Mud Game

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 이동하여 목적지에 도착하는 게임
2. 기능 요구사항
   ① 유저는 체력 20을 가지고 게임시작
   ② 처음 명령문을 입력 받을 때 마다 HP 함께 출력
   ③ 사용자에게 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기
   ④ 상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력
   ⑤ 사용자가 이동할 때 마다 사용자 체력 1씩 감소
   ⑥ 지도 밖으로 나가게 되면 에러 메시지 출력
   ⑦ “지도”를 입력하면 전체 지도와 함께 현재 위치를 출력
   ⑧ “종료”를 입력하면 프로그램 종료
   ⑨ 이 중 다른 것을 입력하면 에러 메시지 출력 후 재입력 요청
   ⑩ 목적지에 도착하면 “성공을 출력하고 종료
   ⑪ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 HP 조정 및 메시지 출력
      · 예) {X}가 있습니다.
      · 적을 만날 경우 HP가 2가 줄어들고 그에 대한 추가 메시지 출력
      · 포션을 만날 경우 HP가 2가 늘어나고 그에 대한 추가 메시지 출력
      · (적이나 포션 등은 사라지지 않음을 전제)
   ⑫ HP가 0가 되면 ”실패“를 출력하고 종료
3. 함수 계획
   ① 메인 함수: 사용자에게 값을 계속 입력받고, 그에 대한 함수 호출
   ② 지도와 현재 위치 출력 함수: displayMap()
   ③ 이동할 위치 유효성 체크 및 동작(HP 감소, 문구 출력) 함수: checkXY()
   ④ 목적지에 도착 체크 함수: checkGoal
   ⑤ 유저 위치의 상태 체크 및 동작(HP 변동, 문구 출력) 함수: checkState()

3. 설계 및 구현

<table>
<tr><td colspan="2">① 유저는 체력 20을 가지고 게임시작</td></tr>
<tr><td rowspan="3"> 
</td><td>1. 입력
· hp: 유저의 체력을 저장하는 변수</td></tr>
<tr><td>2. 결과
· 유저의 체력 20으로 시작</td></tr>
<tr><td>3. 설명
· hp 변수를 20으로 초기화</td></tr>
<tr><td colspan="2">② 처음 명령문을 입력 받을 때 마다 HP 함께 출력</td></tr>
<tr><td rowspan="3"> 
</td><td>1. 입력
· hp: 유저의 체력을 저장하는 변수</td></tr>
<tr><td>2. 결과
· 유저의 현재 체력 표시</td></tr>
<tr><td>3. 설명
· 사용자 입력을 받을 때 hp 변수의 값을 출력</td></tr>
</table>

③ 사용자에게 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기

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

1. 입력
   · user_input: 현재 실행할 행동에 대한 값을 저장하는 변수
2. 결과
   · 실행할 행동에 대한 값을 입력 받음
3. 설명
   · user_input 변수에 다음 행동에 대한 값을 사용자 입력

④ 상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력

⑤ 사용자가 이동할 때 마다 사용자 체력 1씩 감소

⑥ 지도 밖으로 나가게 되면 에러 메시지 출력

```
if (user_input == "상") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    if (checkXY(user_x, mapX, user_y, mapY, hp)) {
        cout << "위로 한 칸 올라갑니다." << endl;
        displayMap(map, user_x, user_y);
    }
    else {
        user_y += 1;
        continue;
    }
}
```

```
else if (user_input == "하") {
    // TODO: 아래로 한 칸 내려가기
    user_y += 1;
    if (checkXY(user_x, mapX, user_y, mapY, hp)) {
        cout << "아래로 한 칸 내려갑니다." << endl;
        displayMap(map, user_x, user_y);
    }
    else {
        user_y -= 1;
        continue;
    }
}
```

```
else if (user_input == "좌") {
    // TODO: 왼쪽으로 이동하기
    user_x -= 1;
    if (checkXY(user_x, mapX, user_y, mapY,  hp)) {
        cout << "왼쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
    }
    else {
        user_x += 1;
        continue;
    }
}
```

```
else if (user_input == "우") {
    // TODO: 오른쪽으로 이동하기
    user_x += 1;
    if (checkXY(user_x, mapX, user_y, mapY, hp)) {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
    }
    else {
        user_x -= 1;
        continue;
    }
}
```

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int user_x, int mapX, int user_y, int mapY, int& hp) {
    bool checkFlag = false;
    if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) {
        checkFlag = true;
        hp--;
    }
    else
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
    return checkFlag;
}
```

1. 입력
   · user_input: 현재 실행할 행동에 대한 값을 저장하는 변수
   · map: 지도의 정보를 담고 있는 2차원 배열(5x5)
   · user_x: 유저의 가로 위치 정보 변수
   · user_y: 유저의 세로 위치 정보 변수
   · mapX: 지도의 가로 크기 변수
   · mapY: 지도의 세로 크기 변수
   · hp: 유저의 체력을 저장하는 변수

2. 결과
   · 사용자의 입력에 따라
     · 유효한 경우: 이동 문구를 출력하고 유저의 위치 이동하여 지도 출력, 체력 1감소
     · 유효하지 않은 경우: 에러 메시지를 출력하고 재입력 요청

3. 설명
   · if문을 통해 상, 하, 좌, 우 이동 방향 체크
   · 해당 방향 이동에 따른 유저 위치 이동
   · 이동한 유저의 위치가 유효한지 체크(checkXY)
   · checkXY(이동 방향 관련x 공통 동작 수행)
     · 유효한 경우: hp 1감소(true 반환)
     · 유효하지 않은 경우: 에러 메시지 출력(false 반환)
   ·if문과 checkXY 반환값(이동 방향에 따른 작업)
     · 유효한 경우: 이동 문구 출력 및 지도 출력
       (지도- 실행순서로 인해 checkXY에서 실행x)
     · 유효하지 않은 경우: 유저 위치 복귀
       continue문으로 재입력 요청

⑦ “지도”를 입력하면 전체 지도와 함께 현재 위치를 출력

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

```
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
            }
            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;
    }
}
```

1. 입력
   · user_input: 현재 실행할 행동에 대한 값을 저장하는 변수
   · map: 지도의 정보를 담고 있는 2차원 배열(5x5)
   · user_x: 유저의 가로 위치 정보 변수
   · user_y: 유저의 세로 위치 정보 변수
   · mapX: 지도의 가로 크기 변수
   · mapY: 지도의 세로 크기 변수

2. 결과
   · 지도의 각 항목과 유저의 위치를 포함한 지도 출력

3. 설명
   · 이중 반복문으로 지도의 정보를 순회
   · if문을 통해 유저의 위치 출력(+가로 칸 구분선 출력)
   · switch문을 통해 각 위치의 항목 출력
   (+가로 칸 구분선 출력)
   · 행이 바뀔 때마다 세로칸 구분선 출력

⑧ “종료”를 입력하면 프로그램 종료

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

1. 입력
   · user_input: 현재 실행할 행동에 대한 값을 저장하는 변수

2. 결과
   · “종료합니다” 문구를 출력하고 프로그램 종료

3. 설명
   · if문을 통해 종료 행동이 입력됐는지 체크
   · 종료 문구 출력
   · break를 통해 while문을 빠져나가 return 도달
   (프로그램 종료)

⑨ 이 중 다른 것을 입력하면 에러 메시지 출력 후 재입력 요청

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

1. 입력

2. 결과
   · 에러 메시지를 출력하고 재입력 요청

3. 설명
   · 상, 하, 좌, 우, 지도, 종료 이외의 명령어 입력 체크
   · 에러 메시지 출력
   · continue문을 통해 while문 시작부로 이동
   (재입력 요청)

⑩ 목적지에 도착하면 “성공을 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;
}
// 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;
}
```

1. 입력
   · map: 지도의 정보를 담고 있는 2차원 배열(5x5)
   · user_x: 유저의 가로 위치 정보 변수
   · user_y: 유저의 세로 위치 정보 변수
   · mapX: 지도의 가로 크기 변수
   · mapY: 지도의 세로 크기 변수

2. 결과
   · 목적지 도달 여부 체크하고 도달 시 성공 문구 출력

3. 설명
   · checkGoal 함수
      · if문을 통해 유저의 위치가 목적지인지 체크
        (true 반환: 목적지 도달, false 반환: 목적지x)
   · if문을 통해 목적지 도달 시 성공 문구 출력
   · break문을 통해 while문을 빠져나가 return 도달
     (프로그램 종료)

⑪ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 HP 조정 및 메시지 출력

```
//무기/갑옷, 포션, 적 만났는지 체크
checkState(map, user_x, user_y, hp);
//유저가 무기/갑옷, 포션, 적을 만났는지 체크하는 함수
void checkState(int map[][mapX], int user_x, int user_y, int& hp) {
    switch(map[user_y][user_x]){
        case 1:
            cout << "아이템이 있습니다." << endl;
            break;
        case 2:
            cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
            hp -= 2;
            break;
        case 3:
            cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;
            hp += 2;
            break;
    }
}
```

1. 입력
   · map: 지도의 정보를 담고 있는 2차원 배열(5x5)
   · user_x: 유저의 가로 위치 정보 변수
   · user_y: 유저의 세로 위치 정보 변수
   · mapX: 지도의 가로 크기 변수
   · mapY: 지도의 세로 크기 변수
   · hp: 유저의 체력을 저장하는 변수

2. 결과
   · 지도의 어떤 항목에 도달했는지에 따라 문구 출력와 hp 조정

3. 설명
   · switch문을 통해 어떤 항목에 도달했는지 체크
   · 해당 case로 이동하여 문구를 출력하고 hp를 조정
   · break문을 통해 switch문을 탈출

⑫ HP가 0가 되면 ”실패“를 출력하고 종료

```
//체력이 다 떨어졌는지 체크
if(hp <= 0){
    cout << "실패";
    break;
}
```

1. 입력
   · hp: 유저의 체력을 저장하는 변수

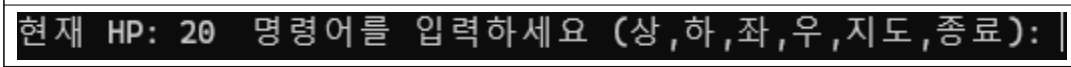
2. 결과
   · 유저의 체력이 0이 되면 실패 문구 출력 및 종료

3. 설명
   · if문으로 hp가 0인지 체크
   · hp가 0이면 실패 문구 출력
   · break문을 통해 while문을 빠져나가 return 도달
     (프로그램 종료)

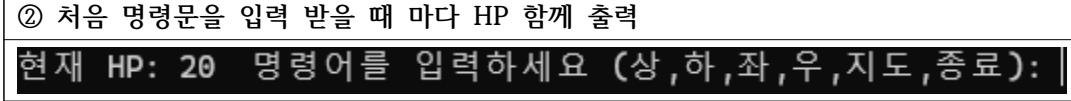
4. 테스트

4-1. 기능 별 테스트 결과

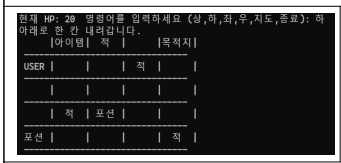
① 유저는 체력 20을 가지고 게임시작

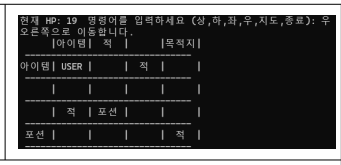


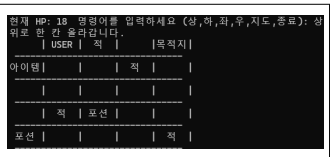
② 처음 명령문을 입력 받을 때 마다 HP 함께 출력

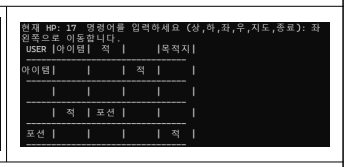


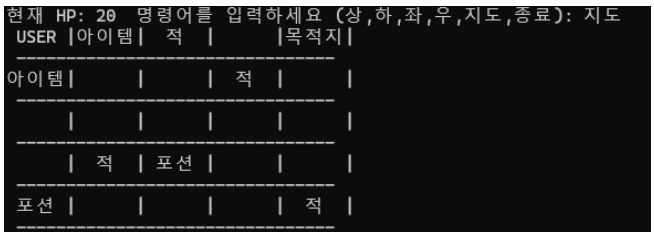
③ 사용자에게 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기

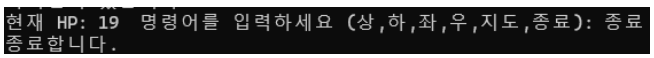




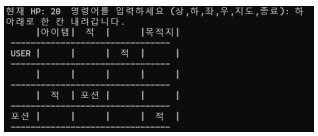


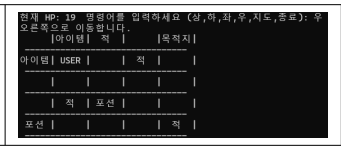


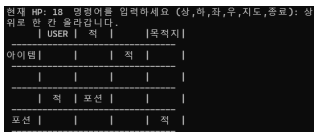


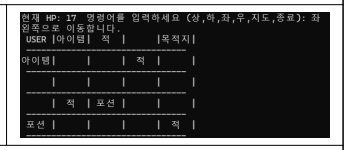


④ 상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력

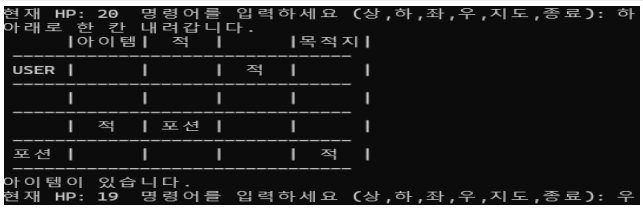




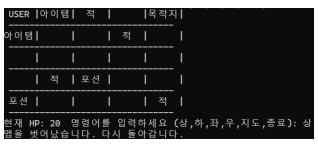


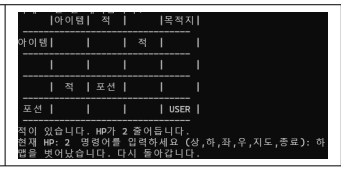


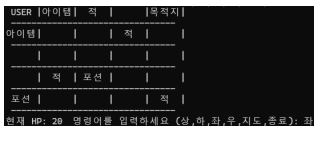
⑤ 사용자가 이동할 때 마다 사용자 체력 1씩 감소

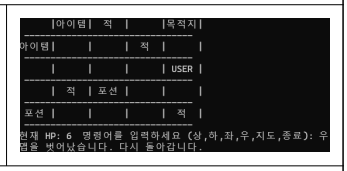


⑥ 지도 밖으로 나가게 되면 에러 메시지 출력

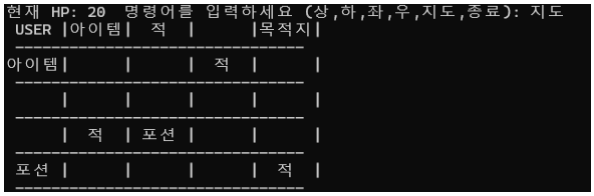




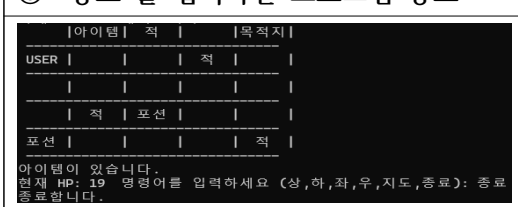




⑦ “지도”를 입력하면 전체 지도와 함께 현재 위치를 출력



⑧ “종료”를 입력하면 프로그램 종료



⑨ 이 중 다른 것을 입력하면 에러 메시지 출력 후 재입력 요청

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

⑩ 목적지에 도착하면 “성공을 출력하고 종료

```
      |아이템|  적   | USER |목적지|
 ------------------------------------
아이템|       |       |  적  |      |
 ------------------------------------
      |       |       |      |      |
 ------------------------------------
      |  적   | 포션  |      |      |
 ------------------------------------
 포션 |       |       |      |  적  |
 ------------------------------------
현재 HP: 15  명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
      |아이템|  적   |      | USER |
 ------------------------------------
아이템|       |       |  적  |      |
 ------------------------------------
      |       |       |      |      |
 ------------------------------------
      |  적   | 포션  |      |      |
 ------------------------------------
 포션 |       |       |      |  적  |
 ------------------------------------
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

⑪ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 HP 조정 및 메시지 출력

```
USER |아이템|  적   |      |목적지|
 ------------------------------------
아이템|       |       |  적  |      |
 ------------------------------------
      |       |       |      |      |
 ------------------------------------
      |  적   | 포션  |      |      |
 ------------------------------------
 포션 |       |       |      |  적  |
 ------------------------------------
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.
```

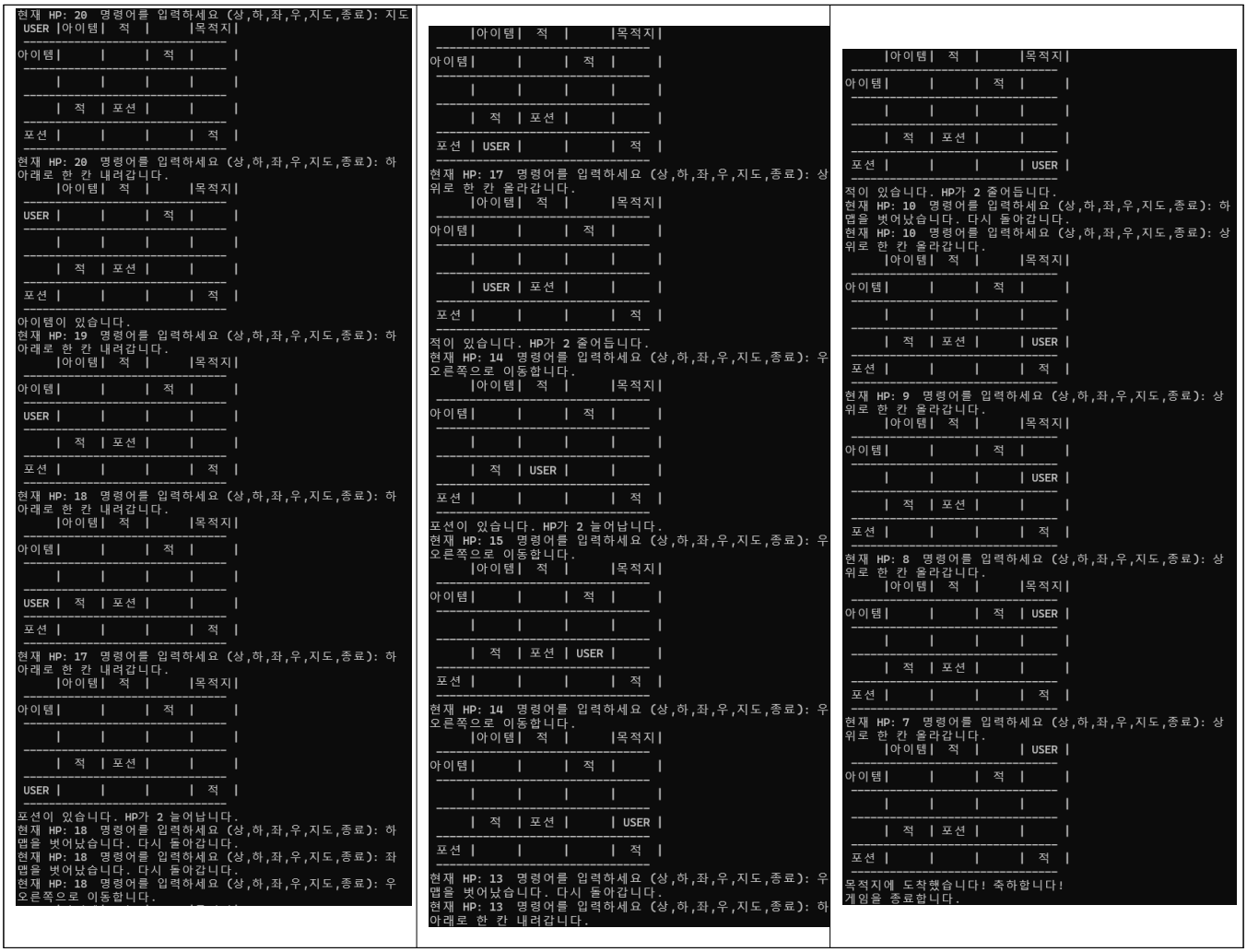
```
      | USER |  적   |      |목적지|
 ------------------------------------
아이템|       |       |  적  |      |
 ------------------------------------
      |       |       |      |      |
 ------------------------------------
      |  적   | 포션  |      |      |
 ------------------------------------
 포션 |       |       |      |  적  |
 ------------------------------------
아이템이 있습니다.
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.
```

```
      |아이템|  적   |      |목적지|
 ------------------------------------
아이템|       |       |  적  |      |
 ------------------------------------
      |       | USER  |      |      |
 ------------------------------------
      |  적   | 포션  |      |      |
 ------------------------------------
 포션 |       |       |      |  적  |
 ------------------------------------
현재 HP: 14  명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로 한 칸 내려갑니다.
      |아이템|  적   |      |목적지|
 ------------------------------------
아이템|       |       |  적  |      |
 ------------------------------------
      |       |       |      |      |
 ------------------------------------
      |  적   | USER  |      |      |
 ------------------------------------
 포션 |       |       |      |  적  |
 ------------------------------------
포션이 있습니다. HP가 2 늘어납니다.
```

⑫ HP가 0가 되면 ”실패“를 출력하고 종료

```
현재 HP: 3  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.
실패
```

4-2. 최종 테스트 스크린샷



5. 결과 및 결론

1. 프로젝트 결과

· 반복문(while, for), 조건문(if, switch), break문, continue문, 2차 배열, 함수를 통해 mud_game을 만들었습니다.

2. 느낀 점

· 코드를 작성하는 과정에서 최적화하는 파트가 있었습니다. 이 과정에서 많은 의문점이 있었던 거 같습니다. 반복되는 코드, 같은 유형의 코드들을 유지보수를 위해 하나의 함수로 묶을 때, 함수 호출 전 조건문을 함수 안에서 한 번 더 하게 되는 경우 발생하였습니다. 이때 유지보수와 성능 저하 둘 중 어떤 것을 선택해야 하느냐에 대한 고민이 들었습니다.

· 코드를 작성하는 과정에서 이동할 위치가 유효한지 체크하는 함수에 중복되는 코드를 넣었습니다. 이 때 함수는 1가지의 명확한 기능을 가지는 것이 좋은 것으로 알고 있었는데 이렇게 하면 함수를 기능을 이름과 주석으로 한 눈에 파악하기 어려워지지 않을까라는 고민이 있었습니다.(코드 복잡해질 경우) 이럴 때는 다른 함수를 따로 만드는 것이 옳은가라는 고민이 있었습니다.