프로젝트형 실습 #02

<Mud Game 게임>

234042 황선하

① 서론

- 목표 : 7주차까지 배운 내용을 활용하여 간단한 Mud 게임을 구현한다.

② 요구사항

- 사용자 요구사항 : 유저가 상하좌우로만 이동하며 목적지에 도착하는 게임

- 기능 계획 :
  1. 사용자에게 "up", "down", "left", "right", "map", "end" 중 하나를 입력 받기
  2. Up/down/left/right 입력 시 해당 방향으로 이동 후 지도 출력
  3. Map 입력 시 전체 지도와 함께 현재 위치 출력
  4. 이 중 다른 것을 입력하면 메시지 출력 후 재 입력 요청
  5. 지도 밖으로 나가게 되면 에러 메시지 출력
  6. 목적지에 도착하면 "성공"을 출력하고 종료
  7. 체력 20을 가지고 게임을 시작하여 이동할 때마다 HP는 1 감소
  8. 처음 명령문을 입력 받을 때마다 HP 함께 출력
  9. HP가 0이 되면 실패를 알리는 문구와 함께 종료
  10. 적, 아이템/포션을 만났을 때 그에 대한 문구 출력하고 각각 HP 2감소, 2증가

- 함수 계획 :
  1. 지도와 사용자의 위치를 출력하는 함수
  2. 이동하려는 곳이 유효한 좌표인지 체크하는 함수
  3. 유저의 위치가 목적지인지 체크하는 함수
  4. 아이템/포션, 적을 만났을 때 그에 대한 문구를 출력하고 HP를 조절하는 함수
  5. 맵을 벗어났는지 확인하고 그에 대한 문구를 출력, 이동 값을 반환하는 함수

③ 설계 및 구현

1.
```
const int mapX = 5;
const int mapY = 5;
int hp = 20; // 현재(기본) HP값(함수에서의 사용을 위해 전역변수로 선언)
```

2. 입력
   - const int mapX = 지도의 x축 개수(2차원 배열의 2차원 값)
   - const int mapY = 지도의 y축 개수(2차원 배열의 1차원 값)
   - int hp = 기본 HP값

3. 설명
   - 세 가지 값을 전역변수로 선언하여 다른 함수들에서의 사용을 용이하게 한다.

1.
```
// 사용자 정의 함수
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 inMapFalse (bool inMap, int user_x, int user_y);
```

2. 입력
   - int user_x = 유저의 x값
   - int user_y = 유저의 y값
   - int mapX = 지도의 x축 개수(2차원 배열의 2차원 값)
   - int mapY = 지도의 y축 개수(2차원 배열의 1차원 값)
   - int map[][] = 전체 지도 배열
   - bool inMap = 이동한 좌표가 지도 안에 위치하는 지 확인

3. 설명
   - 맨 밑에서 정의되는 함수들을 main 함수 앞에서 미리 선언
   - checkXY : 이동하려는 곳이 유효한 좌표인지 체크
   - displayMap : 지도와 사용자의 위치 출력
   - checkGoal : 유저의 위치가 목적지인지 체크
   - checkState : 위치 별 상황에 따른 문구와 hp관리
   - inMapFalse : 맵을 벗어났는지 확인, 문구 출력

1.
```
// 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
```

2. 입력
    - int map[mapY][mapX] = 주어진 지도의 모습을 나타내는 배열
    - int user_x = 유저의 가로 위치를 저장할 변수
    - int user_y = 유저의 세로 위치를 저장할 변수
3. 설명
    - map에서 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지를 나타낸다.
    - while(1)을 통해 게임이 시작되기 전 선언되는 변수들

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

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

    // 새로운 이동 전에 HP값이 0이하인지 확인
    if (hp <= 0) {
        cout << "HP가 0 이하가 되었습니다. 실패했습니다." << endl;
        cout << "게임을 종료합니다.";
        break;
    }

    // 현재 HP값 출력 후 이동 명령어 받기
    cout << "현재 HP: " << hp << "  명령어를 입력하세요 (up,down,left,right,map,end): ";
    cin >> user_input;
```

2. 입력
    - string user_input = 사용자의 입력을 저장할 변수
    - int hp = 현재 HP값
3. 반환값
    - while문을 빠져나가기 위한 break

4. 결과

   - HP가 0이하인 경우 문구를 출력하고 while문을 빠져나간다(종료)
   - 현재 HP를 출력하고 다음에 실행할 명령어를 입력받는다.

5. 설명

   - HP가 0이하가 되면 break를 통해 while문을 빠져나간다.(게임 종료)
   - HP가 1이상인 경우 현재의 HP값을 출력하고 다음 명령어를 입력받는다.

1.
```
if (user_input == "up") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    // 다시 돌아가기 위한 값 받아오기
    int moveValue = inMapFalse(inMap, user_x, user_y);
    // 돌아가는 경우 (1), 그대로 이동하는 경우 (0)
    user_y += moveValue;
    if (moveValue == 0) {
        cout << "위로 한 칸 올라갑니다." << endl;
        // 이동에 따른 HP값 감소
        hp--;
        // 어떤 상태를 만났는지 판단 후 문구 출력, HP 증감
        checkState(map, user_x, user_y);
        displayMap(map, user_x, user_y);
    }
}
```

2. 입력

   - string user_input = 사용자의 입력을 저장할 변수
   - bool inMap = 이동한 좌표가 지도 안에 위치하는 지 확인
   - int user_x = 유저의 가로 위치를 저장할 변수
   - int user_y = 유저의 세로 위치를 저장할 변수
   - int moveValue = 다시 돌아가기 위한 값 저장
   - int map[mapY][mapX] = 주어진 지도의 모습을 나타내는 배열
   - int hp = 현재 HP값

3. 반환값

   - bool inMap : 맵을 벗어난 이동인지 확인 후 T/F 반환
   - inMapFalse : inMap에 따른 다시 돌아갈 값 0/1 반환
   - checkState : 어떤 상태인지 확인 후 hp 조절
   - displayMap : 지도 출력 (+ 현재 위치)

4. 결과

   - y축 값을 1 감소하여 위로 이동한다.

- 맵을 벗어나는지 확인 후 이동할 값(0/1)으로 y축 다시 이동
- 맵을 벗어나지 않는 경우 이동을 나타내는 문구 출력
- 이동에 따른 hp값 감소 후 상태에 따른 hp 조절
- 지도와 USER의 위치 출력

1.
```
else if (user_input == "down") {
    // TODO: 아래로 한 칸 내려가기
    user_y += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    // 다시 돌아가기 위한 값 받아오기
    int moveValue = inMapFalse(inMap, user_x, user_y);
    // 돌아가는 경우 (1), 그대로 이동하는 경우 (0)
    user_y -= moveValue;
    if (moveValue == 0) {
        cout << "아래로 한 칸 내려갑니다." << endl;
        // 이동에 따른 HP값 감소
        hp--;
        // 어떤 상태를 만났는지 판단 후 문구 출력, HP 증감
        checkState(map, user_x, user_y);
        displayMap(map, user_x, user_y);
    }
}
```

2. 입력

- string user_input = 사용자의 입력을 저장할 변수
- bool inMap = 이동한 좌표가 지도 안에 위치하는 지 확인
- int user_x = 유저의 가로 위치를 저장할 변수
- int user_y = 유저의 세로 위치를 저장할 변수
- int moveValue = 다시 돌아가기 위한 값 저장
- int map[mapY][mapX] = 주어진 지도의 모습을 나타내는 배열
- int hp = 현재 HP값

3. 반환값

- bool inMap : 맵을 벗어난 이동인지 확인 후 T/F 반환
- inMapFalse : inMap에 따른 다시 돌아갈 값 0/1 반환
- checkState : 어떤 상태인지 확인 후 hp 조절
- displayMap : 지도 출력 (+ 현재 위치)

4. 결과

- y축 값을 1 증가하여 아래로 이동한다.
- 맵을 벗어나는지 확인 후 이동할 값(0/1)으로 y축 다시 이동
- 맵을 벗어나지 않는 경우 이동을 나타내는 문구 출력

- 이동에 따른 hp값 감소 후 상태에 따른 hp 조절
- 지도와 USER의 위치 출력

1.
```
else if (user_input == "left") {
    // TODO: 왼쪽으로 이동하기
    user_x -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    // 다시 돌아가기 위한 값 받아오기
    int moveValue = inMapFalse(inMap, user_x, user_y);
    // 돌아가는 경우 (1), 그대로 이동하는 경우 (0)
    user_x += moveValue;
    if (moveValue == 0) {
        cout << "왼쪽으로 이동합니다." << endl;
        // 이동에 따른 HP값 감소
        hp--;
        // 어떤 상태를 만났는지 판단 후 문구 출력, HP 증감
        checkState(map, user_x, user_y);
        displayMap(map, user_x, user_y);
    }
}
```

2. 입력

- string user_input = 사용자의 입력을 저장할 변수
- bool inMap = 이동한 좌표가 지도 안에 위치하는 지 확인
- int user_x = 유저의 가로 위치를 저장할 변수
- int user_y = 유저의 세로 위치를 저장할 변수
- int moveValue = 다시 돌아가기 위한 값 저장
- int map[mapY][mapX] = 주어진 지도의 모습을 나타내는 배열
- int hp = 현재 HP값

3. 반환값

- bool inMap : 맵을 벗어난 이동인지 확인 후 T/F 반환
- inMapFalse : inMap에 따른 다시 돌아갈 값 0/1 반환
- checkState : 어떤 상태인지 확인 후 hp 조절
- displayMap : 지도 출력 (+ 현재 위치)

4. 결과

- x축 값을 1 감소하여 왼쪽으로 이동한다.
- 맵을 벗어나는지 확인 후 이동할 값(0/1)으로 x축 다시 이동
- 맵을 벗어나지 않는 경우 이동을 나타내는 문구 출력
- 이동에 따른 hp값 감소 후 상태에 따른 hp 조절
- 지도와 USER의 위치 출력

```
else if (user_input == "right") {
    // TODO: 오른쪽으로 이동하기
    user_x += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    // 다시 돌아가기 위한 값 받아오기
    int moveValue = inMapFalse(inMap, user_x, user_y);
    // 돌아가는 경우 (1), 그대로 이동하는 경우 (0)
    user_x -= moveValue;
    if (moveValue == 0) {
        cout << "오른쪽으로 이동합니다." << endl;
        // 이동에 따른 HP값 감소
        hp--;
        // 어떤 상태를 만났는지 판단 후 문구 출력, HP 증감
        checkState(map, user_x, user_y);
        displayMap(map, user_x, user_y);
    }
}
```

1.

2. 입력

   - string user_input = 사용자의 입력을 저장할 변수
   - bool inMap = 이동한 좌표가 지도 안에 위치하는 지 확인
   - int user_x = 유저의 가로 위치를 저장할 변수
   - int user_y = 유저의 세로 위치를 저장할 변수
   - int moveValue = 다시 돌아가기 위한 값 저장
   - int map[mapY][mapX] = 주어진 지도의 모습을 나타내는 배열
   - int hp = 현재 HP값

3. 반환값

   - bool inMap : 맵을 벗어난 이동인지 확인 후 T/F 반환
   - inMapFalse : inMap에 따른 다시 돌아갈 값 0/1 반환
   - checkState : 어떤 상태인지 확인 후 hp 조절
   - displayMap : 지도 출력 (+ 현재 위치)

4. 결과

   - x축 값을 1 증가하여 오른쪽으로 이동한다.
   - 맵을 벗어나는지 확인 후 이동할 값(0/1)으로 x축 다시 이동
   - 맵을 벗어나지 않는 경우 이동을 나타내는 문구 출력
   - 이동에 따른 hp값 감소 후 상태에 따른 hp 조절
   - 지도와 USER의 위치 출력

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

2. 입력
    - string user_input = 사용자의 입력을 저장할 변수
    - int user_x = 유저의 가로 위치를 저장할 변수
    - int user_y = 유저의 세로 위치를 저장할 변수
    - int map[][] = 주어진 지도의 모습을 나타내는 배열
3. 반환값
    - else if, displayMap : 현재 map의 모습과 USER의 위치를 출력
4. 결과
    - user_input으로 "map"이 들어올 경우 map와 USER의 위치가 출력됨
5. 설명
    - 명령어 입력 시 "map"을 입력하면 전체적인 map의 모습과 함께 USER의 현재 위치를 함께 출력한다.

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

2. 입력
    - string user_input = 사용자의 입력을 저장할 변수
3. 반환값
    - else if : while문을 빠져나가기 위한 break
4. 결과
    - user_input으로 "end"가 들어올 경우 종료 문구와 함께 while문 나가기
5. 설명
    - 명령어 입력 시 "end"를 입력하면 종료 문구를 출력하고 while문을 나가며 게임을 종료한다.

1.
```
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

2. 입력
    - "up", "down", "left", "right", "map", "end"외의 다른 값
3. 반환값
    - else : 반복문의 맨 위로 올라가기 위한 continue
4. 결과
    - "up", "down", "left", "right", "map", "end"외의 다른 값이 입력될 경우 문구를 출력하고 continue를 통해 while문의 맨 앞으로 이동
5. 설명
    - 주어진 명령어 외의 값이 들어올 경우 잘못된 입력임을 알리는 문구를 출력하고 continue를 통해 게임의 시작점으로 돌아가, 새로운 명령어를 입력받는다.

1.
```
// 목적지에 도달했는지 체크
bool finish = checkGoal(map, user_x, user_y);
if (finish == true) {
    cout << "목적지에 도착했습니다! 축하합니다!" << endl;
    cout << "게임을 종료합니다." << endl;
    break;
}
```

2. 입력
    - bool finish = 목적지에 도달했는지 확인
    - int user_x = 유저의 가로 위치를 저장할 변수
    - int user_y = 유저의 세로 위치를 저장할 변수
    - int map[][] = 주어진 지도의 모습을 나타내는 배열
3. 반환값
    - checkGoal : 목적지에 도달했는지 확인을 위한 T/F
    - if : 목적지에 도달했을 경우 while문을 나가기 위한 break

4. 결과

   - checkGoal함수의 반환값이 true인 경우 문구를 출력하고 while문을 빠져나감

5. 설명

   - checkGoal함수를 통해 목적지에 도달했음이 확인된 경우 축하문구와 종료 문구를 출력하고 break문을 통해 게임을 중단한다.

----------함수----------

1.
```
// 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;
    }
}
```

2. 입력

   - int user_x = 유저의 x값
   - int user_y = 유저의 y값
   - int mapX = 지도의 x축 개수(2차원 배열의 2차원 값)
   - int mapY = 지도의 y축 개수(2차원 배열의 1차원 값)
   - int map[][] = 전체 지도 배열

- int posState = 지도의 상태 변수값(0, 1, 2, 3, 4)

3. 반환값
   - displayMap : 맵의 형태와 각 칸 별 상태, USER의 위치 반환
4. 결과
   - map배열을 돌면서 map의 테두리와 칸 별 상태(아이템/포션, 적), USER의 위치를 출력한다.
5. 설명
   - map배열 속 값을 posState에 저장하면서 해당 위치의 상태에 따라 (switch)문을 통해 문구를 출력한다.
   - user의 위치를 map배열에서 확인하여 지도 속에 같이 나타나도록 한다.

1.
```
// 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;
}
```

2. 입력
   - int user_x = 유저의 x값
   - int user_y = 유저의 y값
   - int mapX = 지도의 x축 개수(2차원 배열의 2차원 값)
   - int mapY = 지도의 y축 개수(2차원 배열의 1차원 값)
   - bool checkFlag = 유효한 좌표인지 확인
3. 반환값
   - checkXY : 유효한 좌표인지 나타내는 checkFlag
4. 결과
   - 이동한 좌표가 map배열을 벗어나는 경우 checkFlag를 false에서 true로 변환한다.
   - checkFlag값을 반환하여 상황에 맞는 문구가 발생하도록 한다.
5. 설명

- 이동한 좌표가 지도 내를 벗어나는지 확인하고 변수 checkFlag를 반환하여 상황에 맞는 문구를 출력한다.(경고문 or 진행문)

1.
```
// 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;
}
```

2. 입력
   - int user_x = 유저의 x값
   - int user_y = 유저의 y값
   - int map[][] = 전체 지도 배열
3. 반환값
   - checkGoal : 목적지에 도착했는지 확인하는 T/F값
4. 결과
   - map의 현재 위치의 상태가 도착지를 의미하는 4일 경우 true를 반환
   - map의 현재 위치가 4외의 위치일 경우 false를 반환
5. 설명
   - 지도의 현재 위치를 목적지 상태 값(4)와 비교하여 도착을 의미하는 true를 반환하거나 false를 반환하도록 한다.

1.
```
// 아이템/포션, 적을 만났을 때 그에 대한 메세지 출력 및 HP 증감
void checkState(int map[][mapX], int user_x, int user_y) {
    if (map[user_y][user_x] == 1) {
        hp += 2;
        cout << "아이템이 있습니다. HP가 2 늘어납니다." << endl;;
    }
    else if (map[user_y][user_x] == 2) {
        hp -= 2;
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
    }
    else if (map[user_y][user_x] == 3) {
        hp += 2;
        cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;
    }
}
```

2. 입력
    - int user_x = 유저의 x값
    - int user_y = 유저의 y값
    - int map[][] = 전체 지도 배열
    - int hp = 기본 HP값
3. 반환값
    - if : 상태가 1인 경우 아이템 위치임과 HP 2증가 및 해당 문구 반환
    - else if(1) : 상태가 2인 경우 적 위치임과 HP 2감소 및 해당 문구 반환
    - else if(2) : 상태가 3인 경우 포션 위치임과 HP 2증가 및 해당 문구 반환
4. 결과
    - map에서의 현재 위치를 확인하고 상태 값을 확인하여 올바른 문구를 출력하고 hp값을 조절한다.
5. 설명
    - 지도에서의 현재 위치를 확인하고 상태 값(1, 2, 3)을 확인하여 해당되는 경우 전역변수 hp를 조절하고 문구를 출력한다.

1.
```cpp
// 맵을 벗어났는지 확인하고 그에 대한 메세지 출력
int inMapFalse (bool inMap, int user_x, int user_y) {
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        return 1;
    }
    return 0;
}
```

2. 입력
    - int user_x = 유저의 x값
    - int user_y = 유저의 y값
    - bool inMap = 이동한 좌표가 지도 안에 위치하는 지 확인
3. 반환값
    - inMapFalse : 돌아갈 이동 값 0/1
4. 결과
    - 이동한 좌표가 지도 안에 위치하는지 확인하는 변수 inMap이 false(밖에

위치)일 경우 문구를 출력하고 1을 반환

- 그 외의 경우 0을 반환

5. 설명

- 다시 돌아갈 값을 지도의 안/밖 상황에 따라 다르게 반환하고 맵을 벗어났을 경우에는 문구를 함께 출력

④ 테스트

1. 지도 밖으로 나가게 되면 에러메시지 출력

```
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end): up
맵을 벗어났습니다. 다시 돌아갑니다.
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end):
```

2. "map"을 입력하면 전체 지도와 함께 현재 위치 출력

```
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end): map
 USER |아이템|  적  |     |목적지|
 -------------------------------
아이템|     |     |  적  |     |
 -------------------------------
     |     |     |     |     |
 -------------------------------
     |  적  | 포션 |     |     |
 -------------------------------
 포션 |     |     |     |  적  |
 -------------------------------
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end):
```

3. "end"를 입력하면 게임 종료

```
현재 HP: 16  명령어를 입력하세요 (up,down,left,right,map,end): end
종료합니다.
PS C:\CPP2409>
```

4. 그 외의 값을 입력할 경우 경고 문구 출력

```
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end): side
잘못된 입력입니다.
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end):
```

5. 상태에 따라 hp가 증감되는 모습

```
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end): down
아래로 한 칸 내려갑니다.
아이템이 있습니다. HP가 2 늘어납니다.
     |아이템|  적  |     |목적지|
 -------------------------------
 USER |      |      |  적  |     |
 -------------------------------
      |      |      |      |     |
 -------------------------------
      |  적  | 포션 |      |     |
 -------------------------------
 포션 |      |      |      |  적  |
 -------------------------------
현재 HP: 21  명령어를 입력하세요 (up,down,left,right,map,end):
```

```
현재 HP: 19  명령어를 입력하세요 (up,down,left,right,map,end): right
오른쪽으로 이동합니다.
적이 있습니다. HP가 2 줄어듭니다.
     |아이템|  적  |     |목적지|
 -------------------------------
아이템|      |      | USER |     |
 -------------------------------
      |      |      |      |     |
 -------------------------------
      |  적  | 포션 |      |     |
 -------------------------------
 포션 |      |      |      |  적  |
 -------------------------------
현재 HP: 16  명령어를 입력하세요 (up,down,left,right,map,end):
```

6. 최종 테스트 스크린샷

```
현재 HP: 20  명령어를 입력하세요 (up,down,left,right,map,end): right
오른쪽으로 이동합니다.
아이템이 있습니다. HP가 2 늘어납니다.
     | USER |  적  |     |목적지|
 -------------------------------
아이템|      |      |  적  |     |
 -------------------------------
      |      |      |      |     |
 -------------------------------
      |  적  | 포션 |      |     |
 -------------------------------
 포션 |      |      |      |  적  |
 -------------------------------
현재 HP: 21  명령어를 입력하세요 (up,down,left,right,map,end): right
오른쪽으로 이동합니다.
적이 있습니다. HP가 2 줄어듭니다.
     |아이템| USER |     |목적지|
 -------------------------------
아이템|      |      |  적  |     |
 -------------------------------
      |      |      |      |     |
 -------------------------------
      |  적  | 포션 |      |     |
 -------------------------------
 포션 |      |      |      |  적  |
 -------------------------------
```

```
현재 HP: 18  명령어를 입력하세요 (up,down,left,right,map,end): right
오른쪽으로 이동합니다.
     |아이템|  적  | USER |목적지|
 -------------------------------
아이템|     |     |  적  |     |
 -------------------------------
     |     |     |     |     |
 -------------------------------
     |  적  | 포션 |     |     |
 -------------------------------
 포션 |     |     |     |  적  |
 -------------------------------
현재 HP: 17  명령어를 입력하세요 (up,down,left,right,map,end): right
오른쪽으로 이동합니다.
     |아이템|  적  |     | USER |
 -------------------------------
아이템|     |     |  적  |     |
 -------------------------------
     |     |     |     |     |
 -------------------------------
     |  적  | 포션 |     |     |
 -------------------------------
 포션 |     |     |     |  적  |
 -------------------------------
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
PS C:\CPP2409>
```

⑤ 결과 및 결론

1. 프로젝트를 통해 mud game을 만들어보았다. (특히 함수화를 통해 반복되는 부분을 단순화하였다.)

2. 느낀 점 : 대부분의 부분이 구현되어있던 상태에서 약간의 기능만 추가하는 프로젝트였기 때문에 보다 수월하게 문제를 해결할 수 있었다. 다만 반복되는 명령어를 함수화하는 과정에서 어떻게 하면 더 효율적인 함수를 만들 수 있을 지 고민하는 데에 오랜 시간이 걸렸다. 더 많은 코드에 대한 함수화를 공부해본다면 지금보다 더 효율적인 방법이 빠르게 떠오를 것이라고 생각한다. 또한 코드를 완성시키는 것 뿐만 아니라 직접 보고서를 작성해보면서 코드에 대한 전체적인 이해를 키울 수 있다는 점이 좋았다.