**프로그래밍 기초 및 실습 (나반)**

**팀 프로젝트 보고서**

(소코반 게임 제작)

logo.gif

담당 교수 : 김 명 호

제출 일자 : 2017년 6월 12일

20152994 이진영

20170328 정상아

20170340 최하영

**목차**

1. 프로그램 개요

1) 함수 요약 설명

2) 역할 분담 명시

2. 프로그램 소스

3. 실행 화면

**1. 프로그램 개요**

Github 주소 : https://github.com/mifa031/C\_Project\_2017-1/blob/master/sokoban.c

**1) 함수 요약 설명**

void readMap(int level); // 레벨에 따라 맵 로드

void printMap(); // 화면에 맵 출력

void displayHelp(); // 도움말 보여주는 함수

int getch(); // 키보드로 문자를 입력받아 그 값을 화면에 출력하지 않고 리턴

void screen\_clear(); // 화면을 지우고 ‘Hello 사용자명’ 띄우는 함수

void move(char dir); // 플레이어 이동 구현(인자로는 h,j,k,l의 방향이 들어감)

int isCleared(); // 스테이지 클리어 조건을 확인하여 클리어했으면 TRUE, 못했으면 FALSE 리턴

double time\_diff(struct timeval \*end, struct timeval \*start); // 끝-시작 시간차 리턴

void undo(); // 플레이어 이동을 취소하는 undo 명령 구현

void undo\_record(int iS\_gold\_moved, int current\_gold\_index); // undo 명령 실행에

대비해 move() 함수 내부에서 플레이어 이동 기록에 대한 저장을 수행

void load\_rank(int level); // ranking.txt 파일을 읽고 level에 따라 맵별 랭킹 출력

void save\_rank(int level); // ranking.txt 파일에 정렬된 랭킹 데이터 저장

void save\_game(); // 현재 게임 상태를 sokoban.txt 파일에 저장

void load\_game(); // sokoban.txt 파일에 저장된 데이터를 읽고 게임 저장 시행 시의 게임 상태 복구

**2) 역할 분담 명시**

① 팀원별 참여율

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 과 목 | 프로그래밍기초및실습 | | 팀프로젝트 | | 점수 |  |
| 팀 원1 | 학 번 | 20152994 | 이 름 | 이 진 영 | 참 여 율 | 50% |
| 팀 원2 | 20170328 | 정 상 아 | 30% |
| 팀 원3 | 20170340 | 최 하 영 | 20% |
| 팀 원4 | - | - | - |

② 팀원별 구현 영역

|  |  |  |
| --- | --- | --- |
| 평 가 항 목 | 구현여부 | 담당자 |
| 1. 파일로부터 맵을 정상적으로 읽어드리는가? | O | 이 진 영 |
| 2. 읽어드린 맵의 오류를 처리하는가? | O | 이 진 영 |
| 3. 사용자명을 입력받고 유지하는가? | O | 정 상 아 |
| 4. 모든 맵을 완수한 후 프로그램이 종료되는가? | O | 이 진 영 |
| 5. 순위 처리가 정확히 되는가? | O | 이 진 영 |
| 6. 프로그램이 종료되더라도 순위가 유지되는가? | O | 이 진 영 |
| 7. h, j, k, l 키로 이동이 제대로 되는가? | O | 이 진 영 |
| 8. undo 기능이 제대로 동작하는가? | O | 이 진 영 |
| 9. replay 기능이 제대로 동작하는가? | O | 정 상 아 |
| 10. new 기능이 제대로 동작하는가? | O | 정 상 아 |
| 11. exit 기능이 제대로 동작하는가? | O | 최 하 영 |
| 12. save 기능이 제대로 동작하는가? | O | 이 진 영 |
| 13. file load 기능이 제대로 동작하는가? | O | 이 진 영 |
| 14. display help 기능이 제대로 동작하는가? | O | 정 상 아 |
| 15. top 기능이 제대로 동작하는가? | O | 이 진 영 |
| 16. 맵 별 top 기능이 제대로 동작하는가? | O | 이 진 영 |
| 17. 프로그램 코드는 함수별로 설계 되었는가? | O | - |
| 18. GitHub를 잘 활용했는가? | O | - |
| 19. 출력물에 주석문, 함수 요약 설명, 역할분담 명시가 잘 되어 있는가? | O | - |

**2. 프로그램 소스**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <unistd.h>

#include <sys/time.h>

#include <termio.h>

#include <string.h>

#define TRUE 1

#define FALSE 0

#define MAX\_STAGE 5 //최대 맵 개수

#define MAX\_RC 30 //최대 row 또는 column 개수

char map[MAX\_RC][MAX\_RC]; // 메모리 상에 올려둘 map 배열

int map\_rows; // stage의 최대 행 개수

int map\_cols; // stage의 최대 열 개수

char user\_name[10];

struct timeval start, end; // 스테이지 시작 및 끝 시각

double t[MAX\_STAGE]; // 스테이지별 클리어 시간

int Px,Py; // 플레이어 위치

int stage = 1; // 현재 스테이지

int undo\_point = -1; //플레이 상태를 담은 undo배열들 중 꺼내올 index위치

int input\_index = 0; // undo배열들에 데이터를 담을 index

int undo\_Px[5] = {0};

int undo\_Py[5] = {0}; // undo를 위한 플레이어 위치 저장

int undo\_gold\_x[5] = {-1};

int undo\_gold\_y[5] = {-1};

int undo\_gold\_index[5] = {-1}; //undo를 위한 gold 위치 저장

int undo\_count = 5; // 남은 undo 실행 횟수

int slot\_x[MAX\_RC] = {0};

int slot\_y[MAX\_RC] = {0}; //slot 위치

int slot\_count=0; //slot 개수

int gold\_x[MAX\_RC] = {0};

int gold\_y[MAX\_RC] = {0}; // gold 위치

int gold\_count=0; // gold 개수

int stage\_cleared\_flag[5] = {0}; // 스테이지 클리어 표시

void readMap(int level); // 레벨에 따라 맵 로드

void printMap(); // 화면에 맵 출력

void displayHelp(); // 도움말 보여주는 함수

int getch(); // 키보드 입력을 받는 함수

void screen\_clear(); // 화면을 지우고 플레이어명 띄우는 함수

void move(char dir); // 플레이어 이동 구현 함수

int isCleared(); // 스테이지 클리어했으면 TRUE, 못했으면 FALSE 리턴

double time\_diff(struct timeval \*end, struct timeval \*start); // 끝-시작 시간차 리턴

void undo(); // 플레이어 이동 undo

void undo\_record(int iS\_gold\_moved, int current\_gold\_index); // undo하기전에 필요한 위치정보 기록

void load\_rank(int level); // 레벨에 따라 랭킹 로드

void save\_rank(int level); // 레벨에 따라 랭킹 저장

void save\_game(); // 게임 저장

void load\_game(); // 게임 로드

void main(){

char cmd; // getch()로 문자 1개를 받아올때 쓰는 변수

char cmd\_s[4]; // gets()로 문자열 받아올때 쓰는 변수

char cmd\_t[4]; // gets()로 문자열 받아올때 임시로 쓰는 변수

int cmd\_int = 0; // t명령에서 숫자값 저장할때 쓰는 변수

while(1){

printf("input name : ");

gets(user\_name);

user\_name[10] = '\0';

screen\_clear();

while(1){

for(int i=0; i<MAX\_STAGE; i++){ // flag를 조사하여 클리어한 스테이지는 skip

if(stage\_cleared\_flag[stage-1] == 1){

stage++;

}

if(stage > MAX\_STAGE){ // 마지막 스테이지까지 클리어했다면,

system("clear"); // 축하메시지 출력 후 종료

printf("\nCongratulations!!!\nAll STAGE CLEARED !!!\n");

sleep(10);

exit(0);

}

}

stage\_start:

screen\_clear();

readMap(stage); // stage에 맞게 맵을 읽어온다.

load\_point:

gettimeofday(&start,NULL); // 시간 측정 시작

if(cmd != 'f')

cmd = ' ';

while(1){

if(isCleared()){ // 스테이지 클리어 여부 확인

stage\_cleared\_flag[stage-1] = 1;

break; //클리어 시 flag를 1로 set하고 break

}

if(cmd == ' ' || cmd == 'h' || cmd == 'j' || cmd == 'k' || cmd == 'l' || cmd =='r' || cmd=='u'||cmd=='s'||cmd=='f'){

// 화면에 맵이 출력되어 있을때의 명령입력부분

screen\_clear();

printMap();

if(cmd == 'r' || cmd == 'u' || cmd == 's'||cmd == 'f'){

printf("\n(Command) %c",cmd);

}else{

printf("\n(Command) ");

}

cmd = getch();

if(cmd == 't'){

screen\_clear();

printMap();

printf("\n(Command) %c",cmd);

cmd\_s[0] = cmd;

gets(cmd\_t);

cmd\_s[1] = cmd\_t[0];

cmd\_s[2] = cmd\_t[1];

cmd\_s[3] = cmd\_t[2];

cmd = cmd\_s[0];

}

}else{ // 화면에 맵이 출력되지 않은 경우 명령입력부분

printf("\n(Command) ");

gets(cmd\_s);

cmd = cmd\_s[0];

}

switch(cmd){

case ' ':

case 'h':

case 'j':

case 'k':

case 'l': //방향 이동 구현

move(cmd);

break;

case 'd': //dispaly help

screen\_clear();

displayHelp();

break;

case 'n': //현재까지의 시간기록 삭제 후 첫번째 맵부터 다시시작

system("clear");

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

t[i] = 0;

stage = 1;

goto stage\_start;

case 'e': //현재 상태 파일에 저장하고 종료

system("clear");

printf("\n\n\n\nS E E Y O U %s . . . .\n\n\n\n",user\_name);

printf("(Command) e\n");

gettimeofday(&end,NULL);

t[stage-1] += time\_diff(&end, &start);

save\_game();

sleep(3);

exit(0);

break;

case 't': // 맵별 순위 출력

screen\_clear();

cmd\_int = 0;

if(cmd\_s[1]-48 > 0 && cmd\_s[1]-48 < 6){

switch(cmd\_s[1]){

case '1':

cmd\_int = 1;

break;

case '2':

cmd\_int = 2;

break;

case '3':

cmd\_int = 3;

break;

case '4':

cmd\_int = 4;

break;

case '5':

cmd\_int = 5;

break;

default:

cmd\_int = 0;

}

}else if(cmd\_s[2]-48 > 0 && cmd\_s[2]-48 < 6){

switch(cmd\_s[2]){

case '1':

cmd\_int = 1;

break;

case '2':

cmd\_int = 2;

break;

case '3':

cmd\_int = 3;

break;

case '4':

cmd\_int = 4;

break;

case '5':

cmd\_int = 5;

break;

default:

cmd\_int = 0;

}

}

load\_rank(cmd\_int);

cmd\_int = 0;

for(int i=0; i<4; i++)

cmd\_s[i] = '\0';

break;

case 'r': //게임시간 유지하며 현재 맵 재시작

screen\_clear();

readMap(stage);

break;

case 'u': // 사용자 이동 undo

undo();

break;

case 's': // 게임 저장

gettimeofday(&end,NULL);

t[stage-1] += time\_diff(&end, &start);

save\_game();

break;

case 'f': // 게임 로드

load\_game();

goto load\_point;

}

}

// stage 클리어 시 처음 진입하는 부분

gettimeofday(&end,NULL);

t[stage-1] += time\_diff(&end, &start);

save\_rank(stage);

//걸린 시간을 저장하고, 랭킹 기록

}

}

}

void load\_game(){

FILE\* file;

file = fopen("sokoban.txt","r");

fscanf(file,"%d\n",&stage);

readMap(stage); // 파일에 저장된 스테이지 블러옴

map[Py][Px] = ' '; // 추후 @및 $의 위치 변경을 위해 초기위치 공백처리

for(int i=0; i<gold\_count; i++){

map[gold\_y[i]][gold\_x[i]] = ' ';

}

//파일에 저장된 전역변수들 로드

fscanf(file,"%s\n",&user\_name);

for(int i=0; i<MAX\_STAGE; i++)

fscanf(file,"%lf\n",&t[i]);

fscanf(file,"%d\n",&Px);

fscanf(file,"%d\n",&Py);

fscanf(file,"%d\n",&undo\_point);

fscanf(file,"%d\n",&input\_index);

for(int i=0; i<5; i++)

fscanf(file,"%d\n",&undo\_Px[i]);

for(int i=0; i<5; i++)

fscanf(file,"%d\n",&undo\_Py[i]);

for(int i=0; i<5; i++)

fscanf(file,"%d\n",&undo\_gold\_x[i]);

for(int i=0; i<5; i++)

fscanf(file,"%d\n",&undo\_gold\_y[i]);

for(int i=0; i<5; i++)

fscanf(file,"%d\n",&undo\_gold\_index[i]);

fscanf(file,"%d\n",&undo\_count);

fscanf(file,"%d\n",&slot\_count);

fscanf(file,"%d\n",&gold\_count);

for(int i=0; i<slot\_count; i++)

fscanf(file,"%d\n",&slot\_x[i]);

for(int i=0; i<slot\_count; i++)

fscanf(file,"%d\n",&slot\_y[i]);

for(int i=0; i<gold\_count; i++)

fscanf(file,"%d\n",&gold\_x[i]);

for(int i=0; i<gold\_count; i++)

fscanf(file,"%d\n",&gold\_y[i]);

for(int i=0; i<MAX\_STAGE; i++)

fscanf(file,"%d\n",&stage\_cleared\_flag[i]);

fclose(file);

}

void save\_game(){

FILE\* file;

file = fopen("sokoban.txt","w");

// 게임 save에 필요한 전역변수들을 파일에 저장

fprintf(file,"%d\n",stage);

fprintf(file,"%s\n",user\_name);

for(int i=0; i<MAX\_STAGE; i++)

fprintf(file,"%lf\n",t[i]);

fprintf(file,"%d\n",Px);

fprintf(file,"%d\n",Py);

fprintf(file,"%d\n",undo\_point);

fprintf(file,"%d\n",input\_index);

for(int i=0; i<5; i++)

fprintf(file,"%d\n",undo\_Px[i]);

for(int i=0; i<5; i++)

fprintf(file,"%d\n",undo\_Py[i]);

for(int i=0; i<5; i++)

fprintf(file,"%d\n",undo\_gold\_x[i]);

for(int i=0; i<5; i++)

fprintf(file,"%d\n",undo\_gold\_y[i]);

for(int i=0; i<5; i++)

fprintf(file,"%d\n",undo\_gold\_index[i]);

fprintf(file,"%d\n",undo\_count);

fprintf(file,"%d\n",slot\_count);

fprintf(file,"%d\n",gold\_count);

for(int i=0; i<slot\_count; i++)

fprintf(file,"%d\n",slot\_x[i]);

for(int i=0; i<slot\_count; i++)

fprintf(file,"%d\n",slot\_y[i]);

for(int i=0; i<gold\_count; i++)

fprintf(file,"%d\n",gold\_x[i]);

for(int i=0; i<gold\_count; i++)

fprintf(file,"%d\n",gold\_y[i]);

for(int i=0; i<MAX\_STAGE; i++)

fprintf(file,"%d\n",stage\_cleared\_flag[i]);

fclose(file);

}

void load\_rank(int level){ // level 0은 전체 순위, 1~5는 맵별 순위 출력

FILE\* file;

char temp\_rank[50][50];

char name[5][10];

float time[5];

char current\_map[6];

int user\_num = 0;

char temp\_str[50]; // 파일내 데이터들을 임시로 저장하는 변수들

int is\_no\_user\_map[50]; // 랭킹 기록자가 없는 map은 skip하는데 사용

for(int i=0; i<50; i++){

is\_no\_user\_map[i] = -1;

}

char\* temp\_char;

if((file = fopen("ranking.txt","r")) == NULL){

screen\_clear();

printf("저장된 랭킹 데이터가 없습니다.\n");

sleep(3);

}else if(level == 0){ // 전체 맵 랭킹 출력

int i = 0;

while(!feof(file)){

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(feof(file))

break;

for(char ascii=49; ascii<=53; ascii++){

if((temp\_rank[i][4] == ascii) && (temp\_rank[i][5] == '\n')){

i++;

temp\_char = fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(((temp\_rank[i][4] == (ascii+1)) && (temp\_rank[i][5] == '\n')||

temp\_char == NULL)){

is\_no\_user\_map[i-1] = 1;

is\_no\_user\_map[i] = 1;

break;

}else{

break;

}

}

}

i++;

}

screen\_clear();

for(int a=0; a<i; a++){

if(is\_no\_user\_map[a] < 0)

printf("%s",temp\_rank[a]);

}

}else if(level > 0 && level < 6){ // 맵별 랭킹 출력

int i = 0;

int map\_level = level+48;

char temp\_char2[3];

while(!feof(file)){

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(temp\_rank[i][4] == map\_level && temp\_rank[i][5] == '\n'){

strcpy(current\_map,temp\_rank[i]);

int j = 0;

while(!feof(file)){

fscanf(file,"%s %f%s\n",&name[j],&time[j],&temp\_char2);

user\_num++;

j++;

i++;

if(strcmp(temp\_char2,"sec") != 0){

break;

}

}

break;

}

i++;

}

printf("%s",current\_map);

for(int a=0; a<user\_num-1; a++)

printf("%s %.1fsec\n",name[a],time[a]);

}

}

void save\_rank(int level){

FILE\* file;

FILE\* file2;

char temp\_rank[50][50];

char name[6][10];

float time[6];

char ch\_level = level + 48;

char map\_exp[7] = "map \n";

map\_exp[3] = ' ';

map\_exp[4] = ch\_level;

char temp\_char[4];

int user\_num = 0; // 랭킹파일 데이터를 임시로 저장하는 변수들

if((file = fopen("ranking.txt","r+")) == NULL){

file = fopen("ranking.txt","a+");

}

char tmp\_str[50];

int map\_exp\_check[MAX\_STAGE] = {0};

char tmp\_map\_exp[7] = "map \n";

while(!feof(file)){ // 'map 1' 같은 맵 표시가 있는지 체크, 없으면 0

fgets(tmp\_str,sizeof(tmp\_str),file); //(추후 맵표시를 참고하여 랭킹저장)

for(int a=0; a<MAX\_STAGE; a++){

tmp\_map\_exp[4] = a+49;

if(strcmp(tmp\_str,tmp\_map\_exp) == 0){

map\_exp\_check[a] = a;

}

}

}

fclose(file);

file = fopen("ranking.txt","a"); // 맵 표시가 없으면 추가

for(int a=0; a<MAX\_STAGE; a++){

if(map\_exp\_check[a] == 0){

tmp\_map\_exp[4] = a+49;

fputs(tmp\_map\_exp,file);

}

}

fclose(file);

if((file = fopen("ranking.txt","r+")) == NULL){

file = fopen("ranking.txt","a+");

}

// 레벨에 맞는 플레이어 데이터를 불러온다.

int i = 0;

while(!feof(file)){

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(temp\_rank[i][4] == map\_exp[4] && temp\_rank[i][5] == '\n'){

int j = 0;

while(!feof(file)){

fscanf(file,"%s %f%s\n",&name[j],&time[j],&temp\_char);

j++;

i++;

if(strcmp(temp\_char,"sec") != 0){

user\_num = j-1;

break;

}

}

break;

}

i++;

}

// 현재 플레이어의 데이터를 추가 한다.

user\_num++;

strcpy(name[user\_num-1],user\_name);

time[user\_num-1] = t[stage-1];

// 플레이어 데이터를 시간 기준 오름차순으로 정렬

char temp\_name[10];

float temp\_time;

for(int i=0; i<user\_num; i++){

for(int j=0; j<user\_num; j++){

if(time[i] < time[j]){

strcpy(temp\_name,name[j]);

temp\_time = time[j];

strcpy(name[j],name[i]);

time[j] = time[i];

strcpy(name[i],temp\_name);

time[i] = temp\_time;

}

}

}

if(user\_num > 5)

user\_num = 5; // 최대 5명까지만 보기위한 세팅

fclose(file);

// 바뀐 랭킹 데이터를 다시 쓴다. (ranking2에 임시저장)

file = fopen("ranking.txt","r");

file2 = fopen("ranking2.txt","w");

i = 0;

while(!feof(file)){

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(feof(file))

break;

if(temp\_rank[i][4] == map\_exp[4] && temp\_rank[i][5] == '\n'){

fputs(temp\_rank[i],file2);

i++;

for(int j=0; j<user\_num; j++){

fprintf(file2,"%s %.1fsec\n",name[j],time[j]);

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

}

break;

}

fputs(temp\_rank[i],file2);

i++;

}

for(int i=0; i<50; i++){

for(int j=0; j<50; j++){

temp\_rank[i][j] = '\0';

}

}

fclose(file);

fclose(file2);

file = fopen("ranking.txt","r");

file2 = fopen("ranking2.txt","a");

i = 0;

int j = 1;

while(!feof(file)){

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(feof(file))

break;

if(stage == 5)

j = 0;

if(temp\_rank[i][4] == (map\_exp[4]+j) && temp\_rank[i][5] == '\n'){

fputs(temp\_rank[i],file2);

i++;

while(!feof(file)){

fgets(temp\_rank[i],sizeof(temp\_rank[i]),file);

if(temp\_rank[i][4] == '1' && temp\_rank[i][5] == '\n')

break;

fputs(temp\_rank[i],file2);

i++;

}

break;

}

}

for(int i=0; i<50; i++){

for(int j=0; j<50; j++){

temp\_rank[i][j] = '\0';

}

}

fclose(file);

fclose(file2);

//ranking2의 내용을 ranking파일로 복사한뒤 ranking2 삭제하고 종료

file = fopen("ranking.txt","w");

file2 = fopen("ranking2.txt","r");

while(!feof(file2)){

fgets(temp\_rank[i],sizeof(temp\_rank),file2);

if(feof(file2))

break;

fputs(temp\_rank[i],file);

i++;

}

fclose(file);

fclose(file2);

remove("ranking2.txt");

}

void undo\_record(int is\_gold\_moved, int current\_gold\_index){

// 1번째 undo 기록 (사용자만 움직였을 경우 1번째 undo만 기록)

if(input\_index < undo\_count && !is\_gold\_moved){

undo\_Px[input\_index] = Px;

undo\_Py[input\_index] = Py;

undo\_gold\_x[input\_index] = -1;

undo\_gold\_y[input\_index] = -1;

undo\_gold\_index[input\_index] = -1;

undo\_point = input\_index;

}else if(input\_index >= undo\_count && !is\_gold\_moved){

for(int i=0; i<undo\_count-1; i++){

undo\_Px[i] = undo\_Px[i+1];

undo\_Py[i] = undo\_Py[i+1];

}

undo\_Px[undo\_count-1] = Px;

undo\_Py[undo\_count-1] = Py;

for(int i=0; i<undo\_count-1; i++){

undo\_gold\_x[i] = undo\_gold\_x[i+1];

undo\_gold\_y[i] = undo\_gold\_y[i+1];

undo\_gold\_index[i] = undo\_gold\_index[i+1];

}

undo\_gold\_x[undo\_count-1] = -1;

undo\_gold\_y[undo\_count-1] = -1;

undo\_gold\_index[undo\_count-1] = -1;

undo\_point = undo\_count-1;

}

// 만약 gold의 움직임이 있다면 2번째 undo 기록 수행

if(input\_index < undo\_count && is\_gold\_moved){

undo\_gold\_x[input\_index] = gold\_x[current\_gold\_index];

undo\_gold\_y[input\_index] = gold\_y[current\_gold\_index];

undo\_gold\_index[input\_index] = current\_gold\_index;

}else if(input\_index >= undo\_count && is\_gold\_moved){

undo\_gold\_x[undo\_count-1] = gold\_x[current\_gold\_index];

undo\_gold\_y[undo\_count-1] = gold\_y[current\_gold\_index];

undo\_gold\_index[undo\_count-1] = current\_gold\_index;

}

if(input\_index < undo\_count)

input\_index++;

}

void undo(){

if((undo\_count > 0) && (undo\_point >= 0)){ // undo\_count가 남아 있다면

map[Py][Px] = ' ';

Px = undo\_Px[undo\_point];

Py = undo\_Py[undo\_point]; // 플레이어 위치 undo

if(undo\_gold\_index[undo\_point] >= 0){

map[gold\_y[undo\_gold\_index[undo\_point]]][gold\_x[undo\_gold\_index[undo\_point]]] = ' ';

gold\_x[undo\_gold\_index[undo\_point]] = undo\_gold\_x[undo\_point];

gold\_y[undo\_gold\_index[undo\_point]] = undo\_gold\_y[undo\_point]; //gold 위치 undo

}

undo\_point--;

undo\_count--;

}

}

double time\_diff(struct timeval \*end, struct timeval \*start){

double diff\_sec = difftime(end->tv\_sec,start->tv\_sec); // 초단위 시간차

double diff\_milsec = end->tv\_usec - start->tv\_usec; // 나노초 단위 시간차

diff\_milsec /= (double) 1000000; //나노초를 밀리초 단위로 변경

diff\_sec += diff\_milsec; // 초단위 + 밀리초 단위

return diff\_sec; // 밀리초까지 정확한 시간차

}

int isCleared(){

int matched\_count = 0;

for(int i=0; i<slot\_count; i++){

for(int j=0; j<gold\_count; j++){

if(slot\_x[i] == gold\_x[j] && slot\_y[i] == gold\_y[j])

matched\_count++; // slot과 gold 위치가 같다면 matched\_count 증가

}

}

if(matched\_count == slot\_count) // 서로 위치가 같은 개수가 slot개수와 일치하면

return TRUE;

return FALSE;

}

void move(char dir){

if(dir == ' '){

}else{

undo\_record(FALSE,FALSE);

map[Py][Px] = ' '; // 다음 이동을 위해 현재 @를 화면에서 지움

switch(dir){

case 'h':

Px = Px - 1; // @ 이동 구현

if(map[Py][Px] == '#') //벽을 만나면 @ 이동 취소

Px = Px + 1;

if(map[Py][Px] == '$'){ //@가 이동한 곳에 $가 있다면

for(int i=0; i<gold\_count; i++){

if((gold\_x[i] == Px) && (gold\_y[i] == Py)){ // 해당 $의 index를 찾아내서

undo\_record(TRUE,i);

gold\_x[i] = gold\_x[i] - 1; // @와 동일한 방향으로 이동

if((map[gold\_y[i]][gold\_x[i]] == '#') || //$가 움직인 방향에 #이나

(map[gold\_y[i]][gold\_x[i]] == '$')){ // $가 있다면

gold\_x[i] = gold\_x[i] + 1; // $ 및 @ 이동 취소

Px = Px + 1;

}

}

}

}

break;

case 'j':

Py = Py + 1;

if(map[Py][Px] == '#')

Py = Py - 1;

if(map[Py][Px] == '$'){

for(int i=0; i<gold\_count; i++){

if((gold\_x[i] == Px) && (gold\_y[i] == Py)){

undo\_record(TRUE,i);

gold\_y[i] = gold\_y[i] + 1;

if((map[gold\_y[i]][gold\_x[i]] == '#') ||

(map[gold\_y[i]][gold\_x[i]] == '$')){

gold\_y[i] = gold\_y[i] - 1;

Py = Py - 1;

}

}

}

}

break;

case 'k':

Py = Py - 1;

if(map[Py][Px] == '#')

Py = Py + 1;

if(map[Py][Px] == '$'){

for(int i=0; i<gold\_count; i++){

if((gold\_x[i] == Px) && (gold\_y[i] == Py)){

undo\_record(TRUE,i);

gold\_y[i] = gold\_y[i] - 1;

if((map[gold\_y[i]][gold\_x[i]] == '#') ||

(map[gold\_y[i]][gold\_x[i]] == '$')){

gold\_y[i] = gold\_y[i] + 1;

Py = Py + 1;

}

}

}

}

break;

case 'l':

Px = Px + 1;

if(map[Py][Px] == '#')

Px = Px - 1;

if(map[Py][Px] == '$'){

for(int i=0; i<gold\_count; i++){

if((gold\_x[i] == Px) && (gold\_y[i] == Py)){

undo\_record(TRUE,i);

gold\_x[i] = gold\_x[i] + 1;

if((map[gold\_y[i]][gold\_x[i]] == '#') ||

(map[gold\_y[i]][gold\_x[i]] == '$')){

gold\_x[i] = gold\_x[i] - 1;

Px = Px - 1;

}

}

}

}

break;

}

}

}

void screen\_clear(){

system("clear");

printf("Hello %s\n\n",user\_name);

}

void displayHelp(){

printf("h(왼쪽), j(아래), k(위), l(오른쪽)\n");

printf("u(undo)\n");

printf("r(replay)\n");

printf("n(new)\n");

printf("e(exit)\n");

printf("s(save)\n");

printf("f(file load)\n");

printf("d(display help)\n");

printf("t(top)\n");

}

void readMap(int level){

FILE\* map\_file = NULL;

int map\_level = 0;

int row=0, col=0;

int temp\_col = 0;

char temp\_char = 0;

char temp\_char2 = 0;

slot\_count = 0;

gold\_count = 0;

undo\_count = 5;

// 파일을 읽어서 stage의 크기를 알아냄

map\_file = fopen("map.txt","r");

while((temp\_char = fgetc(map\_file)) != EOF){

if(temp\_char == 'm'|| temp\_char == 'e'){

map\_level++;

}

if(map\_level != level)

continue;

temp\_col++;

if(temp\_char == 0x0D){

temp\_char = fgetc(map\_file);

temp\_col++;

row++;

if(temp\_col > col){

col = temp\_col;

}

temp\_col=0;

}

else if(temp\_char == 0x0A){

row++;

if(temp\_col > col){

col = temp\_col;

}

temp\_col=0;

}

}

map\_rows = row;

map\_cols = col;

fclose(map\_file);

// map 배열 적당히 초기화 한 후, 파일 내용을 읽어옴

for(int i=0; i<MAX\_RC; i++)

for(int j=0; j<MAX\_RC; j++)

map[i][j] = ' ';

map\_file = fopen("map.txt","r");

map\_level = 0;

for(int i=0; i<map\_rows; i++){

for(int j=0; j<map\_cols; j++){

map[i][j] = fgetc(map\_file);

if(map[i][j] == 0x0D){ //윈도우환경 개행(\r\n)에 대비한 루틴

continue;

}

if(map[i][j] == 0x0A && map[i][j-1] == 0x0D ){

map[i][j-1] = ' ';

map[i][j] = 0x0A;

}

if(map[i][j] == '@'){ // 초기 플레이어 위치 세팅

Px = j;

Py = i;

}

if(map[i][j] == 'O'){ // 초기 slot 위치 및 개수 알아내기

slot\_x[slot\_count] = j;

slot\_y[slot\_count] = i;

slot\_count++;

}

if(map[i][j] == '$'){ // 초기 gold 위치 및 개수 알아내기

gold\_x[gold\_count] = j;

gold\_y[gold\_count] = i;

gold\_count++;

}

if(map[i][j] == 'm'){

map[i][j] = fgetc(map\_file);

map\_level++;

}

if(map\_level != level){ // 현재 level에 맞는 map인지 확인

i=0; j=-1;

slot\_count = 0;

gold\_count = 0;

continue;

}

if(map[i][j] == 'a') // 상단에 map표시 안읽어오도록

map[i][j] = fgetc(map\_file);

if(map[i][i] == 'p')

map[i][j] = fgetc(map\_file);

if(map[i][j] == '\n'){ // map배열에 적절한 공백 및 개행처리

for(; j<map\_cols-1; j++)

map[i][j] = ' ';

if(map[i][j] == ' ')

map[i][j] = '\n';

}

}

}

if(gold\_count != slot\_count){ // gold와 slot 개수가 안맞으면 종료

printf("$와 0의 개수가 같지않아 종료합니다.\n");

exit(1);

}

fclose(map\_file);

}

void printMap(){

map[Py][Px] = '@'; // 플레이어의 이동이 현재 구현된 위치에 @ 출력

int row, col;

for(int k=0; k<slot\_count; k++){ // @의 이동으로 인해 slot 위치가 공백이 되지않도록

if(map[slot\_y[k]][slot\_x[k]] == ' ')

map[slot\_y[k]][slot\_x[k]] = 'O';

}

// @의 이동으로 인해 gold 위치가 공백이 되지않고 slot위에 gold가 올라가게 보일 수 있도록

for(int k=0; k<gold\_count; k++){

if(map[gold\_y[k]][gold\_x[k]] == ' '|| map[gold\_y[k]][gold\_x[k]] == 'O')

map[gold\_y[k]][gold\_x[k]] = '$';

}

for(int i=0; i<map\_rows; i++){

for(int j=0; j<map\_cols; j++){

putchar(map[i][j]); // 현재 상태의 map 전체 출력

}

}

printf("\n");

}

int getch(){ // 키보드로 입력받은 변수를 화면에 출력하지않고 리턴

int ch;

struct termios buf;

struct termios save;

tcgetattr(0, &save);

buf = save;

buf.c\_lflag&=~(ICANON|ECHO);

buf.c\_cc[VMIN] = 1;

buf.c\_cc[VTIME] = 0;

tcsetattr(0, TCSAFLUSH, &buf);

ch = getchar();

tcsetattr(0, TCSAFLUSH, &save);

return ch;

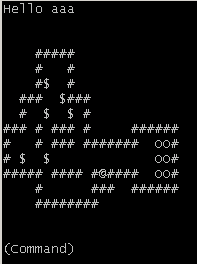
}

**3. 실행 화면**

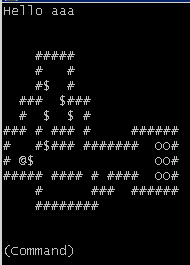
① 게임 파일 실행

1.PNG

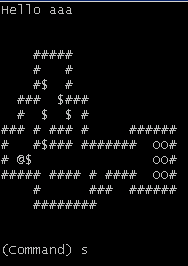
② 게임 실행 초기화면



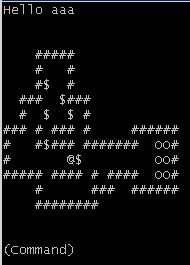
③ 방향키로 플레이어 이동



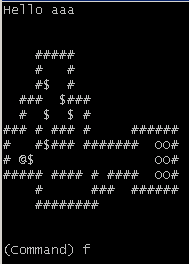
④ 게임 저장



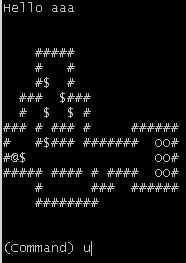
⑤ 플레이어 이동



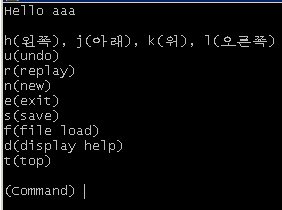
⑥ 게임 로드



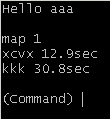
⑦ undo 실행



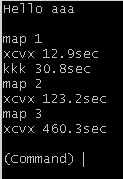
⑧ display help 실행



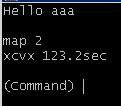
⑨ t 1 실행



⑩ t 실행



⑪ t2 실행 (t 뒤에 바로 숫자를 붙여도, 한 칸 띄고 숫자를 붙여도, 인식 가능)



⑫ exit 실행

