C++ PUSHBOX GAME REPORT

20163081 강신표
20163162 차윤성

Contents

MAP

MAP (map.h)

```
#ifndef __MAP_H__
#define __MAP_H__

class Map{
public:
  int height; // 맵의 세로 간격
  int wide;   // 맵의 가로 간격
  int map[10][10]; // Map을 저장할 2차원 배열
  int nBox; // 각 맵 별 Box의 갯수를 저장하는 변수
  Map();
  void loadMap1();
  void loadMap2();
  void loadMap3();
  void loadMap4();
  void loadMap5();
  //loadMap은 각 Level별 Map을 불러오는 함수이다.
};

#endif
```

Int heigth;
Int wide;
//Map 의 폭과 높이를 설정할 변수를 선언

Int map[10][10]
//map 을 저장할 2 차원 배열을 선언

Int nBox;
// 각 맵에 존재하는 박스의 개수를 저장하는 변수

loadMap() 은 각 level 별 map 을 불러오는 함수

MAP

```
//Map의 크기를 10x10으로 설정
Map::Map(){
  height=10;
  wide=10;
  nBox=0;
}
```

앞서 헤더파일에서 map 에
저장할 2 차원 배열
크기에 맞게
Map 의 폭과 높이의를
10x10 으로 설정

MAP

map1.txt

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 1 2 2 2 1 0 0 0
0 0 1 3 3 3 1 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 4 0 0 1 0 0
0 0 1 1 1 1 1 1 0 0
```

map2.txt

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0 0 0
0 1 2 0 1 1 0 0 0 0
0 1 2 4 0 1 0 0 0 0
0 1 2 0 3 1 0 0 0 0
0 1 1 3 0 1 1 1 0 0
0 0 1 0 3 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 1 1 1 0 0
0 0 1 1 1 1 0 0 0 0
```

map3.txt

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0
0 1 2 0 0 0 0 0 1 0
0 1 0 2 3 3 3 4 1 0
0 1 2 0 0 0 0 0 1 0
0 1 1 1 1 1 0 0 1 0
0 0 0 0 0 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
```

map4.txt

```
0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 0
0 0 1 0 0 0 0 0 1 0
0 0 1 0 2 3 2 0 1 0
0 0 1 0 3 2 3 0 1 0
0 0 1 0 2 3 2 0 1 0
0 0 1 0 3 2 3 0 1 0
0 0 1 0 0 4 0 0 1 0
0 0 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
```

map5.txt

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 0 0 0
0 1 0 0 1 1 1 1 0 0
0 1 0 0 0 0 0 1 1 0
1 1 0 1 1 0 0 0 1 0
1 2 0 2 1 0 4 3 1 1
1 0 0 0 1 0 3 3 0 1
1 0 0 2 1 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
```

map.cpp

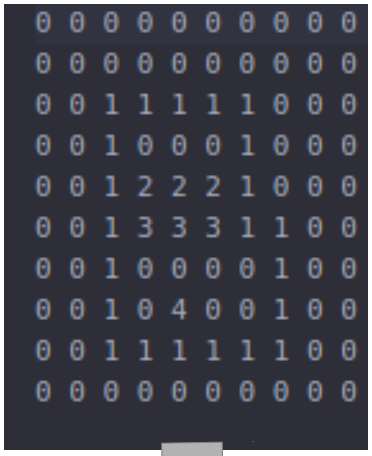
```
// loadMap -> 맵 불러오는 함수
void Map::loadMap1(){
    ifstream infile;
    infile.open("map/map1.txt",ios::in);
    for (int i=0;i<10;i++){
      for(int j=0;j<10;j++){
        infile>>map[i][j];
      }
    }
    infile.close();
}

void Map::loadMap2(){
    ifstream infile;
    infile.open("map/map2.txt",ios::in);
    for (int i=0;i<10;i++){
      for(int j=0;j<10;j++){
        infile>>map[i][j];
      }
    }
    infile.close();
}

void Map::loadMap3(){
    ifstream infile;
    infile.open("map/map3.txt",ios::in);
    for (int i=0;i<10;i++){
      for(int j=0;j<10;j++){
        infile>>map[i][j];
      }
    }
    infile.close();
}

void Map::loadMap4(){
    ifstream infile;
    infile.open("map/map4.txt",ios::in);
    for (int i=0;i<10;i++){
      for(int j=0;j<10;j++){
        infile>>map[i][j];
      }
    }
```

Map 을 불러오는 loadMap() 함수를 만들고 text 파일의 map 을 10x10 2 차원 배열 map 에 저장 시켜 game.cpp 에서 함수를 호출해 사용한다 .

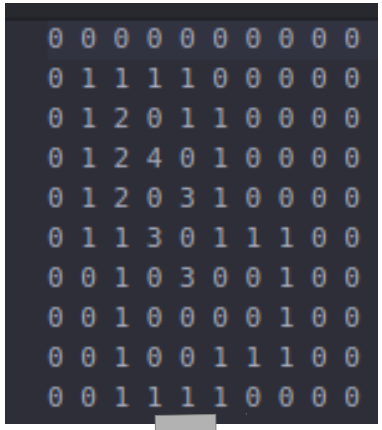
```
void Game::levelList(){
  switch(lev){
  case 1:
    gameMap.loadMap1();
    break;
  case 2:
    gameMap.loadMap2();
    break;
  case 3:
    gameMap.loadMap3();
    break;
  case 4:
    gameMap.loadMap4();
    break;
  case 5:
    gameMap.loadMap5();
    break;
  default:
    break;
  }
}
```

MAP

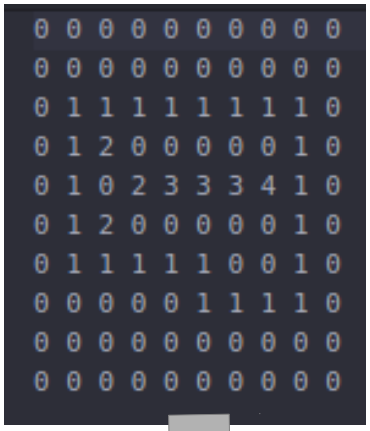
1 단계 Map



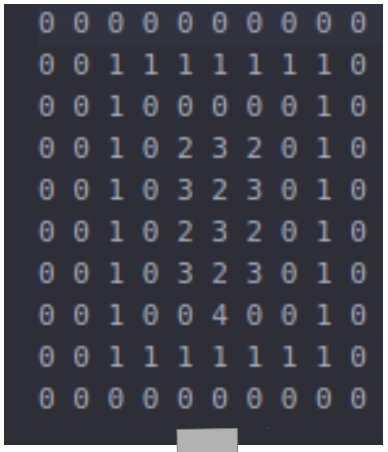
2 단계 Map



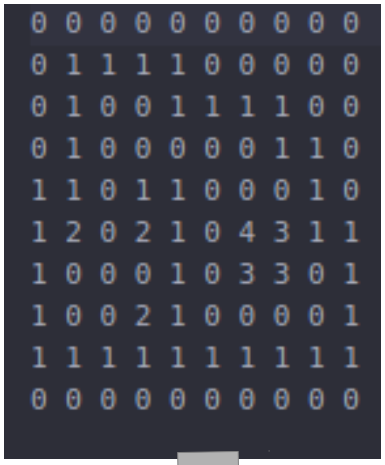
3 단계 Map



4 단계 Map



5 단계 Map



GAME

GAME (game.h)

```
#ifndef __GAME_H__
#define __GAME_H__

struct Object{
  int xPosition;  // 박스와 플레이어의 x좌표를 저장하는 변수
  int yPosition;  // 박스와 플레이어의 y좌표를 저장하는 변수
  unsigned char zn; // 각 객체의 형태를 저장하는 변수
  chtype ozn; // 객체가 이동한 위치의 모양을 저장하는 변수
};
```

int xPosition, int yPosition
// 박스와 플레이어의 x,y 좌표를 저장하는 변수

unsigned char zn;
// 각 객체의 형태를 저장하는 변수

chtype ozn;
// 객체가 이동한 위치의 모양을 저장하는 변

GAME (game.h)

```
class Game {
private:
  const int MAX = 10; //객체의 최대 갯수 저장
  int lev;  // Level을 저장해주는 변수
  int moveCnt;
  int pushCnt;
  Object *obj;  // 각 객체를 저장하는 배열(동적할당)
  Map gameMap;  // level별 맵을 불러오기 위한 Map형 객체
  char undo;  // undo 기능을 위한 플레이어의 이전 위치 저장
  char undobox; // undo 기능을 위한 박스의 이전 위치 저장
public:
  Game();
  int getMoveCnt();
  int getPushCnt();
  void palette();
  void levelUp();
  void levelList();
  void level();
  bool checkClear();
  void play(int input);
};

#endif
```

const int MAX = 10;
// 객체의 최대 갯수를 10 으로 제한

int lev; //level 을 저장하는 변수

int moveCnt;
int pushCnt; //step 과 push
카운트를 저장하는 변수

object *obj;
// 각 객체를 동적할당 하는 배열

map gameMap;
//level 별 맵을 불러오기 위한 객체

char undo;
char undobox;
//undo 기능을 구현하기 위한 변수

GAME

```
Game::Game(){
  moveCnt = 0;
  pushCnt = 0;
  lev = 1;
  obj = new Object[MAX];
}

int Game::getMoveCnt(){
  return moveCnt;
}

int Game::getPushCnt(){
  return pushCnt;
}
```

moveCnt = 0; //step 수를 초기화
pushCnt = 0; //push 수를 초기화
lev = 1; //1 단계 부터 시작
obj = new Object[MAX]; //obj 배열 생성

getMoveCnt() //step 수를 리턴
getPushCnt() //push 수를 리턴

```
void Game::levelUp(){
  lev++;  // 맵 클리어 시 level 올려주는 함수
}
```

Map 클리어 시 다음
level 로 올려주는 함수

GAME (palette)

```
void Game::palette(){
  init_color(COLOR_BLACK,0,0,0);
  init_pair(1,COLOR_BLACK,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_MAGENTA,COLOR_BLACK);
  init_pair(6,COLOR_WHITE,COLOR_BLACK);
}
```

```
for(int y=0;y<gameMap.height;y++){
  for(int x=0;x<gameMap.wide;x++){
    switch(gameMap.map[y][x]){
    case 0:
      mvaddch(y+10,x+33,'-'|COLOR_PAIR(1));
      break;
    case 1:
      mvaddch(y+10,x+33,'@'|COLOR_PAIR(2));
      break;
    case 2:
      mvaddch(y+10,x+33,'x'|COLOR_PAIR(3));
      break;
    case 3:
      mvaddch(y+10,x+33,'-'|COLOR_PAIR(1));
      gameMap.nBox += 1;
      obj[gameMap.nBox].ozn = mvinch(y + 10, x + 33);
      obj[gameMap.nBox].yPosition = y + 10;
      obj[gameMap.nBox].xPosition = x + 33;
      obj[gameMap.nBox].zn = 'O';
      mvaddch(obj[gameMap.nBox].yPosition,obj[gameMap.nBox].xPosition,obj[gameMap.nBox].zn | COLOR_PAIR(5));
      break;
    case 4:
      mvaddch(y+10,x+33,'-'|COLOR_PAIR(1));
      obj[0].ozn = mvinch(y + 10, x + 33);
      obj[0].yPosition = y + 10;
      obj[0].xPosition = x + 33;
      obj[0].zn = '*';
      mvaddch(obj[0].yPosition,obj[0].xPosition,obj[0].zn | COLOR_PAIR(4));
      break;
    }
  }
}
move(obj[0].yPosition,obj[0].xPosition);
}
```

Void palette()
//color_pair 생성
Init_pair(기호 , 글씨색 , 배경색)

Map 에서
0 : blank(-)
1 : wall(@)
2 : destination(x)
3 : O(box)
4 : *(player)
을 나타낸다 .

GAME (levelList)

```
void Game::levelList(){
  switch(lev){
  case 1:
    gameMap.loadMap1();
    break;
  case 2:
    gameMap.loadMap2();
    break;
  case 3:
    gameMap.loadMap3();
    break;
  case 4:
    gameMap.loadMap4();
    break;
  case 5:
    gameMap.loadMap5();
    break;
  default:
    break;
  }
}
```

LevelList()
// 각 lev(단계) 별로 map 을 로드

default :
    map 을 모두 클리어 할 경우

GAME (level)

```
void Game::level(){
  gameMap.nBox = 0;
  clear();
  if(lev<=5){
    levelList();
    mvprintw(5,33,"Level : %d",lev);
    for(int y=0;y<gameMap.height;y++){
      for(int x=0;x<gameMap.wide;x++){
        switch(gameMap.map[y][x]){
        case 0:
          mvaddch(y+10,x+33,'-'|COLOR_PAIR(1));
          break;
        case 1:
          mvaddch(y+10,x+33,'@'|COLOR_PAIR(2));
          break;
        case 2:
          mvaddch(y+10,x+33,'x'|COLOR_PAIR(3));
          break;
```

Level() // 게임을 화면에 표현해주는 함수

gameMap.nBox = 0;
// 맵에서의 박스 개수 초기

if(lev <= 5)
//level 이 5 단계 이전일 때

switch(gameMap.map[y][x]
// map 의 숫자에 따른 기호를 넣고
실제로 화면에 맵을 구현

case 0 : '-' (공백)
case 1 : '@' (벽)
case 2 : 'x' (목적지)

GAME (level)

```
      case 3:
        mvaddch(y+10,x+33,'-'|COLOR_PAIR(1)); //박스가 움직인 자리도 '-'(공백)처리
        gameMap.nBox += 1; //박스의 개수
        obj[gameMap.nBox].ozn = mvinch(y + 10, x + 33); //박스 위치받아오기
        obj[gameMap.nBox].yPosition = y + 10;
        obj[gameMap.nBox].xPosition = x + 33;
        obj[gameMap.nBox].zn = 'O'; //박스를 표현할 기호
        mvaddch(obj[gameMap.nBox].yPosition,obj[gameMap.nBox].xPosition,obj[gameMap.nBox].zn | COLOR_PAIR(5));
        break;
      case 4:
        mvaddch(y+10,x+33,'-'|COLOR_PAIR(1)); //플레이어가 움직인 자리도 '-'(공백)처리
        obj[0].ozn = mvinch(y + 10, x + 33); //플레이어 위치 받아오기
        obj[0].yPosition = y + 10;
        obj[0].xPosition = x + 33;
        obj[0].zn = '*'; //플레이어 표현할 기호
        mvaddch(obj[0].yPosition,obj[0].xPosition,obj[0].zn | COLOR_PAIR(4));
        break;
      }
    }
  }
  move(obj[0].yPosition,obj[0].xPosition); //플레이어를 이동시킨다.
}
```

case 3 : ‘O’ (박스)
case 4 : ‘*’ (플레이어)

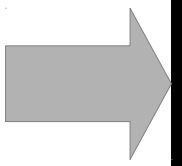
GameMap.nBox +=1
// 맵에서 3 을 읽어들일 때마다 박스 개수 추가

박스와 플레이어가 지나간 자리도 ‘ -’ (공백) 처리되도록 구현

.ozn 에서 mvinch 로 박스와 플레이어의 위치를 받고 .zn 에 저장한 기호를 출력

GAME (level)

```
  //맵을 모두 클리어한 경
  else {
    attron(COLOR_PAIR(4));
    mvprintw(10,31,"Congratulation!");
    mvprintw(12,31,"Clear All Level");
    mvprintw(14,26,"Press q to quit this game");
  }
}
```



주어진 맵을 모두 클리어 할 경우
해당 문구 출력

GAME (checkClear)

```
bool Game::checkClear(){
  chtype check;
  int cnt = 0; //'x'(목적지)에 도착한 박스의 개수
  for(int i=1; i<=gameMap.nBox;i++){
    //박스의 위치를 캐릭터 타입으로 저장
    check = (mvinch(obj[i].yPosition, obj[i].xPosition) & A_CHARTEXT);
    if(check == 120) // 'x'(목적지)와 같을 경우
      cnt++;
  }
  if(cnt == gameMap.nBox) { //전체 박스 개수와 목적지 위의 박스 개수가 같을 때
    return true;
  }
  else
    return false;
}
```

CheckClear()
// 다음 맵으로 넘어갈지 여부 판단

Int cnt; // 목적지에 도착한 박스개수

박스의 위치를 chtype 으로 받은 후
‘x’ (목적지) 의 아스키코드값인
120 과 같을 경우 cnt++

최종적으로 전체 박스의 개수와
목적지에 도달한 박스의 개수가
동일하면 truer 값 리턴

GAME (play)

```
//실제 게임 플레이를 구현
void Game::play(int input){
  bool restart = false;

  char up, left, down, right, oUp, oLeft, oDown, oRight;
  up = (mvinch(obj[0].yPosition - 1,obj[0].xPosition) & A_CHARTEXT);
  left = (mvinch(obj[0].yPosition, obj[0].xPosition - 1) & A_CHARTEXT);
  down = (mvinch(obj[0].yPosition +1, obj[0].xPosition) & A_CHARTEXT);
  right = (mvinch(obj[0].yPosition,obj[0].xPosition + 1)& A_CHARTEXT);
  oUp = (mvinch(obj[0].yPosition - 2,obj[0].xPosition) & A_CHARTEXT);
  oLeft = (mvinch(obj[0].yPosition, obj[0].xPosition - 2) & A_CHARTEXT);
  oDown = (mvinch(obj[0].yPosition + 2, obj[0].xPosition) & A_CHARTEXT);
  oRight = (mvinch(obj[0].yPosition,obj[0].xPosition + 2)& A_CHARTEXT);

  for(int i = 0;i<=gameMap.nBox;i++){
    mvaddch(obj[i].yPosition,obj[i].xPosition,obj[i].ozn);
  }
```

Play() // 게임플레이를 구현

up : 플레이어 위치의 한 칸 위
left : 플레이어 위치의 한 칸 왼쪽
down : 플레이어 위치의 한 칸 아래
right : 플레이어 위치의 한 칸 오른쪽
oUp : 플레이어 위치의 두 칸 위
oLeft ; 플레이어 위치의 두 칸 왼쪽
oDown : 플레이어 위치의 두 칸 아래
oRight : 플레이어 위치의 두 칸 오른쪽

// 플레이어와 박스의 위치마다 해당 기호를 삽

```
for(int i = 0;i<=gameMap.nBox;i++){
    mvaddch(obj[i].yPosition,obj[i].xPosition,obj[i].ozn);
  }
```

GAME (play)

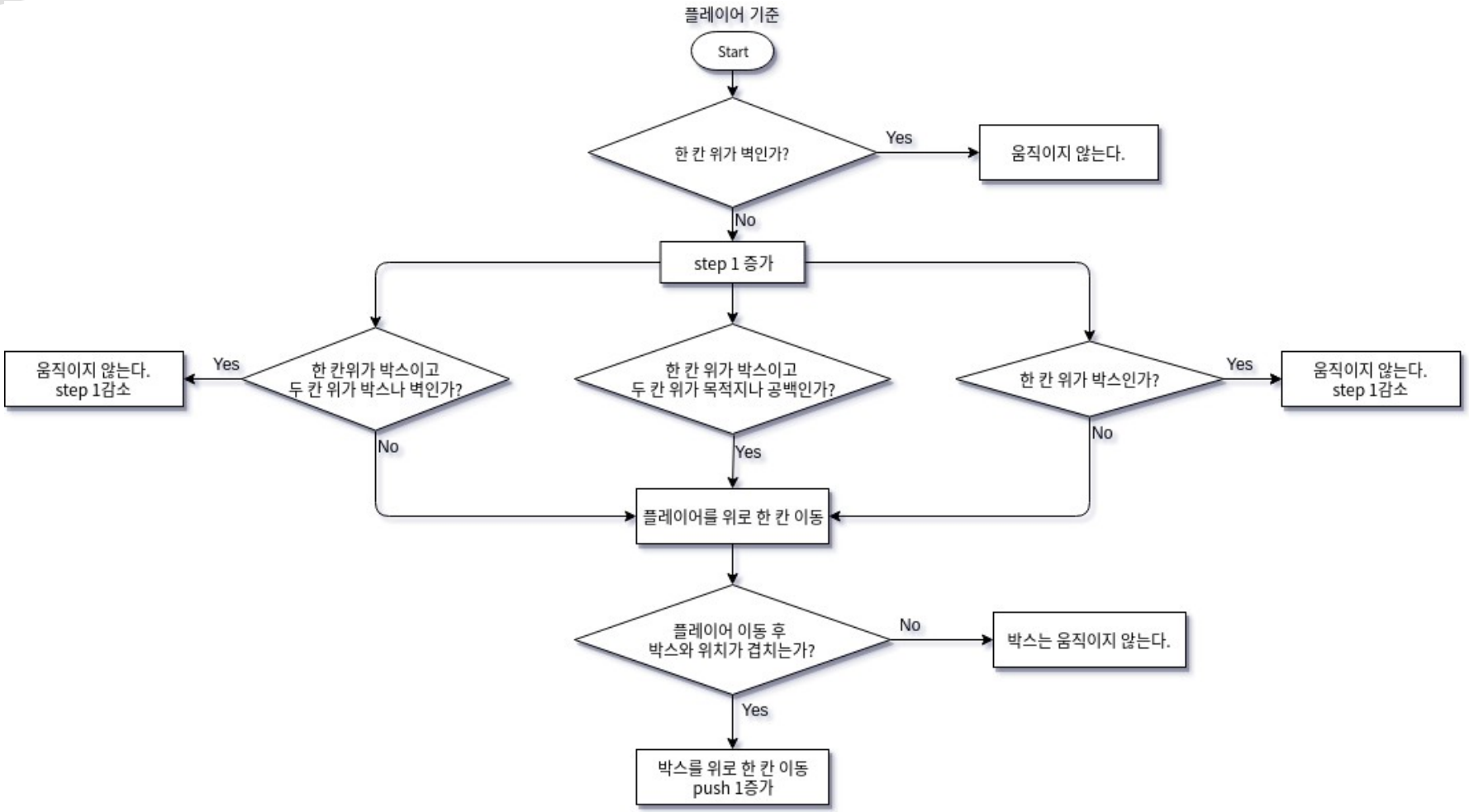
```
// ASCII : 64 = 벽(@) 79 = 박스(O)  45 = 공백  120 = 목적지(x)
switch(input){
case KEY_UP:
  if(up != 64){
    if(up == 79 && (oUp == 45 || oUp == 120)){
      obj[0].yPosition -=1;
      undobox = 0;
      for(int i = 1;i <= gameMap.nBox;i++){
        if((obj[0].yPosition == obj[i].yPosition) && (obj[0].xPosition == obj[i].xPosition)){
          obj[i].yPosition -=1;
          undobox = 'w';
          pushCnt++;
        }
      }
    }
    else if (up == 79 &&(oUp == 64 || oUp == 79))
      moveCnt--;
    else if (up != 79)
      obj[0].yPosition -=1;
    moveCnt++;
  }
  if(checkClear()){
    restart = true;
    levelUp();
    level();
  }
  undo = 'w';
  break;
```

게임 play 할 때 키를 입력받는 switch 문

case KEY_UP: // 윗 방향키
.
. (다음 페이지에서
. 알고리즘 설명)
. (모든 방향키에 동일한
. 알고리즘 적용)

checkClear 함수가 true 일 때
restart = TRUE;
levelUp() 함수 호출 //lev++
level() 다음 레벨에 해당하는 맵을 불러옴 .

GAME (Algorithm)



GAME (Undo)

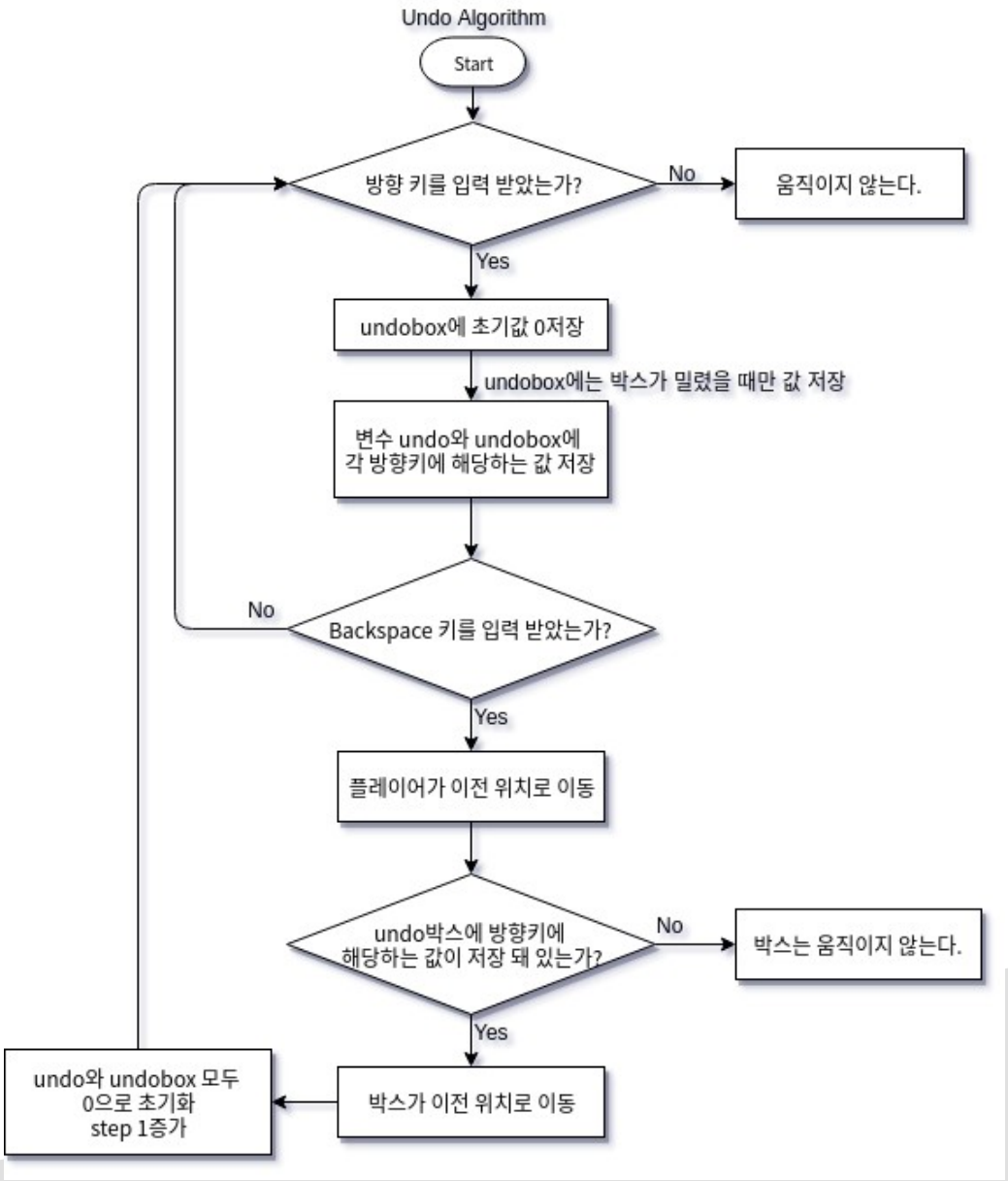
```
case KEY_BACKSPACE:
  switch(undo){
  case 'w':
    obj[0].yPosition +=1;
    for(int i=1;i<=gameMap.nBox;i++){
      if(undobox =='w' && (obj[0].yPosition == (obj[i].yPosition + 2)) && (obj[0].xPosition == obj[i].xPosition)){
        obj[i].yPosition += 1;
      }
    }
    moveCnt++;
    undo = 0;
    undobox = 0;
    break;
```

case KEY_BACKSPACE:
//Backspace 키

실행 취소기능을 구현한
swith(undo) 문

//undo
방향키를 입력 받을 때 마다 변수 undo 에 해당하는 값 저장
backspace 키를 눌렀을 때 , 플레이어를 가장 마지막으로
수행한 행동과 반대로 행동시킴 .
//undobox
가장 마지막으로 수행한 행동이 박스를 민 행동일 경우
박스의 위치도 이전 위치로 돌려놓음 .
*push, pop 을 이용하여 여러 번의 undo 를 구현 할 수 있지만
좀 더 신중하고 긴장되는 게임 플레이를 위해 1 번만 구현

GAME (Undo Algorithm)



GAME (restart)

```
  case 'r':
  case 'R':
    restart = true;
    level();
    break;
  default:
    break;
  }
  if(!restart){
    for(int i=0;i<=gameMap.nBox;i++){
      obj[i].ozn = mvinch(obj[i].yPosition,obj[i].xPosition);
      mvaddch(obj[i].yPosition,obj[i].xPosition,obj[i].zn | ((i == 0)?COLOR_PAIR(4) : COLOR_PAIR(5)));
    }
    move(obj[0].yPosition,obj[0].xPosition);
  }
  else
    restart = false;
}
```

Case ‘r’:
Case ‘R’: //’r’,’R’ 키 입력
restart 값 true 로 변경

If(!restart) //restart true 일때
초기 맵 구성을 불러와
reset 시켜준다 .

MAIN

MAIN

```
int main(){
  Game g;

  int ch;

  initscr();
  noecho();
  resize_term(100,80);
  keypad(stdscr,true);

  start_color();

  g.palette();

  curs_set(0);
```

initscr(); //ncurses TUI 모드 시작

noecho(); // getch() 함수 사용 시
입력받은 문자를 미출력

keypad(stdscr, true);
// 방향키와 Backspace 를 사용
가능하게 해준다 .

palette() //init_pair 저장하는 함수

curs_set(0); // 깜빡이는 커서를 제거

MAIN

```
border('|','|','-','-','*','*','*','*');

mvprintw(11,28,"Press Any Key To Start Game");
refresh();
getch();

g.level();
border('|','|','-','-','*','*','*','*');
refresh();
```

MAIN

```
while((ch = getch()) != 'q'){
  if(ch == 'Q')
    break;
  g.play(ch);
  attron(COLOR_PAIR(6));
  border('|','|','-','-','*','*','*','*');
  mvprintw(1,33,"move : %d",g.getMoveCnt());
  mvprintw(3,33,"push : %d",g.getPushCnt());
  mvprintw(1,1,"Restart : press R");
  mvprintw(2,1,"Undo : press Back Space");
  mvprintw(3,1,"Undo can cancel last behavior");
}

endwin();
```

while((ch = getch()) != ‘q’)
if(ch == ‘Q’)
getch() 함수를 입력받은 값이 q 일 때 프로그램이 종료된다 .

화면 상에 moveCnt 로 step 의 수를 pushCnt 로 push 의 수를 나타낸다

Restart, undo 에 대한 알림

endwin();
//ncurses TUI 모드 종료

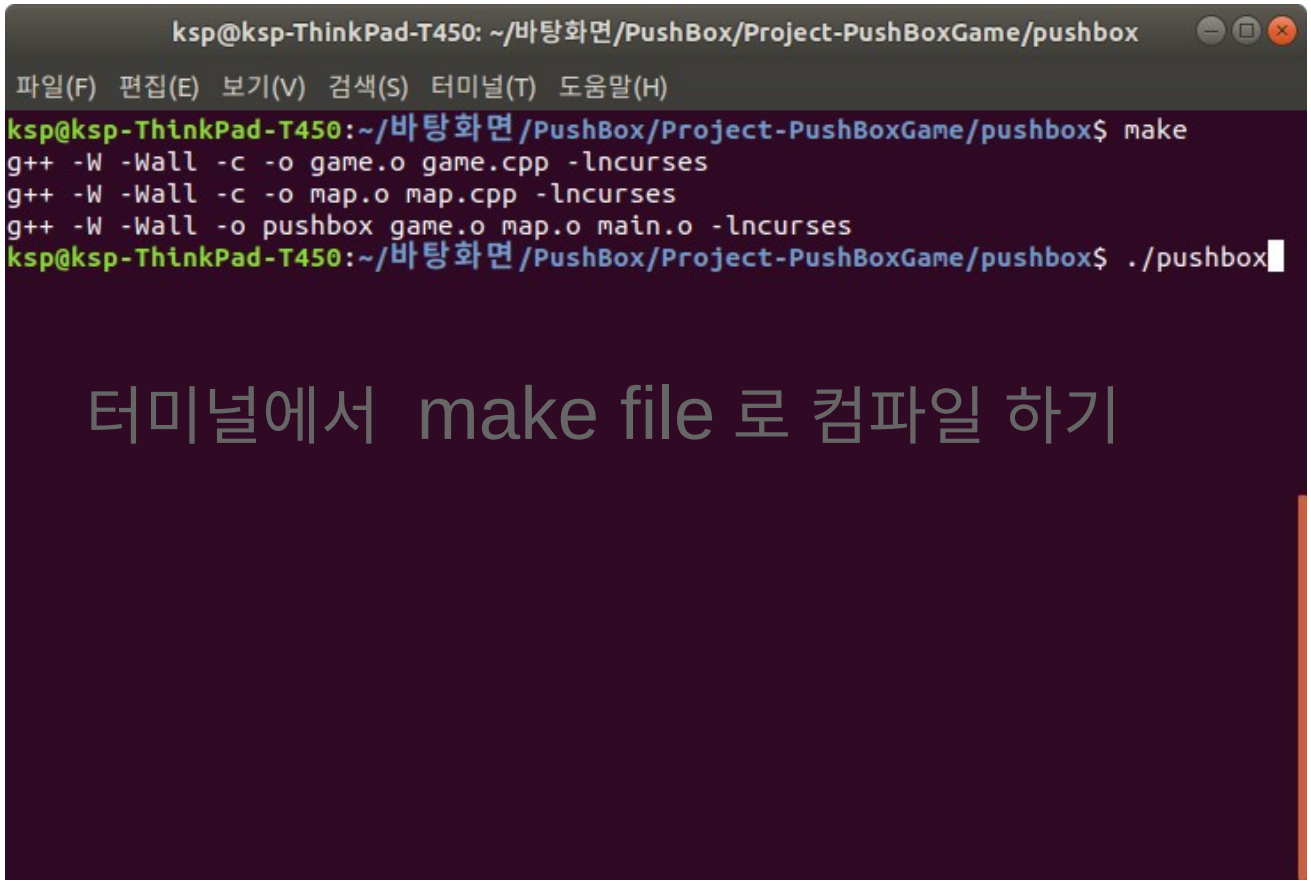
Make File

Make File

Makefile

```
all:pushbox                         //pushbox 실행파일 생성

pushbox:game.o map.o main.o    //실행파일 생성을 위한 오브젝트 파일 검사
  g++ -W -Wall -o pushbox game.o map.o main.o -lncurses

map.o:map.cpp                       //map 오브젝트 파일 생성을 위해 map.cpp 컴파일
  g++ -W -Wall -c -o map.o map.cpp -lncurses

game.o:game.cpp                     //game 오브젝트 파일 생성을 위해 game.cpp 컴파일
  g++ -W -Wall -c -o game.o game.cpp -lncurses

main.o:main.cpp                     //main 오브젝트 파일 생성을 위해 main.cpp 컴파일
  g++ -W -Wall -c -o main.o main.cpp -lncurses

clean:                  //실행파일 생성 후 필요없는 오브젝트 파일을 지워주는 명령어
  rm -rf *.o
```

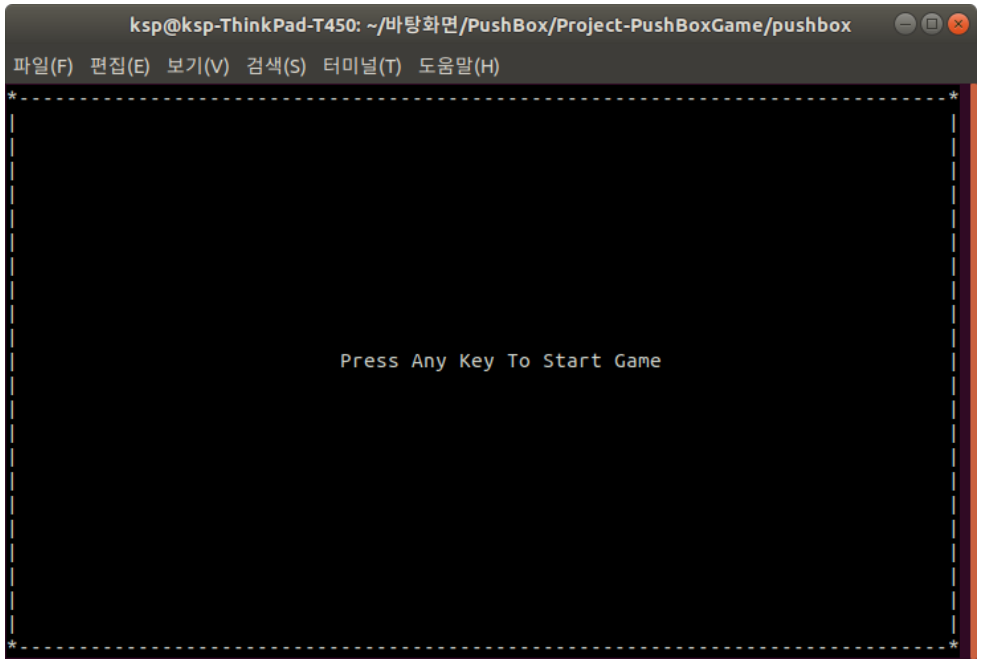
Make File


ksp@ksp-ThinkPad-T450: ~/바탕화면/PushBox/Project-PushBoxGame/pushbox
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
ksp@ksp-ThinkPad-T450:~/바탕화면/PushBox/Project-PushBoxGame/pushbox$ make
g++ -W -Wall -c -o game.o game.cpp -lncurses
g++ -W -Wall -c -o map.o map.cpp -lncurses
g++ -W -Wall -o pushbox game.o map.o main.o -lncurses
ksp@ksp-ThinkPad-T450:~/바탕화면/PushBox/Project-PushBoxGame/pushbox$ ./pushbox
터미널에서 make file 로 컴파일 하기

InGame Play

InGame Play

시작 화면



Level 1

Level 2

InGame Play

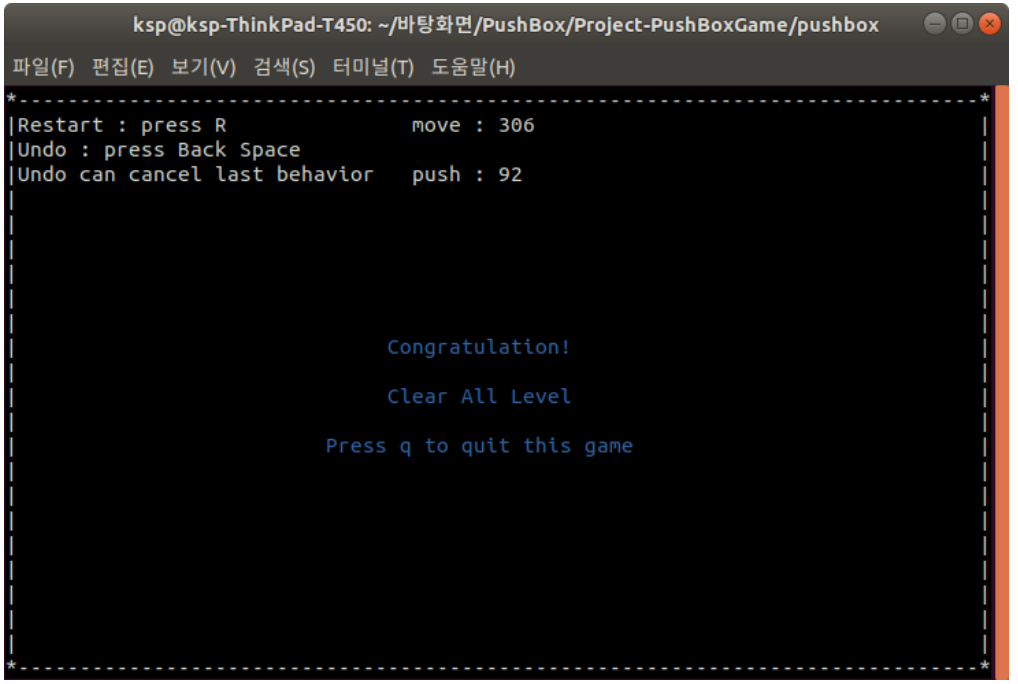
Level 3

Level 4

InGame Play

Level 5

Ending 화면



“Thank you for playing”