# 텍스트로 입력 받는 명령어를 수행하며 즐기는 게임

프로젝트 목적 및 배경: 7주차까지 배운 내용에 대한 실습을 위해 진행

목표: 간단한 Mud게임 구현

사용자 요구사항: 유저가 상하좌우로만 이동하며 목적지에 도착하는 게임

기능 계획:

1. 사용자에게 상, 하, 좌, 우, 지도, 종료 중 하나를 입력받아 입력받은 값으로 지도를 이동시킨다.
2. 지도에는 그 위치마다 아이템, 적, 포션, 등등 이 있음 적을 만나면 포션이 줄고 아이템을 만나면 그대로 이고 포션을 만나면 포션이 늘어난다.
3. 만약 지도의 외부로 간다면 에러메시지를 출력하여 다시 시작한다.
4. 목적지에 도착하면 성공을 출력하고 종료된다.

함수계획:

① 메인 함수: 사용자에게 값을 계속 입력받고, 그에 대한 함수 호출

② 지도와 현재 위치 출력 함수: displayMap()

③ 사용자 위치 체크 함수: checkXY()

④ 목적지에 도착 체크 함수: checkGoal()

사용자가 위치한 장소의 상태를 출력하는 함수: checkState()

# 설계 구현

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

    // 사용자의 입력을 저장할 변수
    string user_input = "";

    cout << "현재 HP: "<<userHP<<"  명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
    cin >> user_input;

    if (user_input == "상") {
        // TODO: 위로 한 칸 올라가기
        user_y -= 1; // 만약 상이면 user_y는 -1이 저장됨
        bool inMap = checkXY(user_x, mapX, user_y, mapY); // 함수에서 반환된 값이 저장됨
        if (inMap == false) {
            cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
            user_y += 1; // 만약 false면 이 문장이 출력되고, user_y에 +1 추가됨
        }
        else {
            cout << "위로 한 칸 올라갑니다." << endl; // true면 이 문장이 출력됨
            displayMap(map, user_x, user_y); // 이 함수 호출
            userHP -= 1; // Hp-1로 계산하여 저장
        }
    }
    else if (user_input == "하") {
        // TODO: 아래로 한 칸 내려가기
        user_y += 1; // user_y에 +1을 함
        bool inMap = checkXY(user_x, mapX, user_y, mapY); // 함수를 호출하여 값을 저장
        if (inMap == false) { // 만약 값이 flase면
            cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
            user_y -= 1;
        }
        else {
            cout << "위로 한 칸 내려갑니다." << endl;
            displayMap(map, user_x, user_y); // 함수를 호출
            userHP -= 1; // hp-1을 하여 저장
        }
    }
    else if (user_input == "좌") {
        // TODO: 왼쪽으로 이동하기
        user_x -= 1;
        bool inMap = checkXY(user_x, mapX, user_y, mapY);

        if (inMap == false) {
            cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
            user_x += 1;
        }
        else {
            cout << "왼쪽으로 이동합니다." << endl;
            displayMap(map, user_x, user_y);
            userHP -= 1;
        }
    }
    else if (user_input == "우") {
        // TODO: 오른쪽으로 이동하기
        user_x += 1;
        bool inMap = checkXY(user_x, mapX, user_y, mapY);
        if (inMap == false) {
            cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
            user_x -= 1;
        }
```

입력(블록/함수에 입력되는 변수, 값들과 설명)

user_input : 명령어를 입력받아 저장할 변수(상, 하, 좌, 우, 지도, 종료)

user_x : 유저의 위치를 저장할 변수 (가로)

user_y : 유저의 위치를 저장할 변수 (세로)

inMap : 맵의 위치가 정확한지 체크하기 위한 변수

userHP : 유저의 HP를 저장하기 위한 변수

반환값 : 없다.

결과 : (블록/ 함수가 종료된 결과)

출력: "현재 HP: userHP 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): "

1. 상

상을 입력하면 "위로 한칸 올라갑니다." 출력된다.

그리고 user_y에 +1을 해서 값을 저장한다.

만약 맵이 벗어난 위치에 있다면 "맵을 벗어났습니다. 다시 돌아갑니다." 출력되고 다시

"현재 HP: userHP 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 가 출력되서 입력받음

➔ 만약 하, 좌, 우 이런 값이 입력될 경우 이런 식의 알고리즘으로 출력된다.

설명(코드 내 작동 순서, 내용 등 추가 설명)

함수를 호출하여 변수의 값이 저장된다.

함수를 호출하여 출력한다.

UserHP는 계속 값이 더하거나 빼져서 초기화된다.

```
    }
    else {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        userHP -= 1;
    }
}
else if (user_input == "지도") {
    // TODO: 지도 보여주기 함수 호출
    displayMap(map, user_x, user_y);
}
else if (user_input == "종료") {
    cout << "종료합니다.";
    break;
}
else { // 이외에 다른 문자를 입력시
    cout << "잘못된 입력입니다." << endl;
    continue; // while문이 다시 반복됨
}

// 사용자의 위치 정보 체크
checkState(map, user_x, user_y); // 이 함수를 호출

// 목적지에 도달했는지 체크
bool finish = checkGoal(map, user_x, user_y); // 함수를 호출하여 저장
if (finish == true) { // ture면
    cout << "목적지에 도착했습니다! 축하합니다!" << endl;
    cout << "게임을 종료합니다." << endl;
    break; // 위 문장이 출력되고 while문이 1이되어 빠져나감
}

// 사용자의 HP 체크
if (userHP <= 0) {
    cout << "HP가 0 이하가 되었습니다. 실패했습니다. " << endl;
    cout << "게임을 종료합니다." << endl;
    break; // 위 문장이 출력되고 while문 빠져나감
}
```

입력:

user_input : 사용자에게 받은 값을 저장하는 변수

bool finish : 목적지에 도달했는지 체크하는 변수

userHP : 사용자의 HP 값 저장 및 초기화 변수

반환값: 없음

결과 :

지도가 입력되면 지도보여주기 함수를 호출하여 지도가 출력된다.

종료가 입력되면 "종료합니다" 출력된다. 그리고 코드가 종료된다.

이외에 다른 문자 입력시 "잘못된 입력입니다." 출력되고 다시 처음으로

while문이 반복된다.

사용자의 위치 정보를 체크하기 위한 함수를 호출해 값을 출력한다.

목적지에 도달했는지 체크하는 함수를 호출하여 변수에 저장한뒤

값이 ture면 위 문장이 출력됨과 동시에 코드가 종료된다.

사용자의 HP 체크를 하여 만약 HP가 0보다 작거나 같으면 위 문장이 출력되고 코드가 종료된다.

```
// 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지
// 지도와 사용자 위치 출력하는 함수
void displayMap(int map[][mapX], int user_x, int user_y) {
    for (int i = 0; i < mapY; i++) {
        for (int j = 0; j < mapX; j++) {
            if (i == user_y && j == user_x) {
                cout << " USER |"; // 양 옆 1칸 공백 // user y와 userx의 값이 같다면 이 문장출력

            }
            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;
    }
}
```

입력: map[][mapX] : 전체지도

User_x : 유저의 좌표의 값을 저장하는 변수 (가로)

User_y : 유저의 좌표의 값을 저장하는 변수 (세로)

pasState : for문을 돈 map[i][j]의 값을 저장

반환: 없음 void형식이다.

결과:

1. for문을 돌면서 처음에는 i가 0으로 초기화되고 내부for문에서 j는 0으로 초기화 된 뒤 만약 i와 j값이 user_y의 값과 user_x의 값이 같으면 위 문장이 출력된다.
2. 그렇지 않으면 else문을 통해 swith문에서 0,1,2,3,4 인 곳에서 그 곳에 맞는 값이면 위 문장 출력되고 다시 내부 for문으로 돌아가 j의 값이 +1로 더해지고 다시 조건에 맞게 출력되고 내부 for문을 다 돌면 줄바꿈뒤 ------ 출력된다.
3. 다시 외부 for문 에서 i가 +1로 증가하며 그렇게 반복되면서 지도가 완성된다.

설명: switch문에서 break를 사용하여 해당값의 알맞은 문장이 출력되면 switch문이 멈추고 빠져나간다.

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int user_x, int mapX, int user_y, int mapY) {
    bool checkFlag = false; // checkFlag flase로 초기화
    if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) {
        checkFlag = true; // 만약 이 조건이 충족될시 true 저장됨
    }
    return checkFlag; // 저장된 값이 반환됨
}
```

입력 : user_x : 유저의 위치를 표현하는 값 (가로)

User_y: 유저의 위치를 표현하는 값의 변수 (세로)

MapX : 5로 선언된 지도에 포함되는 변수

MapY : 5로 선언된 지도에 포함되는 변수

반환 : 변수 checkFlag에 저장된 값이 메인함수에 반환된다.

결과 : 만약 user_x의 값이 0보다 같거나 크고 user_x의값이 mapX의 값보다 더 작고,

동시에 User_y의 값이 0보다 같거나 크고 user_y의 값이 mapY보다 작으면

checkFlaf값이 true로 저장된다.

그렇지 않으면 false로 저장된다.

저장된 값을 반환 시킨다.

```
// 유저의 위치가 목적지인지 체크하는 함수
bool checkGoal(int map[][mapX], int user_x, int user_y) {
    // 목적지 도착하면
    if (map[user_y][user_x] == 4) {
        return true; // 만약 이 조건이 충족시 true 반환
    }
    return false; // 만약 이 조건이 불 충족시 false 반환
}
```

입력:

Map[][mapX]: 지도에 어떤 값이 저장된 상태를 나타내는 변수

User_x: 사용자의 위치

User_y: 사용자의 위치

반환: 매개변수를 통해 false또는 ture의 값을 반환한다.

결과: map[user_y][user_x]의 값이 4의 위치에 있다면 true의 값이 저장되고 이 값이 반환된다.

만약 아닐시 false의 값이 반환된다.

```cpp
// 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지
// 사용자가 위치한 장소의 상태를 출력하는 함수
void checkState(int map[][mapX], int user_x, int user_y) {

    switch (map[user_y][user_x]) {
    case 1:
        cout << "아이템이 있습니다" << endl;
        break; // 1에 있으면 위 문장 출력 하고 멈춤
    case 2:
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
        userHP -= 2; // 2에 있으면 위 문장 출력하고 hp-2 로 계산되고 저장
        break;
    case 3:
        cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;
        // 3에 있으면 위 문장 출력하고 hP +2로 계산됨
        userHP += 2;
        break;
    }
}
```

입력: map[][mapX] : 지도의 상태

User_x : 사용자의 위치 (가로)

User_y : 사용자의 위치 (세로)

Map[user_y][user_x] : 사용자의 위치를 지도의 위치

반환:

Void 반환되지 않는다.

결과:

만약 사용자가 1의 위치에 있다면 위문장 출력한고 멈추고 switch 문을나간다.

2의 위치 : 위 문장 출력한다. 그리고 멈추고 switch 문을나간다.

3의 위치 : 위 문장을 출력한다. 그리고 멈추고 switch 문을나간다.

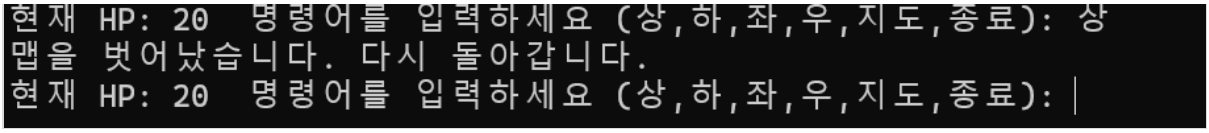
설명:

2의 위치에 있을시 userHP가 -2로 감소하고 저장된다.

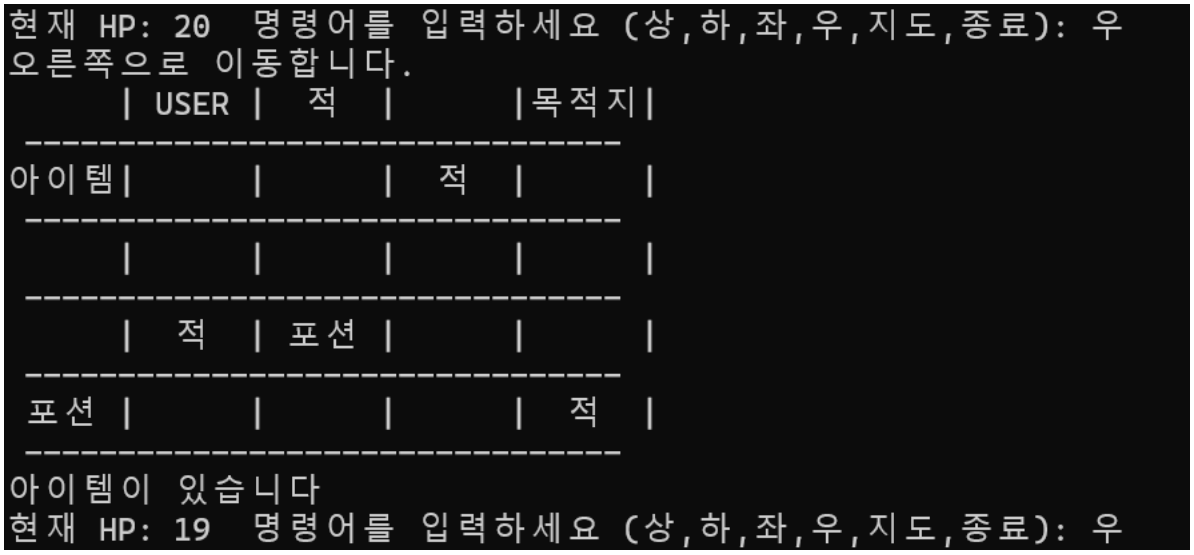
3의 위치에 있을시 userHP가 +2로 늘어나고 저장된다.

# 기능별 테스트 결과

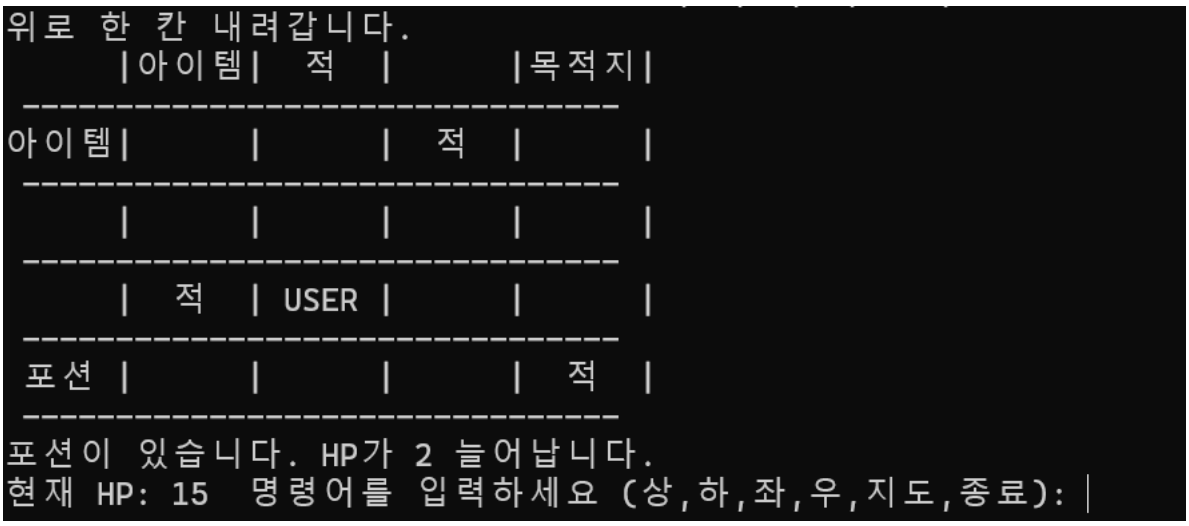
맵을 벗어나면 위 문장 출력된뒤 다시 실행된다.



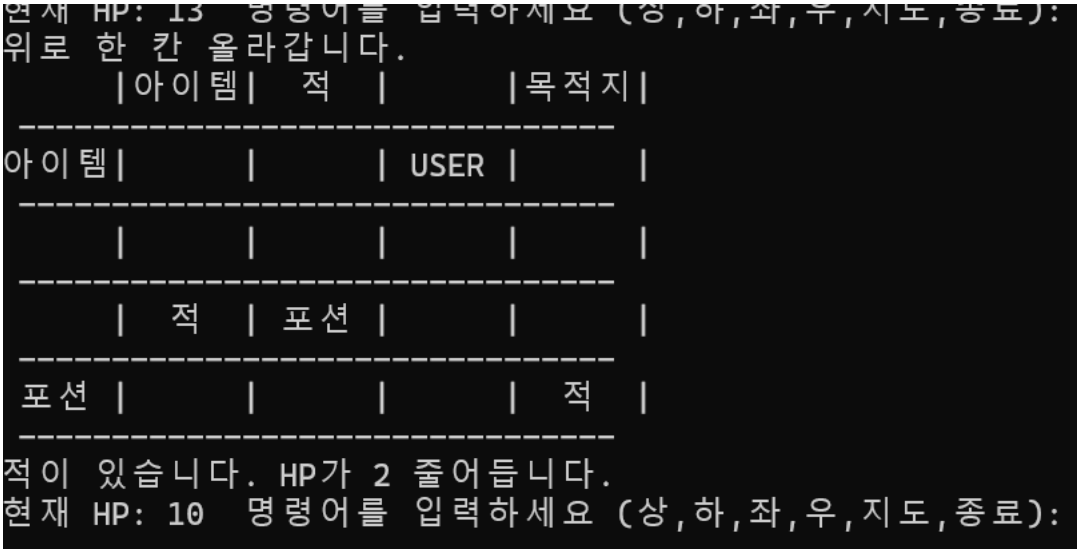

지도에 아이템이 있는 곳으로 가면 아이템이 있습니다가 출력되고 현재 HP까지 출력된다.



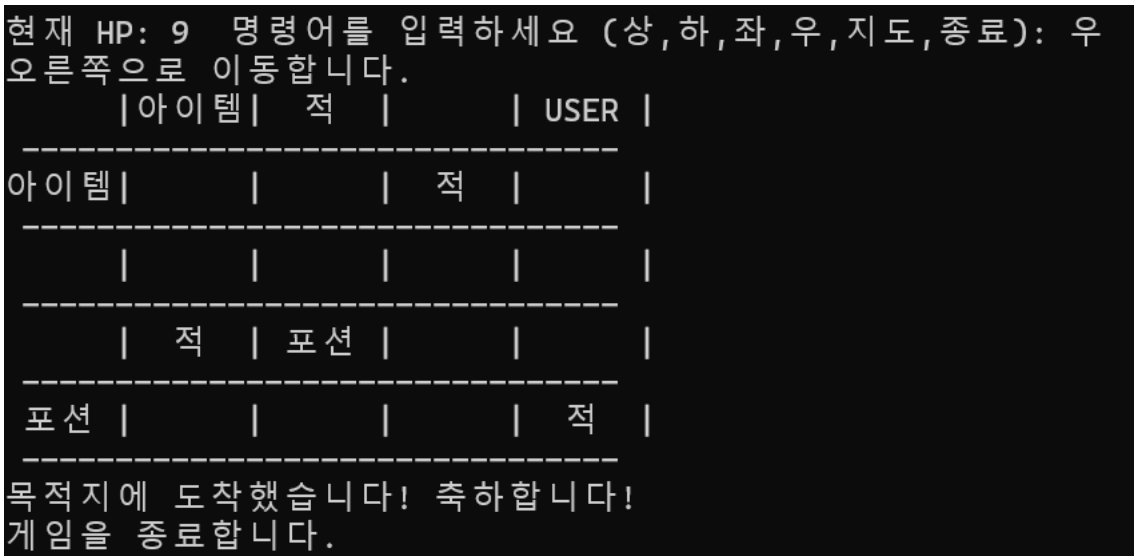

포션이 있는 곳에 도착시 HP가 +2 늘어난다. 그래서 HP가 값이 변경되어 초기화된다.



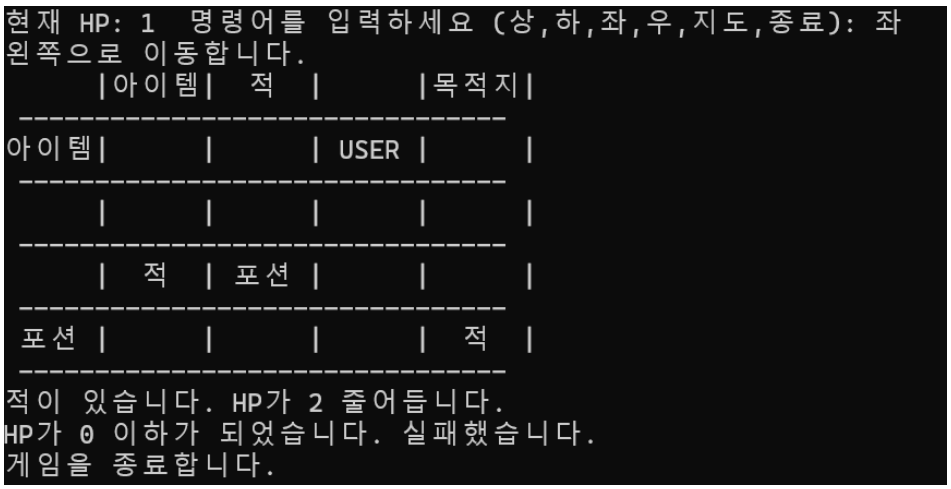
적을 만나면 HP가 -2로 줄어든다.




목적지에 도착하면 게임을 종료된다. (최종 테스트)



HP가 0이하가 되면 종료된다.




## 프로젝트 결과 및 결론

메인 함수와 함수를 이용해 간단하게 즐기는 게임을 만들었다.

더 복잡하게 코드를 써서 더 재밌게 실행시킬 수 있는 게임을 만드는 게 가능 할 수는 있지만 아직 능력이 부족해서 간단하게 실행시키는 게임을 만들었다.

조금 더 코딩 실력이 는다면 복잡하게 만들어서 여러가지 기능들을 추가하고 싶다.