target_x, int target_y, int input){
32     this->map[target_x][target_y] = input;
33     return *this;
34 }
35
36 // 캐릭터의 위치를 (target_x, target_y)로 설정
37 Map& Map::setCharacter(int target_x, int target_y){
38     this->map[target_x][target_y] = 5;
39     return *this;
40 }
```

1.2에서 선언한 멤버 함수들의 구현은 Map.h를 include한 Map.cpp에서 이루어진다.

setElement 함수는 맵의 (target\_x, target\_y)의 값을 input으로 변화시킨다.

map이 Map 객체의 멤버 변수로 들어가 있기 때문에 함수의 반환형은 Map&이며 \*this를 return한다.

setCharacter 함수는 캐릭터의 위치를 이동시키는 함수로 맵의 (target\_x, target\_y) 위치에 캐릭터를 놓는다.

setElement 함수의 인자 input으로 5(캐릭터)를 줘도 되지만, main에서 맵을 만들 때 따로 캐릭터의 시작점을 지정해줘야 해서 setElement 함수와 구분해서 구현했다.

```

60         case 72 : // ↑ 입력
61             if ((this->map[char_row-1][char_col] == 0)|| (this->map[char_row-1][char_col] == 3)){ // 캐릭터 위쪽이 비어있을 때
62                 setCharacter(char_row-1, char_col);
63                 setElement(char_row, char_col, default_map[char_row][char_col]);
64                 increaseNumStep();
65             }
66             else if (this->map[char_row-1][char_col] == 1){ // 캐릭터 위쪽이 벽이면
67                 do_nothing(); // 캐릭터 이동 불가능
68             }
69             else if (this->map[char_row-1][char_col] == 2){ // 캐릭터 위쪽이 박스일 때
70                 if ((this->map[char_row-2][char_col] == 1)|| (this->map[char_row-2][char_col] == 2)){ // 박스 위쪽이 벽이면
71                     do_nothing(); // 캐릭터 이동 불가능
72                 }
73                 else if (this->map[char_row-2][char_col] == 0){ // 박스 앞에 비어있으면
74                     setElement(char_row, char_col, default_map[char_row][char_col]); // 캐릭터 있던 곳은 기본 값
75                     setCharacter(char_row-1, char_col); // 캐릭터와 박스가 함께 위쪽으로 이동
76                     setElement(char_row-2, char_col, 2);
77                     increaseNumStep();
78                     increaseNumPush();
79                 }
80                 else if (this->map[char_row-2][char_col] == 3){ // 박스 앞에 목적이면
81                     setElement(char_row, char_col, default_map[char_row][char_col]); // 캐릭터 있던 곳은 기본 값
82                     setCharacter(char_row-1, char_col); // 캐릭터와 박스가 함께 위쪽으로 이동
83                     setElement(char_row-2, char_col, 6); // 목적지에 박스가 들어갔음을 표시
84                     increaseNumStep();
85                     increaseNumPush();
86                 }
87             }

```

```

42 Map& Map::move(char arrow){
43     /*
44     arrow input -> move
45     front of wall, or over 2 box -> don't move
46     front of one box and not blocked -> move with box
47     */
48
49     find_character(); // 캐릭터 현재 위치 탐색
50     int char_row = this->location_of_character[0]; // 캐릭터의 현재 위치(행 값)
51     int char_col = this->location_of_character[1]; // 캐릭터의 현재 위치(열 값)
52
53     switch(arrow){
54         /*
55         Unicode 72 represents 'up'
56         Unicode 75 represents 'left'
57         Unicode 77 represents 'right'
58         Unicode 80 represents 'down'
59         */
60         case 72 : // ↑ 입력
61             if ((this->map[char_row-1][char_col] == 0)|| (this->map[char_row-1][char_col] == 3)){ // 캐릭터 위쪽이 비어있을 때
62                 setCharacter(char_row-1, char_col);
63                 setElement(char_row, char_col, default_map[char_row][char_col]);
64                 increaseNumStep();
65             }
66             else if (this->map[char_row-1][char_col] == 1){ // 캐릭터 위쪽이 벽이면
67                 do_nothing(); // 캐릭터 이동 불가능
68             }
69             else if (this->map[char_row-1][char_col] == 2){ // 캐릭터 위쪽이 박스일 때
70                 if ((this->map[char_row-2][char_col] == 1)|| (this->map[char_row-2][char_col] == 2)){ // 박스 위쪽이 벽이면
71                     do_nothing(); // 캐릭터 이동 불가능
72                 }
73                 else if (this->map[char_row-2][char_col] == 0){ // 박스 앞에 비어있으면

```

```

88     else if (this->map[char_row-1][char_col] == 6){ // 캐릭터 위쪽이 BoxOnDest이면
89         if (this->map[char_row-2][char_col] == 0){ // BoxOnDest 앞에 비어있으면
90             setElement(char_row, char_col, default_map[char_row][char_col]); // 캐릭터 있던 곳은 기본 값
91             setCharacter(char_row-1, char_col);
92             setElement(char_row-2, char_col, 2);
93             increaseNumStep();
94             increaseNumPush();
95         }
96         else if (this->map[char_row-2][char_col] == 1){ // BoxOnDest 앞에 벽이면
97             do_nothing(); // 캐릭터 이동 불가능
98         }
99         else if (this->map[char_row-2][char_col] == 2){ // BoxOnDest 앞에 박스이면
100             do_nothing(); // 캐릭터 이동 불가능
101         }
102         else if (this->map[char_row-2][char_col] == 3){ // BoxOnDest 앞에 목적지이면
103             setElement(char_row, char_col, default_map[char_row][char_col]); // 캐릭터 있던 곳은 기본 값
104             setCharacter(char_row-1, char_col); // 캐릭터 한칸 전진
105             setElement(char_row-2, char_col, 6); // 박스도 한칸 전진
106             increaseNumStep();
107             increaseNumPush();
108         }
109         else if (this->map[char_row-2][char_col] == 6){ // BoxOnDest 앞에 BoxOnDest이면
110             do_nothing();
111         }
112     }
113     break;

```

다음으로 move 함수는 키보드의 방향키를 입력받았을 때 캐릭터의 움직임을 정의하고 제한사항을 두는 함수로, Push Box Game에서 가장 중요한 로직이 담겨있는 함수이다.

캐릭터의 위치를 갖고있는 멤버 변수 location\_of\_character에서 캐릭터의 위치 (행 값과 열 값)를 참고해서 캐릭터 주변의 상황 (->키를 입력받았다면 캐릭터를 기준으로 오른쪽 두개의 맵 값)을 파악하고 주변의 상황에 따라 캐릭터가 이동을 할 지, 박스를 밀면서 이동을 할 지, 혹은 움직이지 못하게 할 지 결정한다.

만약 캐릭터가 혼자 이동한다면 step횟수만, 캐릭터가 박스를 밀면서 이동한다면 step과 push 횟수를 모두 증가시킨다.

```

284 // Search location of character and store it to location_of_character
285 Map& Map::find_character(){
286     int row, col;
287     for (int i=0; i<10; ++i){
288         for (int j=0; j<10; ++j){
289             if (this->map[i][j] == 5){
290                 row = i;
291                 col = j;
292             }
293         }
294     }
295     this->location_of_character[0] = row;
296     this->location_of_character[1] = col;
297     return *this;
298 }
299
300
301 // If character cannot move, just let it be
302 Map& Map::do_nothing(){
303     return *this;
304 }
305

```

```

306 // Counts number of destinations
307 Map& Map::countDest(){
308     this->numDest = 0;
309     for (int i=0; i<10; ++i){
310         for (int j=0; j<10; ++j){
311             if (this->default_map[i][j] == 3){
312                 this->numDest++;
313             }
314         }
315     }
316     return *this;
317 }
318
319 // Counts number of boxes on destination
320 Map& Map::countBoxOnDest(){
321     this->numBoxOnDest = 0;
322     for (int i=0; i<10; ++i){
323         for (int j=0; j<10; ++j){
324             if (this->map[i][j] == 6){
325                 this->numBoxOnDest++;
326             }
327         }
328     }
329     return *this;
330 }

```

```

332 Map& Map::increaseNumStep(){
333     this->numStep++;
334     return *this;
335 }
336
337 Map& Map::increaseNumPush(){
338     this->numPush++;
339     return *this;
340 }

```

find\_character 함수는 맵에서 캐릭터를 찾아서 location\_of\_character에 캐릭터 위치의

행 값과 열 값을 저장한다. move 함수에서 캐릭터의 위치를 파악하는 데에 사용된다.

do\_nothing 함수는 캐릭터가 움직일 수 없다고 판단될 때, 아무것도 하지 않고 그대로 \*this를 return하는 함수이다. Move 함수에서 사용된다.

countDest와 countBoxOnDest 함수는 각각 목적지와 목적지에 들어간 박스의 개수를 세서 해당 맵 객체의 numDest와 numBoxOnDest에 저장한다. Push Box Game이 종료 되는 기준이 목적지와 목적지에 들어간 박스의 개수가 같은 지이기 때문에, 목적지와 목적지에 들어간 박스의 개수를 세도록 했다.

increaseStep과 increasePush 함수는 각각 step 횟수와 push 횟수를 증가시키는데, Push Box Game 도중 캐릭터가 step을 하거나 push를 할 때 numStep과 numPush의 값을 증가시킬 때 사용된다. move 함수에서 캐릭터가 step이나 push를 한다고 판단 할 때 호출돼서 사용된다.

### 2.3. 3단계 코드 구현

```
167     curs_set(0);
168     int ch;
169
170     while (map1.numDest != map1.numBoxOnDest) {
171         keypad(stdscr, true);
172         //PrintMap();
173         ch = getch();
174         if (ch == KEY_UP) {
175             map1.move(72); // 위
176             // 바뀔 때마다 맵 출력
177             for (int i = 0; i < 10; i++) {
178                 for (int j = 0; j < 10; j++) {
179                     if (map1.map[i][j] == 0) {
180                         attron(COLOR_PAIR(0));
181                         mvprintw(17 + i, 20 + j, " ");
182                         attroff(COLOR_PAIR(0));
183                     }
184                     else if (map1.map[i][j] == 1) {
185                         attron(COLOR_PAIR(1));
186                         mvprintw(17 + i, 20 + j, " ");
187                         attroff(COLOR_PAIR(1));
188                     }
189                     else if (map1.map[i][j] == 2) {
190                         attron(COLOR_PAIR(2));
191                         mvprintw(17 + i, 20 + j, "\u2752");
192                         attroff(COLOR_PAIR(2));
193                     }
194                 }
195             }
196         }
197     }
```



```

194         else if (map1.map[i][j] == 3) {
195             attron(COLOR_PAIR(3));
196             mvprintw(17 + i, 20 + j, "\u2B1A");
197             attroff(COLOR_PAIR(3));
198         }
199         else if (map1.map[i][j] == 4) {
200             attron(COLOR_PAIR(4));
201             mvprintw(17 + i, 20 + j, " ");
202             attroff(COLOR_PAIR(4));
203         }
204         else if (map1.map[i][j] == 5) {
205             attron(COLOR_PAIR(4));
206             mvprintw(17 + i, 20 + j, "C");
207             attroff(COLOR_PAIR(4));
208         }
209         else if (map1.map[i][j] == 6) {
210             attron(COLOR_PAIR(5));
211             mvprintw(17 + i, 20 + j, "\u2752");
212             attroff(COLOR_PAIR(5));
213         }
214     }
215 }
216 mvprintw(9,12,"Step : %d",map1.numStep);
217 mvprintw(9,30,"PUSH : %d",map1.numPush);
218 }

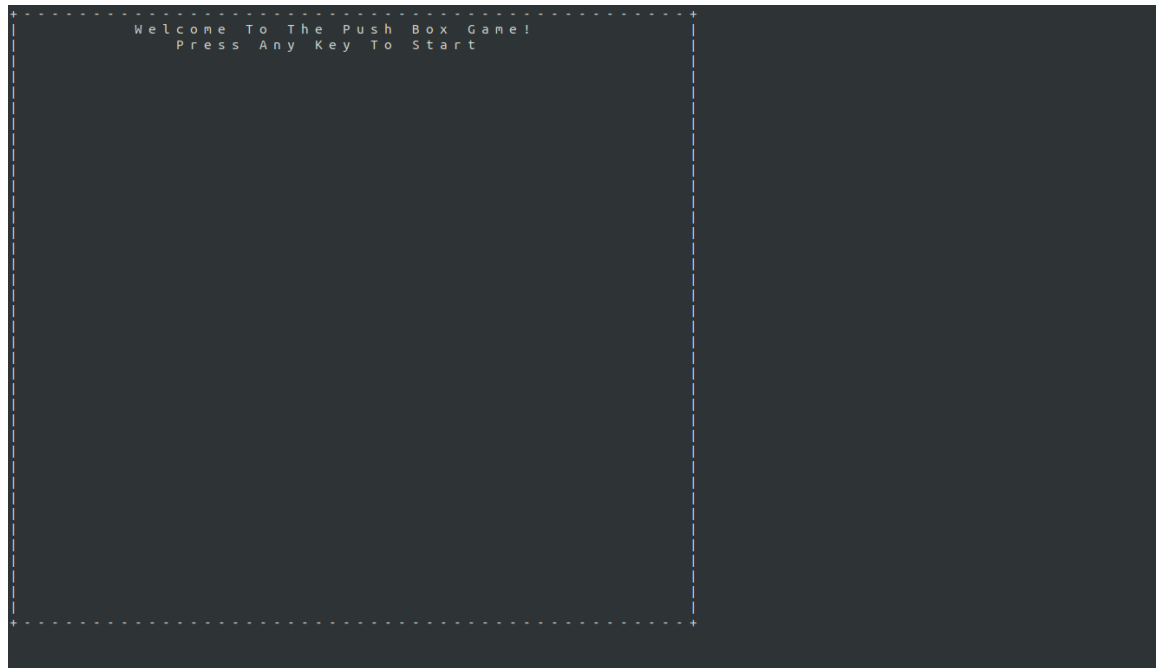
```

현재 맵의 목적지 개수와 목적지에 들어간 박스의 개수가 같아 지기 전까지 while문이 실행된다. 이때 키보드의 입력을 받을 때마다 이에 따라 바뀐 맵과 step, push의 횟수를 가져와서 화면에 출력되도록 구현하였다.

이렇게 Map.cpp에서 구현한 함수들과 ncurses library의 함수들을 이용해서 main에서 키보드 입력에 따라 캐릭터가 움직일 수 있도록 구현했다.

위와 같은 코드가 키보드 위/아래/왼쪽/오른쪽과 Push Box Game 1단계/2단계/3단계로 반복된다.

### 3. 게임 실행



Ubuntu 터미널에서 게임 실행 명령 (`g++ -std=c++11 -o pushbox main.cpp -lcursesw`를 치면 실행 파일 `pushbox`가 생성되고 `./pushbox` 명령으로 게임 실행)을 치면 나오는 처음 게임 시작 화면이다.

Level1

Level 1

Step : 0


PUSH : 0



Level 1

Step : 63

PUSH : 21



Level2

Level 2

Step : 0

PUSH : 0



Level 2

Step : 9

PUSH : 3



### Level3



스테이지 1~3의 실행 모습이다. 박스를 목적지에 모두 넣었을 경우, 엔터키를 입력하면 다음 스테이지로 넘어간다. (getch 함수를 썼기 때문에 엔터키를 눌러줘야 함)



스테이지 3까지 모두 완료한 후 엔터키를 누르면 "SUCCESS!"가 포함된 화면이 출력된다. 이후 아무 키나 누르면 게임이 종료된다.

#### 4. 개선점

추가적인 기능을 넣지 못한 것이 좀 아쉽다. 게임을 실제로 해보니 게임을 완료하지 못했는데 더 이상 진행할 수 없는 상황에서는 터미널을 끄는 법 외에는 게임을 다시 할 수 있는 방법이 없는데, restart 기능을 추가해서 다시 시작하게 하거나 exit 기능을 추가해서 중간에 정상적으로 종료시킬 수 있는 기능을 추가하는 것도 생각해봤다.

또한 이번엔 단순히 텍스트를 기반으로 만들어서 user interface가 별로인데 사진 파일을 불러와서 맵의 요소와 대응되도록 만들면 더욱 멋진 게임을 만들 수 있을 것 같다.

## 5. 기타

### 5.1. Makefile

```
M Makefile
1  HEADERS = Map.h
2  SOURCES = Map.cpp main.cpp
3  CC = g++
4  CFLAGS = -std=c++11
5  EXECUTABLE = pushbox
6  RM = rm
7
8  all : $(CC) $(CFLAGS) -o $(EXECUTABLE) $(SOURCES) -lncursesw # g++ -std=c++11 -o pushbox main.cpp -lncursesw
9
10 clean : $(RM) *.o $(EXECUTABLE)
```

### 5.2. README

```
① README.md ▶ abc # 2019-CPP-term-project
1  # 2019-CPP-term-project
2  2019-1 C++ term project - Push Box Game
3  Team : FRIDAY
```

### 5.3. Github

소스 코드를 Github에 업로드 했습니다. (2019년 6월 18일 이후 공개 범위를 public으로 수정)

Github 주소 : <https://github.com/MaengSanHa/2019-CPP-term-project>