프로젝트 실습(mud_game) 보고서

소프트웨어공학과
190025 안호균

1. 서론

(1) 프로젝트 목적 및 배경: 1주차 ~ 7주차까지 배운 내용을 활용한 실습

(2) 목표: MUD 게임을 구현해보면서, 배운 내용을 직접 적용하며 복습하기

2. 요구사항

(1) 사용자 요구사항:

유저가 상하좌우로 이동하며 제한된 HP내에 목적지에 도착하는 게임

추가 기능 요구사항:

① 유저는 체력 20을 가지고 게임 시작

② 사용자가 이동할 때 마다 체력 1 감소(맵을 벗어나는 경우 제외)

③ 명령문을 입력 받기 직전 HP를 출력하여 사용자에게 안내

④ HP가 0 이하가 되면 실패 메세지를 출력하고 종료

⑤ 아이템, 적, 포션을 만났을 때에 대한 메시지를 출력

(2) 기능 계획:

① 사용자에게 좌표 이동(상, 하, 좌, 우), 지도, 종료 중 입력 받기

- 좌표 이동(상, 하, 좌, 우) 입력 시 좌표 이동 후 지도 출력
- 지도 입력 시 현재 위치를 지도에 표시하여 출력
- 존재하지 않는 명령 입력시 에러 메시지 및 재입력 요청 출력

② 지도에서 벗어날 시 오류 메시지 출력 후 재입력 요청

③ 제한된 HP 이내에 목적지에 도착하면 성공 메시지 출력

(3) 함수 계획

① 메인 함수: 사용자에게 입력을 받고, 입력에 따른 함수 호출

② displayMap: 지도에 현재 위치를 표시하여 출력하는 함수

③ checkXY: 사용자의 현재 위치를 체크하는 함수

④ checkGoal: 사용자가 목적지에 도착했는지 체크하는 함수

⑤ checkState: 사용자가 아이템, 적, 포션을 만났을 때의 상황 처리

3. 설계 및 구현

(1) 기능 별 구현 사항

- **기능 구현에 쓰일 상수와 변수 선언 및 초기화**

```
const int mapX = 5; // 지도의 가로축 크기
const int mapY = 5; // 지도의 세로축 크기

// 추가 기능 요구사항 1. 유저는 체력 20을 가지고 게임 시작
int user_hp = 20;
```

- **사용자 정의 함수에 대한 함수 원형**

```
// 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);
// 추가 기능 요구사항 5에 대한 함수 원형
void checkState(int map[][mapX], int user_x, int user_y);
```

- **게임을 플레이할 공간을 나타내는 2차원 배열 map 선언, 각 위치에 항목 추가**

```
// 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 좌표를 나타내는 user_x, user_y 변수 선언 및 초기화**

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

- **사용자에게 명령(상, 하, 좌, 우, 지도, 종료) 입력 받기**

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


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

    // 명령어를 입력할 것을 안내
    cout << "명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
    cin >> user_input;
```

▶ 입력

1. user_input = 사용자의 입력을 저장하는 변수

▶ 결과

1. 명령어를 입력 할 것을 안내
2. 사용자의 명령을 입력 받음

▶ 설명: 사용자의 입력을 저장하기 위해 string 타입 변수 user_input을 선언 및 초기화하고, 사용자의 명령을 입력 받아 user_input에 저장한다.

- **사용자에게 명령으로 “상”을 입력 받은 경우 처리**

```
if (user_input == "상") { // 사용자가 입력으로 "상"을 입력했다면
    // 위로 한 칸 올라가기
    user_y -= 1; // 사용자의 y 좌표를 -1만큼 변경하여 위치를 위로 올림
    bool inMap = checkXY(user_x, mapX, user_y, mapY); // 사용자의 위치에 대한 유효성을 bool 변수로 저장함
    // 지도를 벗어났을 시 에러 메세지 출력 후 위치 복구
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y += 1;
    }
    // 정상적인 입력이라면 위치 이동
    else {
        cout << "위로 한 칸 올라갑니다." << endl;
        displayMap(map, user_x, user_y);
        // 추가 기능 요구사항 5. 아이템, 적, 포션이 있을 때 출력
        checkState(map, user_x, user_y);
        // 추가 기능 요구사항 2, 3. 이동할 때마다 체력 1씩 감소, HP 출력
        user_hp--;
        cout << "현재 HP: " << user_hp << " ";
    }
}
```

▶ 입력

1. user_input = 사용자의 입력
2. user_y = 사용자의 y 좌표
3. inMap = 사용자의 위치에 대한 유효성을 나타내는 변수
4. user_hp = 사용자의 현재 체력

▶ 결과

1. 사용자의 위치를 한 칸 위로 올림
2. 변경된 위치에 대한 유효성 체크
3. 변경된 위치가 유효하지 않다면 오류 메시지 출력 후 위치 원복
4. 정상적인 입력이라면 위치 이동 및 안내 메시지, 지도 출력 후 체력 표시

▶ 설명: 입력 받은 사용자의 입력 user_input 값이 “상”이라면 사용자의 y 좌표를 -1만큼 변경하여 위치를 한 칸 위로 올린다. 변경된 좌표를 checkXY 함수를 통해 유효성을 체크하여 유효하지 않은 위치라면 오류 메시지를 출력한 후 위치를 원복하고, 정상적인 이동이라면 위치를 이동한 후 displayMap 함수를 통해 이동된 위치를 지도에 표시하여 출력한다. 또한, checkState 함수를 통해 위치를 이동하며 아이템, 적, 포션을 마주쳤을 경우에 대한 처리를 해준다. 모든 과정이 끝나면, 이동을 마쳤으므로 hp를 1만큼 감소하여 현재 HP를 출력한다.

● **사용자에게 명령으로 “하”를 입력 받은 경우 처리**

```
else if (user_input == "하") { // 사용자가 입력으로 "하"를 입력했다면
    // 아래로 한 칸 내려가기
    user_y += 1; // 사용자의 y 좌표를 +1만큼 변경하여 위치를 아래로 옮김
    bool inMap = checkXY(user_x, mapX, user_y, mapY); // 사용자의 위치에 대한 유효성을 bool 변수로 저장함
    // 지도를 벗어났을 시 에러 메세지 출력 후 위치 복구
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y -= 1;
    }
    // 정상적인 입력이라면 위치 이동
    else {
        cout << "아래로 한 칸 내려갑니다." << endl;
        displayMap(map, user_x, user_y);
        // 추가 기능 요구사항 5. 아이템, 적, 포션이 있을 때 출력
        checkState(map, user_x, user_y);
        // 추가 기능 요구사항 2, 3. 이동할 때마다 체력 1씩 감소, HP 출력
        user_hp--;
        cout << "현재 HP: " << user_hp << " ";
    }
}
```

▶ 입력

1. user_input = 사용자의 입력
2. user_y = 사용자의 y 좌표
3. inMap = 사용자의 위치에 대한 유효성을 나타내는 변수
4. user_hp = 사용자의 현재 체력

▶ 결과

1. 사용자의 위치를 한 칸 아래로 내림
2. 변경된 위치에 대한 유효성 체크
3. 변경된 위치가 유효하지 않다면 오류 메시지 출력 후 위치 원복
4. 정상적인 입력이라면 위치 이동 및 안내 메시지, 지도 출력 후 체력 표시

▶ 설명: 입력 받은 사용자의 입력 user_input 값이 “하”라면 y 좌표를 1만큼 변경하여 위치를 한 칸 아래로 내린다. 이후 실행되는 checkXY를 통한 유효성 체크 및 오류 메시지 표시, 정상적인 이동일 때 displayMap 함수를 통한 지도 출력, checkState 함수를 통한 상태 변화 처리, 현재 HP 출력 등 모든 동작은 앞서 설명한 방식과 동일하게 실행된다.

- **사용자에게 명령으로 “좌”를 입력 받은 경우 처리**

```
else if (user_input == "좌") { // 사용자가 입력으로 "좌"를 입력했다면
    // 왼쪽으로 이동하기
    user_x -= 1; // 사용자의 x 좌표를 -1만큼 변경하여 위치를 왼쪽으로 이동함
    bool inMap = checkXY(user_x, mapX, user_y, mapY); // 사용자의 위치에 대한 유효성을 bool 변수로 저장함
    // 지도를 벗어났을 시 에러 메세지 출력 후 위치 복구
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_x += 1;
    }
    // 정상적인 입력이라면 위치 이동
    else {
        cout << "왼쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        // 추가 기능 요구사항 5. 아이템, 적, 포션이 있을 때 출력
        checkState(map, user_x, user_y);
        // 추가 기능 요구사항 2, 3. 이동할 때마다 체력 1씩 감소, HP 출력
        user_hp--;
        cout << "현재 HP: " << user_hp << " ";
    }
}
```

▶ 입력

1. user_input = 사용자의 입력
2. user_x = 사용자의 x 좌표
3. inMap = 사용자의 위치에 대한 유효성을 나타내는 변수
4. user_hp = 사용자의 현재 체력

▶ 결과

1. 사용자의 위치를 한 칸 왼쪽으로 변경
2. 변경된 위치에 대한 유효성 체크
3. 변경된 위치가 유효하지 않다면 오류 메시지 출력 후 위치 원복
4. 정상적인 입력이라면 위치 이동 및 안내 메시지, 지도 출력 후 체력 표시

▶ 설명: 입력 받은 사용자의 입력 user_input 값이 “좌”라면 x 좌표를 -1만큼 변경하여 위치를 한 칸 왼쪽으로 바꾼다. 이후 실행되는 checkXY를 통한 유효성 체크 및 오류 메시지 표시, 정상적인 이동일 때 displayMap 함수를 통한 지도 출력, checkState 함수를 통한 상태 변화 처리, 현재 HP 출력 등 모든 동작은 앞서 설명한 방식과 동일하게 실행된다.

● **사용자에게 명령으로 “우”를 입력 받은 경우 처리**

```
else if (user_input == "우") { // 사용자가 입력으로 "우"를 입력했다면
    // 오른쪽으로 이동하기
    user_x += 1; // 사용자의 x 좌표를 +1만큼 변경하여 위치를 오른쪽으로 이동함
    bool inMap = checkXY(user_x, mapX, user_y, mapY); // 사용자의 위치에 대한 유효성을 bool 변수로 저장함
    // 지도를 벗어났을 시 에러 메세지 출력 후 위치 복구
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_x -= 1;
    }
    // 정상적인 입력이라면 위치 이동
    else {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        // 추가 기능 요구사항 5. 아이템, 적, 포션이 있을 때 출력
        checkState(map, user_x, user_y);
        // 추가 기능 요구사항 1-2, 1-3. 이동할 때마다 체력 1씩 감소, HP 출력
        user_hp--;
        cout << "현재 HP: " << user_hp << " ";
    }
}
```

▶ 입력

1. user_input = 사용자의 입력
2. user_x = 사용자의 x 좌표
3. inMap = 사용자의 위치에 대한 유효성을 나타내는 변수
4. user_hp = 사용자의 현재 체력

▶ 결과

1. 사용자의 위치를 한 칸 오른쪽으로 변경
2. 변경된 위치에 대한 유효성 체크
3. 변경된 위치가 유효하지 않다면 오류 메시지 출력 후 위치 원복
4. 정상적인 입력이라면 위치 이동 및 안내 메시지, 지도 출력 후 체력 표시

▶ 설명: 입력 받은 사용자의 입력 user_input 값이 “우”라면 x 좌표를 1만큼 변경하여 위치를 한 칸 오른쪽으로 바꾼다. 이후 실행되는 checkXY를 통한 유효성 체크 및 오류 메시지 표시, 정상적인 이동일 때 displayMap 함수를 통한 지도 출력, checkState 함수를 통한 상태 변화 처리, 현재 HP 출력 등 모든 동작은 앞서 설명한 방식과 동일하게 실행된다.

● **사용자에게 명령으로 “지도”를 입력 받은 경우 처리**

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

▶ 입력

1. user_input = 사용자의 입력

▶ 결과

1. 지도상에 나타나는 사용자의 현재 위치를 출력

▶ 설명: displayMap 함수를 통해 사용자의 현재 위치를 지도에 표시하여 출력한다.

- **사용자에게 명령으로 “종료”를 입력 받은 경우 처리**

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

▶ 입력

1. user_input = 사용자의 입력

▶ 결과

1. 종료 메시지 출력

▶ 설명: 사용자의 입력이 “종료”라면 종료를 위해 while문을 탈출한다.

- **사용자에게 명령으로 존재하지 않는 입력을 받은 경우 처리**

```
else {
    // 에러 메세지 출력 후 재입력 받도록 continue
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

▶ 입력

1. user_input = 사용자의 입력(존재하지 않는 명령이 입력된 경우)

▶ 결과

1. 오류 메시지 출력

▶ 설명: 사용자의 입력이 존재하지 않는 입력이라면 오류 메시지를 출력하고, 명령 재입력을 위해 continue를 실행한다.

- **사용자의 HP가 0 이하가 된 경우 처리**

```
// 추가 기능 요구사항 4. HP가 0 이하가 되면 "실패"를 출력하고 종료
if (user_hp <= 0) {
    cout << "HP가 0 이하가 되었습니다. 실패했습니다." << endl;
    cout << "게임을 종료합니다." << endl;
    return 0;
}
```

▶ 입력

1. user_hp = 사용자의 현재 체력

▶ 결과

1. 실패 메시지 출력
2. 게임 종료 메시지 출력

▶ 설명: 사용자의 체력이 0 이하가 되면, 게임에 실패한 것이므로 실패 메시지를 출력한 후, 게임 종료 메시지를 출력하며 프로그램을 종료한다.

- **사용자가 목적지에 도달한 경우 처리**

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

▶ 입력

1. finish = 사용자가 목적지에 도달했는지 여부

▶ 결과

1. 사용자가 목적지에 도달한 것으로 확인되면 축하 메시지 출력
2. 게임 종료 메시지 출력

▶ 설명: checkGoal 함수를 통해 사용자가 목적지 좌표에 도달한 것이 확인되면 bool 타입 변수 finish에 값을 저장하고, 목적지에 도달했을 시 축하 메시지와 게임 종료메시지를 출력하며 반복문에서 break하여 프로그램을 종료한다.

- **특정 조건에 의해 무한 반복 while문을 벗어난 경우 처리**

```
    }
    return 0;
}
```

▶ 입력

없음

▶ 결과

1. 프로그램 종료

▶ 설명: 입력을 받기 위해 무한 반복하는 while문을 벗어난 경우, 이후 바로 return 0을 통해 프로그램을 종료하도록 한다.

● 사용자 정의 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++) {
        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

1. map = 지도를 나타내는 2차원 배열
2. user_x = user의 x 좌표
3. user_y = user의 y 좌표

▶ 결과

1. 형식에 맞춰 지도를 표시

▶ 설명: 함수의 매개변수로 지도를 나타내는 2차원 배열 map, user의 x 좌표 user_x, user의 y 좌표 user_y를 건네받아, map 배열의 해당 인덱스의 값에 맞추어 지도를 표시한다. (0이라면 공백 표시, 1이라면 아이템 표시, 2라면 적 표시, 3이라면 포션 표시, 4라면 목적지 표시)

● 사용자 정의 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

1. user_x = user의 x 좌표
2. mapX = 지도의 x 좌표
3. user_y = user의 y 좌표
4. mapY = 지도의 y 좌표
5. checkFlag = 좌표의 유효성

▶ 결과

1. 해당 좌표의 유효성 반환

▶ 설명: 함수의 매개 변수로 user의 x 좌표, 지도의 x 좌표, user의 y 좌표, 지도의 y 좌표를 건네받아 사용자의 이동이 지도를 벗어나는지를 검사하고, 벗어난다면 checkFlag를 false로, 벗어나지 않는다면 checkFlag를 true로 설정하여 checkFlag 값을 반환한다.

● 사용자 정의 함수 checkGoal

```
// 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

1. map = 지도를 나타내는 2차원 배열
2. user_x = user의 x 좌표
3. user_y = user의 y 좌표

▶ 결과

1. 해당 좌표가 목적지 여부를 반환

▶ 설명: 사용자가 이동한 좌표에 '4' 값이 들어있다면, 목적지에 도착한 것이므로 true를 반환하고, 이 경우가 아니라면 false를 반환하여 목적지 여부를 판별한다.

- **사용자 정의 함수 checkState**

```
// 추가 기능 요구사항 5. 아이템, 적, 포션이 있을 때 출력하는 함수
void checkState(int map[][mapX], int user_x, int user_y) {

    if (map[user_y][user_x] == 1) { // 아이템이 있을 때
        cout << "아이템이 있습니다." << endl;
    }
    else if (map[user_y][user_x] == 2) { // 적이 있을 때
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
        user_hp -= 2;
    }
    else if (map[user_y][user_x] == 3) { // 포션이 있을 때
        cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;
        user_hp += 2;
    }
}
```

▶ 입력

1. map = 지도를 나타내는 2차원 배열
2. user_x = user의 x 좌표
3. user_y = user의 y 좌표

▶ 결과

1. 사용자가 도착한 좌표에 들어있는 값에 따른 상황별 메시지 출력 및 HP 변화

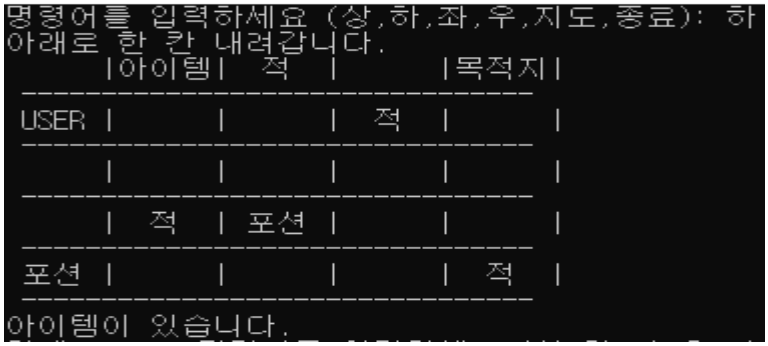
▶ 설명: 사용자가 이동한 좌표에 '1' 값이 들어있다면, 아이템을 만난 것이므로 아이템을 만났다는 메시지를 출력한다. '2' 값이 들어있다면, 적을 만난 것이므로 적을 만났다는 메시지를 출력한 후, hp를 2만큼 감소시킨다. '3' 값이 들어있다면, 포션을 만난 것이므로 포션을 만났다는 메시지를 출력한 후, hp를 2만큼 증가시킨다.

4. 테스트

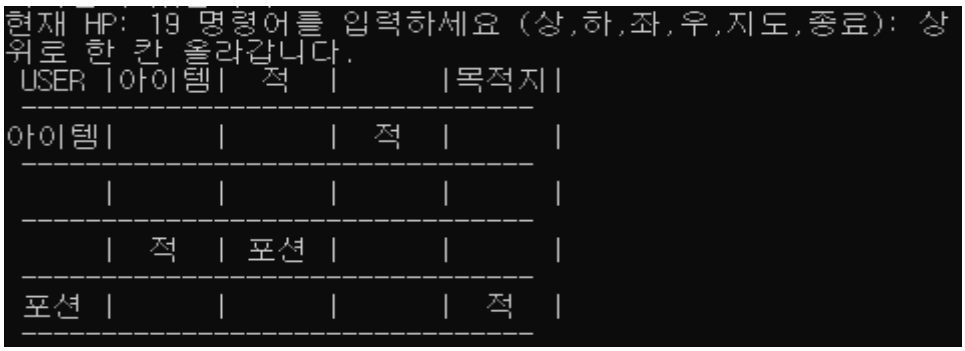
(1) 기능별 테스트 결과

① 사용자에게 좌표 이동(상, 하, 좌, 우), 지도, 종료 중 입력 받기

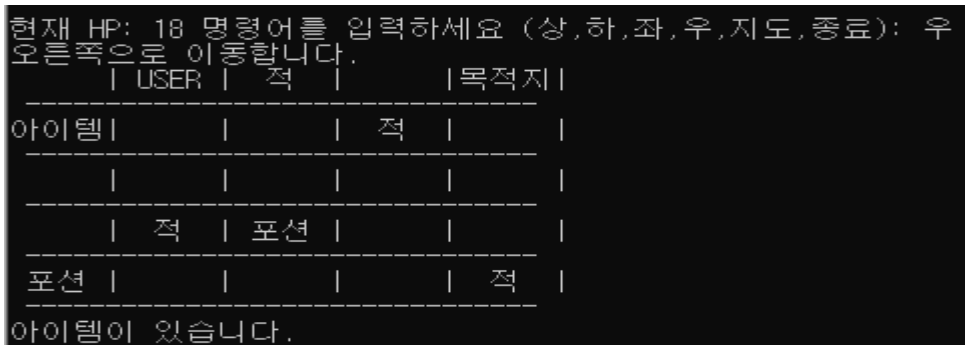
<입력으로 “하”를 입력 받은 경우>



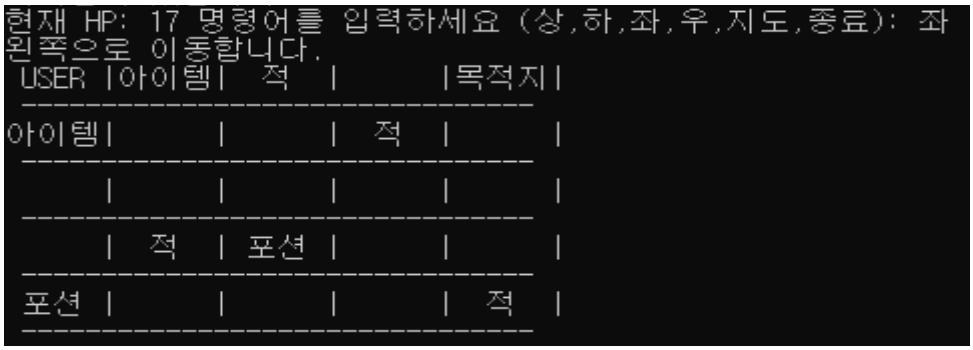
<입력으로 “상”을 입력 받은 경우>



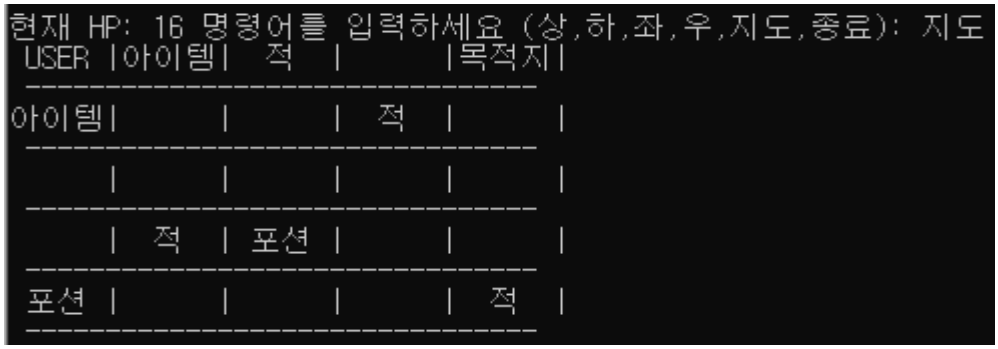
<입력으로 “우”를 입력 받은 경우>



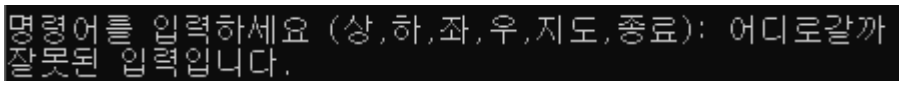
<입력으로 “좌”를 입력 받은 경우>



<입력으로 “지도”를 입력 받은 경우>

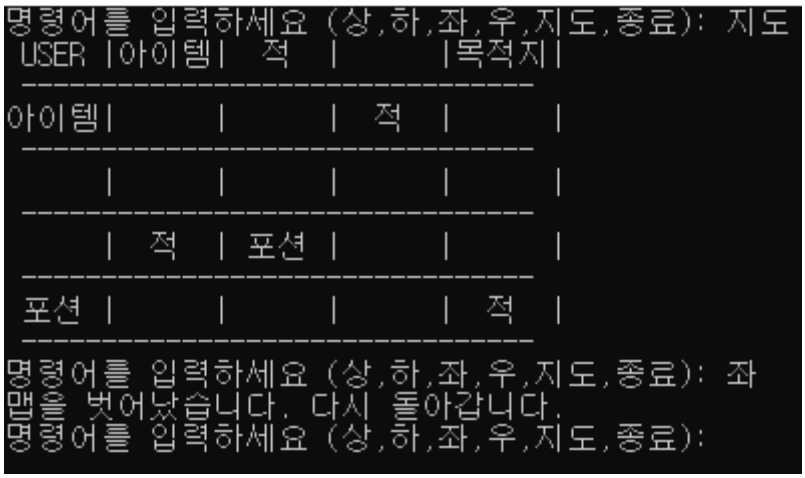


<입력으로 존재하지 않는 명령어를 입력 받은 경우>



<입력으로 “종료”를 입력 받은 경우>

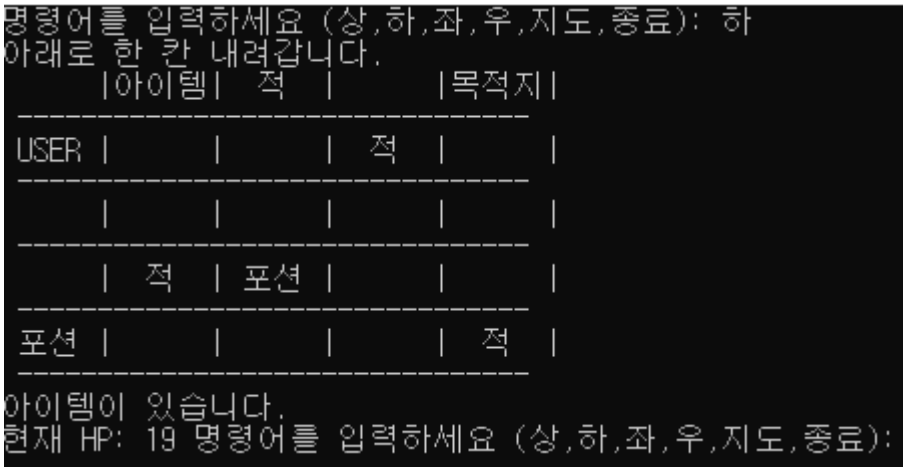
② 지도에서 벗어날 시 오류 메시지 출력 후 재입력 요청



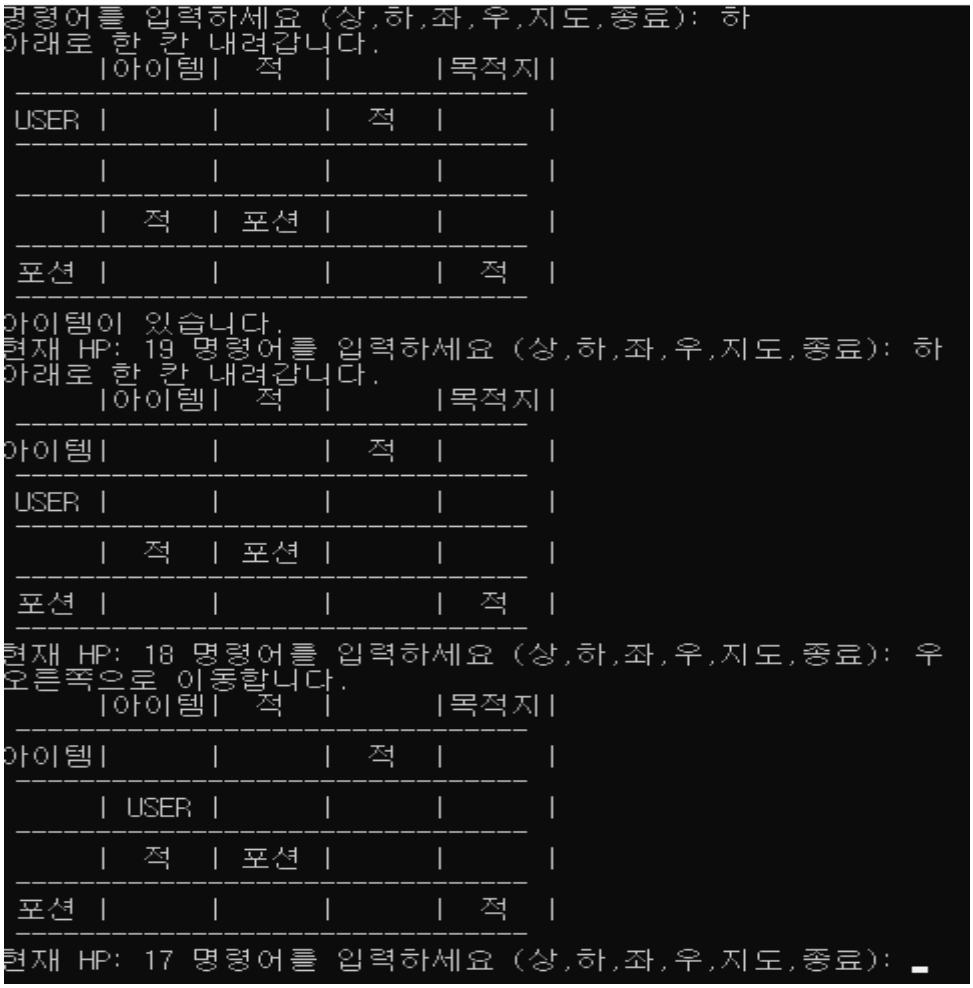
③ 제한된 HP 이내에 목적지에 도착하면 성공 메시지 출력

```
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): 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): 우
오른쪽으로 이동합니다.
      |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  |
 ---------------------------------
현재 HP: 14 목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

추가 기능 요구사항 ① 유저는 체력 20을 가지고 게임 시작

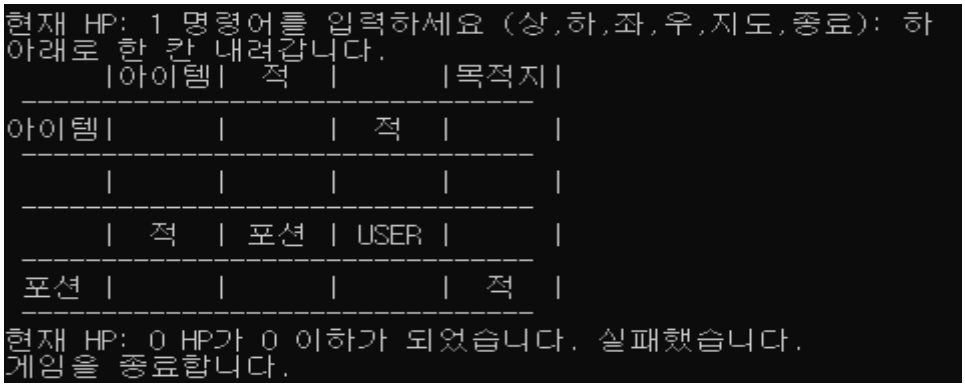


추가 기능 요구사항 ② 사용자가 이동할 때 마다 체력 1 감소(맵을 벗어나는 경우 제외)



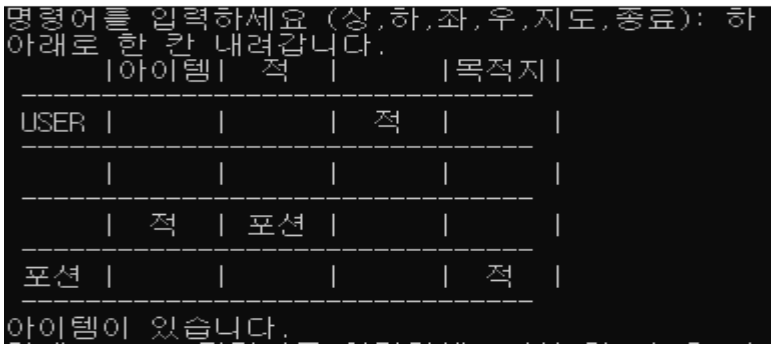
추가 기능 요구사항 ③ 명령문을 입력 받기 직전 HP를 출력하여 사용자에게 안내

추가 기능 요구사항 ④ HP가 0 이하가 되면 실패 메세지를 출력하고 종료

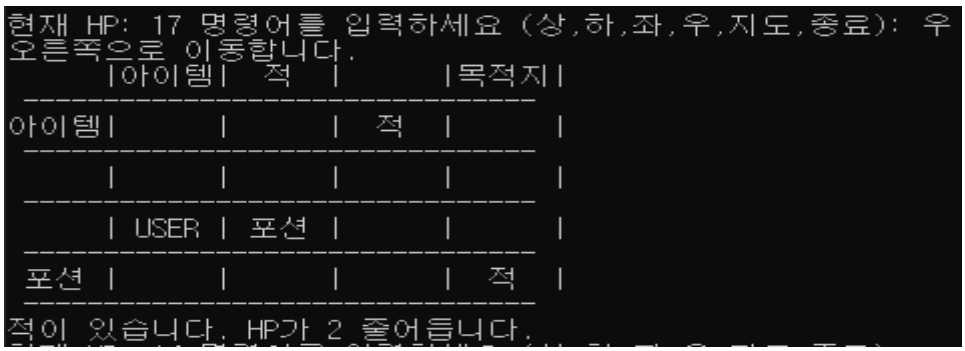


추가 기능 요구사항 ⑤ 아이템, 적, 포션을 만났을 때에 대한 메시지를 출력

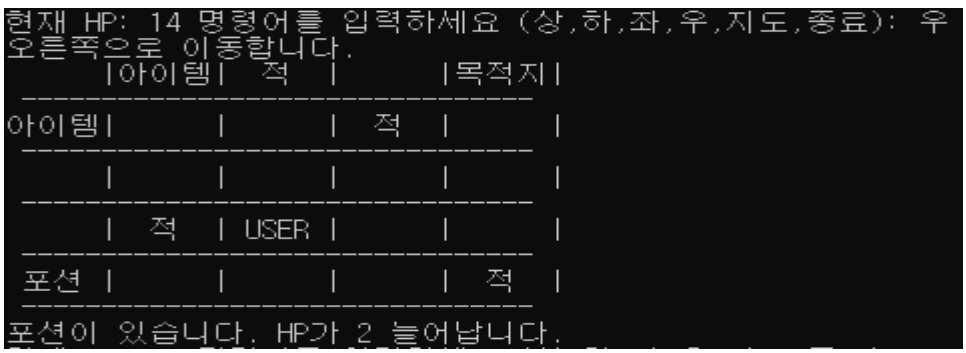
<아이템을 만난 경우>



<적을 만난 경우>



<포션을 만난 경우>



5. 결과 및 결론

1. 프로젝트 결과: 7주차까지 배운 내용을 응용해 간단한 MUD 게임을 제작했다.

2. 느낀 점: 지난 프로젝트처럼 배운 내용을 토대로 간단한 게임을 구현하며 지금까지 수업 시간에 배운 모든 내용을 종합적으로 활용 해볼 수 있었다. 다양한 자료형과 변수를 다루며 입력을 받아 그에 따른 출력을 해보고, 입력을 함수를 통해 가공하여 알맞은 값을 도출하기도 했다. 또한, 조건문과 반복문을 적극적으로 활용했고, 프로그램에 진행에 따라 변화하는 값을 2차원 배열을 통해 관리하며 프로젝트의 목표에 도달할 수 있었다. 내가 배운 이론을 실습을 통해 직접 익힐 수 있었던 좋은 기회였다.