mud_game

1. 서론

- 프로젝트 목적 및 배경 : 7주차까지 배운 내용에 대한 실습을 위해 진행
- 목표 : 간단한 Mud 게임 구현

2. 요구사항

- 사용자 요구사항
  - 유저가 상하좌우로만 이동하며 목적지에 도착하는 게임
  - 유저가 이동하며 아이템 / 적을 만나면 HP의 변화가 일어나는 게임

- 기능 계획

1. 기본적인 설정 사항 설정
   a. 유저의 기본 HP
   b. 보드판 크기
   c. 유저의 위치
2. 처음 명령문을 입력 받을 때 마다 HP 함께 출력
3. 사용자에게 “상”, ”하”, ”좌”, ”우”, “종료”, ‘’지도’’ 중 하나를 입력 받기 & 이동 유효성 판단
   - 상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력
   - 지도를 입력하면 전체 지도와 함께 현재 위치를 출력

- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청

4. 사용자가 이동할 때 마다 사용자 체력 1씩 감소
5. HP가 0이 되면 “실패”를 출력하고 종료
6. 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력
    - 예) {X}가 있습니다.
    - 적을 만날 경우 HP가 2가 줄어들고 그에 대한 추가 메시지 출력
7. 목적지에 도착하면 “성공”을 출력하고 종료

- 함수 계획
    - main( ) : 사용자에게 값을 계속 입력 받고, 그에 대한 함수 호출
    - displayMap( ) : 지도와 현재 위치 출력 함수
    - checkXY( ) : 사용자 위치 체크 함수
    - checkGoal( ) : 목적지에 도착 체크 함수
    - checkState( ) : 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력함수

3. 설계 및 구현

- 함수 설명

작성한 코드 설명의 이해를 높이기 위해 함수에 관한 내용 먼저 설명하겠습니다.

- 함수 종류

```
bool checkXY(int user_x, int mapX, int user_y, int mapY);
void displayMap(int map[][mapX], int user_x, int user_y);
bool checkGoal(int map[][mapX], int user_x, int user_y);
void checkState(int map[][mapX], int user_x, int user_y);
```

```
// 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;
```

- checkXY( )
    1. 사용자의 이동이 유효한지 확인
    2. 사용자의 좌표 및 크기 비교하여 유효한 위치인 경우 true 반환
    3. 유효한 위치가 아니라면 false 반환

```
// 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
            }
            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;
    }
}
```

- displayMap( )
    1. 현재 맵, 사용자의 위치 표시
    2. 맵 위치의 상태에 따라 “아이템”, “적”, ‘’포션’’, ‘’목적지’’로 표시

```
// 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) { // 목적지는 4로 미리 설정
        return true;
    }
    return false;
```

- checkGoal( )
    1. 유저의 현재 위치를 기반으로 목적지에 도착 여부 확인
    2. 목적지에 도착 : True 반환
    3. 목적지에 도착x : False 반환

```
// 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력하는 함수
void checkState(int map[][mapX], int user_x, int user_y) {
    int positionState = map[user_y][user_x];

    // switch문을 사용하여 위치에 따른 메시지 출력 및 HP 업데이트
    switch (positionState) {
    case 1:
        cout << "아이템이 있습니다." << endl;
        break;
    // 적을 만났을 경우 hp 2 감소
    case 2:
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
        user_hp -= 2;
        break;
        // 포션을 만났을 경우 hp 2 증가
    case 3:
```

```
        cout << "포션이 있습니다. HP가 2 증가합니다." << endl;
        user_hp += 2;
        break;
    default:
        break;
    }
}
```

- checkState( )
    1. 유저가 현재 위치한 맵의 위치에 따른 메시지 출력
    2. 도메인 별 발생하는 HP 증감에 따른 유저의 HP 업데이트
    3. positionState : 현재 유저의 위치에 해당하는 맵 위치 나타내는 변수

- 기능 별 구현 사항 (요구사항 별 코드)

```
int user_hp = 20; // 유저의 기본 체력 20 설정
const int mapX = 5; // 게임을 진행할 보드판 가로 크기 설정
const int mapY = 5; // 게임을 진행할 보드판 세로 크기 설정

// 사용자 정의 함수 작성을 위한 코드
bool checkXY(int user_x, int mapX, int user_y, int mapY);
void displayMap(int map[][mapX], int user_x, int user_y);
bool checkGoal(int map[][mapX], int user_x, int user_y);
void checkState(int map[][mapX], int user_x, int user_y);

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
```

1. **기본적인 설정 사항 설정**

2. 입력
    a. user_ hp : 유저의 기본 체력
    b. mapX : 보드판 가로 크기
    c. mapY : 보드판 세로 크기
    d. 네 가지 함수원형 설정
    e. user_x : 유저 x 위치 저장 변수
    f. user_y : 유저 y 위치 저장 변수
3. 결과
    a. 게임 진행에서 사용할 변수 및 함수 설정 완료
4. 설명
    a. user_hp를 전역 변수로 사용하여 게임을 진행하면서 hp 증감 표현
    b. mapX, mapY를 상수로 설정하여 보드판 크기 고정
    c. 함수원형을 작성하여 main ( )를 맨 위에 작성하도록 설정
    d. 변수 user_x, user_y를 설정하여 유저의 위치를 저장

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

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

        // 유저의 HP 표시 및 명령어 입력 받기
        cout << "현재 HP: " << user_hp << " 명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
        cin >> user_input;
```

1. **처음 명령문을 입력 받을 때 마다 HP 함께 출력**
2. 입력
    a. string user_input : string 변수 user_input을 통해 사용자의 입력 저장
3. 결과

a. 전역 변수로 설정한 user_hp를 출력

b. 이동할 방향을 입력 후 user_input에 저장

4. 설명

a. while (1)을 통해 조건이 성립할 때까지 게임 계속 진행

b. 방향을 저장하는 변수를 설정하여 추후에 사용

```
if (user_input == "상") {
            // 위로 한 칸 올라가기
            user_y -= 1;
            // checkXY함수를 이용해서 좌표의 유효성 판별 -> bool형
            bool inMap = checkXY(user_x, mapX, user_y, mapY);
            // 만약 checkXY의 반환이 False라면 테이블에서의 좌표가 유효하지 않기 때문에
            //맵을 벗어난 것이다.
            if (inMap == false) {
                cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
                user_y += 1;
            }
            else {
                cout << "위로 한 칸 올라갑니다." << endl;
                displayMap(map, user_x, user_y); // 새롭게 정의된 맵 표기
            }
        }
        else if (user_input == "하") {
            // TODO: 아래로 한 칸 내려가기
            user_y += 1;
            // checkXY함수를 이용해서 좌표의 유효성 판별 -> bool형
            bool inMap = checkXY(user_x, mapX, user_y, mapY);
            // 만약 checkXY의 반환이 False라면 테이블에서의 좌표가 유효하지 않기 때문에
            //맵을 벗어난 것이다.
            if (inMap == false) {
                cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
                user_y -= 1;
            }
            else {
                cout << "위로 한 칸 내려갑니다." << endl;
                displayMap(map, user_x, user_y); // 새롭게 정의된 맵 표기
            }
        }
        else if (user_input == "좌") {
            // TODO: 왼쪽으로 이동하기
            user_x -= 1;
            // checkXY함수를 이용해서 좌표의 유효성 판별 -> bool형
            bool inMap = checkXY(user_x, mapX, user_y, mapY);
            // 만약 checkXY의 반환이 False라면 테이블에서의 좌표가 유효하지 않기 때문에
            // 맵을 벗어난 것이다.
            if (inMap == false) {
```

```
            cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
            user_x += 1;
        }
        else {
            cout << "왼쪽으로 이동합니다." << endl;
            displayMap(map, user_x, user_y); // 새롭게 정의된 맵 표기
        }
    }
    else if (user_input == "우") {
      // TODO: 오른쪽으로 이동하기
      user_x += 1;
      // checkXY함수를 이용해서 좌표의 유효성 판별 -> bool형
      bool inMap = checkXY(user_x, mapX, user_y, mapY);
      // 만약 checkXY의 반환이 False라면 테이블에서의 좌표가 유효하지 않기 때문에
      // 맵을 벗어난 것이다.
      if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_x -= 1;
      }
      else {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);// 새롭게 정의된 맵 표기
      }
    }
    // 상하좌우가 아닌 종료를 입력시 게임이 종료
    else if (user_input == "종료") {
        cout << "종료합니다." << endl;
        break;
    }
    // 상하좌우가 아닌 지도를 입력 시 지도 보여주는 기능
    else if (user_input == "지도") {
        // TODO: 지도 보여주기 함수 호출
        displayMap(map, user_x, user_y);
    }
    // 위의 상하좌우와 종료 외에 다른 입력값을 작성시 잘못된 입력입니다 출력
    else {
        cout << "잘못된 입력입니다." << endl;
        continue;
    }
```

1. 사용자에게 “상”, ”하”, ”좌”, ”우”, “종료”, ‘’지도’’ 중 하나를 입력 받기 & 이동 유효성 판단
2. 입력
    a. 사용자가 “상”, ”하”, ”좌”, ”우”, “종료”, ‘’지도’’ 중 하나를 입력
3. 결과
    a. 유저의 위치가 각각 방향으로 한 칸씩 이동하며 맵 업데이트
    b. 유효한 위치이면 “{방향}으로 한 칸 내려갑니다.” 메시지 출력

c. “종료’ 입력 시 “종료합니다.” 메시지 출력
d. ‘’지도” 입력 시 현재 맵의 상태 출력
e. 위의 5가지 입력이 아닌 다른 값을 입력할 시 “잘못된 입력입니다.” 메시지 출력

4. 반환값
    a. checkXY( )
        i. 위치가 유효하면 True 반환
        ii. 위치가 유효하지 않다면 False 반환
    b. displayMap( )
        i. 사용자의 위치 반환
        ii. 맵 위치에 따른 도메인 반환
5. 설명
    a. 사용자가 방향 중 하나를 입력할 때마다 유저 이동
    b. 유효한 위치인지 확인
    c. 결과에 따른 메시지 출력 후 맵 업데이트

```
user_hp -= 1; // 이동할 때마다 hp가 1 감소하기 위한 설정
```

1. 사용자가 이동할 때 마다 사용자 체력 1씩 감소
2. 입력
    a. user_hp : user의 hp를 담고 있는 변수
3. 결과
    a. 유저가 이동 후 hp 1 감소
4. 설명
    a. while문 안에 존재하며 “사용자에게 상, 하, 좌, 우, 종료 중 하나를 입력 받기“ 기능 아래에 코드를 작성하여 전역 변수 user_hp가 이동한 뒤 감소

```
// 만약 user_hp가 0이 되버리면 게임을 종료하고 아래와 같은 문장을 출력
        if (user_hp <= 0) {
            cout << "HP가 0 이하가 되었습니다. 실패했습니다." << endl;
            cout << "게임을 종료합니다." << endl;
            break;
        }
```

1. HP가 0이 되면 “실패”를 출력하고 종료
2. 입력
    a. user_hp : user의 hp를 담고 있는 변수
3. 결과 및 설명
    a. 이동과 적을 만나 hp가 0이 되면 종료 메시지 및 게임이 종료

```
// checkState 함수 호출하여 아래에 작성한 checkState 활성화
        checkState(map, user_x, user_y);
```

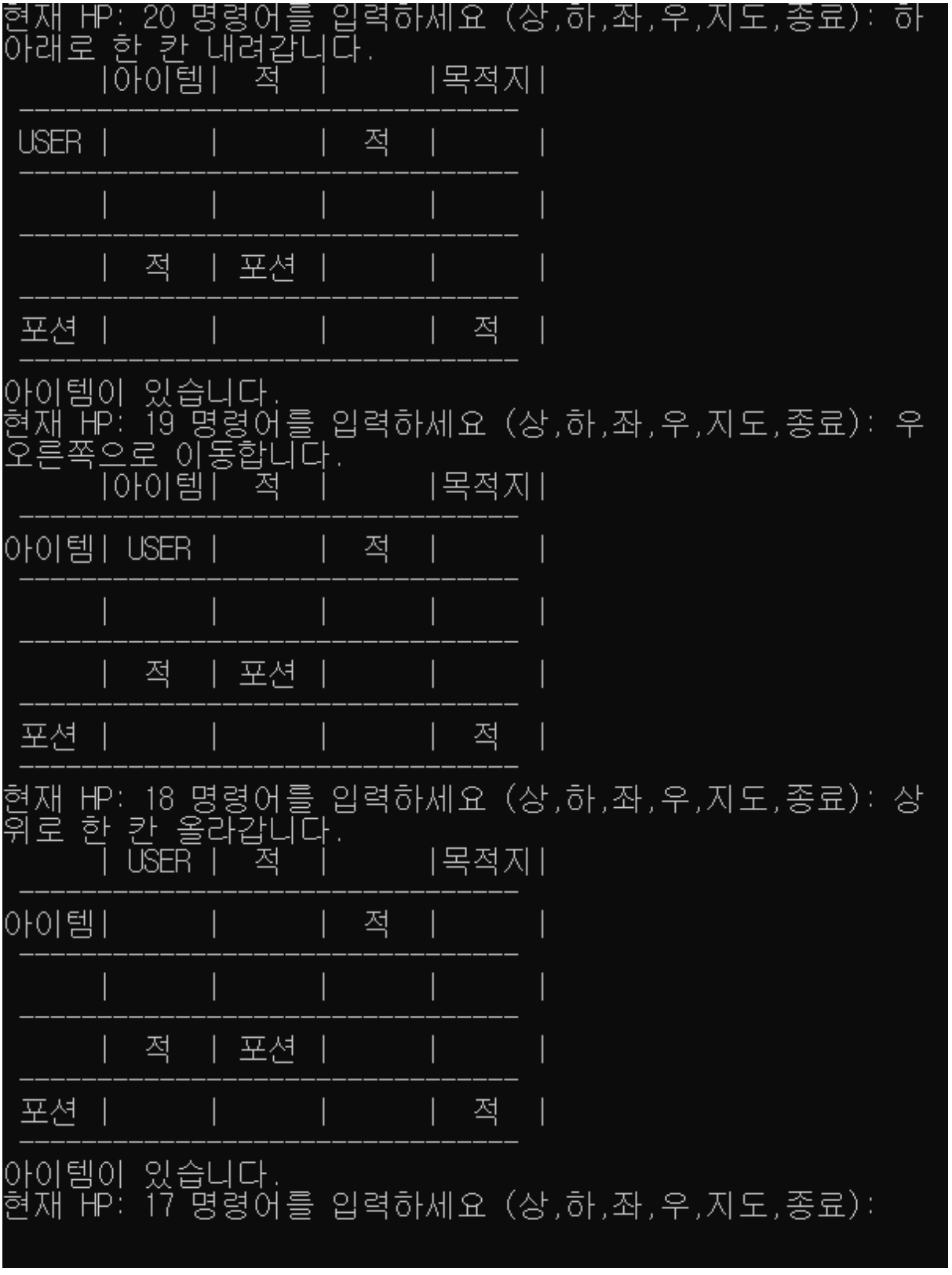
1. 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력
2. 입력
    a. checkState( ) 호출
3. 반환값
    a. postionState == 1 : “아이템” 메시지 출력
    b. positionState == 2 : “적” 메시지 출력 및 user_hp 2 감소
    c. positionState == 3 : “ 포션” 메시지 출력 및 user_hp 2 증가
4. 결과 및 설명
    a. positionState 변수 값에 따른 출력

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

1. 목적지에 도착하면 “성공”을 출력하고 종료
2. 입력
    a. checkGoal( ) 호출한 값을 가지는 finish 변수
3. 반환값
    a. 목적지 도착(4에 도달) : True 반환
    b. 목적지 도착x(4에 도달x) : False 반환
4. 결과
    a. "목적지에 도착했습니다! 축하합니다!", "게임을 종료합니다." 메시지 출력
    b. 게임 종료
5. 설명
    a. 게임에서 유저가 목적지에 도착했는지에 대한 여부를 확인하고, 도착이라면 게임을 종료

4. 테스트

- 기능 별 테스트 결과 : (요구사항 별 스크린샷)

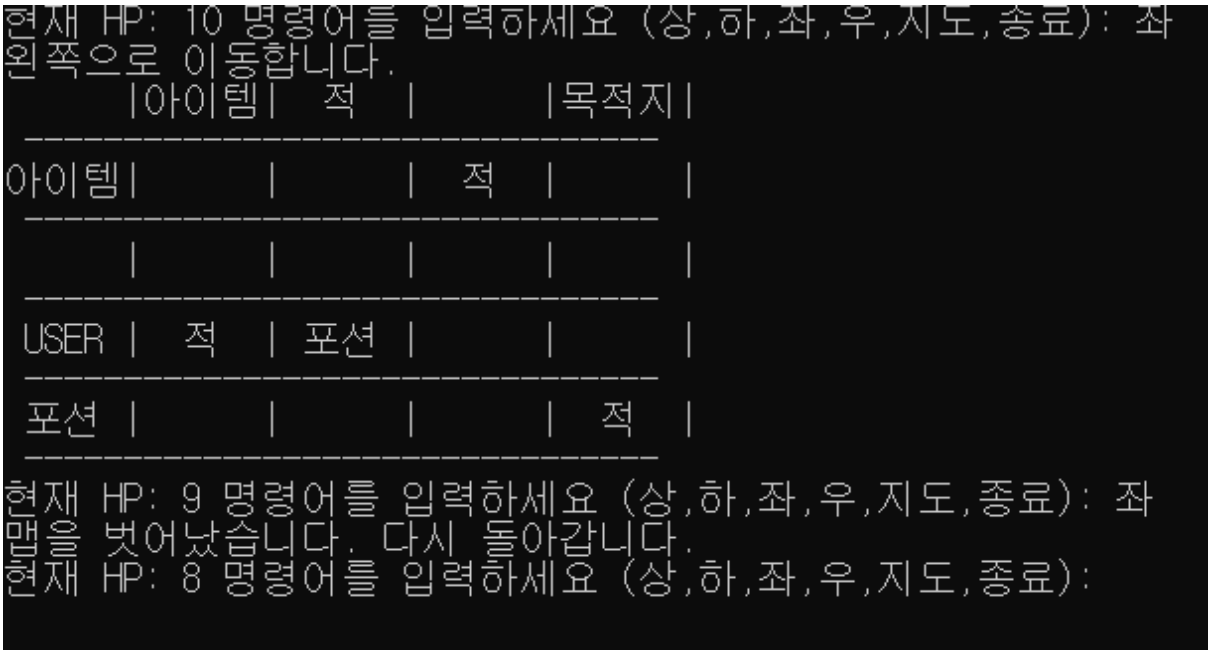


- 처음 명령문을 입력 받을 때 마다 HP 함께 출력
- 사용자에게 “상”, ”하”, ”좌”, ”우”, “종료”, ‘’지도’’ 중 하나를 입력 받기

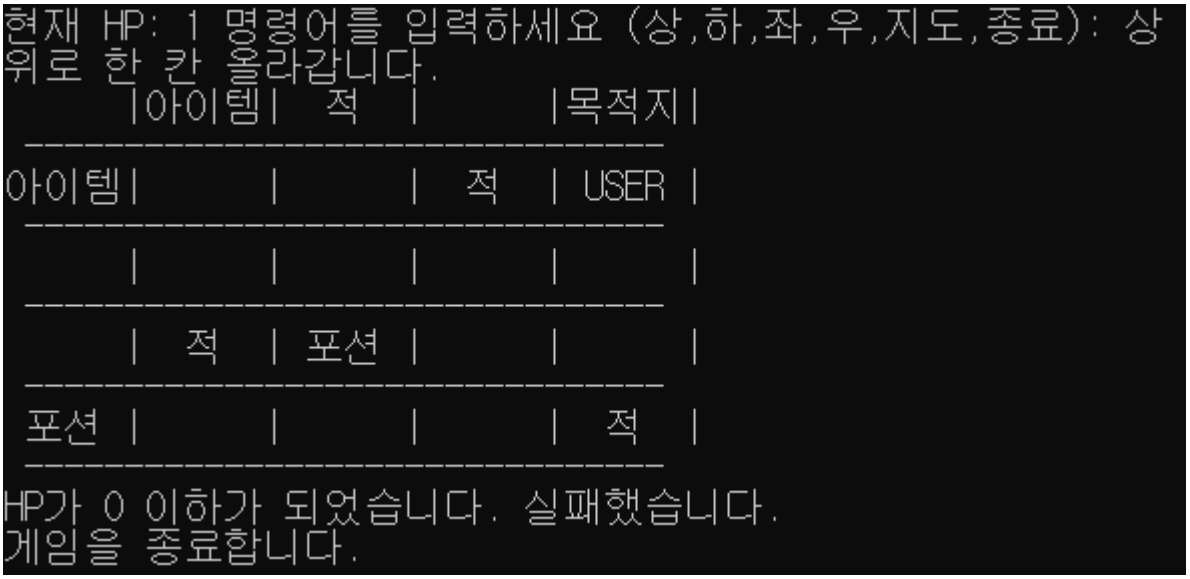
- 사용자가 이동할 때 마다 사용자 체력 1씩 감소

```
현재 HP: 16 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로 한 칸 내려갑니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      | USER |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
현재 HP: 15 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로 한 칸 내려갑니다.
      |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): 우
오른쪽으로 이동합니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | USER |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
포션이 있습니다. HP가 2 증가합니다.
현재 HP: 13 명령어를 입력하세요 (상,하,좌,우,지도,종료):
```

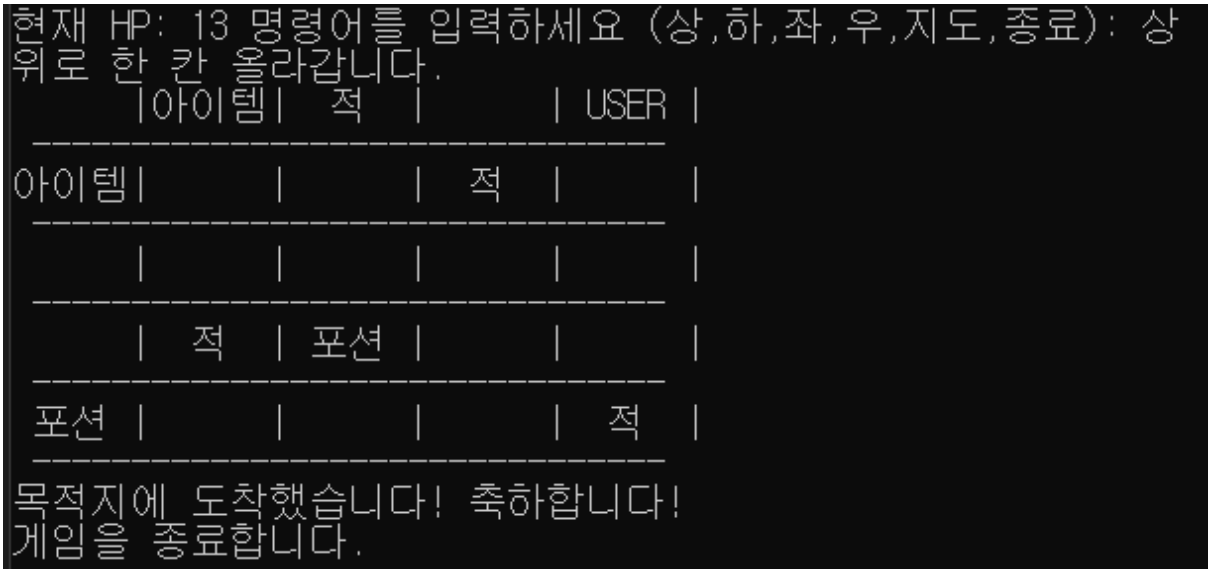
- 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력



- 이동 유효성 판단



- HP가 0이 되면 “실패”를 출력하고 종료



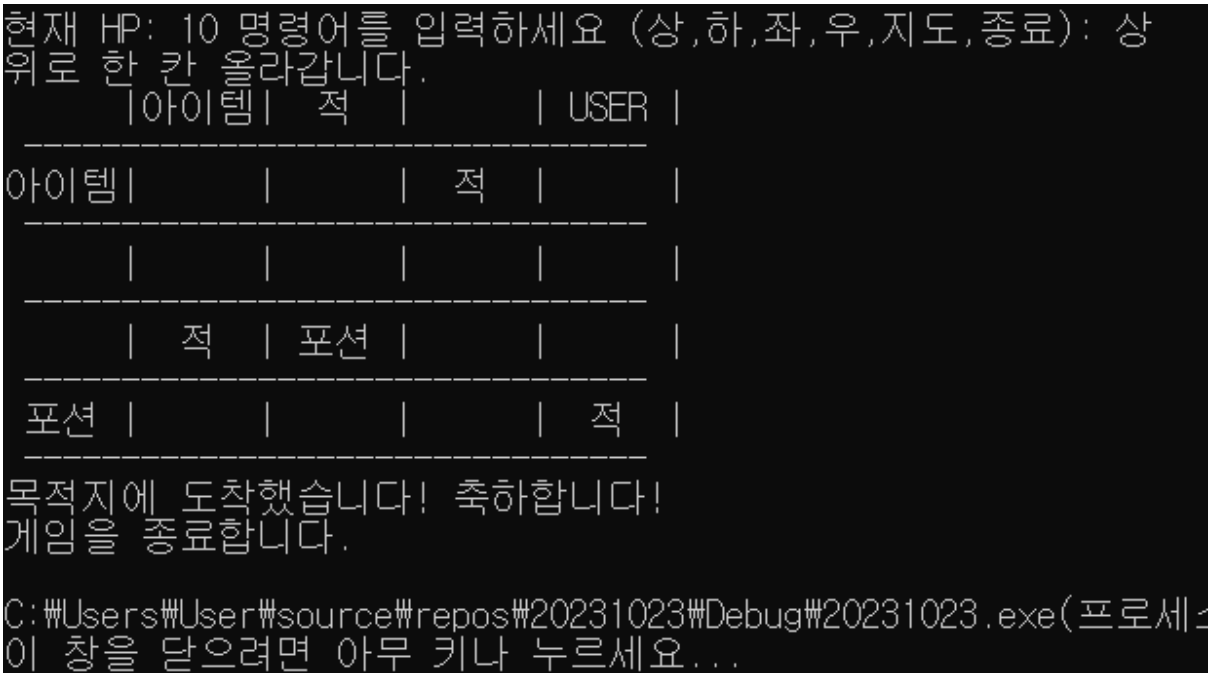
- 목적지에 도착하면 “성공”을 출력하고 종료

- 최종 테스트 스크린샷 : (프로그램 전체 동작 스크린샷)

```
현재 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로 한 칸 내려갑니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
 USER |      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
아이템이 있습니다.
현재 HP: 19 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로 한 칸 내려갑니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
 USER |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
현재 HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
아래로 한 칸 내려갑니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
 USER |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
```

```
현재 HP: 17 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
      |아이템|  적   |      |목적지|
 ---------------------------------
아이템|      |       |  적  |      |
 ---------------------------------
      |      |       |      |      |
 ---------------------------------
      | USER | 포션  |      |      |
 ---------------------------------
 포션 |      |       |      |  적  |
 ---------------------------------
적이 있습니다. HP가 2 줄어듭니다.
현재 HP: 14 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
      |아이템|  적   |      |목적지|
 ---------------------------------
아이템|      |       |  적  |      |
 ---------------------------------
      |      |       |      |      |
 ---------------------------------
      |  적  | USER  |      |      |
 ---------------------------------
 포션 |      |       |      |  적  |
 ---------------------------------
포션이 있습니다. HP가 2 증가합니다.
현재 HP: 15 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
      |아이템|  적   |      |목적지|
 ---------------------------------
아이템|      |       |  적  |      |
 ---------------------------------
      |      |       |      |      |
 ---------------------------------
      |  적  | 포션  | USER |      |
 ---------------------------------
 포션 |      |       |      |  적  |
 ---------------------------------
```

```
현재 HP: 14 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
      |아이템|  적  |      |목적지|
 ---------------------------------
아이템|      |      |  적  |      |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      | USER |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
현재 HP: 13 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
맵을 벗어났습니다. 다시 돌아갑니다.
현재 HP: 12 명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
위로 한 칸 올라갑니다.
      |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  |      |목적지|
 ---------------------------------
아이템|      |      |  적  | USER |
 ---------------------------------
      |      |      |      |      |
 ---------------------------------
      |  적  | 포션 |      |      |
 ---------------------------------
 포션 |      |      |      |  적  |
 ---------------------------------
```



5. 결과 및 결론

- 프로젝트 결과 : C++ 언어를 통해 mud game 게임을 구현

- 느낀 점

저번 Tic-Tac-Toe 게임 구현과는 달리 이번 mud game에서는 함수를 이용한 코드를 구현하는데 초점을 두었다. 함수를 이용하여 main( ) 함수에서 호출만 하면 기능을 사용할 수 있어 코드를 작성할 때에는 난이도가 있어도 막상 작성하고 나면 간결하다는 생각을 하였다. 기능에 대한 코드를 함수로 작성하였는데 추후에 클래스 개념을 학습한 후 게임 요소 도메인들을 클래스로 나누고, 클래스 내부에서 기능들을 함수로 호출하여 작성한다면 코드의 간결성이 더 높아질 것이라고 생각한다.