MazeRPG(project)

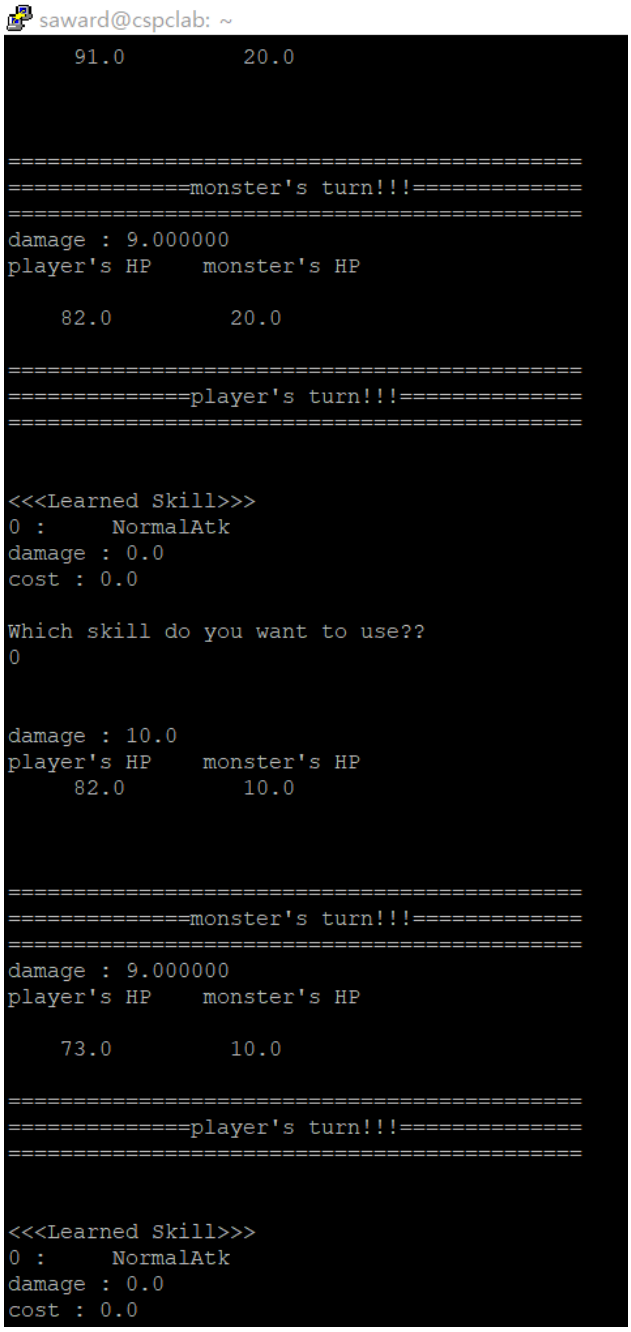
컴퓨터공학과 20171646 박태윤

컴퓨터공학과 20171686 장은상

1. 목표

maze를 기반으로 한 RPG게임을 만든다. 미로에는 벽과 통로, 몬스터와 아이템 상자가 존재한다. 플레이어는 커맨드 입력을 통해 이동, 전투 시 스킬을 사용할 수 있으며 만약 플레이어의 체력이 0이하가 되면 게임 오버로 프로그램이 종료된다. 출구에 존재하는 최종 몬스터를 처치하면 게임을 클리어 할 수 있으며 클리어 시 미로의 정보, 처치 했던 몬스터의 정보, 플레이어의 스탯을 출력한다.

2. 실행 결과



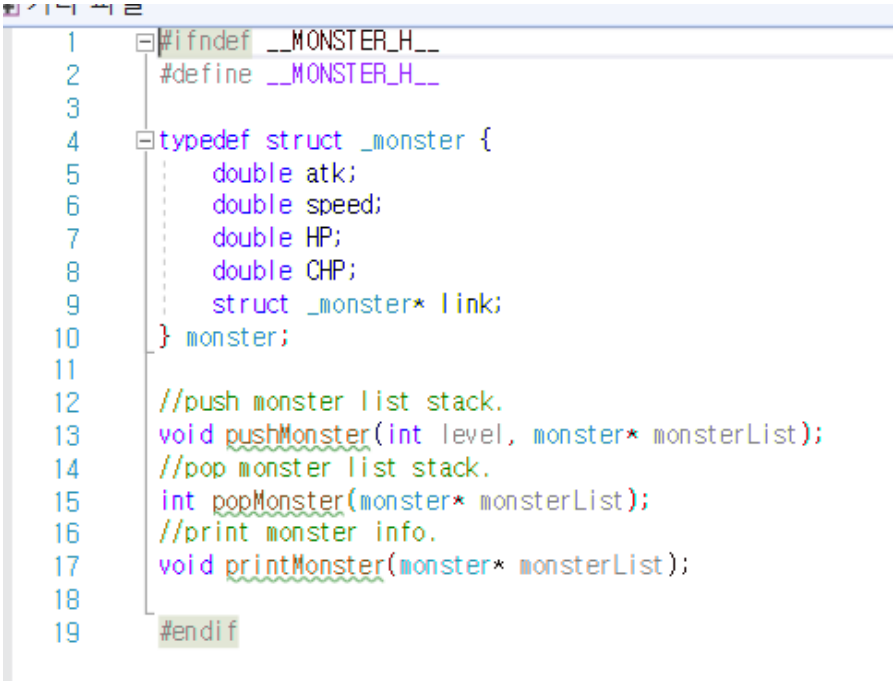
saward@cspclab: ~

```
win!!!!!!!!!!!!!!


==========================================
=============player's status=============
level : 1
attack : 10.0
HP : 100.0
speed : 10.0
==========================================

The number : 1
Attack : 3
HP : 9
Speed : 9


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

------------------------------------
            clear!!!!!!
------------------------------------
```

3. 코드 및 알고리즘

- 헤더 파일 및 함수

(1) monster.h : 몬스터 관련한 함수들이 존재하는 헤더파일이다.



```
1   #ifndef __MONSTER_H__
2   #define __MONSTER_H__
3
4   typedef struct _monster {
5       double atk;
6       double speed;
7       double HP;
8       double CHP;
9       struct _monster* link;
10  } monster;
11
12  //push monster list stack.
13  void pushMonster(int level, monster* monsterList);
14  //pop monster list stack.
15  int popMonster(monster* monsterList);
16  //print monster info.
17  void printMonster(monster* monsterList);
18
19  #endif
```

monster구조체를 선언하였다. 구조체 안에는 몬스터의 공격력인 atk, 스피드인 speed, 체력인 HP, 현재 체력인 CHP가 존재하며 링크를 선언하였다.

-> pushMonster : monsterList스택에 플레이어가 몬스터를 마주칠 때 마다 해당 몬스터의 정보를 push해준다.

-> popMonster : monsterList스택에 들어있는 노드들을 Pop해주면서 해당 노드를 리턴해준다.

-> printMonster : 몬스터의 정보를 출력해준다.

(2) player.h : 플레이어 관련한 함수들이 존재하는 헤더파일이다.

```c
#ifndef __PLAYER_H__
#define __PLAYER_H__
#include <stdio.h>

//status
typedef struct _status {
    double atk;
    double speed;
    double HP;
    double CHP;
    int level;
} stat;

typedef struct _item {
    stat s;
    char* name;
    struct _item* link;
} item;

typedef struct _skill {
    stat s;
    char* name;
} skill;

typedef struct _player {
    stat s;
    //itemlist
    item* pItem;
    //skill list
    skill* pSkill;
    int skillNumber;
} player;

//text file to itemList.
void itemInput(FILE* ifp, item* itemList);
//text file to skillList.
skill* skillInput(FILE* sfp);
//assign one item to player.
void getItem(player* p, item* itemList);
//assign one skill to player, level up.
void getSkill(player* p, skill* skillList);


//show skill list
void showSkill(player* p);
//show item list
void showItem(player* p);
//initiate player stat
void playerInit(player*, skill*);
//print player info.
void printPlayer(player* p);


#endif
```

구조체로 stat, item, skill, player를 선언하였다. stat은 플레이어와 아이템, 스킬의 스탯을 나타내며 item은 스탯, 이름, 링크를 가진다. skill은 스탯과 이름을 가지며 player는 스탯, 아이템창, 스킬창, 스킬의 번호를 가진다.

-> itemInput : 텍스트 파일에 들어있는 아이템의 정보를 itemList에 넣어준다.

-> skillInput : 텍스트 파일에 들어있는 스킬의 정보를 skillList에 넣어준다.

-> getItem : 플레이어가 아이템 상자를 발견할 시 itemList에 있는 아이템 중 하나를 랜덤으로 골라 플레이어에게 할당한다.

-> getSkill : 플레이어가 몬스터를 처치하고 레벨업을 할 시 트리로 구현된 skillList에 있는 스킬 중 하나를 배울 수 있게 하는 역할을 한다.

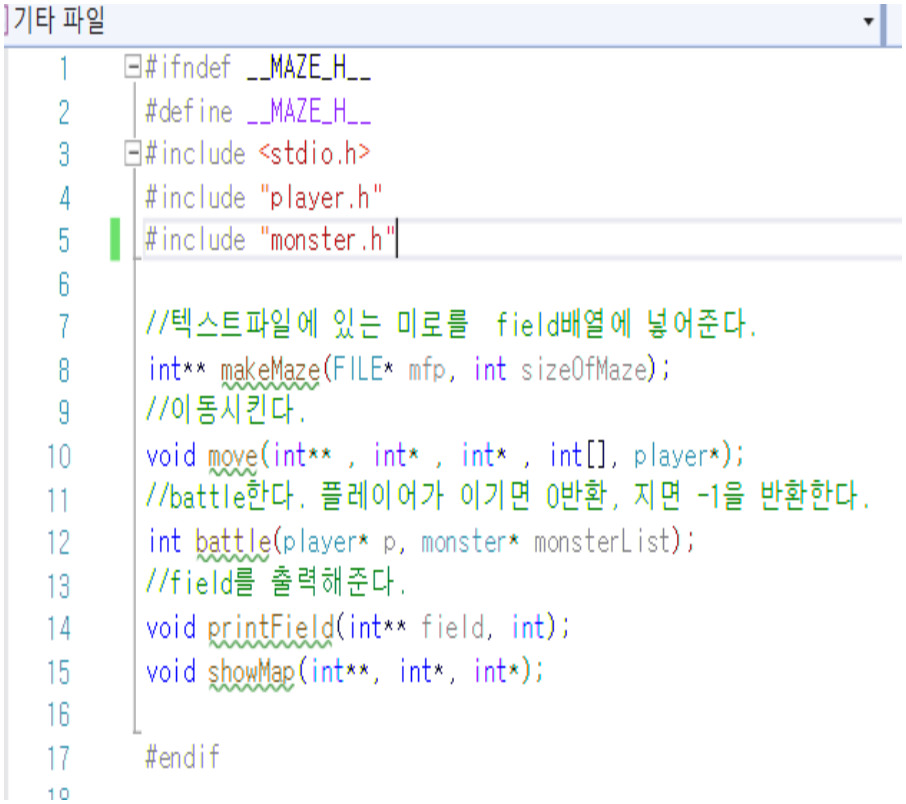
-> showSkill : 플레이어의 스킬창에 있는 스킬들의 정보를 출력한다.

-> showItem : 플레이어의 아이템창에 있는 아이템의 이름을 출력한다.

-> playerInit : 플레이어의 스탯과 스킬창, 아이템창을 초기화해준다.

-> printPlayer : 플레이어의 정보를 출력해준다.

(3) maze.h : 미로에 관련한 함수들이 존재하는 헤더파일이다.



-> makeMaze : 텍스트 파일에 들어있는 미로의 정보를 받아 field라는 미로를 만들고 리턴한다.

-> move : 플레이어의 이동을 담당하며, 스킬창과 아이템창을 볼 수 있도록 하는 함수이다.

-> battle : 플레이어가 몬스터를 마주칠 시 전투를 담당하는 함수이다.

-> printField : 미로의 총 정보를 출력해주는 함수이다.

-> showMap : 플레이어 현재 위치를 기준으로 미로의 정보를 출력해주는 함수이다.

- 메인 함수

```c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include "maze.h"
#include "player.h"
#include "monster.h"


void main() {
    FILE* mfp = fopen("MAZE.txt","r");
    FILE* ifp = fopen("item.txt", "r");
    FILE* sfp = fopen("skill.txt", "r");
    int sizeOfMaze;
    int **field = NULL;
    int can[4];

    fscanf(mfp, "%d", &sizeOfMaze);

    field = makeMaze(mfp, sizeOfMaze);
    item* itemList = (item*)malloc(sizeof(item));
    skill* skillList = NULL;
    monster* monsterList = (monster*)malloc(sizeof(monster));
    player *p = (player*)malloc(sizeof(player));
    itemInput(ifp, itemList);
    skillList = skillInput(sfp);
    int currentRow = 1, currentCol = 1;
    int res=0;
    playerInit(p, skillList);

    //game play
    while (1) {

        if (p->s.CHP <= 0) {
            printf("\n-----------------------------------\n");
            printf("            Game Over!!!!!!         \n");
            printf("-----------------------------------\n");
            break;
        }

        //move
        if (field[currentRow][currentCol] == 0) {
            move(field, &currentRow, &currentCol, can, p);
        }
        //monster
        else if (field[currentRow][currentCol] == 2) {
```

```c
    //game play
    while (1) {

        if (p->s.CHP <= 0) {
            printf("\n-----------------------------------\n");
            printf("            Game Over!!!!!!         \n");
            printf("-----------------------------------\n");
            break;
        }

        //move
        if (field[currentRow][currentCol] == 0) {
            move(field, &currentRow, &currentCol, can, p);
        }
        //monster
        else if (field[currentRow][currentCol] == 2) {
            //randomly monster
            pushMonster(p->s.level, monsterList);
            res = battle(p, monsterList);
            field[currentRow][currentCol] = 0;
            if (res == 1) {
                //스킬을 넣어주고 레벨을 1 높인다.
                getSkill(p, skillList);
            }
        }
        //item
        else if (field[currentRow][currentCol] == 4) {
            //randomly item
            getItem(p,itemList);
            field[currentRow][currentCol] = 0;
        }
        //boss
        else if (field[currentRow][currentCol] == 3) {
            pushMonster(p->s.level, monsterList);
            res = battle(p, monsterList);

            printPlayer(p);
            printMonster(monsterList);
            printField(field, sizeOfMaze);

            if (res == 1) {
                /*엔딩*/
                printf("\n-----------------------------------\n");
                printf("            clear!!!!!!             \n");
                printf("-----------------------------------\n");
                break;
            }
```

```
76                  break;
77              }
78
79              else if(res == -1){
80                  /*배드엔딩*/
81                  printf("₩n-----------------------------------₩n");
82                  printf("            SOOO Close              ₩n");
83                  printf("            Try Again!!!!           ₩n");
84                  printf("-----------------------------------₩n");
85                  break;
86              }
87          }
88
89      }
90
91      fclose(mfp); fclose(ifp); fclose(sfp);
92
93      for (int i = 0; i < sizeOfMaze; i++) {
94          free(field[i]);
95      }
96      free(field);
97
98      item *x,*y;
99      y = itemList;
100
101     while (y != NULL) {
102         x = y;
103         y = y->link;
104         free(x);
105     }
106
107     free(p->pSkill);
108     free(skillList);
109
110     y = p->pItem;
111
112     while (y != NULL) {
113         x = y;
114         y = y->link;
115         free(x);
116     }
117
118
119     free(p);
120 }
121
```

메인함수에서 이 게임에 존재하는 모든 스킬, 아이템의 정보가 들어있는 itemList, skillList를 만들어준다. 이 때, itemList는 연결리스트로, skillList는 배열을 기반으로 한 트리로 구현한다. 또한 makeMaze함수를 통해 int** field변수에 미로의 정보를 저장해준다. 이후 playerInit을 통해 플레이어의 정보를 초기화해준다. while(1)반복문을 이용하여 게임의 동작을 구현하였다. 반복문 안에서 플레이어의 체력이 0이하로 떨어지면 게임 오버를 출력하고 반복문을 종료시킨다. 조건문으로 현재 플레이어의 위치인 field[currentRow][currentCol]이 0, 2, 3, 4인 경우를 나누었는데, 0인 경우 move함수를 통해 이동을 진행하며 이 move함수 안에서 플레이어가 field값이 1인 곳으로 이동을 시도할 경우 갈 수 없음을 나타내었다. 2인 경우는 battle함수를 통해 전투를 진행하였으며 4인 경우 getItem을 통해 아이템을 하나 받아 플레이어에게 부여하였고 3인 경우는 최종 전투를 진행하여 최종 몬스터를 잡았을 경우 클리어 메시지를, 잡는데 실패하였을 경우 게임 오버 메시지와 함께 미로의 총 정보, 지금까지 잡았던 몬스터들의 정보, 플레이어의 정보를 출력해준다. 이후 반복문이 종료되면 메모리 해제를 진행하였다.

- monster.c

```
void pushMonster(int level, monster* monsterList) {
    /*add monster to stack accoring to player level*/
    monster* pMonster = (monster*)malloc(sizeof(monster));
    pMonster->atk = level * 9;
    pMonster->HP = -20 + level * 60;
    pMonster->CHP = -20 + level * 60;
    pMonster->speed = level * 9;

    if (level == 1) {
        monsterList->link = pMonster;
        pMonster->link = NULL;
    }
    else {
        pMonster->link = monsterList->link;
        monsterList->link = pMonster;
    }
}

int popMonster(monster* monsterList) {
    /*monsterList stack Pop and return monster*/
    monster* dMonster;
    int mLevel;

    if (monsterList->link != NULL) {
        dMonster = monsterList->link;
        monsterList->link = dMonster->link;
        mLevel = (int)dMonster->atk / 9;
        free(dMonster);
        return mLevel;
    }
    else
        return -1;
}
```

mosterList에 몬스터 노드를 push하고 pop하는 함수이다. pushMonster에서 몬스터의 스탯을 정하였는데 이는 플레이어의 레벨에 비례한 값을 넣어주었다. popMonster에서는 몬스터 노드 하나를 Pop하고 몬스터의 레벨에 해당하는 mLevel을 리턴하였다.

```c
void printMonster(monster* monsterList) {
    /*print monster stack*/
    /*get return of popMonster to print*/
    int res;

    while (1) {
        res = popMonster(monsterList);

        if (res == -1)
            break;
        else {
            printf("The number : %d₩n", res);
            printf("Attack : %d₩n", res * 9);
            printf("HP : %d₩n", -20 + res * 60);
            printf("Speed : %d₩n₩n₩n", res * 9);
        }
    }
}
```

지금까지 잡았던 몬스터들의 모든 정보를 출력해주는 함수이다. popMonster를 통해 받은 res값을 이용하여 res가 -1인 경우 더 pop할 노드가 없다는 뜻이므로 break를 시키고 그렇지 않은 경우 res값에 따른 몬스터의 번호와 스탯을 출력해준다.

- player.c

```c
void itemInput(FILE* ifp, item* itemList) {
    /*item.txt to itemList*/
    int i,j;
    double a, s, h, c;
    item* temp;
    itemList->link = NULL;
    char* iname;
    char tempname='a';
    char trash[20];
    for (i = 0; i < 20; i++) {
        temp = itemList;
        while (temp->link != NULL) {
            temp = temp->link;
        }
        iname = (char*)malloc(sizeof(char) * 20);
        j = 0;
        fgets(trash, sizeof(trash), ifp);
        while (tempname != '₩n') {
            fscanf(ifp, "%c", &tempname);
            if (tempname != '₩n') {
                iname[j] = tempname;
                j++;
            }
        }
        iname[j] = '₩0';
        tempname = 'a';
        fscanf(ifp, "%lf %lf %lf %lf", &a, &s, &h, &c);
        item* newitem = (item*)malloc(sizeof(item));
        newitem->name = iname;
        newitem->s.atk = a;
        newitem->s.speed = s;
        newitem->s.HP = h;
        newitem->s.CHP = c;
        newitem->link = NULL;
        temp->link = newitem;
    }

    free(iname);
}
```

item.txt에 있는 아이템의 정보들을 연결리스트인 itemList에 저장을 해주는 함수이다. tempname변수를 통해 한 글자씩 받아 iname배열에 넣어준다. 이름을 다 받은 후에 a, s, h, c 변수를 통해 아이템의 스탯을 받고 해당 정보들을 newItem노드에 모두 넣어준다. 이후 temp의 링크가 newitem을 가리키도록 만들었는데, temp는 itemList에서 마지막 노드의 위치를 가리키고 있다. 이 과정을 이 게임에 존재하는 모든 아이템 개수인 20개 만큼 반복한다. 함수의 동작이 모두 끝나면 동적할당을 한 iname의 메모리 해제를 진행한다.

```
skill* skillInput(FILE* sfp) {
    /*make skillList as a tree.*/
    int skillsize;
    char* sname = NULL;
    char trash[20];
    char temp = 'a';
    double a, s, h, c;
    int j;
    fscanf(sfp, "%d", &skillsize);
    skill* skillList = (skill*)malloc(sizeof(skill) * skillsize);
    for (int i = 0; i < skillsize; i++) {
        sname = (char*)malloc(sizeof(char) * 20);
        j = 0;
        fgets(trash, sizeof(trash), sfp);
        while (temp != '₩n') {
            fscanf(sfp, "%c", &temp);
            if (temp != '₩n') {
                sname[j] = temp;
                j++;
            }
        }
        sname[j] = '₩0';
        temp = 'a';
        fscanf(sfp, "%lf %lf %lf %lf", &a, &s, &h, &c);
        skillList[i].name = sname;
        skillList[i].s.atk = a;
        skillList[i].s.speed = s;
        skillList[i].s.HP = h;
        skillList[i].s.CHP = c;
    }

    free(sname);


    return skillList;
}
```

skill.txt에 있는 스킬들의 정보를 받아 skill* skillList에 넣어 이를 반환해주는 함수이다. skillList는 배열을 이용한 이진 트리로 구현하였다. 스킬의 이름을 sname에 저장하였으며 회복 스킬은 c가 1, 공격 스킬을 c가 -1을 나타내게끔 텍스트파일을 작성하였다.

```
void getItem(player *p, item* itemList) {
    /*Give random item to player*/
    int assign, a;
    item* i = (item*)malloc(sizeof(item));
    item* temp = itemList;
    srand(time(0));
    assign = rand() % 19;

    for (a = 0; a <= assign; a++) {
        temp = temp->link;
    }

    printf("================================================================================\n");
    printf("You get the %s!!\n\tatk : %.2lf\tspd : %.2lf\tHP : %.2lf\n", temp->name, temp->s.atk, temp->s.speed, temp->s.HP);
    printf("================================================================================\n");
    p->s.HP += temp->s.HP;
    p->s.atk += temp->s.atk;
    p->s.speed += temp->s.speed;
    p->s.CHP += temp->s.CHP;

    i->s.atk = temp->s.atk;
    i->s.HP = temp->s.HP;
    i->s.CHP = temp->s.CHP;
    i->s.speed = temp->s.speed;
    i->name = temp->name;

    if (p->pItem == NULL) {
        p->pItem = i;
        p->pItem->link = NULL;
        return;
    }
    else {
        temp = p->pItem;
        while (temp->link != NULL) {
            temp = temp->link;
        }
        temp->link = i;
        i->link = NULL;
    }

}
```

플레이어가 아이템 상자를 발견했을 시 itemList에 있는 아이템들 중 하나를 랜덤으로 받는 함수이다. 어떤 아이템을 받았는지를 출력하였으며 아이템의 스탯 정보에 따라 플레이어의 스탯을 변경하였다. i라는 item형 노드를 하나 만들어 해당 아이템의 정보를 넣어준 뒤 플레이어의 아이템창인 p->pItem연결리스트에 연결해주었다.

```
void getSkill(player *p, skill* skillList) {
    /*Assign skill to player, and level up proccess*/
    p->s.level++;
    int node = p->skillNumber;
    int choose = 3;
    int leftchild = node * 2 +1;
    int rightchild = node * 2 + 2;

    if (p->s.level < 5) {
        printf("Choose Skill\n\t1 : %s\t2 : %s\n", skillList[leftchild].name, skillList[rightchild].name);
        while (!(choose == 1 || choose == 2)) {
            scanf("%d", &choose);
            if (!(choose != 1 || choose != 2))
                printf("Please choose 1 or 2\n");
        }
        if (choose == 1) {
            p->pSkill[p->s.level - 1] = skillList[leftchild];
            p->skillNumber = leftchild;
            printf("You learn %s.\n", skillList[leftchild].name);
        }
        else if (choose == 2) {
            p->pSkill[p->s.level - 1] = skillList[rightchild];
            p->skillNumber = rightchild;
            printf("You learn %s\n", skillList[rightchild].name);
        }
    }

    else
        printf("\n\nskill master.\n\n\n");
    p->s.atk = p->s.atk * 1.5;
    p->s.HP = p->s.HP * 1.5;
    p->s.speed = p->s.speed * 1.5;
    p->s.CHP = p->s.HP;

}
```

플레이어가 몬스터를 처치했을 시 레벨업을 하여 스킬을 하나 배우도록 하는 함수이다. 이진 트리로 skillList가 구현되었기 때문에 플레이어는 가장 최근에 배운 스킬을 부모로 하여 자식들에 해당하는 스킬만 배울 수 있도록 구현하였다. 이는 1 또는 2를 입력하여 1을 입력할 시 왼쪽 자식에 해당하는 스킬, 2를 입력할 시 오른쪽 자식에 해당하는 스킬을 배울 수 있으며 그 이외의 값을 입력하면 Please choose 1 or 2를 출력하고 다시 입력을 받게끔 구현하였다. 트리의 레벨이 4이기 때문에 플레이어의 레벨이 5가 넘어가는 경우 스킬을 더 배우지 않도록 하였으며 이 때는 skill master를 출력하도록 구현하였다. 이후 플레이어가 레벨 업을 하였으므로 각 스탯을 모두 증가시킨 뒤 전투 이후 플레이어의 현재 체력이 감소된 상태일 것이므로 현재 체력을 플레이어의 체력 스탯으로 초기화 시켜주었다.

```
void showSkill(player* p) {
    //Show skill list
    int i;
    int k;
    if (p->s.level < 5) {
        k = p->s.level;
    }
    else
        k = 4;
        printf("₩n₩n<<<Learned Skill>>>₩n");
        for (i = 0; i < k; i++) {
            printf("%d : ₩t%s₩n", i, p->pSkill[i].name);
            printf("damage : %.1f₩n", p->pSkill[i].s.atk);
            printf("cost : %.1f₩n₩n", p->pSkill[i].s.speed);

        }
}
void showItem(player* p) {
    //show item list
    int i;
    item* temp;
    temp = p->pItem;
    if (temp->link == NULL)
        printf("No item₩n");
    else {
        printf("₩n₩n<<<Item List>>>₩n");
        temp = temp->link;
        while (temp != NULL) {
            printf("%s₩n", temp->name);
            temp = temp->link;
        }
    }
}
```

플레이어의 스킬창과 아이템창을 출력하는 함수이다. k만큼 반복문을 작동시켜 스킬의 정보를 출력하도록 구현하였는데 이 때 플레이어의 레벨이 5가 넘어가는 경우 k = 4로 할당하였다.

아이템은 연결리스트로 구현하였기 때문에 temp = p->pItem으로 설정하여 temp->link가 NULL인 경우 No item을 출력하였으며 그렇지 않은 경우는 temp가 NULL일 때까지 반복하여 아이템의 이름을 출력하였다.

```c
void playerInit(player* p, skill* s) {
    /*initiate player stat.*/
    p->s.atk = 10;
    p->s.HP = 100;
    p->s.CHP = 100;
    p->s.level = 1;
    p->s.speed = 10;

    p->pItem = (item*)malloc(sizeof(item));
    p->pItem->link = NULL;
    p->pSkill = (skill*)malloc(sizeof(skill) * 4);
    p->pSkill[0] = s[0];
    p->skillNumber = 0;

}
void printPlayer(player* p) {
    /*print player info*/
    printf("=============player's status=============₩n");
    printf("level : %d₩n", p->s.level);
    printf("attack : %.1f₩n", p->s.atk);
    printf("HP : %.1f₩n", p->s.HP);
    printf("speed : %.1f₩n", p->s.speed);
    printf("========================================₩n₩n");
}
```

플레이어의 스탯과 아이템창, 스킬창을 초기화 해주는 함수와 플레이어의 스탯 정보를 출력해주는 함수이다. playerInit에서 플레이어는 스킬을 최대 4가지를 배울 수 있기 때문에 pSkill를 4만큼 동적할당 하였으며 플레이어의 첫 번째 스킬로 기본 공격인 NormalAtk를 p->pSkill[0]에 넣어주었다.

- maze.c

```c
#define LEFT 97
#define RIGHT 100
#define UP 119
#define DOWN 115

int getch(void){
    char ch;
    int error;
    static struct termios Otty, Ntty;

    fflush(stdout);
    tcgetattr(0,&Otty);
    Ntty = Otty;
    Ntty.c_iflag = 0;
    Ntty.c_oflag = 0;
    Ntty.c_lflag &= ~ICANON;
#if 1
    Ntty.c_lflag &= ~ECHO;
#else
    Ntty.c_lflag |= ECHO;
#endif
    Ntty.c_cc[VMIN] = 0;
    Ntty.c_cc[VTIME] = 1;
#if 1
#define FLAG TCSAFLUSH
#else
#define FLAG TCSANOW
#endif
    if(0 == (error = tcsetattr(0, FLAG, &Ntty))){
        error = read(0, &ch, 1);
        error += tcsetattr(0,FLAG, &Otty);
    }
    return (error == 1 ? (int)ch : -1);
}
```

플레이어가 커맨드를 입력할 시 계속해서 엔터를 입력하는 것을 방지하기 위해 선언한 getch함수이다.

```
int** makeMaze(FILE* mfp, int sizeOfMaze) {
    /*maze.txt to field */

    int** field = (int**)malloc(sizeof(int*) * sizeOfMaze);

    for (int j = 0; j < sizeOfMaze; j++)
        field[j] = (int*)malloc(sizeof(int) * sizeOfMaze);

    for (int i = 0; i < sizeOfMaze; i++) {
        for (int j = 0; j < sizeOfMaze; j++) {
            fscanf(mfp, "%d", &field[i][j]);
        }
    }
    return field;
}
```

MAZE.txt에 저장된 미로의 정보를 받아 int** field에 저장하여 이를 리턴해주는 함수이다. 미로에서 0은 갈 수 있는 길, 1은 벽, 2는 몬스터, 3은 최종 몬스터, 4는 아이템 상자를 의미한다.

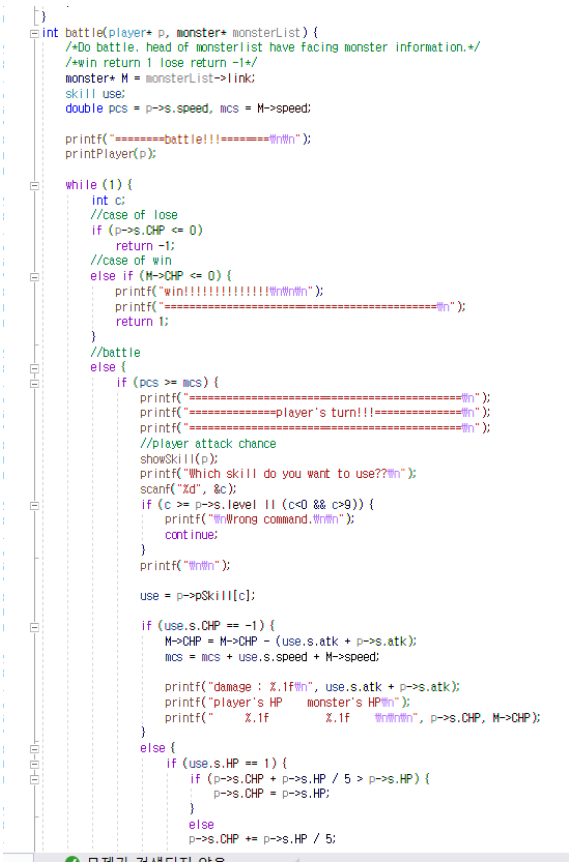
```
void move(int** field, int* currentRow, int* currentCol, int can[], player* p) {
    /*Get command and move to up down left right 'i'show itemlist, 'k' show skilllist.*/
    /*when get i and k do not shut down function*/

    int c;

    can[0] = field[*currentRow - 1][*currentCol];
    can[1] = field[*currentRow + 1][*currentCol];
    can[2] = field[*currentRow][*currentCol-1];
    can[3] = field[*currentRow][*currentCol+1];
            c = getch();
            switch (c) {
            case LEFT:
                printf("move left\n");
                if (can[2] != 1) {
                    (*currentCol)--;
                }
                else {
                    printf("can't move.\n");
                    move(field, currentRow, currentCol, can, p);
                }
                break;
            case RIGHT:
                printf("move right\n");
                if (can[3] != 1) {
                    (*currentCol)++;
                }
                else {
                    printf("can't move.\n");
                    move(field, currentRow, currentCol, can, p);
                }
                break;
            case UP:
                printf("move up\n");
                if (can[0] != 1) {
                    (*currentRow)--;
                }
                else {
                    printf("can't move.\n");
                    move(field, currentRow, currentCol, can, p);
                }
                else {
                    printf("can't move.\n");
                    move(field, currentRow, currentCol, can, p);
                }
                break;
            case UP:
                printf("move up\n");
                if (can[0] != 1) {
                    (*currentRow)--;
                }
                else {
                    printf("can't move.\n");
                    move(field, currentRow, currentCol, can, p);
                }
                break;
            case DOWN:
                printf("move down\n");
                if (can[1] != 1) {
                    (*currentRow)++;
                }
                else {
                    printf("can't move.\n");
                    move(field, currentRow, currentCol, can, p);
                }
                break;
            case 'i':
                showItem(p);
                move(field, currentRow, currentCol, can, p);
                break;
            case 'k':
                showSkill(p);
                move(field, currentRow, currentCol, can, p);
                break;
            case 'm':
                showMap(field, currentRow, currentCol);
                move(field, currentRow, currentCol, can, p);
                break;
            }
}
```

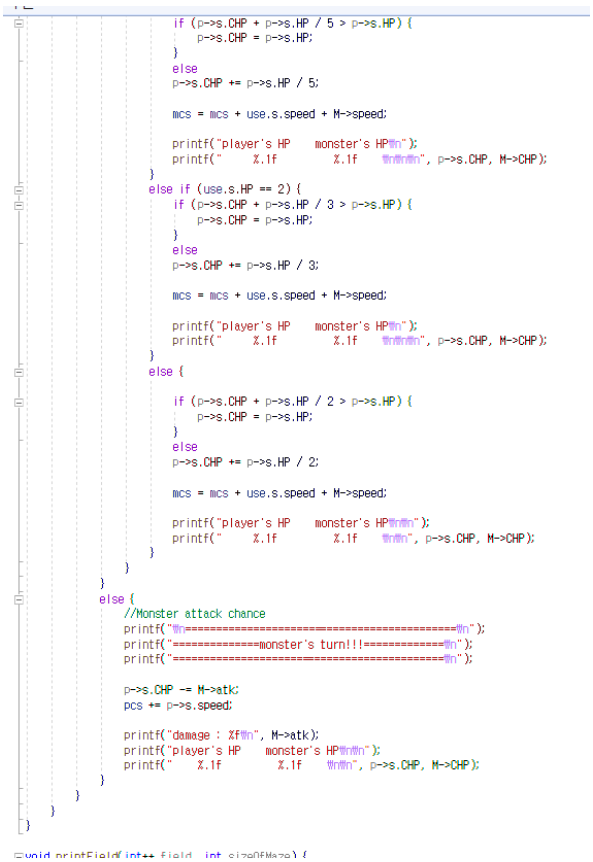
커맨드를 입력 받아 플레이어를 이동시켜주는 함수이다. can배열은 현재 플레이어 위치에서 상, 하, 좌, 우의 정보를 받아 해당 위치로 플레이어가 이동할 수 있는지 없는지를 알려준다. 방향키로 이동을 하면서 갈 수 없는 위치로 이동을 시도할 경우 can't move를 출력한 이후 다시 move함수를 호출하였으며 이동이 완료되면 현재 플레이어의 위치를 나타내는 *currentRow, *currentCol을 변경하였다. 상, 하, 좌, 우 이동은 각각 w, a, s, d 입

력을 할 시 진행하였으며 그 이외에 k를 입력하면 플레이어의 스킬창, i를 입력하면 플레이어의 아이템창, m을 입력하면 현재 플레이어의 위치를 기준으로 미로의 정보를 출력해주는 함수를 호출하였으며 move함수를 재호출하여 이동을 계속 진행할 수 있도록 구현하였다.

```
void showMap(int** field,int* cr, int* cc) {

    for (int i = 0; i <= *cr; i++) {
        for (int j = 0; j <= *cc; j++) {
            printf("%d ", field[i][j]);
        }
        printf("₩n");
    }
}
```

플레이어의 현재 위치를 나타내는 *cr, *cc를 기준으로 미로(field)의 정보를 출력해주는 함수이다.





전투를 진행하는 함수이다. monsterList에서 가장 최근에 Push가 된 몬스터의 정보를 M에 할당하였으며 pcs와 mcs는 각각 현재 플레이어와 몬스터의 스피드 값을 나타내는데, 이는 현재 누구의 턴인지를 구분하기 위해 사용된다. 몬스터는 플레이어에게 몬스터의

공격력만큼의 데미지를 주며 플레이어는 현재 플레이어의 공격력에 스킬의 공격력 값을 더한 만큼의 데미지를 몬스터에게 준다. 플레이어가 턴을 진행하면 mcs에 해당 스킬의 코스트 값인 스피드에 현재 몬스터의 스피드 스탯만큼을 더해주었으며 몬스터가 턴을 진행할 시 pcs에 플레이어의 스피드 스탯만큼을 더해주었다. 턴은 mcs가 pcs보다 크면 몬스터가 가져가며 반대로 pcs가 mcs보다 큰 경우 플레이어가 가져간다. 전투를 진행하면서 데미지만큼 체력이 줄어든 것을 출력하였으며 플레이어가 스킬을 사용할 수 있도록 어떤 스킬이 있는지를 보여주고 입력을 받았다. 몬스터의 체력이 0이하가 되는 경우 전투에서 이겼음을 알려주었고 반대로 플레이어의 체력이 0이하가 되는 경우 -1을 리턴하여 플레이어가 사망했음을 알려주었다.

```
void printField(int** field, int sizeOfMaze) {
    /*print field.*/
    for (int i = 0; i < sizeOfMaze; i++) {
        for (int j = 0; j < sizeOfMaze; j++)
            printf("%d ", field[i][j]);

        printf("₩n");
    }
}
```

미로의 전체 정보를 출력해주는 함수이다.

4. 기여도

박태윤 : 보고서 작성, 헤더파일 포함 전체적인 코드 구현.(50%)

장은상 : 발표 준비 + 진행 및 아이템 스킬 관련 구현, putty에서 구현, 디버깅, 코드 수정.(50%)