Mud Game Project

182671 김준석

1. 서론

A. 프로젝트 목적 및 배경

: 7주차까지 배운 내용에 대한 실습을 위해 진행(함수, 배열 등)

B. 목표

: 간단한 Mud 게임 구현

2. 요구사항

A. 사용자 요구사항

: 유저가 체력을 고려하며 상하좌우로 이동해 목적지에 도착하는 게임

B. 기능 계획

i. 사용자에게 현재 HP를 출력하고 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기

ii. 사용자의 입력에 따라 정해놓은 조작 실시
- “상/하/좌/우” 입력시 ①해당 방향으로 이동 후 체력 1감소 ②전체 지도 출력 ③이동한 좌표에서 아이템,포션,적을 만났을 때 그에 대한 메시지 출력 및 HP증감
- “지도”를 입력시 전체 지도와 함께 현재 위치를 출력
- “종료”를 입력시 게임을 즉시 종료
- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재입력 요청

iii. 지도 밖으로 벗어나게 되면 에러 메시지 출력

iv. 목적지에 도착하면 “성공”을 출력하고 종료

v. 유저는 체력 20을 가지고 게임을 시작

vi. HP가 0이 되면 “실패”를 출력하고 종료

C. 함수 계획

i. 메인 함수 : 지도의 요소들을 채워놓고 유저의 초기 좌표를 설정. 사용자에게 값을 계속 입력받고, 그에 대한 함수 호출

ii. displayMap : 유저의 위치와 함께 지도의 내용을 출력하는 함수

iii. checkXY : 이동하려는 좌표가 유효한 좌표인지 체크하는 함수

iv. checkGoal : 현재 유저가 목적지에 도달했는지 체크하는 함수

v. moveUser : 유저가 이동시 체력을 감소시키는 함수

vi. showHealth : 현재 유저의 체력을 출력하는 함수

vii. isOver : 유저의 HP가 0이 되어 게임에 실패했는지 체크하는 함수

viii. checkState : 현재 좌표에서의 상태를 체크하는 함수(아이템/포션/적)

3. 설계 및 구현

*기능단위로 우선 서술하고 기능의 설명란에서 함수에 대해 서술

A. 사용자에게 현재 HP를 출력하고 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기

i.

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

// showHealth 함수를 호출해 현재 HP를 출력
showHealth();

cout << "명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): ";
cin >> user_input; // 사용자 입력을 저장
```

ii. 입력

1. user_input : 유저가 입력한 명령어(상, 하, 좌, 우, 지도, 종료)

iii. 결과

1. 사용자에게 현재 HP를 보여준다.
2. 사용자에게 명령어 입력을 안내한다.
3. 사용자에게 명령어를 입력 받고 이를 저장한다.

iv. 설명

1. 입력한 명령어를 저장할 변수 user_input 선언
2. showHealth 함수를 호출해 현재 HP를 출력
3. 명령어를 입력하라는 문자열을 출력하고 입력값을 변수 user_input에 저장
4. showHealth : 사용자의 현재 HP를 출력하는 함수

A.

```
// 사용자의 HP를 출력하는 함수
void showHealth() {

    // 전역변수 userHealth 출력
    cout << "현재 HP: " << userHealth << "  ";
}
```

B. 입력

i. 없음

C. 반환값

i. 없음

D. 결과

i. 사용자의 현재 HP를 출력

E. 설명

i. 전역변수 userHealth에 접근해 문자열과 함께 출력

B. 사용자의 입력에 따라 정해놓은 조작 실시

- "상/하/좌/우" 입력시 ①해당 방향으로 이동 후 체력 1감소 ②전체 지도 출력
  ③이동한 좌표에서 아이템,포션,적을 만났을 때 그에 대한 메시지 출력 및 HP증감
- "지도"를 입력시 전체 지도와 함께 현재 위치를 출력
- "종료"를 입력시 게임을 즉시 종료
- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재입력 요청

```
// 입력이 "상"
if (user_input == "상") {
    user_y--; // 사용자의 y좌표를 1감소시켜 위로 이동
    if (checkXY(user_x, mapX, user_y, mapY)) {
        moveUser(); // 이동했으므로 체력 1 감소
        cout << "위로 한 칸 올라갑니다." << endl;
        // 지도 출력
        displayMap(map, user_x, user_y);
        // 현재 좌표 상태 확인
        checkState(map, user_x, user_y);
    }
```

```
// 입력이 "하"
else if (user_input == "하") {
    user_y++; // 사용자의 y좌표를 1증가시켜 아래로 이동
    if (checkXY(user_x, mapX, user_y, mapY)) {
        moveUser(); // 이동했으므로 체력 1 감소
        cout << "아래로 한 칸 올라갑니다." << endl;
        // 지도 출력
        displayMap(map, user_x, user_y);
        // 현재 좌표 상태 확인
        checkState(map, user_x, user_y);
    }
```

```
// 입력이 "좌"
else if (user_input == "좌") {
    user_x--; // 사용자의 x좌표를 1감소시켜 좌로 이동
    if (checkXY(user_x, mapX, user_y, mapY)) {
        moveUser(); // 이동했으므로 체력 1 감소
        cout << "왼쪽으로 이동합니다." << endl;
        // 지도 출력
        displayMap(map, user_x, user_y);
        // 현재 좌표 상태 확인
        checkState(map, user_x, user_y);
    }
```

```
// 입력이 "우"
else if(user_input == "우") {
    user_x++; // 사용자의 x좌표를 1증가시켜 우로 이동
    if (checkXY(user_x, mapX, user_y, mapY)) {
        moveUser(); // 이동했으므로 체력 1 감소
        cout << "오른쪽으로 이동합니다." << endl;
        // 지도 출력
        displayMap(map, user_x, user_y);
        // 현재 좌표 상태 확인
        checkState(map, user_x, user_y);
    }
```

```
// 입력이 "지도"
else if (user_input == "지도") {
    // 지도를 보여주는 displayMap 함수 호출
    displayMap(map, user_x, user_y);
}
// 입력이 "종료"
else if (user_input == "종료") {
    cout << "종료합니다.";
    break; // 무한 loop 종료
}
// 그 외에 다른 잘못된 입력
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

i. **입력**

1. **user_input : 유저가 입력한 명령어를 저장한 변수**
2. **map : 지도의 각 좌표의 요소들을 나타내는 2차원 배열**

3. user_x, user_y : 현재 유저가 위치한 x좌표, y좌표를 저장한 변수

ii. 결과

1. 입력이 “상/하/좌/우”인 경우

   A. 지도를 벗어나지 않는다면 유저의 좌표 증감시켜 이동

   B. 이동한 후에는 유저의 HP 1감소

   C. 이동을 안내하는 문자열 출력

   D. 전체 지도를 출력

   E. 현재 좌표에 아이템, 포션, 적이 있는지 확인 후 문자열 출력

   F. 포션이 있다면 HP 2증가, 적이 있다면 HP 2감소

2. 입력이 “지도”인 경우

   A. 전체 지도를 출력

3. 입력이 “종료”인 경우

   A. 게임을 즉시 종료

4. 그 외에 다른 입력인 경우

   A. 잘못된 입력임을 안내 후 다시 입력받기

iii. 설명

1. 입력이 “상/하/좌/우”인 경우

   A. user_x를 1증가시켜 오른쪽으로 이동 / 1감소시켜 왼쪽으로 이동
   user_y를 1증가시켜 아래로 이동 / 1감소시켜 위로 이동

   B. checkXY 함수를 호출해 증감시킨 좌표가 지도를 벗어나는지 확인

   C. 지도를 벗어나지 않으면 moveUser 함수를 호출해 HP를 1감소

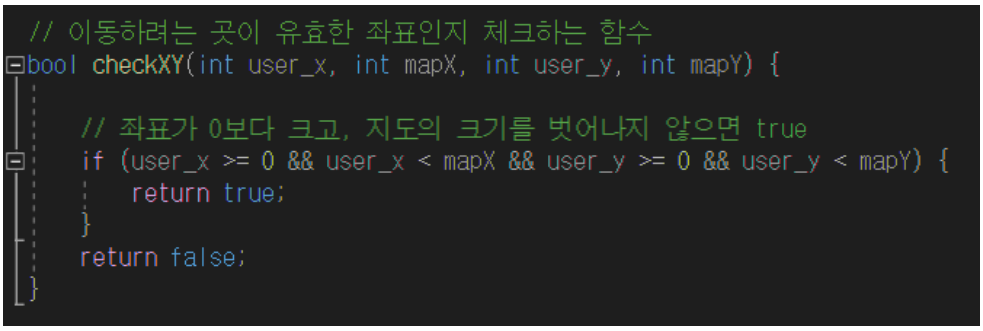
   D. 이동한 방향을 안내하는 문자열 출력

   E. displayMap 함수를 호출해 전체 지도를 출력

   F. checkState 함수를 호출해 현재 유저의 좌표에 아이템, 포션, 적이 있는지 확인 후 그에 따른 문자열 출력 및 HP 증감

   * 아래부터는 호출한 함수에 대한 설명

G. checkXY : 유저가 이동하려는 좌표가 지도를 벗어나는지 체크하는 함수

i.


ii. 입력

1. user_x, user_y : 유저가 이동하려는 곳의 x, y 좌표
2. mapX, mapY : 지도의 가로/세로 크기

iii. 반환값

1. 지도를 벗어나는지의 여부 boolean값(true/false)

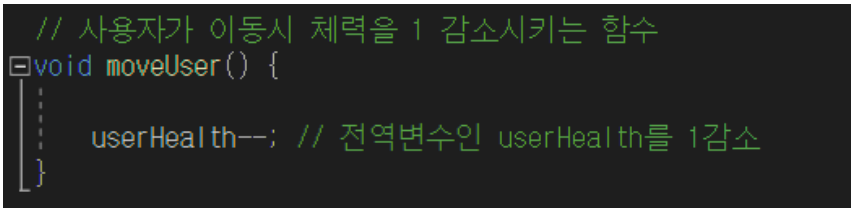
iv. 결과

1. 이동하려는 곳이 유효하다면 true 반환
2. 이동하려는 곳이 유효하지 않다면 false 반환

v. 설명

1. 유저가 이동하려는 곳의 좌표가 양수여야 유효하다.
2. 유저가 이동하려는 곳의 x좌표가 지도의 가로 크기보다 작아야 유효하다. (좌표는 0부터 시작하므로)
3. 유저가 이동하려는 곳의 y좌표가 지도의 세로 크기보다 작아야 유효하다. (좌표는 0부터 시작하므로)

H. moveUser : 이동한 유저의 HP를 1감소시키는 함수

i.


ii. 입력

1. 없음

iii. 반환값

1. 없음

iv. 결과

1. 이동에 따른 유저의 HP 1감소

v. 설명

1. 사용자가 이동했으므로 전역변수 userHealth에 접근해 1감소
2. 원래는 user_input, user_x, user_y를 파라미터로 하고, 좌표 연산 및 문자열 출력을 이 함수에서 모두 수행하려고 했으나 우선 생략

I. displayMap : 사용자의 위치를 포함한 전체 지도를 출력하는 함수

i.

```
// 지도와 사용자의 위치를 출력하는 함수
void displayMap(int map[][mapX], int user_x, int user_y) {
    // 지도 전체를 순회
    for (int i = 0; i < mapY; i++) {
        for (int j = 0; j < mapX; j++) {
            // 사용자의 위치인 좌표이면 USER 출력
            if (i == user_y && j == user_x) {
                cout << " USER |";
            }
            // 해당 요소의 값에 따라 문자열 출력
            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;
                }
            }
        }
        // 줄 바꾸고 구분선 출력
        cout << endl;
        cout << " ---------------------------- " << endl;
    }
}
```

ii. 입력

1. map : 지도의 좌표별 요소들을 저장한 2차원 배열
2. user_x, user_y : 현재 유저가 위치한 x, y좌표

iii. 반환값

1. 없음

iv. 결과

1. 좌표별 요소들을 포함한 전체 지도를 출력
2. 사용자의 위치 또한 지도에 출력(사용자의 위치에 요소가 있더라도 USER로 덮음)

v. 설명

1. 2차원 배열인 map을 2중 for문으로 순회
2. if문을 사용해 현재 유저의 좌표와 일치하면 USER 출력
3. 유저의 위치가 아닌 경우 map의 해당 인덱스의 요소(int값)를 posState라는 변수에 저장
4. posState의 값에 따라 switch문을 통해 공백, 아이템, 적, 포션, 목적지를 출력

J. checkState : 현재 좌표에서의 상태를 체크하는 함수(아이템/포션/적)

i.

```
// 현재 좌표에서의 상태를 체크하는 함수(아이템, 포션, 적)
void checkState(int map[][mapX], int user_x, int user_y) {

    int posState = map[user_y][user_x]; // 현재 유저가 위치한 좌표의 요소 값을 저장
    switch (posState) {
    case 1: // 아이템이 있는 경우
        cout << "아이템이 있습니다." << endl;
        break;
    case 2: // 적이 있는 경우
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
        userHealth -= 2; // HP 2 감소
        break;
    case 3: // 포션이 있는 경우
        cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;
        userHealth += 2; // HP 2 증가
        break;
    }
}
```

ii. 입력

1. map : 지도의 좌표별 요소들을 저장한 2차원 배열
2. user_x, user_y : 현재 유저가 위치한 x, y좌표

iii. 반환값

1. 없음

iv. 설명

1. 유저가 위치한 곳에 요소가 있다면 그것을 안내
2. 포션이나 적이 있다면 HP를 증감 연산

v. 결과

1. 유저가 위치한 좌표에 있는 요소(int값)를 posState 변수에 저장
2. posState의 값에 따라 switch문을 통해 문자열 출력 및 전역변수 userHealth에 접근해 HP 증감 연산
3. 적(2)이 있다면 userHealth 2감소
4. 포션(3)이 있다면 userHealth 2증가

2. 입력이 "지도"인 경우

A. displayMap 함수를 호출해 전체 지도를 출력

3. 입력이 "종료"인 경우

A. 종료를 안내하는 문자열 출력

B. break로 while loop를 탈출해 게임 즉시 종료

4. 그 외에 다른 입력인 경우

A. 잘못된 입력을 안내하는 문자열 출력

B. continue로 while loop의 초기로 이동(다시 명령어 입력받기)

C. 지도 밖으로 벗어나게 되면 에러 메시지 출력

i.

```
// 지도를 벗어나면 다시 복귀
else {
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    user_y++;
}
```

```
// 지도를 벗어나면 다시 복귀
else {
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    user_y--;
}
```

```
// 지도를 벗어나면 다시 복귀
else {
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    user_x++;
}
```

```
// 지도를 벗어나면 다시 복귀
else {
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    user_x--;
}
```

ii. 입력

1. checkXY 함수의 반환값 : 유저가 이동하려는 좌표가 유효한지에 대한 boolean값(false)

iii. 결과

1. 맵을 벗어났다는 에러 메시지를 출력
2. 이동한 좌표를 다시 복구

iv. 설명

1. checkXY 함수의 파라미터로 주어진 유저가 이동하려는 x 또는 y좌표값이 유효하지 않으므로 이를 원래대로 복구시킨다.

D. 목적지에 도착하면 “성공”을 출력하고 종료

i.
```
// 목적지에 도달했는지 체크
if (checkGoal(map, user_x, user_y)) {
    // 문자열을 출력하고 loop 종료
    cout << "목적지에 도착했습니다! 축하합니다!" << endl;
    cout << "게임을 종료합니다." << endl;
    break;
}
```

ii. 입력

1. map : 지도의 좌표별 요소들을 저장한 2차원 배열
2. user_x, user_y : 현재 유저가 위치한 x, y좌표

iii. 결과

1. 목적지에 도달했다면 이를 안내하는 메시지 출력
2. 게임을 종료를 안내하는 메시지 출력
3. 게임을 종료

iv. 설명

1. checkGoal 함수를 호출해 현재 유저의 좌표가 목적지에 도달했는지에 대한 boolean값 반환
2. 도달했다면 문자열을 출력하고 break로 while loop를 탈출해 게임을 종료
3. checkGoal : 현재 유저가 목적지에 도달했는지 체크하는 함수

A.

```
// 유저의 위치가 목적지인지 체크하는 함수
bool checkGoal(int map[][mapX], int user_x, int user_y) {

    // 목적지인 (4, 4)에 도달했다면 true
    if (map[user_y][user_x] == 4) {
        return true;
    }
    return false;
}
```

B. 입력

i. map : 지도의 좌표별 요소들을 저장한 2차원 배열

ii. user_x, user_y : 현재 유저가 위치한 x, y좌표

C. 반환값

i. 목적지에 도달했는지에 대한 boolean값

D. 결과

i. 입력받은 사용자의 좌표가 목적지라면 true 반환

ii. 아니라면 false 반환

E. 설명

i. 사용자의 현재 위치의 좌표를 인덱스 값으로 map의 요소에 접근

ii. 해당 인덱스의 요소값이 4 즉, 목적지를 나타내는지를 확인

E. 유저는 체력 20을 가지고 게임을 시작

i.
```
int userHealth = 20; // 유저의 체력. 초기값은 20
```

ii. 입력

1. 없음.

iii. 결과

1. 유저의 현재 HP를 저장하는 userHealth를 선언하고 20으로 초기화

iv. 설명

1. userHealth를 전역변수로 사용하여 함수의 파라미터로 주지 않더라도 함수에서 사용 가능

F. HP가 0이 되면 “실패”를 출력하고 종료

i.
```
// 체력이 1 아래가 되었는지 체크
if (isOver()) {
    // 문자열을 출력하고 loop 종료
    cout << "HP가 0 이하가 되었습니다. 실패했습니다." << endl;
    cout << "게임을 종료합니다." << endl;
    break;
}
```

ii. 입력

1. isOver함수의 반환값 : true/false

iii. 결과

1. 체력이 1 아래가 되어 게임에 실패했다면 이를 안내하는 메시지 출력
2. 게임 종료를 안내하는 메시지 출력 및 게임 종료

iv. 설명

1. isOver 함수를 호출해 유저의 현재 HP가 1 아래인지를 확인
2. HP가 1 아래가 되었다면 문자열을 출력
3. break로 while loop를 탈출해 게임 종료
4. isOver : 유저의 HP가 0이 되어 게임에 실패했는지 체크하는 함수

A.

```
// 사용자의 HP가 0이 되어 게임에 실패했는지를 체크하는 함수
bool isOver() {

    // HP가 1 아래가 되면 true. 즉, 게임 실패
    if (userHealth < 1) {
        return true;
    }
    return false;
}
```

B. 입력

i. 없음

C. 반환값

i. true/false : 체력이 1 아래인지에 대한 boolean값

D. 결과

i. true 또는 false를 반환

E. 설명

i. 전역변수인 userHealth에 접근해 1보다 작은지 if문 수행

ii. 1보다 작다면 게임에 실패했으므로 true반환 아니면 false반환

4. 테스트

A. 사용자에게 현재 HP를 출력하고 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기

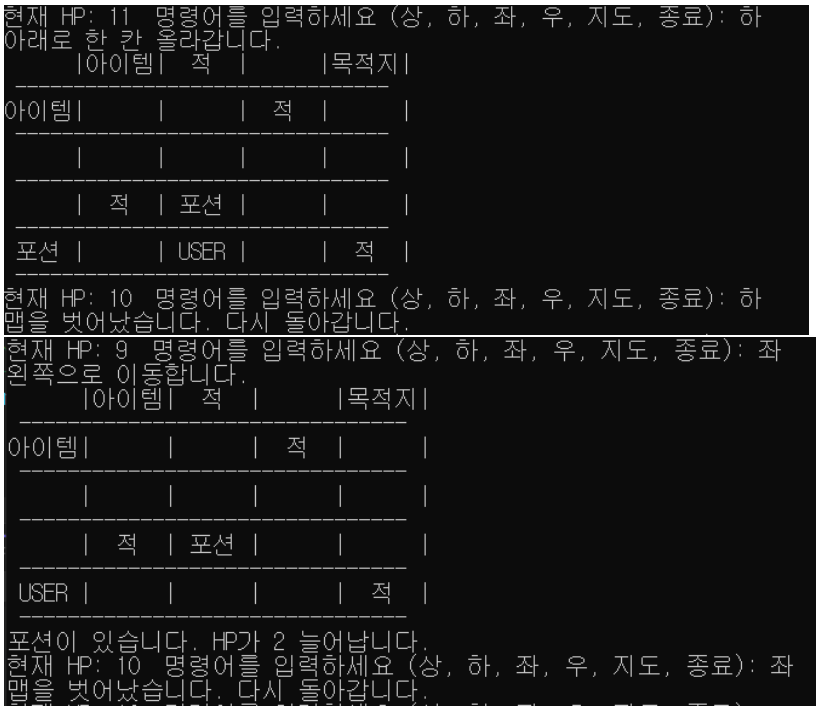
```
현재 HP: 20  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료):
```

B. 사용자의 입력에 따라 정해놓은 조작 실시

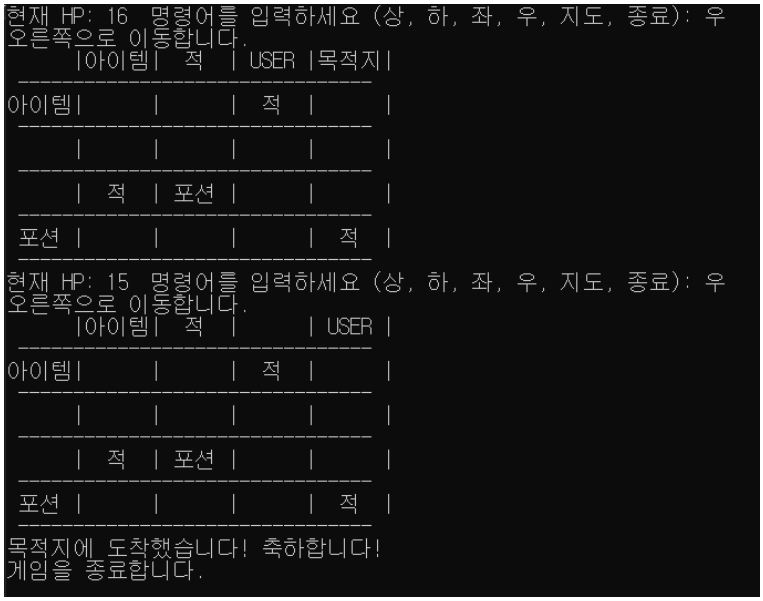
- “상/하/좌/우” 입력시 ①해당 방향으로 이동 후 체력 1감소 ②전체 지도 출력 ③이동한 좌표에서 아이템,포션,적을 만났을 때 그에 대한 메시지 출력 및 HP증감
- “지도”를 입력시 전체 지도와 함께 현재 위치를 출력
- “종료”를 입력시 게임을 즉시 종료
- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재입력 요청

```
현재 HP: 20  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 하
아래로 한 칸 올라갑니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
 USER |      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
아이템이 있습니다.
현재 HP: 19  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 상
위로 한 칸 올라갑니다.
 USER |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
현재 HP: 18  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 우
오른쪽으로 이동합니다.
      | USER |  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
아이템이 있습니다.
현재 HP: 17  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 좌
왼쪽으로 이동합니다.
 USER |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
현재 HP: 15  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 우
오른쪽으로 이동합니다.
      |아이템| USER |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
적이 있습니다. HP가 2 줄어듭니다.
현재 HP: 12  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료):
현재 HP: 10  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 하
아래로 한 칸 올라갑니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | USER |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
포션이 있습니다. HP가 2 늘어납니다.
현재 HP: 11  명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료):
```

C. 지도 밖으로 벗어나게 되면 에러 메시지 출력

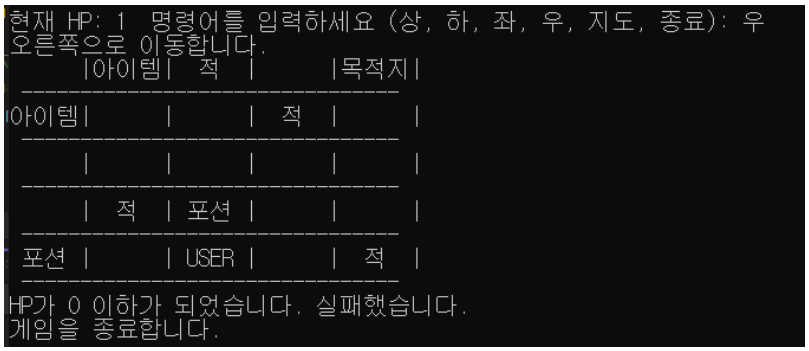


D. 목적지에 도착하면 “성공”을 출력하고 종료

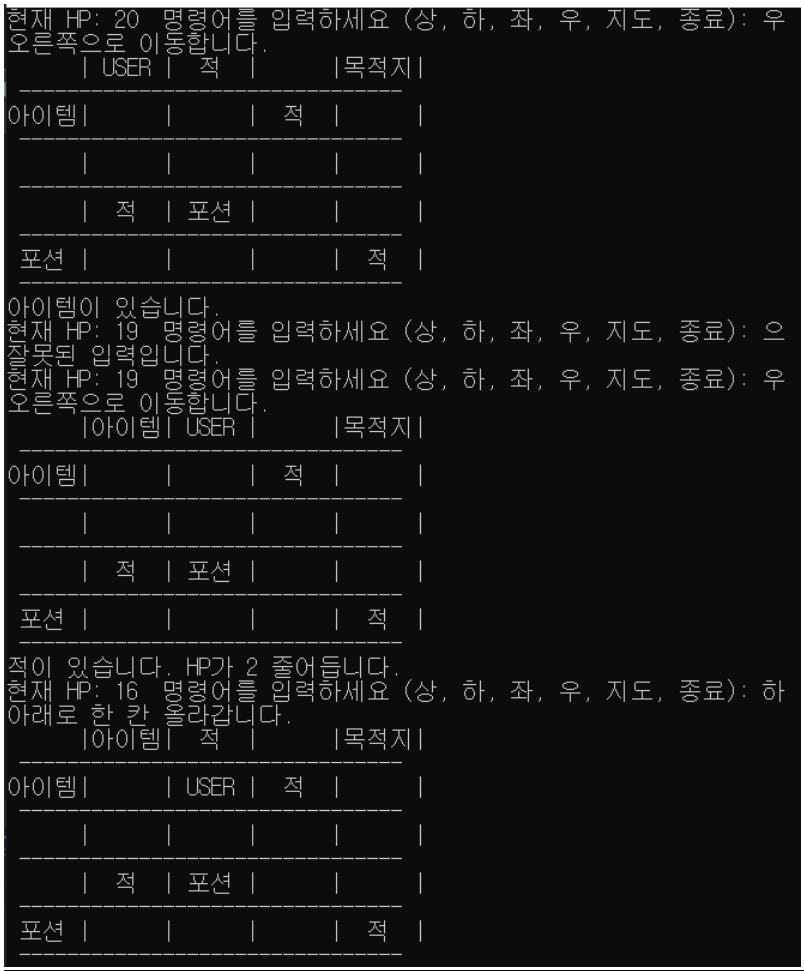


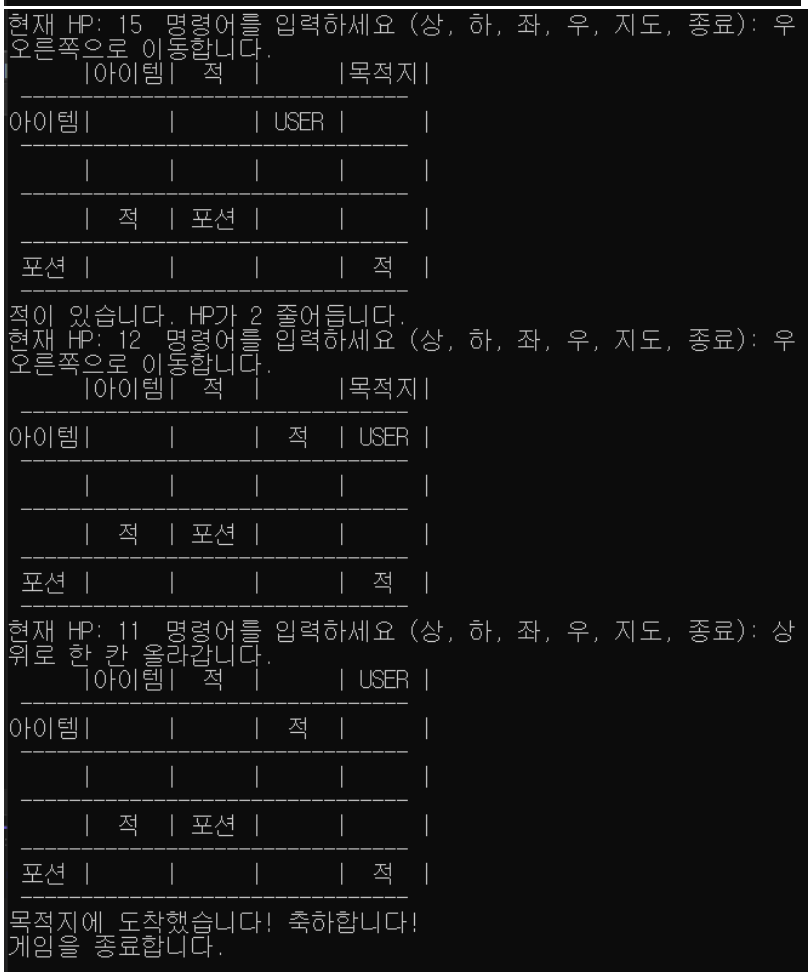
E. 유저는 체력 20을 가지고 게임을 시작

F. HP가 0이 되면 “실패”를 출력하고 종료



G. 최종 테스트


현재 HP: 20 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 우
오른쪽으로 이동합니다.
| USER | 적 | |목적지|
아이템| | | 적 | |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
아이템이 있습니다.
현재 HP: 19 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 으
잘못된 입력입니다.
현재 HP: 19 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 우
오른쪽으로 이동합니다.
|아이템| USER | |목적지|
아이템| | | 적 | |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
적이 있습니다. HP가 2 줄어듭니다.
현재 HP: 16 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 하
아래로 한 칸 올라갑니다.
|아이템| 적 | |목적지|
아이템| | USER | 적 | |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |


현재 HP: 15 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 우
오른쪽으로 이동합니다.
|아이템| 적 | |목적지|
아이템| | | USER | |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
적이 있습니다. HP가 2 줄어듭니다.
현재 HP: 12 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 우
오른쪽으로 이동합니다.
|아이템| 적 | |목적지|
아이템| | | 적 | USER |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
현재 HP: 11 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 상
위로 한 칸 올라갑니다.
|아이템| 적 | | USER |
아이템| | | 적 | |
| | | | |
| 적 | 포션 | | |
포션 | | | | 적 |
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.

5. 결과 및 결론

A. 프로젝트 결과 : Mud game을 만들었다. 기능별 구현 및 몇몇 기능에 대한 함수화를 진행하였다.

B. 느낀 점 : 코드에 대해 반복해서 볼수록 점점 더 세세한 부분에 대한 함수화를 떠올릴 수 있었다. 함수화가 많이 이루어질수록 확실히 유지보수에 유리할 것 같다는 것이 느껴졌다.