1. 서론

(1). 프로젝트 목적 및 배경: 7주차까지 배운 내용에 대한 실습을 위해 진행

(2). 목표: 간단한 Mud 게임 구현

2. 요구사항

(1). 사용자 요구사항: 유저가 상하좌우로만 이동하며 목적지에 도착하는 게임

(2). 기능 계획

① 사용자에게 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받기

- HP 출력
- 상/하/좌/우 입력 시 해당 방향으로 이동 후 지도 출력
- 지도 밖으로 나가게 되면 에러 메시지 출력
- "지도"를 입력하면 전체 지도와 함께 현재 위치를 출력
- "종료"를 입력하면 게임 종료
- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청

② 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력

③ HP가 0이 되면 "실패"를 출력하고 종료

④ 목적지에 도착하면 "성공"을 출력하고 종료

(3). 함수 계획

① 메인 함수: 사용자에게 값을 계속 입력 받고, 그에 대한 함수 호출

② 지도와 현재 위치 출력 함수: displayMap()

③ 사용자 위치 체크 함수: checkXY()

④ 목적지에 도착 체크 함수: checkCoal()

⑤ 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력 함수: checkState()

⑥ 지도를 벗어날 때 출력 함수: printFalse()

⑦ 이동 방향 출력 함수: printTrue()

3. 설계 및 구현

(1). 기능 별 구현 사항

main() 함수

① 지도 만들기

```
// 메인 함수
int main() {
    // 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지
    int map[mapY][mapX] = { {0, 1, 2, 0, 4},
                            {1, 0, 0, 2, 0},
                            {0, 0, 0, 0, 0},
                            {0, 2, 3, 0, 0},
                            {3, 0, 0, 0, 2} };


    // 유저의 위치를 저장할 변수
    int user_x = 0; // 가로 번호
    int user_y = 0; // 세로 번호
```

입력:

- mapY = 지도의 세로 칸의 개수 (행의 개수)
- mapX = 지도의 가로 칸의 개수 (열의 개수)

결과:

- 이차원 배열 map[mapY][mapX] 선언, 초기화

설명:

- 이차원 배열에 각각의 원소를 집어 넣어 지도를 만든다.
- 이때 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지를 나타낸다.

② 사용자에게 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받기

- HP 출력

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

        cout << endl;

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

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

입력:

- user_input = 사용자의 입력을 저장할 변수
- hp = 현재 체력

결과:

- 현재 HP를 출력한다.
- 상, 하, 좌, 우, 지도, 종료 중 하나를 사용자에게 입력 받아 user_input에 저장한다.

설명:

- while(1)을 통해 무한 루프로 사용자에게서 user_input 값을 입력 받는다.

  -> 이 게임(while)을 빠져나오기 위해서는 break;이 필요하다.

- 상/하/좌/우 입력 시 해당 방향으로 이동 후 지도 출력
- 지도 밖으로 나가게 되면 에러 메시지 출력

```
if (user_input == "상") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        printFalse();
        user_y += 1;
        continue;
    }
    else {
        string move = "위";
        printTrue(move);
        displayMap(map, user_x, user_y);
    }
}
else if (user_input == "하") {
    // 아래로 한 칸 내려가기
    user_y += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        printFalse();  // 맵을 벗어날 때 출력할 함수 호출
        user_y -= 1;
        continue;
    }
    else {
        string move = "아래";
        printTrue(move);  // 맵을 벗어나지 않을 때 출력할 함수 호출
        displayMap(map, user_x, user_y);
    }
}
```

```
else if (user_input == "좌") {
    // 왼쪽으로 이동하기
    user_x -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        printFalse();
        user_x += 1;
        continue;
    }
    else {
        string move = "왼";
        printTrue(move);
        displayMap(map, user_x, user_y);
    }
}
else if (user_input == "우") {
    // 오른쪽으로 이동하기
    user_x += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        printFalse();
        user_x -= 1;
        continue;
    }
    else {
        string move = "오른";
        printTrue(move);
        displayMap(map, user_x, user_y);
    }
}
```

입력:

- user_input = 사용자가 입력한 문자열
- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치

- inMap = 현재 사용자의 위치가 지도를 벗어나는지 나타내는 bool 타입 변수
- mapX = 지도의 가로 칸 수
- mapY = 지도의 세로 칸 수
- move = 사용자의 움직임을 나타내는 string 타입 변수 (”위”, ”아래”, ”왼”, ”오”)
- map = 지도에 대한 이차원 배열

결과:

- 사용자가 입력한 문자열에 따라 현재 사용자의 위치를 한 칸 이동한다.
- 함수 checkXY()를 호출한다.
- inMap이 false일 경우 printFalse() 함수를 호출하고, 앞에서 이동하였던 사용자의 위치를 원상복구 시켜준다. 그리고 다시 while의 첫 문장으로 돌아가 사용자에게서 입력을 다시 받는다.
- inMap이 true일 경우 printTrue() 함수를 호출하고, displayMap() 함수를 통해 이동한 위치의 지도를 보여준다.

설명:

- if – else if 문을 이용하여 사용자가 입력한 값이 ”상”, ”하”, ”좌”, ”우”일 경우를 나눈다.
- 사용자가 입력한 방향으로 한 칸 움직여 사용자의 현재 위치를 변경한다.

”상” -> 위로 한 칸 이동 -> 사용자의 현재 세로 위치가 1 감소 -> user_y -= 1

”하” -> 아래로 한 칸 이동 -> 사용자의 현재 세로 위치가 1 증가 -> user_y += 1

”좌” -> 왼쪽으로 한 칸 이동 -> 사용자의 현재 가로 위치가 1 감소 -> user_x -= 1

”우” -> 오른쪽으로 한 칸 이동 -> 사용자의 현재 가로 위치가 1 증가 -> user_x += 1

- inMap 변수에 이동하려는 위치가 유효한지 체크하기 위해 호출한 함수 checkXY()의 반환값을 저장한다. (bool 타입 변수로 true나 false를 값으로 가짐)
- inMap이 false일 경우 해당 위치가 지도에서 벗어난다는 의미로, 그에 대한 메시지를 출력할 printFalse() 함수를 호출한다. 전에 이동시켜준 현재 위치를 다시 원상복구 한다. 그리고 continue;를 통해 while문의 첫 문장으로 돌아간다.

”상” -> 다시 아래로 한 칸 이동 -> user_y += 1

”하” -> 다시 위로 한 칸 이동 -> user_y -= 1

"좌" -> 다시 오른쪽으로 한 칸 이동 -> user_x += 1

"우" -> 다시 왼쪽으로 한 칸 이동 -> user_x += 1

- inMap이 true일 경우, 해당 위치로 이동할 수 있다. 따라서 이동 방향에 대한 메시지를 출력하는 printFalse() 함수를 호출한다. displayMap() 함수를 호출하여 현재 지도를 출력한다.

- "지도"를 입력하면 전체 지도와 함께 현재 위치를 출력

```
else if (user_input == "지도") {
    // 지도 보여주기 함수 호출
    displayMap(map, user_x, user_y);
    continue;
}
```

입력:

- user_input = 사용자가 입력한 문자열
- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치
- map = 지도에 대한 이차원 배열

결과:

- displayMap()을 호출하고, 다시 while의 첫 문장으로 돌아간다.

설명:

- user_input이 "지도"일 경우, 현재 사용자의 위치를 지도상으로 출력해준다.
- continue;를 통해 while문의 첫 문장으로 돌아가 또 다른 사용자 입력을 받는다. 이때 continue;를 입력하지 않으면 만약 현재 위치에 적이나 포션이 있을 경우, checkState() 함수의 호출에 의해 HP의 변동이 생기게 된다.

- "종료"를 입력하면 게임 종료

```
else if (user_input == "종료") {
    cout << "종료합니다.";
    break;
}
```

입력:

- user_input = 사용자가 입력한 문자열

결과:

- 종료 메시지를 출력하고, 게임을 종료한다.

설명:

- user_input이 "종료"일 경우 종료 메시지를 출력하고, break;을 통해 while문을 벗어난다.

- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청

```
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

입력:

결과:

- 잘못된 입력이라는 메시지를 출력하고, 다시 사용자에게 새로운 입력을 받는다.

설명:

- user_input이 상, 하, 좌, 우, 지도, 종료 중 어느 것에도 해당하지 않는다면, 잘못된 입력이라는 메시지를 출력하고, continue;를 통해 while문의 첫 문장으로 다시 돌아가 새로운 사용자 입력을 받는다.

③ 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력

```
// 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력 & HP 감소/증가
checkState(map, user_x, user_y);
```

입력:

- map = 지도에 대한 이차원 배열
- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치

결과:

- checkState() 함수를 호출한다.

설명:

- checkState() 함수는 아이템, 포션, 적을 만났을 때 그에 대한 메시지를 출력하고, 그에 따른 HP를 감소/증가 한다.

④ HP가 0이 되면 "실패"를 출력하고 종료

```
        // HP가 0 이하가 되면 게임 종료
        if (hp <= 0) {
            cout << "HP가 0 이하가 되었습니다. 실패했습니다." << endl;
            cout << "게임을 종료합니다." << endl;
            break;
        }
```

입력:

- hp = 현재 체력

결과:

- hp의 값이 0 이하일 경우, 그에 따른 메시지를 출력하고 게임을 종료한다.

설명:

- hp의 값이 0 이하가 되면 해당 메시지를 출력하고, break;을 통해 while문을 빠져나가며 게임을 종료한다.

⑤ 목적지에 도착하면 "성공"을 출력하고 종료

```
        // 목적지에 도달했는지 체크
        bool finish = checkGoal(map, user_x, user_y);
        if (finish == true) {
            cout << "목적지에 도착했습니다! 축하합니다!" << endl;
            cout << "게임을 종료합니다." << endl;
            break;
        }
    }
    return 0;
}
```

입력:

- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치
- map = 지도에 대한 이차원 배열

- finish = 목적지에 도착했는지 여부를 저장하는 bool 타입 변수

결과:

- 목적지에 도착했을 경우, 그에 대한 메시지를 출력하고 게임을 종료한다.

설명:

- finish가 true일 경우, 즉 목적지에 도착했을 경우 목적지에 도착했다는 메시지를 출력하고, break;을 통해 게임을 종료한다.

함수

① 지도와 현재 위치 출력 함수: displayMap()

```
// 지도와 사용자 위치 출력하는 함수
void displayMap(int map[][mapX], int user_x, int user_y) {
    for (int i = 0; i < mapY; i++) {
        cout << " -------------------------------- " << endl;
        cout << "|";
        for (int j = 0; j < mapX; j++) {
            if (i == user_y && j == 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;
}
```

입력:

- int map[][] = 2차원 배열 지도

- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치

반환값:

- void 타입의 함수로 반환값이 없다. 그러나 함수 내에서 행해지는 출력이 존재한다.

결과:

- 전체 지도를 출력한다.
- 현재 사용자의 위치 뿐만 아니라 아이템, 적, 포션, 목적지에 대한 위치도 지도에 표지 한다.

설명:

- 2차원 배열의 원소들을 출력한다.
- 사용자의 위치와 동일한 좌표를 발견할 경우 사용자의 정보를 출력한다.
- map의 원소를 출력할 때 0은 빈 공간(" "), 1은 "아이템", 2는 "적", 3은 "포션", 4는 "목적지"로 바꾸어 출력한다.

② 사용자 위치 체크 함수: checkXY()

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int user_x, int mapX, int user_y, int mapY) {
    bool checkFlag = false;
    if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) {
        checkFlag = true;
    }
    return checkFlag;
}
```

입력:

- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치
- mapX = 지도의 가로 칸 수
- mapY = 지도의 세로 칸 수

반환값:

- bool 타입의 checkFlag, 이때 checkFlag는 현재 사용자의 위치가 유효한지 나타내는 변수의 값

결과:

- 사용자가 이동하고자 하는 위치가 지도상에서 유효한 좌표라면 true를 반환하고, 아니면 false를 반환한다.

설명:

- 사용자가 이동하고자 하는 좌표가 지도상에서 유효한지 확인하기 위해서는 사용자의 현재 가로와 세로의 위치가 각각 0이상 5미만에 속해야 한다. (map의 크기가 5*5)
- user_x와 user_y의 값이 모두 0이상 5미만일 경우 checkFlag에 true가 대입되고, 그게 아니면 checkFlag는 false의 값을 가진다.

③ 목적지에 도착 체크 함수: checkCoal()

```
// 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;
    }
    return false;
}
```

입력:

- int map[][] = 2차원 배열 지도
- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치

반환값:

- bool 타입의 true/false, 이들은 현재 사용자의 위치가 목적지인지/아닌지 나타낸다.

결과:

- 현재 사용자의 위치가 목적지와 같다면 true를 반환, 아니면 false를 반환한다.

설명:

- 현재 사용자의 위치, 즉 map[user_y][map_x]의 값이 4일 경우 true를 반환한고, 현재

위치의 2차원 배열 map의 원소가 4가 아닐 경우 false를 반환한다. (이때 4는 목적지를 나타냄)

④ 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력 함수: checkState()

```
// 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력 & HP 감소/증가 함수
void checkState(int map[][mapX], int user_x, int user_y) {
    // 적을 만났을 때 HP 2 감소, 포션을 만났을 때 HP 2 증가
    switch (map[user_y][user_x]) {
    case 1:
        cout << "아이템이 있습니다." << endl;
        break;
    case 2:
        hp -= 2;
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
        break;
    case 3:
        hp += 2;
        cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;
        break;
    }
}
```

입력:

- int map[][] = 2차원 배열 지도
- user_x = 사용자 현재 위치의 가로 위치
- user_y = 사용자 현재 위치의 세로 위치

반환값:

- void 타입의 함수로 반환값이 없다. 그러나 함수 내에서 행해지는 출력이 존재한다.

결과:

- 현재 지도에서 사용자가 위치한 곳의 원소(아이템/적/포션/)를 출력한다.
- 현재 사용자가 적을 만났을 경우 HP가 2 감소하고, 포션을 만났을 경우 HP가 2 증가한다.

설명:

- switch문을 통해 현재 사용자의 위치(map[user_y][user_y])가 지도상에서 무엇인지를 출력한다. (2차원 배열 map의 원소가 1일 땐 아이템/ 2일 땐 적/ 3일 땐 포션이 존재한다.)
- 현재 위치가 map의 원소에서 2의 값을 가질 때, 즉 적과 만났을 경우 hp가 2 감소

한다.

- 현재 위치가 map의 원소에서 3의 값을 가질 때, 즉 포션과 만났을 경우 hp가 2 증가한다.

⑤ 지도를 벗어날 때 출력 함수: printFalse()

```
//맵을 벗어날 때 출력할 함수
void printFalse() {
    cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
}
```

입력: X

반환값:

- void 타입의 함수로 반환값이 없다. 그러나 함수 내에서 행해지는 출력이 존재한다.

결과:

- 맵을 벗어났다고 출력된다.

설명:

- 이는 사용자가 이동하고 싶어하는 좌표가 지도 내에서 유효하지 않을 때 호출되는 함수로, 맵을 벗어났다는 메시지를 출력한다.

⑥ 이동 방향 출력 함수: printTrue()

```
//맵을 벗어나지 않고 원하는 방향으로 이동 가능할 때 출력할 함수
void printTrue(string move) {
    cout << move << "쪽으로 한 칸 이동합니다." << endl;
    hp -= 1;  // 이동할 때 체력 1 감소
}
```

입력:

- move = 사용자의 움직임

반환값:

- void 타입의 함수로 반환값이 없다. 그러나 함수 내에서 행해지는 출력이 존재한다.

결과:

- 사용자의 이동을 출력한다.

- hp의 값이 1 감소한다.

설명:

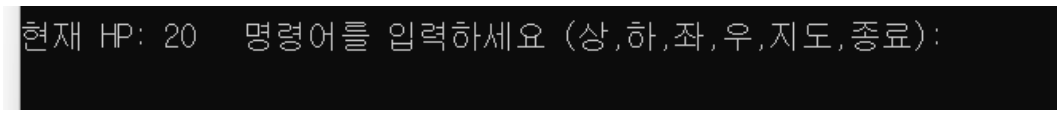
- 사용자의 움직임(move 값)에 따른 출력이 이루어진다. 이는 사용자가 유효한 좌표 내에서 이동했다는 의미로 HP의 값이 1 감소해야 한다.

4. 테스트

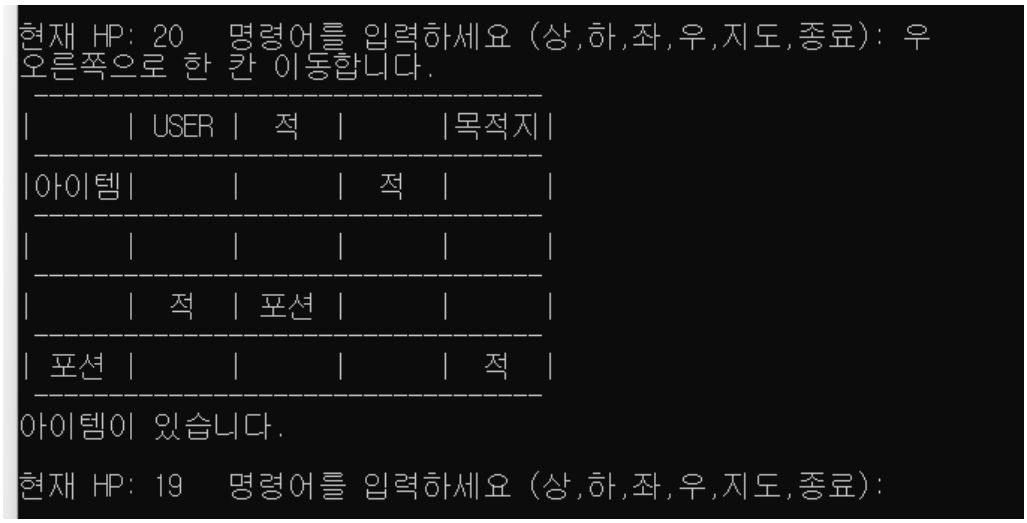
(1). 기능 별 테스트 결과

① 사용자에게 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받기

- HP 출력

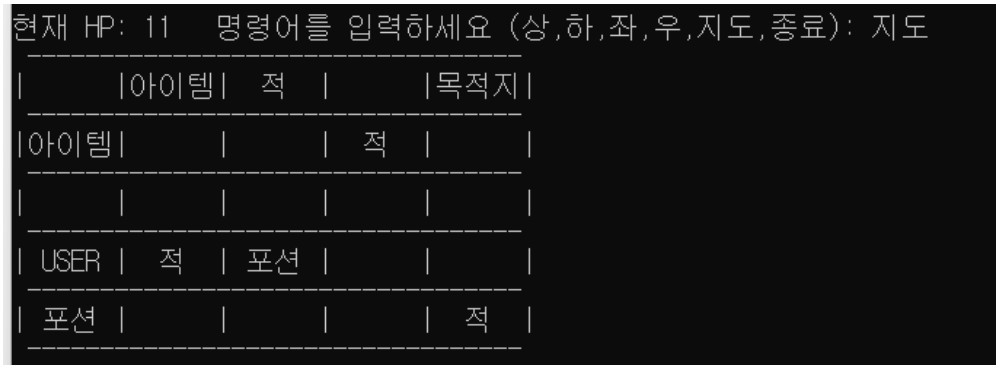


- 상/하/좌/우 입력 시 해당 방향으로 이동 후 지도 출력



- 지도 밖으로 나가게 되면 에러 메시지 출력

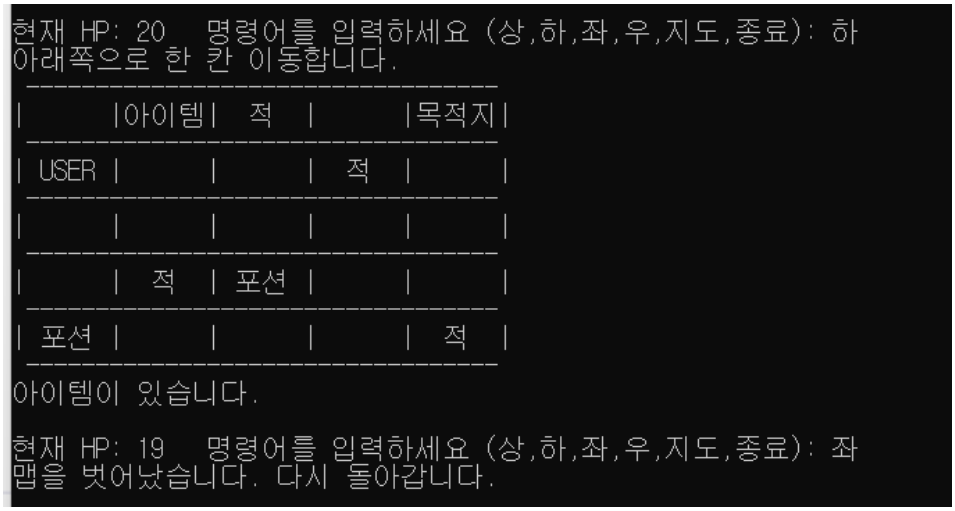
• "지도"를 입력하면 전체 지도와 함께 현재 위치를 출력



• "종료"를 입력하면 게임 종료

• 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청

② 아이템, 포션, 적을 만났을 때 그에 대한 메시지 출력



```
현재 HP: 15   명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      | USER | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
적이 있습니다. HP가 2 줄어듭니다.

현재 HP: 12   명령어를 입력하세요 (상,하,좌,우,지도,종료): 지도
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
|아이템|      |      |  적  |      |
```

```
현재 HP: 11   명령어를 입력하세요 (상,하,좌,우,지도,종료): 좌
왼쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| USER |      |      |      |  적  |
 ---------------------------------
포션이 있습니다. HP가 2 늘어납니다.

현재 HP: 12   명령어를 입력하세요 (상,하,좌,우,지도,종료): 좌
맵을 벗어났습니다. 다시 돌아갑니다.
```

③ HP가 0이 되면 "실패"를 출력하고 종료

```
현재 HP: 1   명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      | USER |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
HP가 0 이하가 되었습니다. 실패했습니다.
게임을 종료합니다.
```

④ 목적지에 도착하면 "성공"을 출력하고 종료

```
현재 HP: 9  명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  |      | USER |
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

(2). 최종 테스트 스크린샷

```
현재 HP: 7  명령어를 입력하세요 (상,하,좌,우,지도,종료): 좌
왼쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  |      |목적지|
 ---------------------------------
|아이템|      |      | USER |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
적이 있습니다. HP가 2 줄어듭니다.

현재 HP: 4  명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
위쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  | USER |목적지|
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------

현재 HP: 3  명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 한 칸 이동합니다.
 ---------------------------------
|      |아이템|  적  |      | USER |
 ---------------------------------
|아이템|      |      |  적  |      |
 ---------------------------------
|      |      |      |      |      |
 ---------------------------------
|      |  적  | 포션 |      |      |
 ---------------------------------
| 포션 |      |      |      |  적  |
 ---------------------------------
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

5. 결과 및 결론

(1). 프로젝트 결과: mud game을 만들었다.

(2) 느낀 점: 초반에 코드를 받았을 때, 해당 코드에서 사용되는 함수에 대해 모두 정의해야 되는 줄 알고 눈 앞이 깜깜했는데, HP를 추가하고 그에 대한 함수만 만들면 돼서 별로 어렵지 않았던 것 같다.

Mud 게임을 처음 봤을 때 너무 어려울 것 같고, 어떻게 게임을 진행시켜야 되는지 막막했는데, 주어진 코드를 해석하고 내가 추가로 만든 코드를 더해 실행하는 과정에서 해당 게임에 대한 이해도와 많은 뿌듯함을 얻었다.