mud_game 프로젝트 보고서

소프트웨어공확과 214953 기도현

1. 서론

1. 프로젝트 목적 및 배경: 7주차까지 배운 내용을 복습, 점검, 활용하여 실습을 진행

2. 목표: class선언과 class 정의를 분리해 간단한 mud_game 만들기

2. 요구사항

1. 사용자 요구사항: 유저가 up, down, right, left만 사용하여 한정된 hp 안에서 목적지에 도착하는 게임

2. 기능 계획:
① class선언과 class 정의를 분리해 user 객체 만들기
② 현재 체력 출력
③ 사용자가 이동하고 싶은 방향 or 지도 출력 or 게임 끝내기 중 선택받기
④ 입력 받은 방향의 좌표 유효성 체크 후 유효하지 않으면 입력 다시 받기
⑤ 유효한 좌표라면 해당 좌표로 user 이동
⑥ map에 아이템, 적, 포션, 목적지 배치
⑦ 적or 아이템or 포션을 만나면 만났다고 출력,
적을 만나면 체력 -2, 포션을 만나면 체력 +2
⑧ 한 칸 이동할 때마다 체력 -1
⑨ 체력이 0이되면 게임 끝
⑩ 사용자 입력에서 지도 출력을 고르거나 사용자가 한칸 이동할 때마다 지도 출력
⑪ 사용자가 목적지에 도착하면 게임 승리

⑫ 사용자가 map을 입력하면 지도를 출력하고 end를 입력하면 게임 종료

⑬ 사용자가 잘못된 값을 입력하면 값 다시 받기

3. 함수 계획: ① 사용자가 선택한 좌표가 유효한지 확인하는 checkXY 함수

② 사용자가 목적지에 도착했는지 확인하는 checkGoal 함수

③ 사용자가 이동한 곳에 적or 아이템 or 포션이 있는지 확인하는 checkState 함수

④ 지도를 출력해주면 displayMap 함수

⑤ 메인 내에서 중복된 코드, 함수 사용을 줄이기 위한 game_excution 함수

⑥ 사용자의 입력 좌표가 유효하지 않았을 때 user의 좌표와 hp를 좌표를 받기 전 상태로 되돌리기 위한 out_board 함수

3. 설계 및 구현

1-1 기능 별 구현 사항: (요구사항 힘수 코드)

① 사용자가 선택한 좌표가 유효한지 확인하는 checkXY 함수

1. 함수 스크린샷

```
bool user::checkXY(user &user_A,int mapX, int mapY){
    bool checkFlag = false;
    if (user_A.user_x >= 0 && user_A.user_x < mapX && user_A.user_y >= 0 && user_A.user_y < mapY) {
        checkFlag = true;
    }
    return checkFlag;
}
```

2. 입력

- user_A = user_A 객체의 주소
- mapX = 맵의 최대 x값-1
- mapY = 맵의 최대 y값 -1

3. 반환값

- checkFlag = 유저가 이동하고 싶어하는 좌표가 유효한지 참 거짓을 저장

4. 결과

- 유저가 이동하고 싶어하는 좌표가 유효하면 ture를 반환, 유효하지 않으면 false를 반환

5. 설명

- 객체의 주소를 매개변수로 받아 해당 객체의 맴버변수에 자유롭게 접근
- checkFlag값을 false로 초기화
- if문을 통해 user_x, user_y값이 지도 안에서 유효한지 확인하고 유효하면 checkFlag 값을 ture로 변경
- checkFlag를 반환

② 사용자가 목적지에 도착했는지 확인하는 checkGoal 함수

1. 함수 스크린샷

```
bool user::checkGoal(user &user_A,int map[][mapX]){
    // 목적지 도착하면
    if (map[user_A.user_y][user_A.user_x] == 4) {
        return true;
    }
    return false;
}
```

2. 입력

- user_A = user_A 객체의 주소
- map = 전체지도
- user_x = 유저의 x값
- user_y = 유저의 y값

3. 반환값

- ture or false

4. 결과

- user가 목적지에 도착하면 true를 반환

5. 설명

- 객체의 주소를 매개변수로 받아 해당 객체의 맴버변수에 자유롭게 접근
- if문을 활용해 user의 현재 위치에 해당하는 배열에 저장된 값이 4이면 목적지에 도착한 것으로 간주하고 true를 반환, 4가 아니라면 false를 반환

③ 사용자가 이동한 곳에 적or 아이템 or 포션이 있는지 확인하는 checkState 함수

1. 함수 스크린샷

```
void user::checkState(user &user_A,int map[][mapX]){
    if(map[user_A.user_y][user_A.user_x]==1){
        cout << "{아이템}이 있습니다."<<endl;
    }
    else if(map[user_A.user_y][user_A.user_x]==2){
        cout << "{적}이 있습니다"<<endl;
        user_A.user_hp-=2;
    }
    else if(map[user_A.user_y][user_A.user_x]==3){
        cout << "{포션}이 있습니다"<<endl;
        user_A.user_hp+=2;
    }
}
```

2. 입력

- user_A = user_A 객체의 주소
- map = 전체 지도
- user_x = 유저의 x값
- user_y = 유저의 y값
- user_hp = 유저의 체력

3. 반환값

- 없음

4. 결과

- 이동한 좌표에 어떤 것이 있는지 출력
- 적이 있다면 유저의 체력 -2
- 포션이 있다면 유저의 체력 +2

5. 설명

- 객체의 주소를 매개변수로 받아 해당 객체의 맴버변수에 자유롭게 접근
- if문을 통해 유저가 지도에 어디 위치해 있는지 확인하고 위치해 있는 배열 안의 값에 따라 해당 위치에 무엇이 있는지 출력
- 해당 위치에 적이 있었다면 체력을 -2 하고, 포션이 있었다면 체력을 +2

④ 지도를 출력해주면 displayMap 함수

1. 함수 스크린샷

```
void displayMap(user &user_A,int map[][mapX]) {
    for (int i = 0; i < mapY; i++) {
        for (int j = 0; j < mapX; j++) {
            if (i == user_A.user_y && j == user_A.user_x) {
                cout << " USER |"; // 양 옆 1칸 공백
            }
            else {
                int posState = map[i][j];
                switch (posState) {
                case 0:
                    cout << "      |"; // 6칸 공백
                    break;
                case 1:
                    cout << "아이템|";
                    break;
                case 2:
                    cout << "  적  |"; // 양 옆 2칸 공백
                    break;
                case 3:
                    cout << " 포션 |"; // 양 옆 1칸 공백
                    break;
                case 4:
                    cout << "목적지|";
                    break;
                }
            }
        }
        cout << endl;
        cout << " ------------------------------ " << endl;
    }
}
```

2. 입력

    - user_A = user_A 객체의 주소

    - map = 전체 지도

    - user_x = 유저의 x값

    - user_y = 유저의 y값

3. 반환값

    - 없음

4. 결과

    - 전체 지도를 출력

    - user 위치를 출력

5. 설명

- 이중 포문을 통해 2차원 배열의 각 인덱스에 접근
- 각 인덱스에 저장된 값이 0이면 빈칸 1이면 아이템, 2이면 적, 3이면 포션, 4이면 목적지를 출력
- 이중배열 도중 user_x와 user_y가 있는 곳에 접근하면 user를 출력

⑤ 메인 내에서 중복된 코드, 함수 사용을 줄이기 위한 game_excution 함수

1. 코드 스크린샷

```
void game_excution(user &user_A,int map[][mapX],int num){
    if(num==8){
            user_A.user_y -= 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,0,1);
            }
            else {
                cout << "위로 한 칸 올라갑니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
    else if(num==2){
            user_A.user_y += 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,0,-1);
            }
            else {
                cout << "아래로 한 칸 내려갑니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
```

```
    else if(num==4){
            user_A.user_x -= 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,1,0);
            }
            else {
                cout << "왼쪽으로 이동합니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
    else if(num==6){
            user_A.user_x += 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,-1,0);
            }
            else {
                cout << "오른쪽으로 이동합니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }

}
```

2. 입력

- user_A = user_A 객체의 주소
- map = 전체 지도
- user_x = 유저의 x값
- user_y = 유저의 y값
- num = 유저가 이동하고 싶어하는 방향
- user_hp = 유저의 체력
- inMap = 좌표의 유효성을 저장
- mapX = 지도의 최대 x값 -1
- mapY = 지도의 최대 y값 -1

3. 반환값

-없음

4. 결과

- 입력받은 이동방향에 따라 유저의 x,y값 변경
- 유저의 hp -1
- 이동하고싶은 좌표의 유효성 검사
- 유효하지 않은 좌표라면 유저의 x, y, hp값을 되돌리고 함수 끝내기
- 유효한 주소라면 어디로 이동했는지 출력
- 유저가 이동한 후 지도를 출력
- 위치에 뭐가 있다면 뭐가 있는지 출력하고 적이라면 hp-2, 포션이라면 hp+2

5. 설명

- 객체의 주소를 매개변수로 받아 해당 객체의 맴버변수에 자유롭게 접근
- 사용자가 up을 입력하면 num을 8, down이면 2, left면 4, right면 6을 함수 인자에 넣음
- if문을 통해 유저가 이동하고 싶은 방향을 알아냄
- 이동하고 싶은 방향에 맞게 유저의 x,y값을 조정
- 이동했으니 유저의 hp-1
- checkXY 함수를 활용해 이 좌표가 유효한지 확인
- 유효하지 않다면 out_board 함수를 호출해 이동한 x, y 좌표를 원상복구 하고 hp도 원상복구
- 유효한 좌표라면 이동한 방향에 대해 출력
- dispalyMap 함수를 활용해 유저가 이동한 후 지도를 출력

- checkState 함수를 활용해 해당 위치에 뭐가 있다면 뭐가 있는지 출력하고 적이라면 hp-2, 포션이라면 hp+2

⑥ 사용자의 입력 좌표가 유효하지 않았을 때 user의 좌표와 hp를 좌표를 받기 전 상태로 되돌리기 위한 out_board 함수

1. 함수 스크린샷

```
void out_board(user &user_A, int plus_x,int plus_y){
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    user_A.user_y += plus_y;
    user_A.user_x += plus_x;
    user_A.user_hp += 1;

}
```

2. 입력

- user_A = user_A 객체의 주소
- plus_x = 이동한 방향의 x좌표
- plus_y = 이동한 방향의 y좌표

3. 반환값

-없음

4. 결과

- 이동한 좌표가 유효하지 않을 때 호출하는 함수이기 때문에 이동하기 전 상태로 x, y좌표와 hp를 원상복구
- 맵을 벗아났습니다. 다시 돌아값니다 출력

5. 설명

- 객체의 주소를 매개변수로 받아 해당 객체의 맴버변수에 자유롭게 접근
- 매개변수로 방금 이동한 방향의 x, y 값을 전달받음
- 전달받은 x, y값을 바탕으로 좌표 원상복구
- hp도 +1 해서 hp 원상복구
- cout 명령어로 원하는 문구 출력

1-2 기능 별 구현 사항: (요구사항 별 코드)

① class선언과 class 정의를 분리해 user 객체 만들기

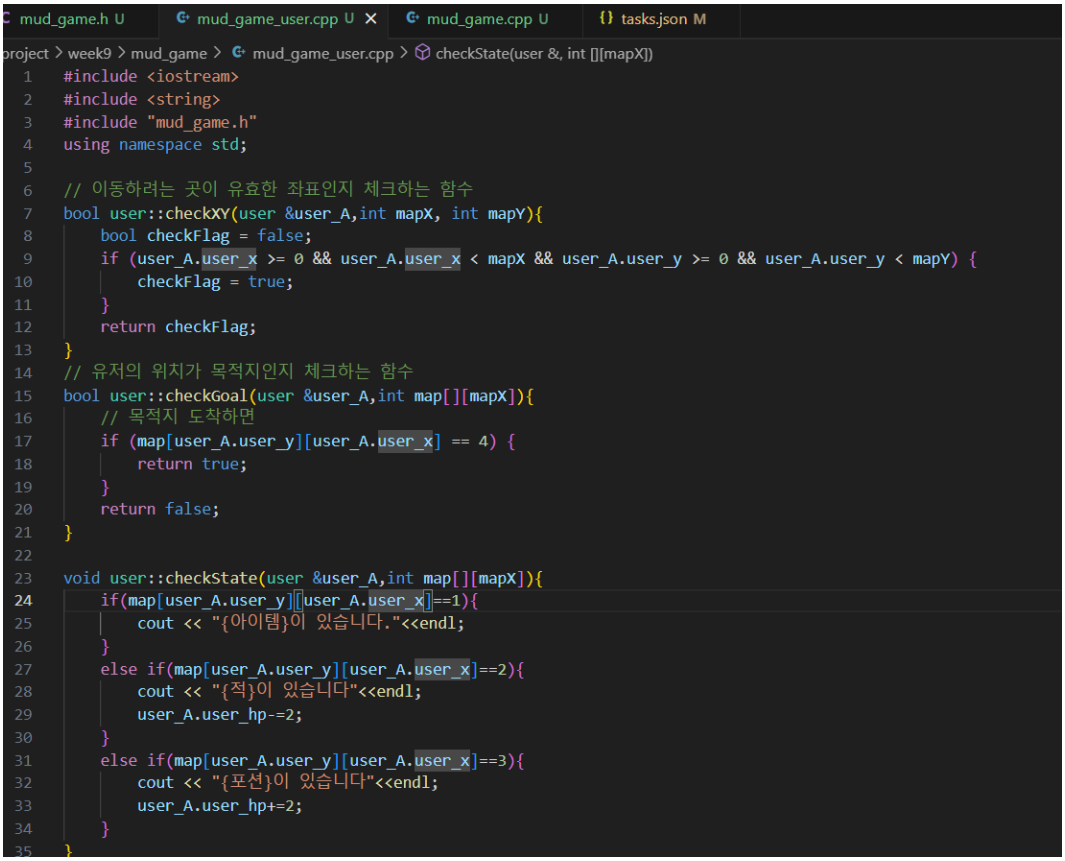
1. 코드 스크린샷

```
class user
{

 static const int mapX = 5;
 static const int mapY = 5;

public:
    // 유저의 위치를 저장할 변수
    int user_x = 0; // 가로 번호
    int user_y = 0; // 세로 번호
    int user_hp = 20;
    bool checkXY(user &user_A, int mapX, int mapY);
    bool checkGoal(user &user_A, int map[][mapX]);
    void checkState(user &user_A, int map[][mapX]);
};
```

(user 인터페이스를 저장한 mud_game.h)



(user class를 정의한 mud_game_user.cpp)

(user class를 활용하는 mud_game.cpp)

```
user user_A;
```

(main 함수 내에서 user_A 객체 선언)

설명:

- user가 사용할 변수와 함수를 user 클래스를 만들어서 관리
- 함수의 매개변수로 필요한 mapX와 mapY는 static 접근지정자를 사용해 관리
- 변수에 대한 알맞은 초기값으로 초기화

- main 함수 내에서 user_A 객체를 만들어 플레이어 정보를 관리

- mud_game.h에 user 인터페이스를 저장하고 mud_game_user.cpp에서 user class 정의

mud_game.cpp의 main 함수에서 user class 사용

② 현재 체력 출력

1. 코드 스크린샷

```
while (1) { // 사용자에게 계속 입력받기 위해 무한 루프

    // 사용자의 입력을 저장할 변수
    string user_input = "";
    cout <<"현재 HP는 "<<user_A.user_hp<<"입니다."<<endl;
    cout << "명령어를 입력하세요 (up,down,left,right,map,end): ";
    cin >> user_input;
```

2. 입력:

- user_A.user_hp = user_A 객체의 현재 체력을 저장하는 변수

3. 결과:

- user_A의 현재 체력이 출력

4. 설명:

- 유저에게 명령어를 입력받기 전에 hp를 알려주어 명령어 선택을 용이하게 함

- user_A 객체의 맴버변수인 user_hp에 접근해 그 값을 출력

③ 사용자가 이동하고 싶은 방향 or 지도 출력 or 게임 끝내기 중 선택 받기

1. 코드 스크린샷

```
string user_input = "";
cout <<"현재 HP는 "<<user_A.user_hp<<"입니다."<<endl;
cout << "명령어를 입력하세요 (up,down,left,right,map,end): ";
cin >> user_input;
```

2. 입력

- user_input = 사용자가 입력한 명령어를 저장하는 변수

3. 결과

- user가 값을 입력하고 입력한 값을 user_input에 저장

4. 설명

- user가 입력한 문자열을 cin 명령어를 통해 user_input에 저장

④ 입력 받은 방향의 좌표 유효성 체크 후 유효하지 않으면 입력 다시 받기

1. 코드 스크린샷

```
if (user_input == "up") {
    // 위로 한 칸 올라가기
    game_excution(user_A,map,8);
}
else if (user_input == "down") {
    // TODO: 아래로 한 칸 내려가기
    game_excution(user_A,map,2);
}
else if (user_input == "left") {
    // TODO: 왼쪽으로 이동하기
    game_excution(user_A,map,4);
}
else if (user_input == "right") {
    // TODO: 오른쪽으로 이동하기
    game_excution(user_A,map,6);
}
else if (user_input == "map") {
    // TODO: 지도 보여주기 함수 호출
    displayMap(user_A,map);
}
else if (user_input == "end") {
    cout << "종료합니다.";
    break;
}
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

```
bool user::checkXY(user &user_A,int mapX, int mapY){
    bool checkFlag = false;
    if (user_A.user_x >= 0 && user_A.user_x < mapX && user_A.user_y >= 0 && user_A.user_y < mapY) {
        checkFlag = true;
    }
    return checkFlag;
}
```

```
bool inMap = user_A.checkXY(user_A,mapX,mapY);
if (inMap == false) {
    out_board(user_A,0,1);
}
else {
    cout << "위로 한 칸 올라갑니다." << endl;
    displayMap(user_A,map);
    user_A.checkState(user_A ,map);
}
```

```
void out_board(user &user_A, int plus_x,int plus_y){
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    user_A.user_y += plus_y;
    user_A.user_x += plus_x;
    user_A.user_hp += 1;

}
```

2. 입력

- user_input = user가 입력한 문자열을 저장한 string 변수
- checkFlag = 해당 좌표의 유효성을 나타내는 bool 변수
- user_x = user의 x좌표를 의미하는 변수
- user_y = user의 y좌표를 의미하는 변수
- mapX = map의 최대 x좌표-1을 의미하는 변수
- mapY = map의 최대 y좌표-1을 의미하는 변수
- inMap = 해당좌표의 유효성을 나타내는 bool 변수
- plus_x = 입력한 좌표가 map을 넘어갔을 시 이동한 x좌표
- plus_y = 입력한 좌표가 map을 넘어갔을 시 이동한 y좌표
- user_hp = user의 현재 체력

3. 결과

- user_input이 up, down, right, left, map, end가 아니면 잘못된 입력입니다를 출력하고 다시 user_input 입력받기

- 입력받은 방향의 좌표가 map을 넘어가는 좌표라면 체력을 원상복구 시키고 맵을 벗어났습니다. 다시 돌아갑니다를 출력 후 다시 user_input 입력받기

4. 설명

- if문을 통해 user_input값을 확인하고 다른 문자열이 들어오면 잘못된 입력입니다를 출력 후 while문을 이용한 무한루프를 통해 문자열을 다시 입력받음

- 1. user_input값이 옳바른 좌표이동 입력값이라면 user class에 정의되어 있는 checkXY 함수를 사용
  2. if문을 활용해 좌표값이 옳바른지 확인하고 참 것짓값인 check_Flag 변수를 return
  3. 그 bool 값을 main 함수의 변수인 inMap에 저장
  4. inMap이 false 즉 움직인 좌표가 map을 벗어난다면 out_board 함수 호출
  5. 맵을 벗어났습니다. 다시 돌아갑니다를 출력 후 좌표와 체력을 원상복구 시키기
  6. 무한루프를 이용해 다시 user_input 받기

⑤ 유효한 좌표라면 해당 좌표로 user 이동

1. 코드 스크린샷

```
if (user_input == "up") {
    // 위로 한 칸 올라가기
    game_excution(user_A,map,8);
}
else if (user_input == "down") {
    // TODO: 아래로 한 칸 내려가기
    game_excution(user_A,map,2);
}
else if (user_input == "left") {
    // TODO: 왼쪽으로 이동하기
    game_excution(user_A,map,4);
}
else if (user_input == "right") {
    // TODO: 오른쪽으로 이동하기
    game_excution(user_A,map,6);
}
```

```
void game_excution(user &user_A,int map[][mapX],int num){
    if(num==8){
            user_A.user_y -= 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,0,1);
            }
            else {
                cout << "위로 한 칸 올라갑니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
    else if(num==2){
            user_A.user_y += 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,0,-1);
            }
            else {
                cout << "아래로 한 칸 내려갑니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
```

```
else if(num==4){
        user_A.user_x -= 1;
        user_A.user_hp-=1;
        bool inMap = user_A.checkXY(user_A,mapX,mapY);
        if (inMap == false) {
            out_board(user_A,1,0);
        }
        else {
            cout << "왼쪽으로 이동합니다." << endl;
            displayMap(user_A,map);
            user_A.checkState(user_A ,map);
        }
}
else if(num==6){
        user_A.user_x += 1;
        user_A.user_hp-=1;
        bool inMap = user_A.checkXY(user_A,mapX,mapY);
        if (inMap == false) {
            out_board(user_A,-1,0);
        }
        else {
            cout << "오른쪽으로 이동합니다." << endl;
            displayMap(user_A,map);
            user_A.checkState(user_A ,map);
        }
}
```

2. 입력

- user_input = user가 입력한 문자열을 저장한 string 변수
- user_x = user의 x좌표를 의미하는 변수
- user_y = user의 y좌표를 의미하는 변수
- inMap = 해당좌표의 유효성을 나타내는 bool 변수
- user_A = user class의 객체

3. 결과

- user_input이 유효한 곳으로 이동하는 입력이라면 user_x와 user_y 값을 조작해 user의 위치를 이동

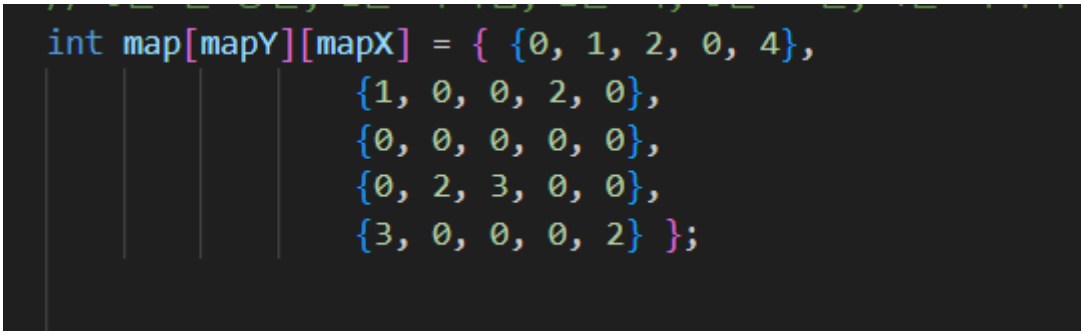
4. 설명

-

1. user_iput이 up이면 game_excution 함수의 num 매개변수에 8을, down이면 2를, right면 6을, left면 4를 전달
2. game_excution에서 if문을 활용해 num값에 해당하는 user_input을 해석
3. 각각의 user_input에 알맞은 user_x, user_y값을 변경
4. 각각의 user_input에 맞는 이동결과를 cout을 통해 알려줌

⑥ map에 아이템, 적, 포션, 목적지 배치

1. 코드 스크린샷



설명

- 2차원 배열의 각각 인덱스 값을 활용해 0은 빈칸 1은 아이템, 2는 적, 3은 포션, 4는 목적지라고 가정하고 배열을 초기화 함

⑦ 적or 아이템or 포션을 만나면 만났다고 출력,

적을 만나면 체력 -2, 포션을 만나면 체력 +2

1. 코드 스크린샷

```
void user::checkState(user &user_A,int map[][mapX]){
    if(map[user_A.user_y][user_A.user_x]==1){
        cout << "{아이템}이 있습니다."<<endl;
    }
    else if(map[user_A.user_y][user_A.user_x]==2){
        cout << "{적}이 있습니다"<<endl;
        user_A.user_hp-=2;
    }
    else if(map[user_A.user_y][user_A.user_x]==3){
        cout << "{포션}이 있습니다"<<endl;
        user_A.user_hp+=2;
    }
}
```

2. 입력

- userA = user_A객체의 주소
- map = 전체 지도
- user_x = 유저의 x값
- user_y = 유저의 y값
- user_hp = 유저의 체력

3. 반환값

- 없음

4. 결과

- 이동한 좌표에 어떤 것이 있는지 출력

- 적이 있다면 유저의 체력 -2

- 포션이 있다면 유저의 체력 +2

5. 설명

- if문을 통해 유저가 지도에 어디 위치해 있는지 확인하고 위치해 있는 배열 안의 값에 따라 해당 위치에 무엇이 있는지 출력

- 해당 위치에 적이 있었다면 체력을 -2 하고, 포션이 있었다면 체력을 +2

⑧ 한 칸 이동할 때마다 체력 -1

1. 스크린샷

```
void game_excution(user &user_A,int map[][mapX],int num){
    if(num==8){
            user_A.user_y -= 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,0,1);
            }
            else {
                cout << "위로 한 칸 올라갑니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
```

2. 입력

- user_A = 객체 user_A의 주소

- user_hp = 유저의 체력

3. 결과

- 유저가 한 칸 이동할 때마다 체력 -1

4. 설명

- 유저가 right, left, up, down 중 하나를 입력하면 체력 -1

⑨ 체력이 0이되면 게임 끝

1. 코드 스크린샷

```
if(user_A.user_hp==0){
    cout << "실패";
    break;
}
```

2. 입력

- user_hp = 유저의 체력

3. 결과

- 유저의 hp가 0이되면 실패를 출력하고 게임 끝

4. 설명

- if문을 활용해 user_hp가 0이면 실패를 출력하고 break로 무한루프 탈출

⑩ 사용자 입력에서 지도 출력을 고르거나 사용자가 한칸 이동할 때마다

지도 출력

1. 스크린샷

```
void game_excution(user &user_A,int map[][mapX],int num){
    if(num==8){
            user_A.user_y -= 1;
            user_A.user_hp-=1;
            bool inMap = user_A.checkXY(user_A,mapX,mapY);
            if (inMap == false) {
                out_board(user_A,0,1);
            }
            else {
                cout << "위로 한 칸 올라갑니다." << endl;
                displayMap(user_A,map);
                user_A.checkState(user_A ,map);
            }
    }
```

2.. 입력

- user_A = user_A 객체의 주소

- map = 전체 지도

3. 결과

- 한 칸 이동할 때마다 지도 출력

4. 설명

- num값이 8, 6, 4, 2중 하나고 inMap이 ture이면 displayMap 함수를 실행해 지도를 출력, displayMap함수에 관한 설명은 위쪽에 있음

⑪ 사용자가 목적지에 도착하면 게임 승리

1. 코드 스크린샷

```
bool finish = user_A.checkGoal(user_A,map);
if (finish == true) {
    cout << "목적지에 도착했습니다! 축하합니다!" << endl;
    cout << "게임을 종료합니다." << endl;
    break;
}
```

2. 입력

- finish = 목적지에 도착했는지 확인하는 변수

- user_A = user_A 객체의 주소

- map = 전체 지도

3. 결과

- user가 목적지에 도착했으면 목적지에 도착했습니다! 축하합니다! 게임을 종료합니다.를 출력하고 게임 종료

4. 설명

- 유저가 목적지에 도착했는지 확인하는 함수 check_Goal을 활용하여 fisnish에 도착했으면 참 아니면 거짓을 저장

- checkGoal 함수는 위에 자세히 설명

- fisnish가 true면 축하한다는 맨트를 출력하고 break문을 통해 무한루프를 빠져나옴

⑫ 사용자가 map을 입력하면 지도를 출력하고 end를 입력하면 게임 종료

1. 스크린샷

```
else if (user_input == "map") {
    // TODO: 지도 보여주기 함수 호출
    displayMap(user_A,map);
}
else if (user_input == "end") {
    cout << "종료합니다.";
    break;
}
```

2. 입력

- user_input : 사용자가 입력한 값

- user_A = user_A 객체의 주소

- map = 전체 지도

3. 결과

- map을 입력하면 지도를 출력

- end를 입력하면 게임을 종료

4. 설명

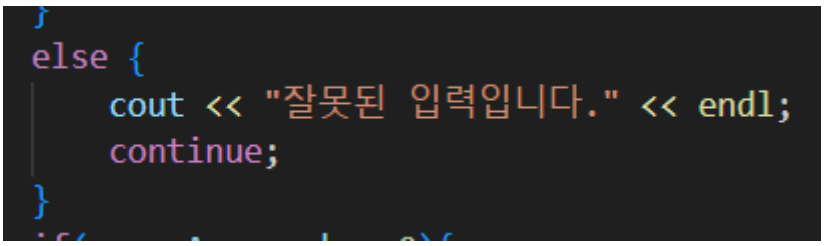
- if문을 활용해 user_input의 값을 확인

- map이면 지도를 출력하는 함수 실행

- end면 종료합니다를 출력하고 break를 통해 무한루프 탈출

⑬ 사용자가 잘못된 값을 입력하면 값 다시 받기

1. 코드 스크린샷



2. 입력

-user_input = 유저가 입력한 값

3. 결과

- 잘못된 입력입니다 출력

- while문 시작으로 돌아가기

4. 설명

- if문으로 user_input을 검사하다가 어떤 조건에도 해당하지 않으면 잘못된 입력으로 간주하고 continue를 사용해 while문의 시작으로 되돌림

4. 테스트

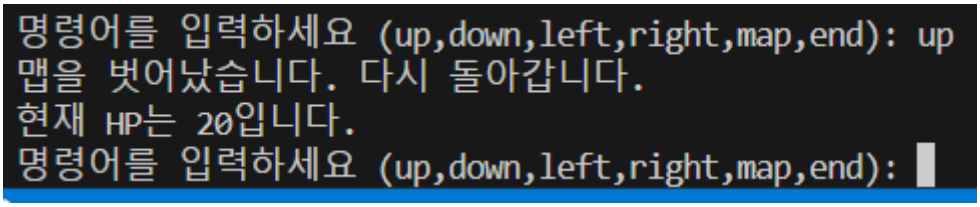
1. 기능 별 테스트 결과: (요구사항 별 스크린샷)

① class선언과 class 정의를 분리해 user 객체 만들기

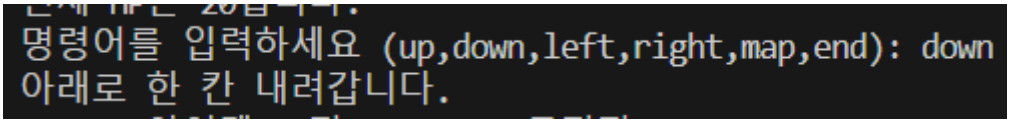
② 현재 체력 출력

③ 사용자가 이동하고 싶은 방향 or 지도 출력 or 게임 끝내기 중 선택 받기

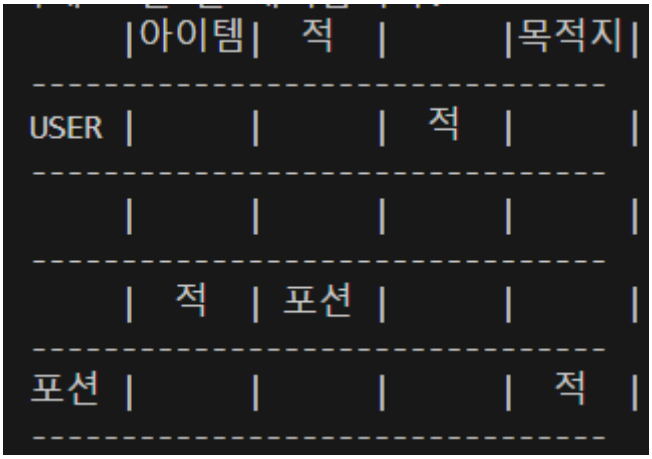
④ 입력 받은 방향의 좌표 유효성 체크 후 유효하지 않으면 입력 다시 받기



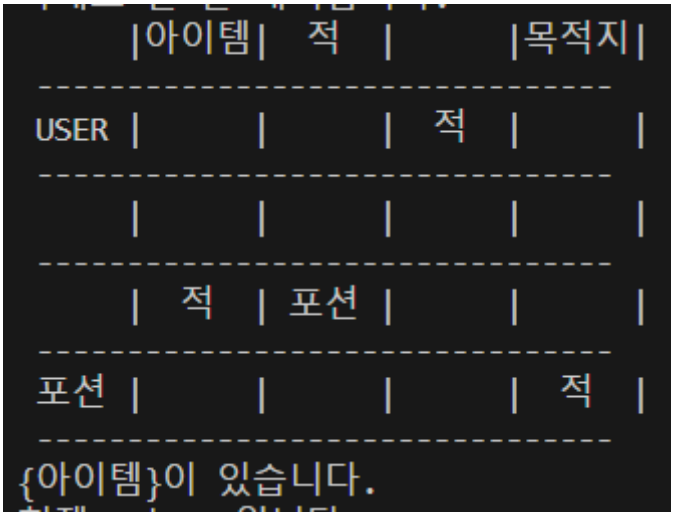
⑤ 유효한 좌표라면 해당 좌표로 user 이동

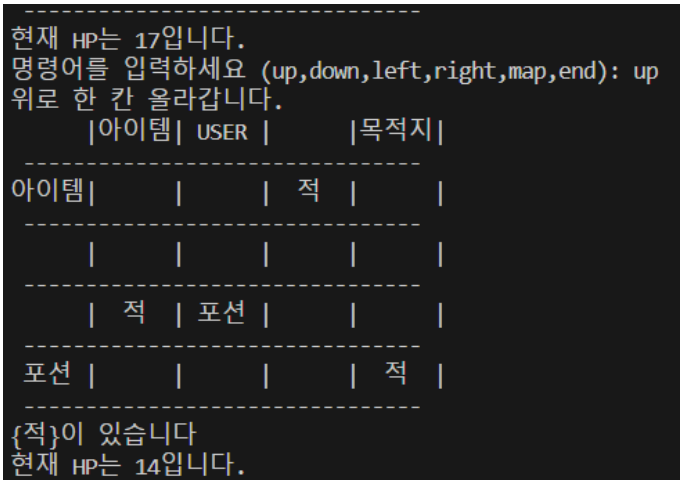


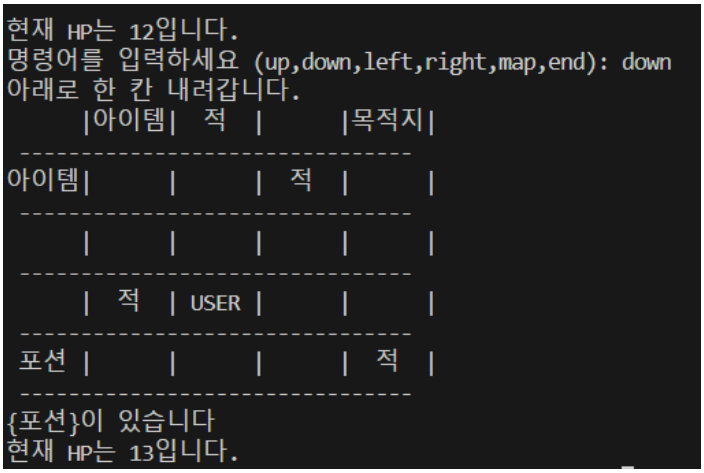
⑥ map에 아이템, 적, 포션, 목적지 배치



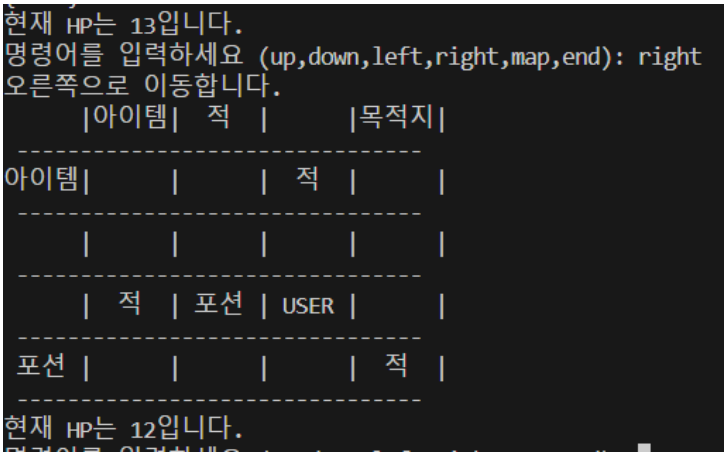
⑦ 적or 아이템or 포션을 만나면 만났다고 출력, 적을 만나면 체력 -2, 포션을 만나면 체력 +2



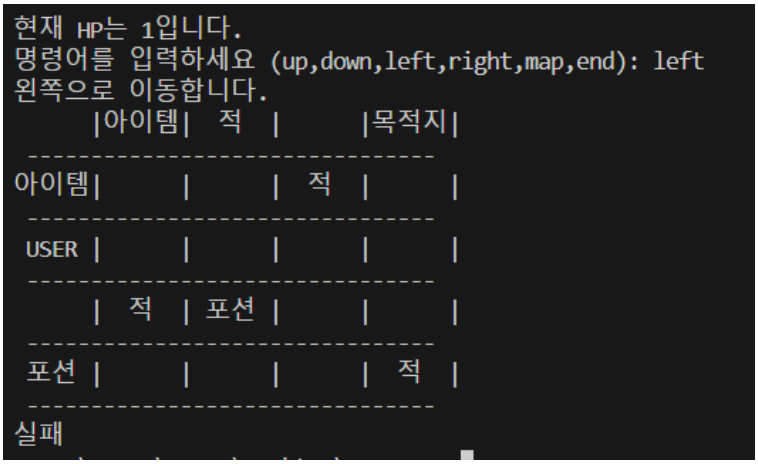




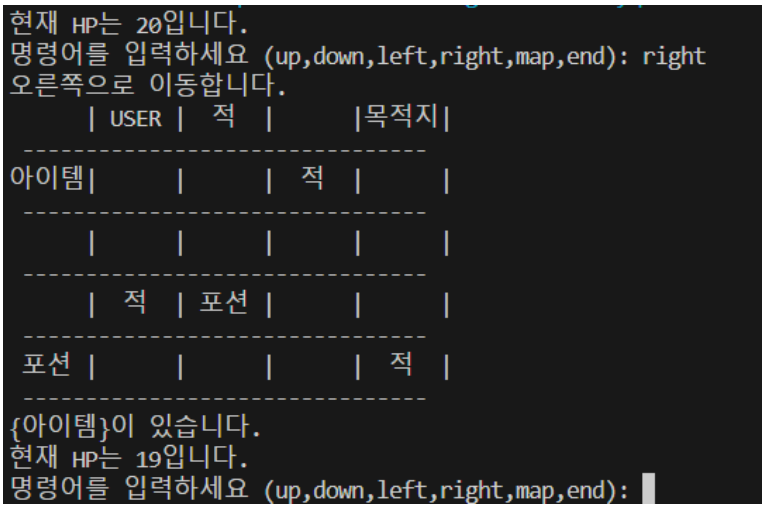
⑧ 한 칸 이동할 때마다 체력 -1



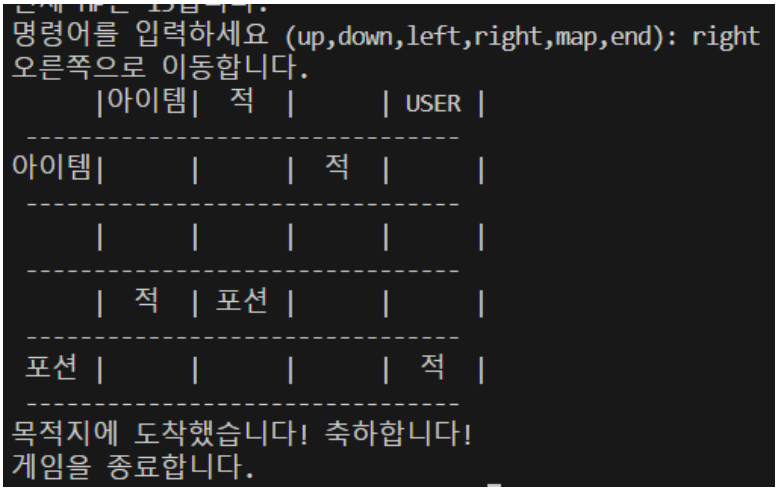
⑨ 체력이 0이되면 게임 끝



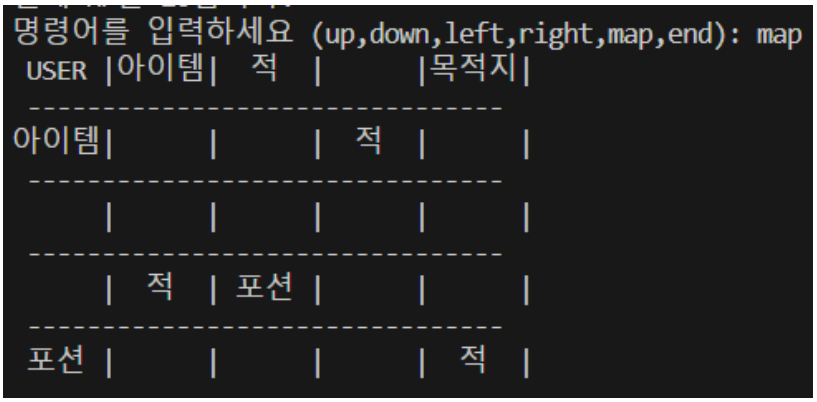
⑩ 사용자 입력에서 지도 출력을 고르거나 사용자가 한 칸 이동할 때마다 지도 출력

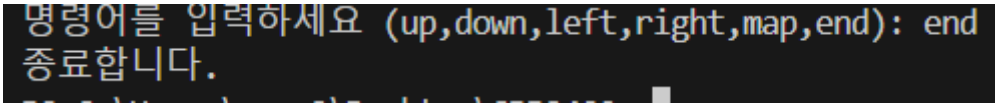


⑪ 사용자가 목적지에 도착하면 게임 승리

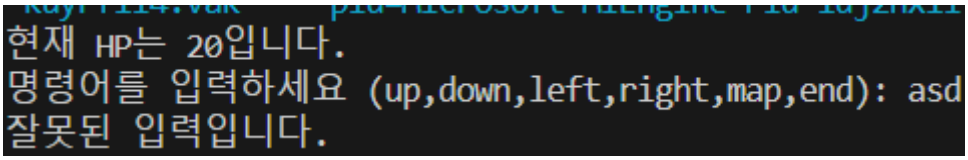


⑫ 사용자가 map을 입력하면 지도를 출력하고 end를 입력하면 게임 종료

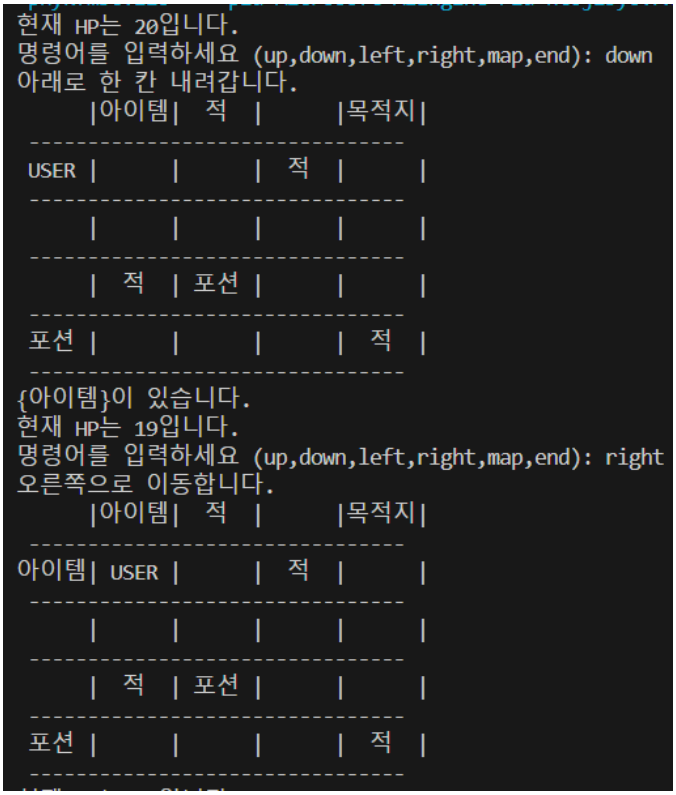


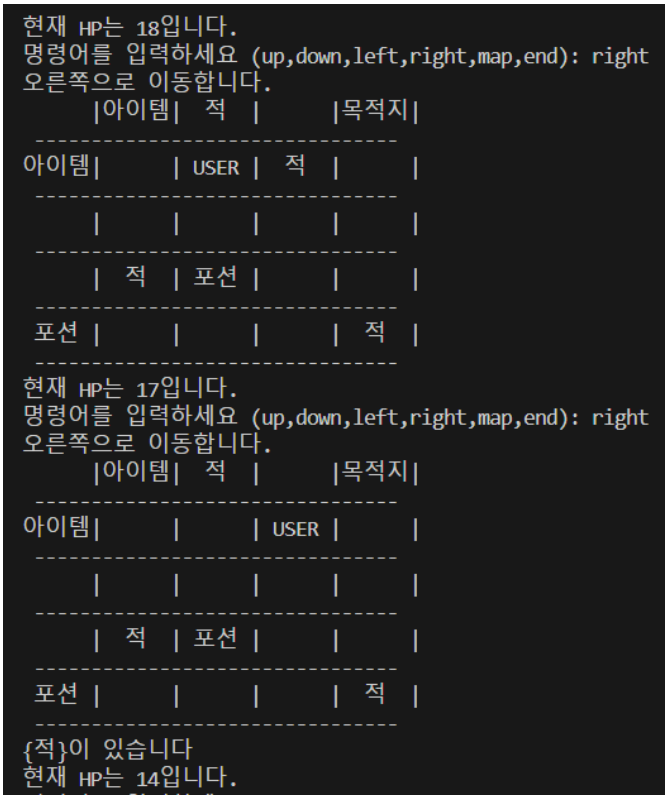


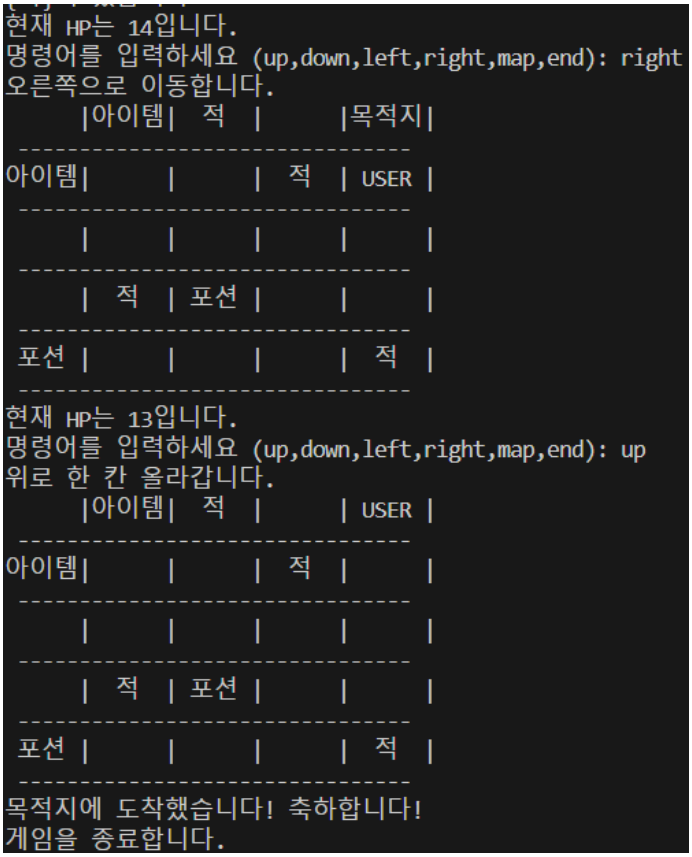
⑬ 사용자가 잘못된 값을 입력하면 값 다시 받기



2. 최종 테스트 스크린샷: (프로그램 전체 동작 스크린샷)







5. 결과 및 결론

1. 프로젝트 결과: class와 객체를 활용해 mud game을 만들었다.

2. 느낀 점: 자바를 배울 때 class 개념을 배웠지만 짜여진 코드를 읽기만 했지 헤더파일을 만들고, 거기에 클래스를 선언하고, 다른 cpp 파일에서 클래스를 정의하고, main 함수에서 class를 사용해 코드를 짜보는 건 처음 해봤다. class를 이용해 코드를 직접 짜보니 막연하게 알고있던 class 개념을 더 확실히 이해할 수 있게 되었다. 처음에 포인터를 이용해 매개변수를 넘기며 코드를 짰었는데 클레스를 선언하고 객체를 이용하니 코드가 훨씬 깔끔해졌다. 수업중에 교수님이 하신 말씀처럼 player가 늘어나도 객체를 이용한 코딩을 한다면 여러 개의 객체를 만들면 되니 훨씬 관리하기 쉬울 것이다. 나는 기말 프로젝트로 체스게임을 만들고 싶은데 거기에 응용하면 제격일 것 같다.

막연하게 짤 수 있겠지 했던 mud_game 코드였는데 막상 짜보니 오류가 많이 발생했다. 특히 객체를 매개변수로 넘기는 부분에서 오류가 많이 발생했는데, 인터넷 서칭과 수업자료를 통해 해결했다. 시간은 많이 썼지만 시간에 비해 얻은 게 많은 과제였다.