mud_game.pdf

학번 : 175989

이름 : 백성준

1. 서론
2. 요구사항
3. 설계 및 구현
4. 테스트
5. 결과 및 결론

1. 서론

-프로젝트 목적 및 배경 : 함수 사용의 숙달을 위하여 틱택토를 심화하지 않고 함수 숙달에 도움이 되는 게임에 대해 알아보고 실습한다.

-목표: 좌표이동과 상호작용 사물이 있는 Mud 게임을 구현하여 본다.

2. 요구사항

-사용자 요구사항 : 상하좌우 이동이 있으며 오브젝트와의 상호작용이 가능하며, 결과적으로 목적지에 도착하는 것이 목적인 게임

-기능 계획

① 사용자에게 상, 하, 좌, 우, 지도, 종료 중 하나를 입력받기

-상,하,좌,우 입력 시 이동과 지도 표시

-지도를 입력하면 전체지도와 위치를 표시

-다른 값이 입력되면 메시지 표시

② 지도 밖으로 나가면 에러 메시지 출력 및 원상복귀

③ 목적지에 도달하면 성공의 출력과 게임의 종료

④ 체력의 구현(HP 20 시작)

⑤ 체력이 모두 소모하면 게임 종료

⑥ 이동 시 체력 감소

⑦ 처음 명령문 입력 시 HP 출력

⑧ 오브젝트와 상호작용 및 메시지 출력

-함수 계획

① 메인함수: 사용자의 입력을 받기 및 게임 진행 동안 무한반복

② 지도와 현재 위치 출력 : displayMap()

③ 사용자 위치 체크 : checkXY()

④ 목적지 도착 체크 : checkGoal()

⑤ 오브젝트와 상호 작용 : checkState()

⑥ 유효하지 않은 값 되돌리기 : opti()

3. 설계 및 구현

-기능 별 구현 사항

① 사용자에게 상, 하, 좌, 우, 지도, 종료 중 하나를 입력받기

가) 스크린샷

```
// 사용자의 입력을 저장할 변수
string user_input = "";
cout << "현재 HP: " << Hp;
cout << "명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
cin >> user_input;
```

나) 입력

String user_input : 사용자의 입력을 저장할 변수

다) 반환값

없음

라) 결과

안내문구 출력

user_input 에 유저가 입력한 구문이 저장됨.

마) 설명

유저에게 안내 멘트를 하고 그 입력을 입력받는다.

-상,하,좌,우 입력 시 이동과 지도 표시

가)스크린샷

"상"입력

```
if (user_input == "상") {//입력 받은 값이 상일 경우
    // 위로 한 칸 올라가기
    user_y -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);//이동한 값이 타당한지를 확인하는 함수. 잘못
    if (inMap == false) {//위의 함수가 false를 출력한 경우
        opti(user_x, mapX, user_y, mapY);
    }
    else {//입력 받은 이동 값이 타당한 경우
        cout << "위로 한 칸 올라갑니다." << endl;//메세지와 맵을 출력하는 함수를 실행
        displayMap(map, user_x, user_y);
        Hp--;//이동이 유효하여 이동 했으니 감소시킴 HP를
        checkState(map, user_x, user_y);//적, 포션 등을 만났을 때 반응및 메세지를 위한 함수
    }
}
```

"하"입력

```
else if (user_input == "하") {//상과 원리가 동일. x나 y
    // TODO: 아래로 한 칸 내려가기
    user_y += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        opti(user_x, mapX, user_y, mapY);
    }
    else {
        cout << "위로 한 칸 내려갑니다." << endl;
        displayMap(map, user_x, user_y);
        Hp--;
        checkState(map, user_x, user_y);
    }
}
```

"좌"입력

```
else if (user_input == "좌") {//상과 원리가 동일. x나 y의
    // TODO: 왼쪽으로 이동하기
    user_x -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);

    if (inMap == false) {
        opti(user_x, mapX, user_y, mapY);
    }
    else {
        cout << "왼쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        Hp--;
        checkState(map, user_x, user_y);
    }
```

"우"입력

```
else if (user_input == "우") {//상과 원리가 동일. x나 y의 값만 그에 맞게 설정
    // TODO: 오른쪽으로 이동하기
    user_x += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        opti(user_x, mapX, user_y, mapY);
    }
    else {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        Hp--;
        checkState(map, user_x, user_y);
    }
}
```

나) 입력

User_input : user 가 입력한 값

User_x : 유저의 x 위치 값

User_y : 유저의 y 위치 값

displayMap(map, user_x, user_ y) : 지도 출력을 위한 함수

다) 반환값

없음

라) 결과

이동에 맞는 좌표에 맞는 값이 되도록 좌표에 가감을 실시. 좌(user_x 에 -1), 우(user_x 에 +1), 상(usre_y 에 +1), 하(user_y 에 -1)

이동에 맞는 메시지 출력

이동 후 지도 출력

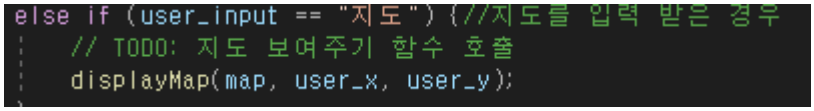
마) 설명

사용자가 입력한 값에 따라 이동(사용자 좌표의 수정), 그 후, 함수를 통한 지도 출력을 한다.

-지도를 입력하면 전체지도와 위치를 표시

가) 스크린샷

"지도"입력



나) 입력

User_input : 사용자의 입력을 확인하기 위함

displayMap(map, user_x, user_y) : 지도와 유저 위치를 표시하기 위한 함수

다) 반환값

없음

라) 결과

displayMap()의 실행

마) 설명

지도가 입력 되었을 때, diplayMap()의 실행을 통한 지도와 유저위치 출력.

-다른 값이 입력되면 메시지 표시

가) 스크린샷

```
else {//위의 경우가 아닌 다른 것을 입력한경우
    cout << "잘못된 입력입니다." << endl;
    continue;//다시 위로 돌아감
}
```

나) 입력

없음

다) 반환값

없음

라) 결과

잘못된 입력이라는 메시지를 출력.

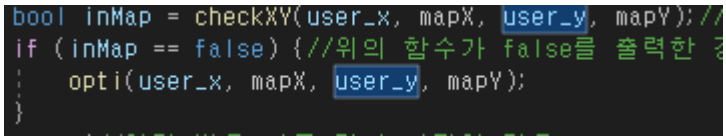
무한 루프의 다음 반복으로 넘어감.

마) 설명

프로그램되지 않은 입력이 왔을 때, 다음의 메시지를 출력하고 무한루프의 처음으로 돌아간다.

② 지도 밖으로 나가면 에러 메시지 출력 및 원상복귀

가) 스크린샷



나) 입력

inMap : checkXY 에서 반환한 값을 받기 위한 변수

checkXY(user-x,mapX,user_y,mapY) : 유저가 이동하려는 방향이 유효한지 확인하는 함수. 잘못되었다면 false 를 출력.

opti(user-x,mapX,user_y,mapY) : 요류 메시지의 출력과 유저 위치 좌표를 되돌리기 위한 함수.

다) 반환값

checkXY()의 참 거짓 출력. 이동 방향의 유효성에 따라 달라진다.

라) 결과

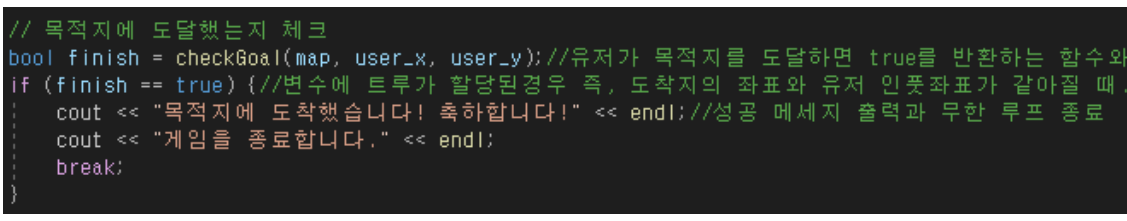
유효하지 않은 이동이 입력되었을 때, 오류메시지를 출력하고 이동했던 방향으로 회귀한다.

마) 설명

유효하지 않은 방향 입력을 막고(유효성 검사) 오류 메시지를 출력한다.

③ 목적지에 도달하면 성공의 출력과 게임의 종료

가) 스크린샷



나) 입력

Finish : 함수에서 나오는 반환되는 값을 받기 위한 변수

checkGoal(map, user_x, user_y) : 목적지에 도달하였는지 확인하는 함수

다) 결과

checkGoal()에서 true 를 반환하면 도착 메시지를 출력하고 무한루프 탈출

라) 설명

checkGoal()을 통해 목적지에 도착했는지 확인하고, 도착했다면 도착 메시지를 출력하고 종료한다.

④ 체력의 구현(HP 20 시작)

가) 스크린샷

나) 입력

Hp : 사용자의 체력

다) 설명

사용자의 체력을 전역변수로 선언하여 체력을 구현하였다.

⑤ 체력이 모두 소모하면 게임 종료

가) 스크린샷

```
//Hp가 0이면 종료를 위함
if (Hp == 0)//0일 경우
{
    cout << "Hp가 0이 되었습니다." << endl << "게임을 종료합니다.";
    break;
}
```

나) 입력

Hp : 사용자의 체력

다) 결과

체력이 모두 소모되었음을 출력하고, 게임을 종료한다.

라) 설명

사용자의 체력이 0이 되면 메시지를 출력하고 무한루프를 탈출한다.

⑥ 이동 시 체력 감소

가) 스크린샷

```
else {//입력 받은 이동 값이 타당한 경우
    cout << "위로 한 칸 올라갑니다." << endl;//메세지와 맵을 출력하는 함수를 실행
    displayMap(map, user_x, user_y);
    Hp--;//이동이 유효하여 이동 했으니 감소시킴 HP를
    checkState(map, user_x, user_y);//적, 포션 등을 만났을 때 반응및 메세지를 위한
}
```

```
else {
    cout << "위로 한 칸 내려갑니다." << endl;
    displayMap(map, user_x, user_y);
    Hp--;
    checkState(map, user_x, user_y);
}
else {
    cout << "왼쪽으로 이동합니다." << endl;
    displayMap(map, user_x, user_y);
    Hp--;
    checkState(map, user_x, user_y);
}
else {
    cout << "오른쪽으로 이동합니다." << endl;
    displayMap(map, user_x, user_y);
    Hp--;
    checkState(map, user_x, user_y);
}
```

나) 입력

Hp : 사용자의 체력

다) 결과

Hp 가 1 떨어진다.

라) 설명

이동하고 지도를 표시하는 코드블록 내에 hp 를 1 떨어뜨려주어 이동 시에 체력이 감소하는 것을 구현하였다.

⑦ 처음 명령문 입력 시 HP 출력

가) 스크린샷

```
cout << "현재 HP: " << Hp;
```

나) 입력

Hp : 사용자의 현재 Hp

다) 결과

현재 Hp 가 메시지와 함께 출력된다.

라) 설명

무한루프의 시작점에 넣어서 무한루프의 시작 시에 Hp 가 출력된다.

⑧ 오브젝트와 상호작용 및 메시지 출력

가) 스크린샷

```
else {//입력 받은 이동 값이 타당한 경우
    cout << "위로 한 칸 올라갑니다." << endl;//메세지와 맵을 출력하는 함수를 실행
    displayMap(map, user_x, user_y);
    Hp--;//이동이 유효하여 이동 했으니 감소시킴 HP를
    checkState(map, user_x, user_y);//적, 포션 등을 만났을 때 반응및 메세지를 위한 함수
}
```

```
else {
    cout << "위로 한 칸 내려갑니다." << endl;
    displayMap(map, user_x, user_y);
    Hp--;
    checkState(map, user_x, user_y);
}
else {
    cout << "왼쪽으로 이동합니다." << endl;
    displayMap(map, user_x, user_y);
    Hp--;
    checkState(map, user_x, user_y);
}
else {
    cout << "오른쪽으로 이동합니다." << endl;
    displayMap(map, user_x, user_y);
    Hp--;
    checkState(map, user_x, user_y);
}
```

나) 입력

checkState(map, user_x, user_y) :

지도의 해당위치에 있는 물체와 상호작용을 위한 함수

다) 결과

상호작용을 하고(Hp 의 감소 등) 그 결과를 메시지로 출력한다.

라) 설명

이동의 결과와 지도의 표시를 하는 코드 블록 내에 넣어 해당 행동을 할 때, 지도 상의 물체와의 상호작용을 하고 관련 메시지를 출력한다.

-함수 계획

① 메인함수: 사용자의 입력을 받기 및 게임 진행 동안 무한반복

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

    //Hp가 0이면 종료를 위함
    if (Hp == 0)//0일 경우
    {
        cout << "Hp가 0이 되었습니다." << endl << "게임을 종료합니다.";
        break;
    }
    // 사용자의 입력을 저장할 변수
    string user_input = "";
    cout << "현재 HP: " << Hp;
    cout << "명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
    cin >> user_input;

    if (user_input == "상") {//입력 받은 값이 상일 경우
        // 위로 한 칸 올라가기
        user_y -= 1;
        bool inMap = checkXY(user_x, mapX, user_y, mapY);//이동한 값이 타당한지를 확인하는 함수. 잘못되면 flase를 출력. inMap은 그 값을 받기 위한 변수
        if (inMap == false) {//위의 함수가 false를 출력한 경우
            opti(user_x, mapX, user_y, mapY);
        }
        else {//입력 받은 이동 값이 타당한 경우
            cout << "위로 한 칸 올라갑니다." << endl;//메세지와 맵을 출력하는 함수를 실행
            displayMap(map, user_x, user_y);
            Hp--;//이동이 유효하여 이동 했으니 감소시킴 HP를
            checkState(map, user_x, user_y);//적, 포션 등을 만났을 때 반응할 메세지를 위한 함수
        }
    }
    else if (user_input == "하") {//상과 원리가 같음. x나 y의 값만 그에 맞게 설정
        // TODO: 아래로 한 칸 내려가기
        user_y += 1;
        bool inMap = checkXY(user_x, mapX, user_y, mapY);
        if (inMap == false) {
            opti(user_x, mapX, user_y, mapY);
        }
        else {
            cout << "위로 한 칸 내려갑니다." << endl;
            displayMap(map, user_x, user_y);
            Hp--;
            checkState(map, user_x, user_y);
```

```
            }
        }
        else if (user_input == "좌") {//상과 원리가 같음. x나 y의 값만 그에 맞게 설정
            // TODO: 왼쪽으로 이동하기
            user_x -= 1;
            bool inMap = checkXY(user_x, mapX, user_y, mapY);

            if (inMap == false) {
                opti(user_x, mapX, user_y, mapY);
            }
            else {
                cout << "왼쪽으로 이동합니다." << endl;
                displayMap(map, user_x, user_y);
                Hp--;
                checkState(map, user_x, user_y);
            }
        }
        else if (user_input == "우") {//상과 원리가 같음. x나 y의 값만 그에 맞게 설정
            // TODO: 오른쪽으로 이동하기
            user_x += 1;
            bool inMap = checkXY(user_x, mapX, user_y, mapY);
            if (inMap == false) {
                opti(user_x, mapX, user_y, mapY);
            }
            else {
                cout << "오른쪽으로 이동합니다." << endl;
                displayMap(map, user_x, user_y);
                Hp--;
                checkState(map, user_x, user_y);
            }
        }
        else if (user_input == "지도") {//지도를 입력 받은 경우
            // TODO: 지도 보여주기 함수 호출
            displayMap(map, user_x, user_y);
        }
        else if (user_input == "종료") {//종료를 입력한 경우
            cout << "종료합니다.";
            break;//break로 무한 루프를 탈출
        }
        else {//위의 경우가 아닌 다른 것을 입력한경우
            cout << "잘못된 입력입니다." << endl;
            continue;//다시 위로 올라감
        }

        // 목적지에 도달했는지 체크
        bool finish = checkGoal(map, user_x, user_y);//유저가 목적지를 도달하면 true를 반환하는 함수와 그 값을 받기 위한 변수
        if (finish == true) {//변수에 트루가 할당된경우 즉, 도착지의 좌표와 유저 인덱좌표가 같아질 때.
            cout << "목적지에 도착했습니다! 축하합니다!" << endl;//성공 메세지 출력과 무한 루프 종료
            cout << "게임을 종료합니다." << endl;
            break;
        }

    }
    return 0;
}
```

② 지도와 현재 위치 출력 : displayMap()

가)스크린샷

```
// 지도와 사용자 위치 출력하는 함수
void displayMap(int map[][mapX], int user_x, int user_y) {//배열과 입력 좌표를 매개변수로 받음
    for (int i = 0; i < mapY; i++) {//맵의 세로만큼 반복
        for (int j = 0; j < mapX; j++) {//맵의 각 세로에서 가로만큼 반복
            //각 배열을 탐색하되 그 배열에 유저가 있을 때에는 유저를 출력, 그 배열에 유저가 존재하지 않으면
            //각 메세지를 출력할 때 가로를 나누는 칸도 출력
            if (i == user_y && j == user_x) {//그 반복 시기 때 좌표와 입력좌표가 같으면
                cout << " USER |"; // 양 옆 1칸 공백 //유저를 출력
            }
            else {//유저 존재하지 않는 칸의 반복을 시행하고 있을 때
                int posState = map[i][j];//이때는 지도의 배열에 해당하는 값(아이템, 적 등)을 변수에 받아옴
                switch (posState) {
                case 0://빈 공간일 경우
                    cout << "      |"; // 6칸 공백
                    break;
                case 1://아이템을 경우
                    cout << "아이템|";
                    break;
                case 2://적일 경우
                    cout << "  적  |"; // 양 옆 2칸 공백
                    break;
                case 3://포션일 경우
                    cout << " 포션 |"; // 양 옆 1칸 공백
                    break;
                case 4://목적지일 경우
                    cout << "목적지|";
                    break;
                }
            }
        }
        cout << endl;
        cout << " ------------------------------ " << endl;//세로의 반복문을 실행할 때 1회 시행할 때마다
    }
}
```

나) 입력

- Int map[][mapX] 전체 위치 정보를 담고 있는 배열
- user_x : 유저의 x 위치 값
- user_y : 유저의 y 위치 값

다) 반환값

없음

라) 결과

지도의 틀을 출력 하는 동시에 중간에 지도의 정보를 출력.

각 배열의 x 와 y 축 인덱스를 탐색하다가 유저의 위치 값과 일치하면 "user"를 출력하고 그것이 아니라면 그 위치에 할당된 오브젝트나 공란을 출력.

마) 설명

지도의 틀과 그 안의 내용을 출력한다. 유저의 위치가 나오면 유저의 출력이 우선시 된다.

③ 사용자 위치 체크 : checkXY()

가) 스크린샷

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int user_x, int mapX, int user_y, int mapY) {
    bool checkFlag = false;//반환값을 위한 변수를 선언
    if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) {//입력 좌표가 0이상이여야 하고 맵의 크기보다 작을 경
        checkFlag = true;//트루를 할당하여 유효한 좌표임을 출력
    }
    return checkFlag;// 받은 값 출력
}
```

나) 입력

User_x : 유저 위치의 x 값

User_y : 유저 위치의 y 값

mapX : 지도의 x 값

mapY : 지도의 y 값

checkFlag : 함수의 반환 값을 저장하기 위함.

다) 반환값

기본으로 false 를 출력. 만약 사용자의 좌표가 지도를 벗어나지 않았다면 true 를 반환한다.

라) 결과

지도 내에 유저가 존재하면 true, 아니면 false 를 반환한다.

마) 설명

사용자의 좌표가 지도 내에 존재한다면 true 를 출력한다. 그 외의 경우(좌표가 유효하지 않음)에는 checkFlag 의 값이 불변하여 false 가 출력된다.

④ 목적지 도착 체크 : checkGoal()

가) 스크린샷

```
// 유저의 위치가 목적지인지 체크하는 함수
bool checkGoal(int map[][mapX], int user_x, int user_y) {// 입력 좌표값과 배열을 매개변수로
    // 목적지 도착하면
    if (map[user_y][user_x] == 4) {//입력 좌표의 값이 들어간 배열에 목적지가 있으면
        return true;//트루를 출력
    }
    return false;//if 가 실행되지 않은 목적지가 아니면 false를 출력
}
```

나) 입력

Map[][mapX] : 지도의 실질적 역할을 하는 배열

User_x : 유저 위치의 x 값

User_y : 유저 위치의 y 값

다) 반환값

목적지에 유저가 위치하면 true, 목적지에 그 외에는 false 를 반환한다.

라) 결과

유저의 좌표와 지도 상의 좌표를 비교하여 유저의 위치에 목적지가 존재하면 true, 아니면 false 를 반환한다.

마) 설명

유저의 좌표값을 배열에 넣고, 그 위치의 배열(지도의 역할)이 4(목적지)의 값을 갖는지 확인. 4(목적지)가 맞다면 true 를 반환한다.

⑤ 오브젝트와 상호 작용 : checkState()

가) 스크린샷

```
void checkState(int map[][mapX], int user_x, int user_y)
{
    int posState = map[user_y][user_x];//각 배열 인덱스에 있는 값을 받을 변수를 선언 및 그 배열의 값으로 초기화
    switch (posState)//즉, 그 배열에 어떤 것이 들어있는지 확인하고 그에대한 반응을 함.
    {
    case 0://공란일 경우, 아무것도 하지 않음
        break;
    case 1://아이템일 경우
        cout << "아이템을 만났습니다.₩n";//아이템을 만났다는 함수 출력
        break;
    case 2://적을 만난 경우
        cout << "적을 만났습니다. Hp가 2 줄어듭니다.₩n";//메세지 출력
        Hp -= 2;//데미지를 줌
        break;
    case 3://포션을 만난경우
        cout << "포션을 획득했습니다. Hp가 2 늘어납니다.₩n";//메세지를 출력하고 회복을함
        Hp += 2;
    case 4://목적지인 경우 아무것도 하지 않음
        break;
    }
}
```

나) 입력

Map[][mapX] : 지도의 실질적 역할을 하는 배열

User_x : 유저 위치의 x 값

User_y : 유저 위치의 y 값

posState : 배열에 저장된 값을 함수 내에서 가지고 있기 위한 변수.

Hp : 유저의 체력

다) 반환값

Void

라) 결과

유저의 좌표 값을 배열에 할당하여 그곳의 배열의 값에 따라 각 case 를 실행한다.

0 일 경우 switch 문을 탈출.

1 일 경우 아이템 관련 메시지 출력.

2 일 경우 유저의 체력을 2 줄이고, 적을 만났음을 출력.

3 일 경우 유저의 체력을 2 늘리고, 포션을 먹었음을 출력.

4 일 경우 switch 문을 탈출.

마) 설명

유저의 좌표 값을 map[][]에 넣고 그 값을 posState 에 할당하여, 이를 가지고 switch 문을 실행 시킨다. 공란(0), 목적지(4)일 경우 아무 행동도 취하지 않고 함수가 종료된다. 2(적), 3(포션)이 할당되어 있다면 그에 맞는 체력 변수의 값을 변경하고 메시지를 출력한다.

⑦ 유효하지 않은 값 되돌리기 : opti()

가) 스크린샷

```
//오류 메세지 출력 및 이동 취소
void opti(int& user_x, int mapX, int& user_y, int mapY)
{//맵 바깥으로 나가는 네 가지의 상황에 맞는 좌표 값의 수정
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;//오류 메세지를 출력하기 위함
    if (user_x >= mapX)//맵보다 좌표가 커지면
    {
        user_x -= 1;
    }
    else if (user_x < 0)//맵의 바깥으로 나가면
    {
        user_x += 1;
    }
    else if (user_y >= mapY)//맵보다 좌표가 커지면
    {
        user_y -= 1;
    }
    else if (user_y < 0)//좌표가 바깥으로 나가면
    {
        user_y += 1;
    }
}
```

나) 입력

User_x : 유저 위치의 x 값

User_y : 유저 위치의 y 값

mapX : 지도의 x 값

mapY : 지도의 y 값

다) 반환값

void

라) 결과

유효하지 않음을 뜻하는 메시지 출력.

좌표가 지도의 크기와 같거나 크면 각 좌표(x or y)의 값에서 1 이 차감된다.

좌표가 0 보다 작으면 (x or y)의 값에서 1 이 증가된다.

마) 설명

좌표가 지도의 크기와 같거나 크다는 것은 지도(map[][])를 양의 방향으로 벗어났다는 것이고, 좌표가 0 보다 작다는 것은 지도(map[][])를 음의 방향으로 벗어났다는 것이다. 따라서 각각 좌표 값의 가감을 통해서 범위 내로 넣어준다는 것은 유효하지 않은 좌표 값을 원래대로 돌리는 효과가 있는 것이다.

4. 테스트

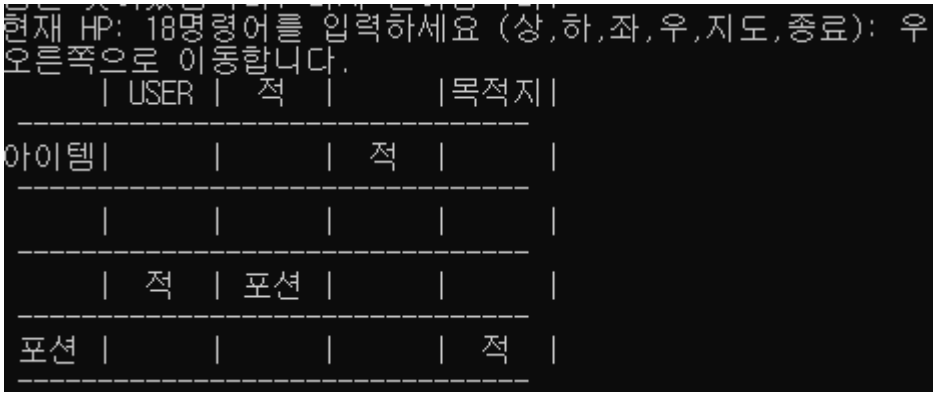
1. 기능별 테스트

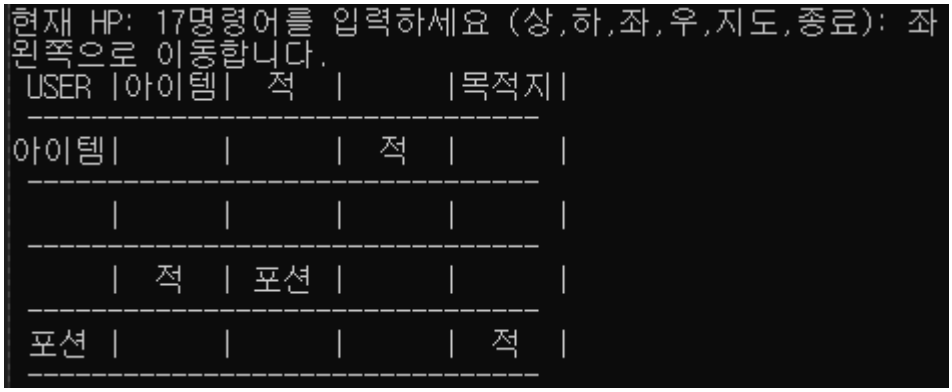
① 사용자에게 상, 하, 좌, 우, 지도, 종료 중 하나를 입력받기

```
현재 HP: 20명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
현재 HP: 20명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
현재 HP: 19명령어를 입력하세요 (상,하,좌,우,지도,종료): 좌
현재 HP: 19명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
현재 HP: 20명령어를 입력하세요 (상,하,좌,우,지도,종료): 지도
현재 HP: 18명령어를 입력하세요 (상,하,좌,우,지도,종료): 종료
```

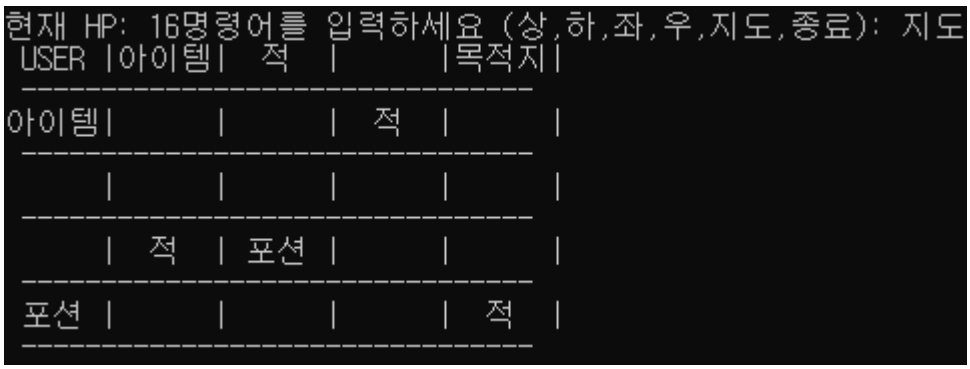
-상,하,좌,우 입력 시 이동과 지도 표시

```
현재 HP: 20명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로로 한 칸 내려갑니다.
     |아이템|  적  |      |목적지|
 ---------------------------------
 USER |      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
현재 HP: 19명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
위로 한 칸 올라갑니다.
 USER |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
```

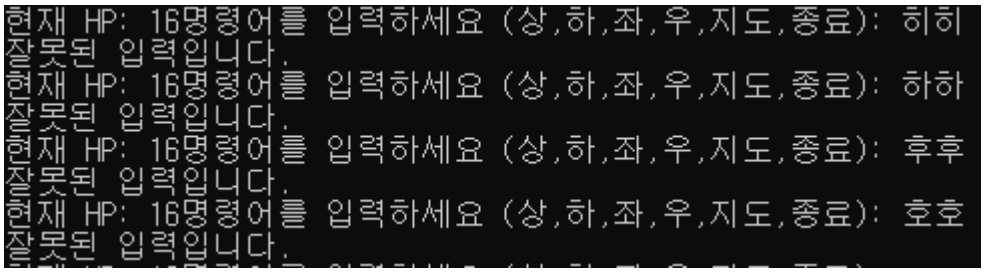




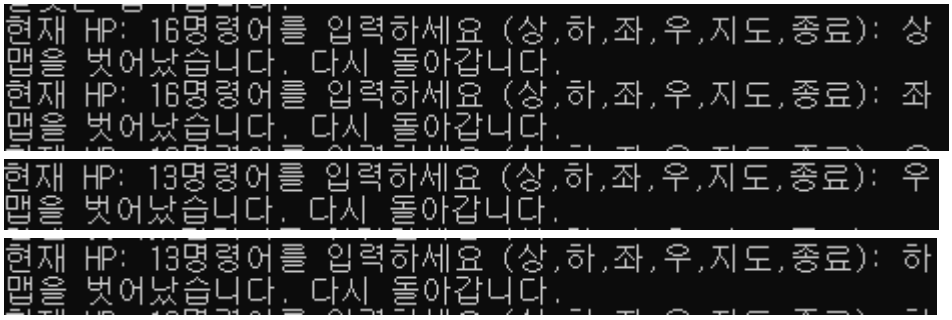
-지도를 입력하면 전체지도와 위치를 표시



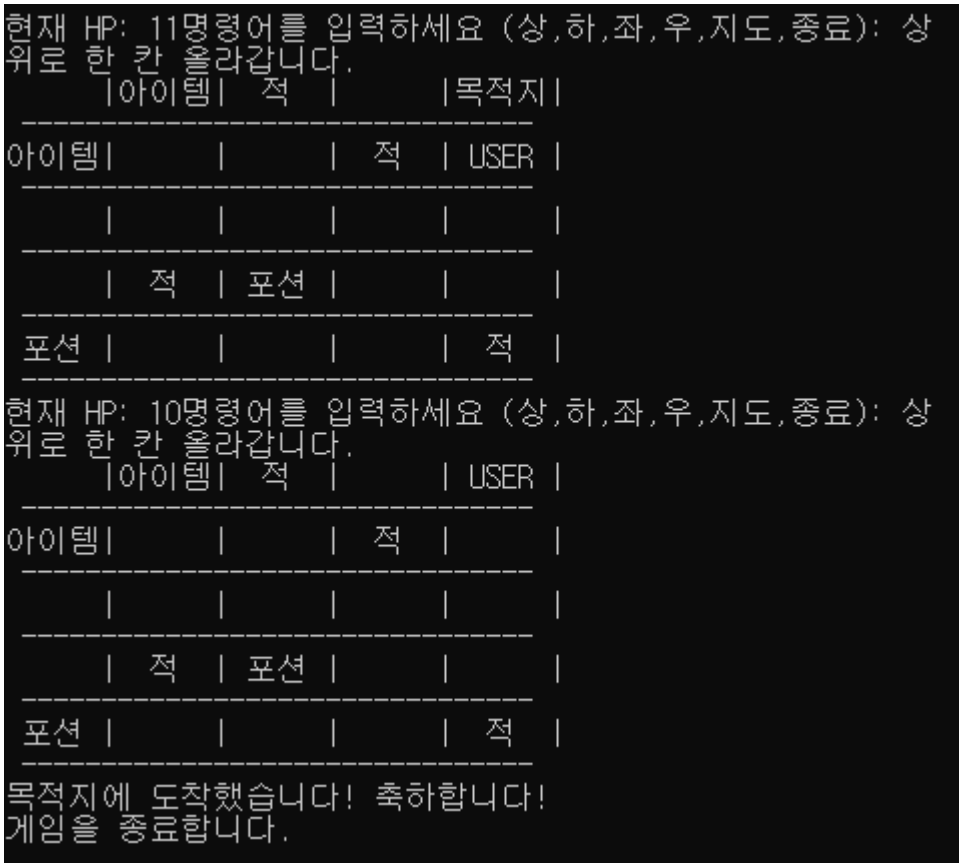
-다른 값이 입력되면 메시지 표시



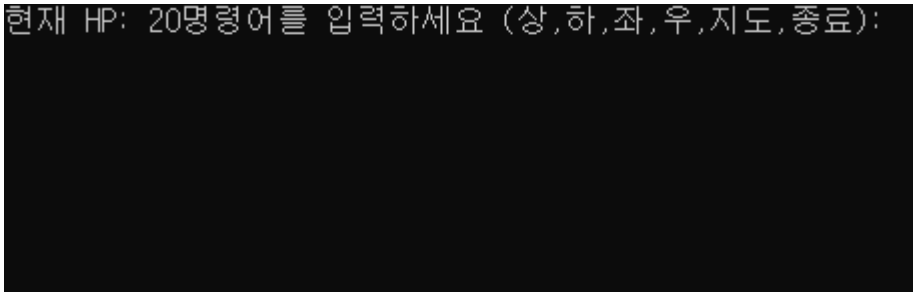
② 지도 밖으로 나가면 에러 메시지 출력 및 원상복귀



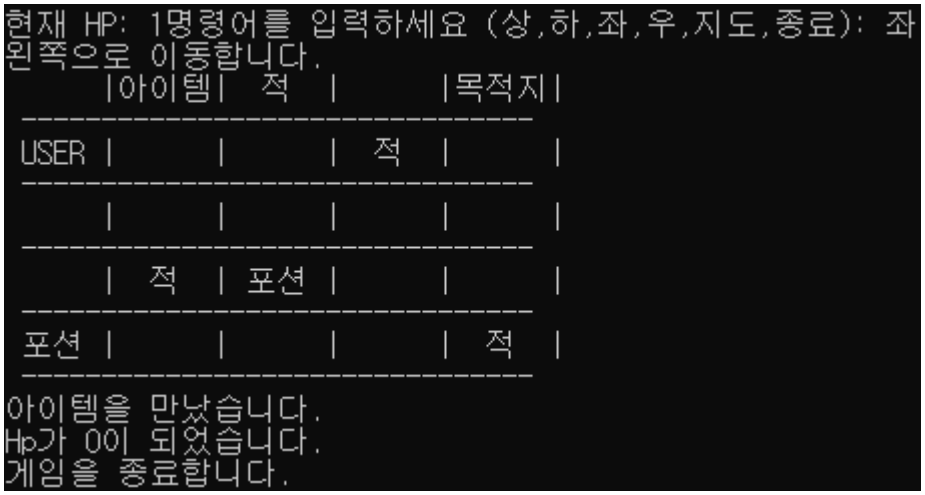
③ 목적지에 도달하면 성공의 출력과 게임의 종료



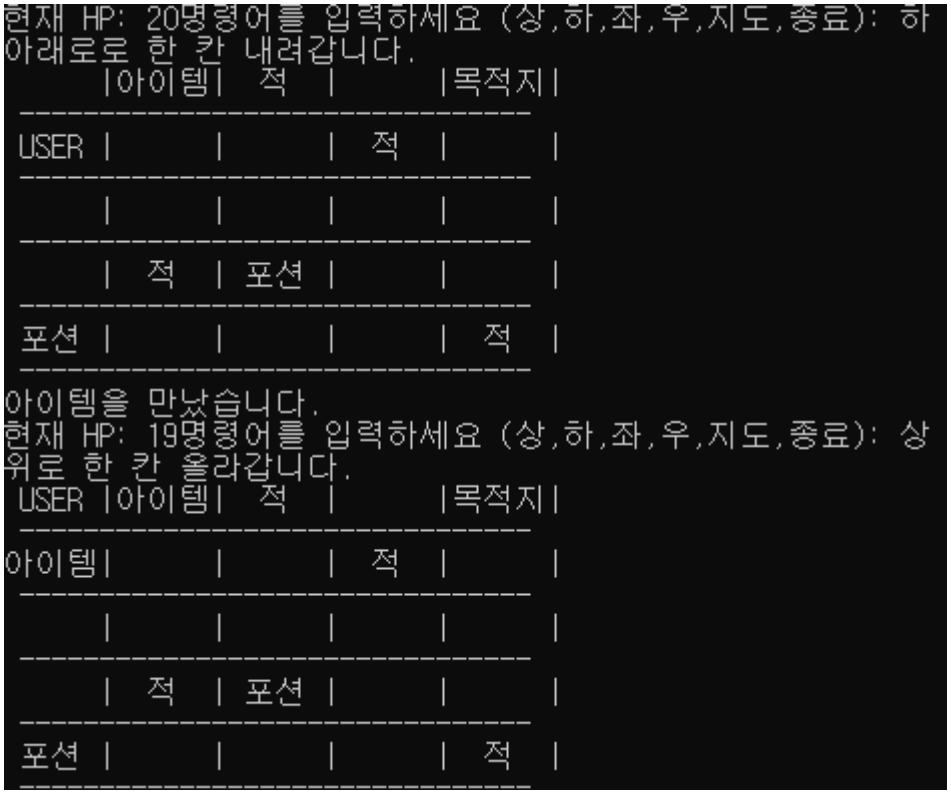
④ 체력의 구현(HP 20 시작)

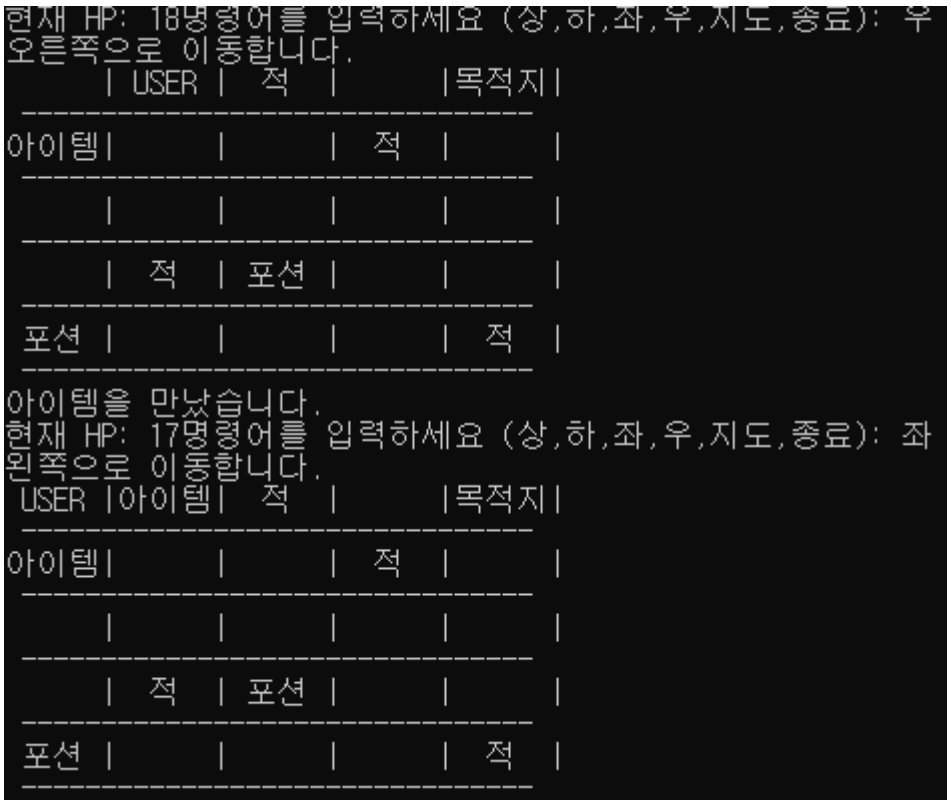


⑤ 체력이 모두 소모하면 게임 종료



⑥ 이동 시 체력 감소





⑦ 처음 명령문 입력 시 HP 출력

```
현재 HP: 4명령어를 입력하세요 (상,하,좌,우,지도,종료):
```

```
현재 HP: 3명령어를 입력하세요 (상,하,좌,우,지도,종료):
```

⑧ 오브젝트와 상호작용 및 메시지 출력

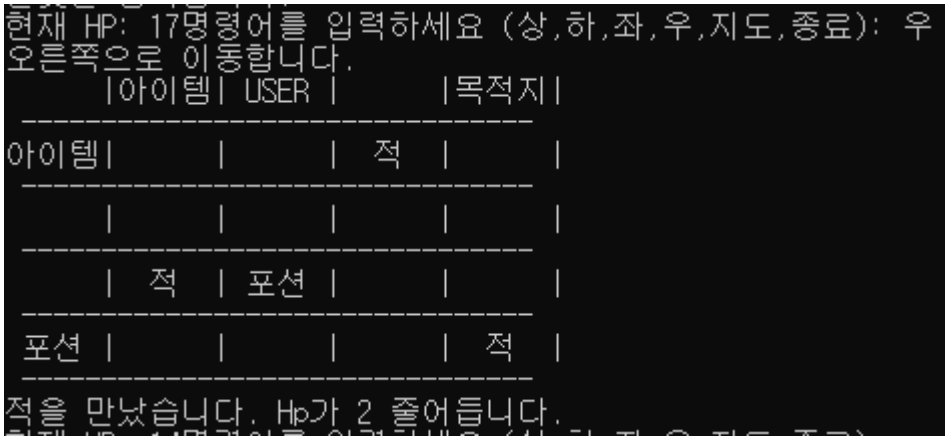
공란

```
현재 HP: 19명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
     |아이템|  적  |      |목적지|
 ------------------------------
아이템| USER |      |  적  |      |
 ------------------------------
     |      |      |      |      |
 ------------------------------
     |  적  | 포션 |      |      |
 ------------------------------
 포션 |      |      |      |  적  |
 ------------------------------
```

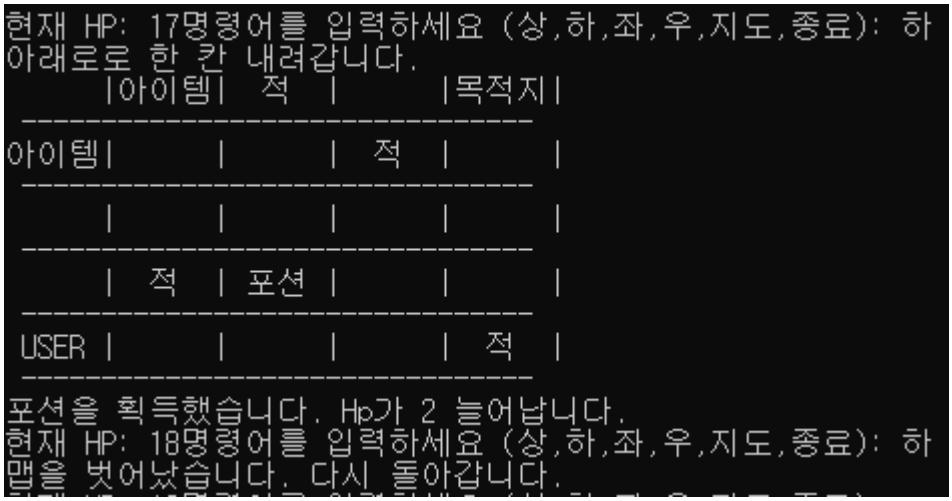
아이템

```
현재 HP: 18명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
위로 한 칸 올라갑니다.
     | USER |  적  |      |목적지|
 ------------------------------
아이템|      |      |  적  |      |
 ------------------------------
     |      |      |      |      |
 ------------------------------
     |  적  | 포션 |      |      |
 ------------------------------
 포션 |      |      |      |  적  |
 ------------------------------
아이템을 만났습니다.
```

적



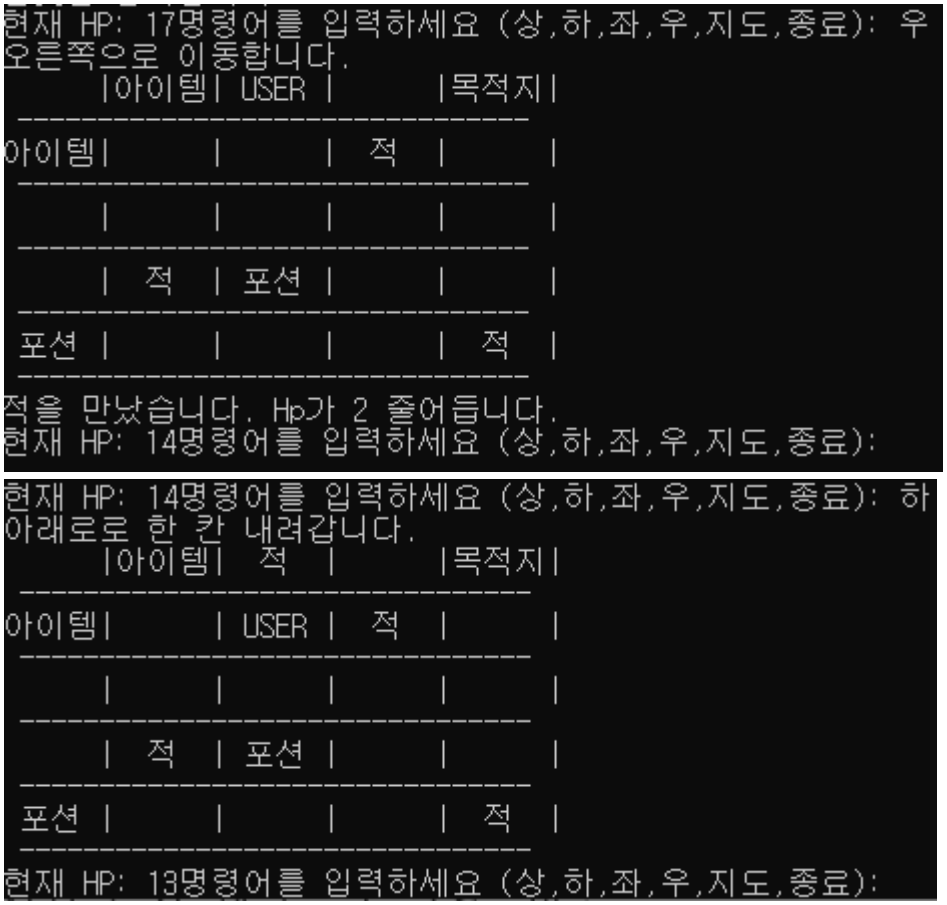
포션



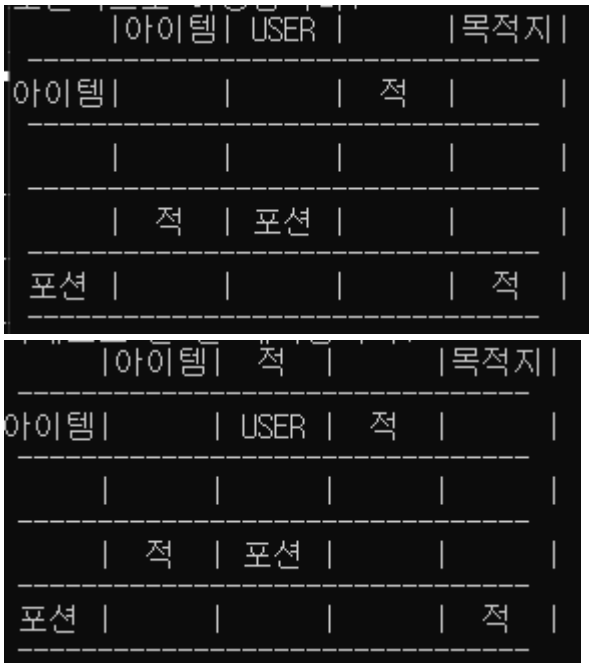
2. 함수 테스트

① 메인함수: 사용자의 입력을 받기 및 게임 진행 동안 무한반복

게임진행간 무한반복

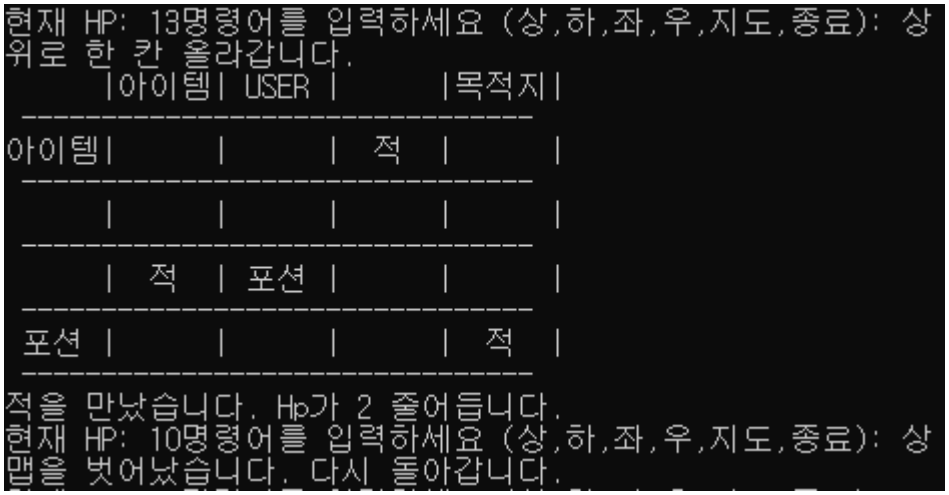


② 지도와 현재 위치 출력 : displayMap()

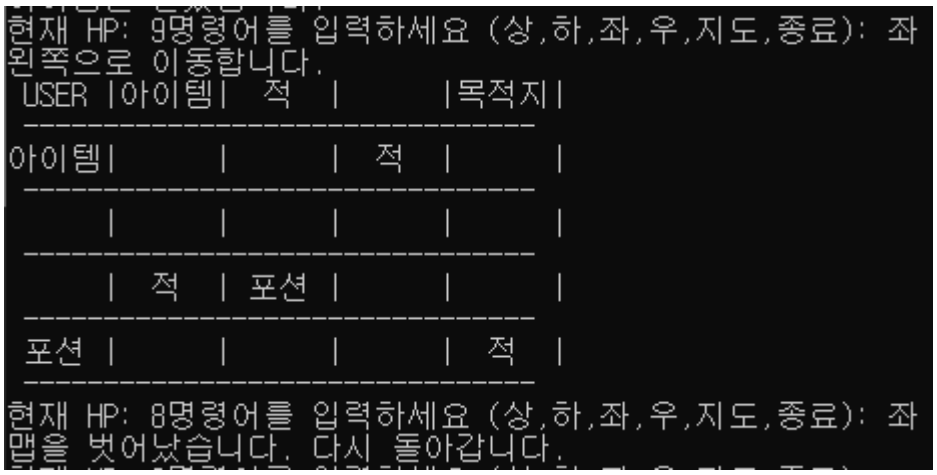


③ 사용자 위치 체크 : checkXY()

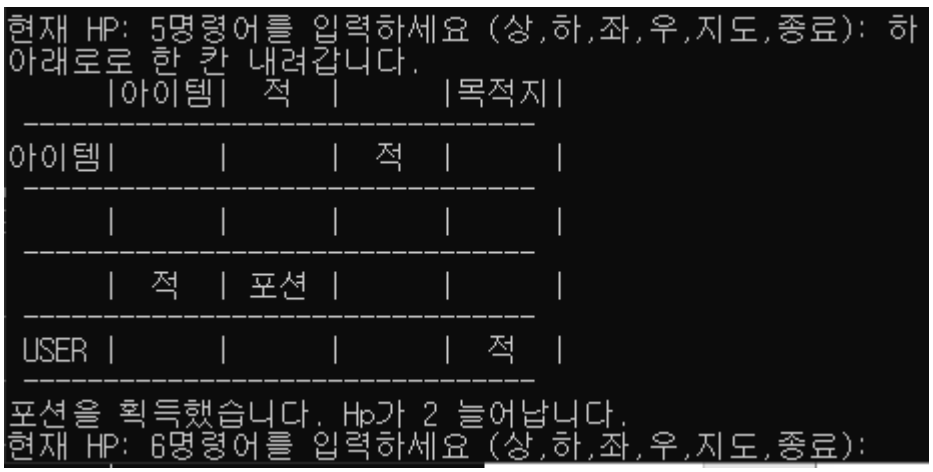
'상'이 유효한 경우, 아닌경우



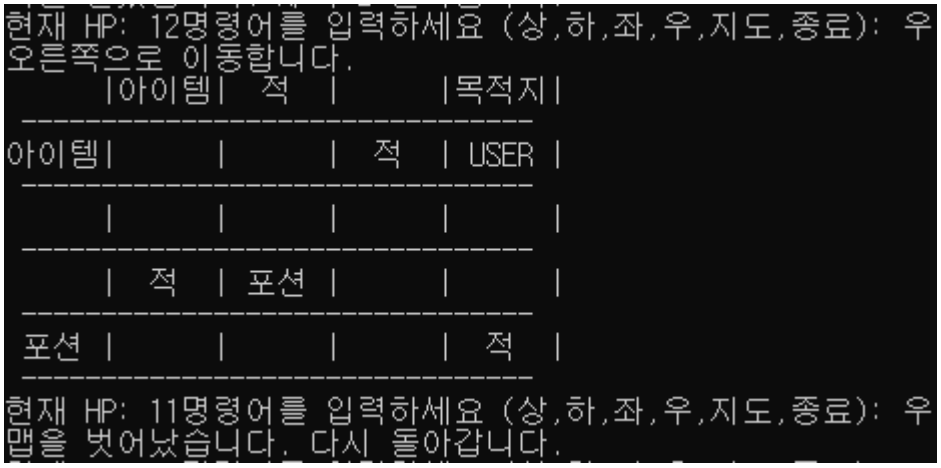
'좌'가 유효한 경우, 아닌경우



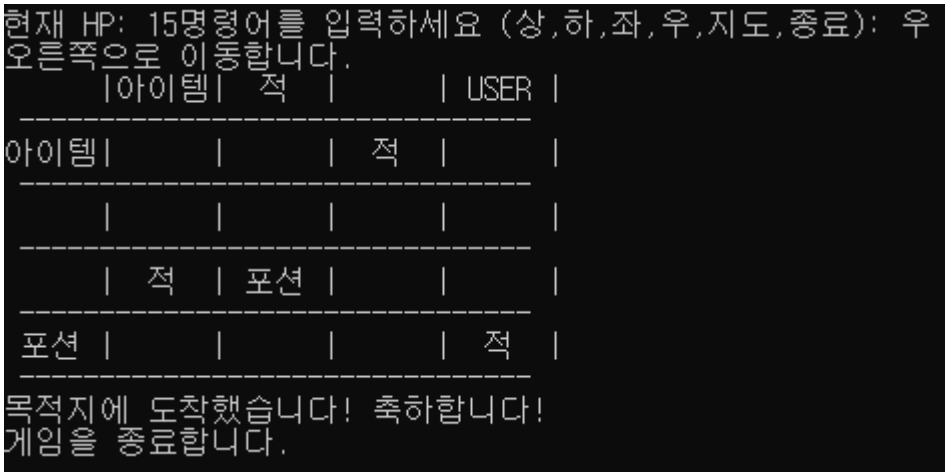
'하'가 유효한 경우, 아닌경우



'우'가 유효한 경우, 아닌경우

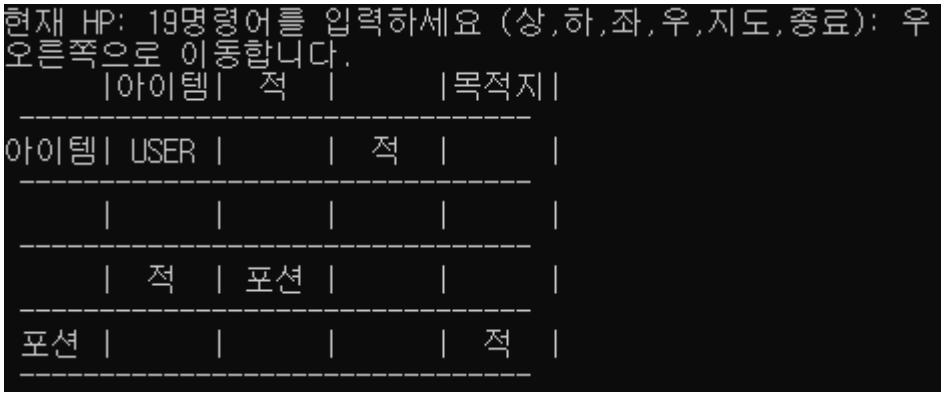


④ 목적지 도착 체크 : checkGoal()

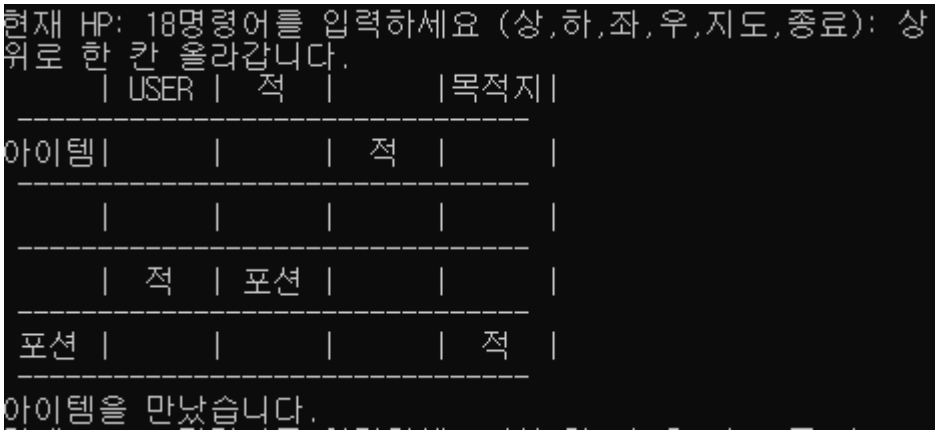


⑤ 오브젝트와 상호 작용 : checkState()

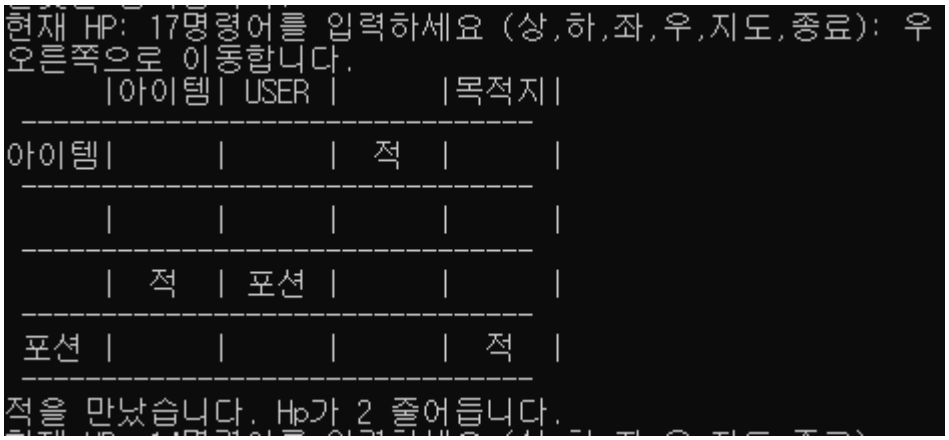
공란


현재 HP: 19명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
|아이템| 적 | |목적지|
아이템| USER | | 적 |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |

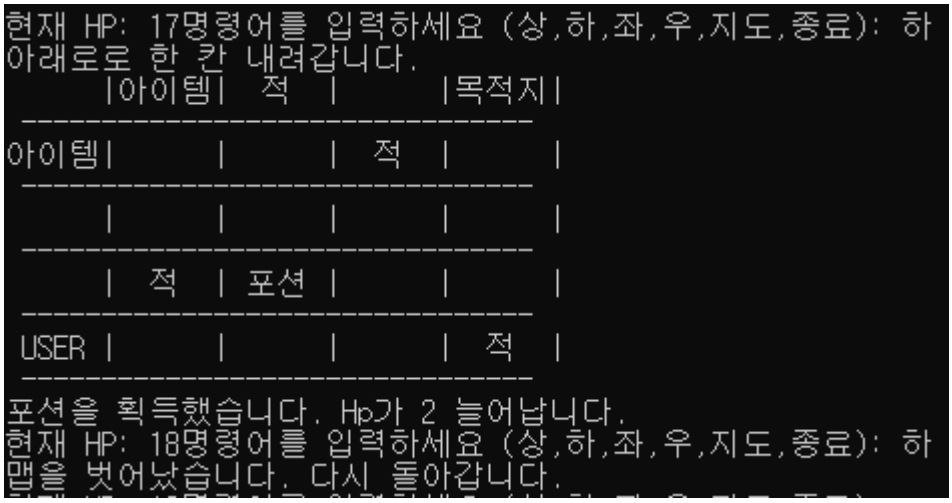
아이템


현재 HP: 18명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
위로 한 칸 올라갑니다.
| USER | 적 | |목적지|
아이템| | | 적 |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
아이템을 만났습니다.

적

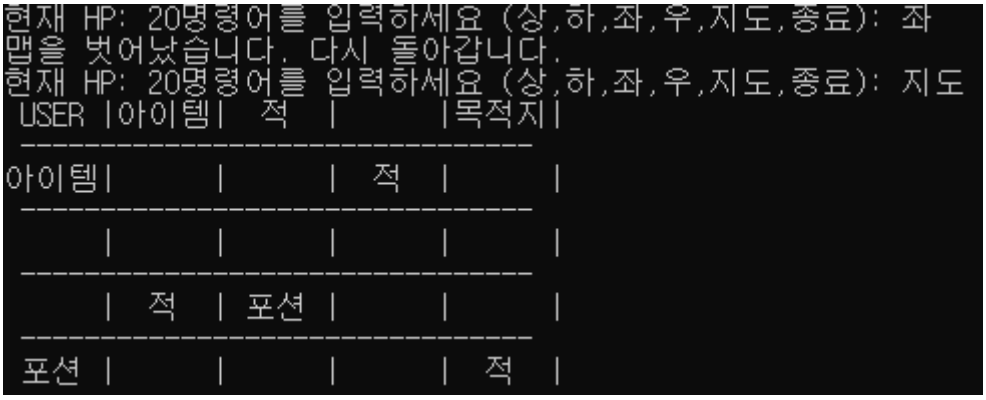

현재 HP: 17명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
|아이템| USER | |목적지|
아이템| | | 적 |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
적을 만났습니다. Hp가 2 줄어듭니다.

포션

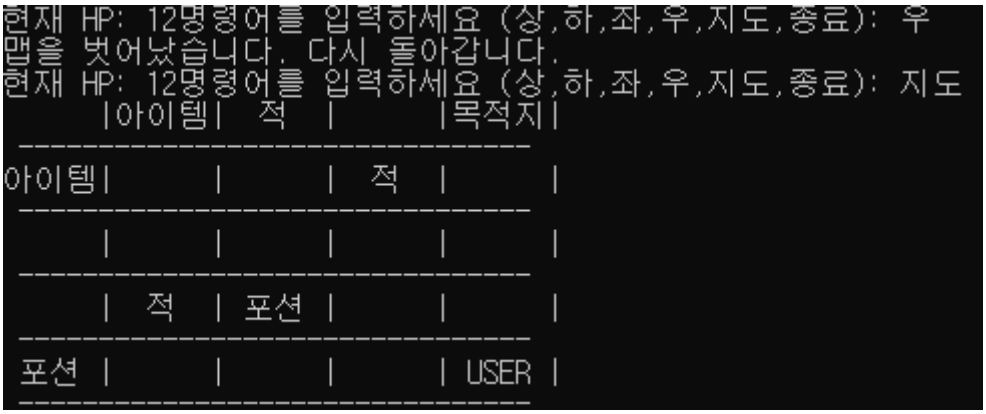


⑦ 유효하지 않은 값 되돌리기 : opti()

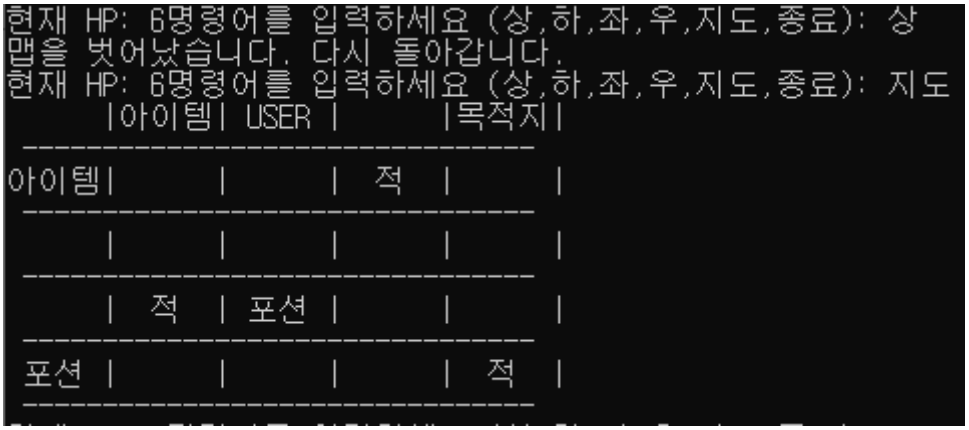
좌로 벗어났을 때



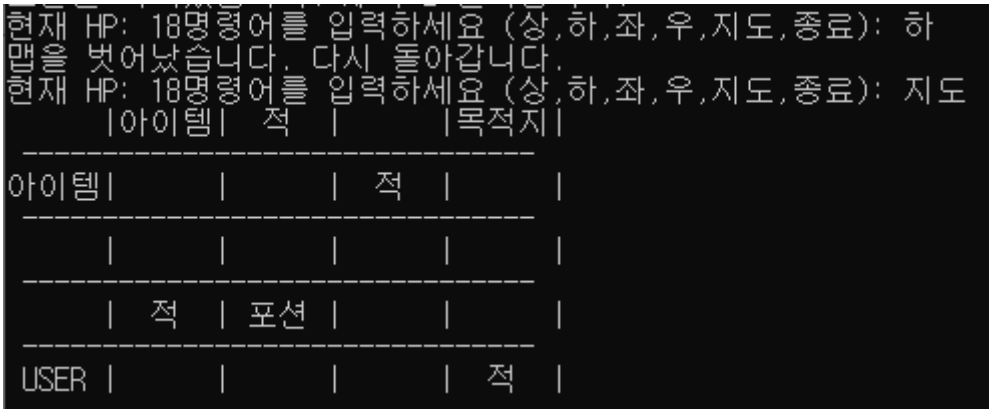
우로 벗어났을 때



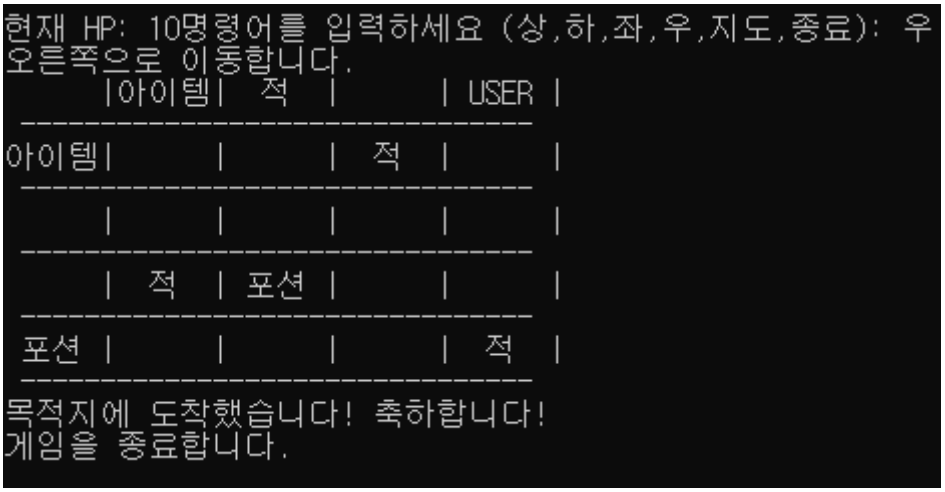
상으로 벗어났을 때



하로 벗어났을 때



3. 최종테스트



5. 결과 및 결론

프로젝트 결과 : 추가 기능이 포함된 mud game 을 드디어 완성하였다. 함수와 배열에 대한 이해가 심화되었다.

느낀점 : 함수에 대해 생각하다가 우리 인생과 비슷한 점이 있다는 것을 알게 되었다. 함수는 저마다의 반환형을 가지고 있다. 이는 자손, 업적, 이야기, 세상에 남긴 물리적 변화(조각상, 고대 벽화, 비석)와 같이 사람이 죽으면서 남기는 무언가로 볼 수 있다. 사람은 함수와 같이 모두 return 즉, 죽음(종료)을 알면서도 살아간다(함수의 실행). 또한, 함수가 선언이 없다면 예기치 못한 statement 를 만나면 오류가 나듯이 사람도 어떤 일이 바로 앞까지 다가오기 전까지는 알지 못한다.

하지만, 선언(미리 준비)을 한다면 어려운 상황에서도 대처할 수 있을 것이다.

이러한 재미있는 상상을 통해 교훈을 얻었다. 인생도 항상 준비(선언)가 되어있어야 기회를 잡을 수 있고, 이미 정해져 있을지도 모르는 반환형(유전자에 각인되어 있는)이 무엇인가를 알아내기 위해 끝까지 살아내는 것이 인생인가 싶다.