I 서론

7주차까지 배운 내용에 대한 실습과 추후 기말 프로젝트 진행에 대한 힌트를 얻기 위해 Mud 게임 구현을 진행했습니다.

II 요구사항

사용자 요구사항

게임 목표: 상하좌우로 이동하며 무사히 목적지에 도착하기

맵: 2차원 배열 (5 x 5) (코드 내에 미리 선언 후 초기화)

기능 계획

1. 사용자에게 “상”, ”하”, ”좌”, ”우”, “지도”, “종료” 중 하나를 입력 받기
   - 상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력
   - “지도”를 입력하면 전체 지도와 함께 현재 위치를 출력
   - 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청
2. 지도 밖으로 나가게 되면 에러 메시지 출력
3. 목적지에 도착하면 “성공”을 출력하고 종료

추가 기능 요구사항

4.. 유저는 체력 20을 가지고 게임 시작

5. 사용자가 이동할 때 마다 사용자 체력 1씩 감소

6. 처음 명령문을 입력 받을 때 마다 HP 함께 출력

7. HP가 0이 되면 “실패”를 출력하고 종료

8. 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

포션을 만날 경우 HP가 2가 늘어나고 그에 대한 추가 메시지 출력

- (적이나 포션 등은 사라지지 않음을 전제)

함수 계획

9.void checkState(int map[][mapX], int user_x, int user_y): 함수 생성 후 호출

10.계속 반복되는 bool inMap 함수를 최적화

11. 메인 함수: 사용자에게 값을 계속 입력받고, 그에 대한 함수 호출

12. 지도와 현재 위치 출력 함수: displayMap()

13. 사용자 위치 체크 함수: checkXY()

14. 목적지에 도착 체크 함수: checkGoal()

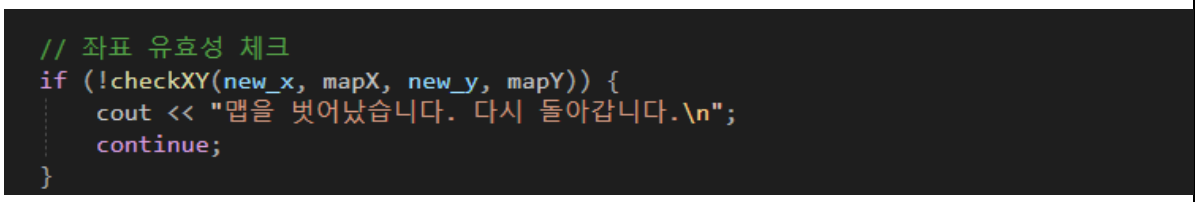
III 설계 및 구현

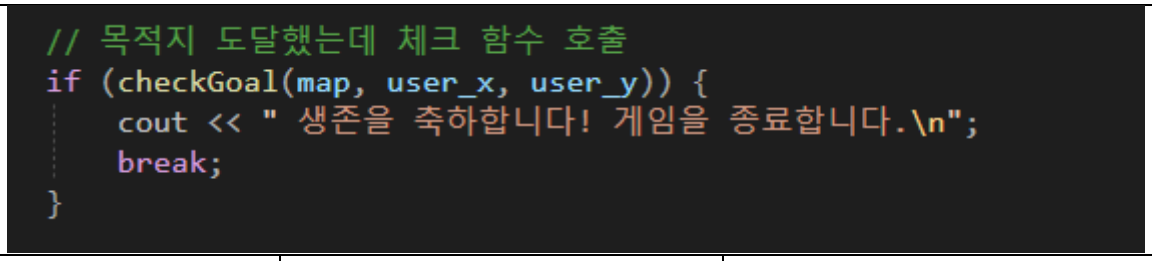
위의 순서에 따라 표로 나타냈습니다.

	코드블록 스크린샷
1	

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

while (1) {
    //사용자에게 계속 입력받으며 루프
    cout << "HP: " << hp << "\n명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
    string user_input = "";
    cin >> user_input;// 사용자의 입력 저장

    int new_x = user_x, new_y = user_y; // new_x, new_y에 기존 좌표 저장

    if (user_input == "상") new_y -= 1; // 위로 한 칸 이동
    else if (user_input == "하") new_y += 1; // 아래로 한 칸 이동
    else if (user_input == "좌") new_x -= 1; // 왼쪽으로 한 칸 이동
    else if (user_input == "우") new_x += 1;// 오른쪽으로 한 칸 이동
    else if (user_input == "지도") {// 지도 펼치기
        displayMap(map, user_x, user_y);
        continue;
    }
    else if (user_input == "종료") break;
    else {
        cout << "잘못된 입력입니다.\n";
        continue;
    }
```

입력	결과	설명
user_input: 사용자의 입력을 저장	사용자의 입력을 검사한 후 올바른 명령어를 입력했으며, 좌표 유효성 검사를 통과한 경우에 다음 라인으로 넘어갑니다..	사용자의 입력에 따라 if문으로 분기하여 다른 행동을 합니다. 상 하 좌 우 입력일 때 checkXY함수를 호출해서 좌표의 유효성을 검사한 후 지도를 펼칩니다. 지도가 입력일 때는 지도를 펼치고 종료가 입력일 때는 종료 메시지를 출력하며 while문을 탈출합니다. 다른 입력이 들어왔을 경우에는 잘못된 입력입니다 메시지를 출력 후 입력을 받는 것부터 다시 반복합니다. **1.**

	코드블록 스크린샷		
2			
	입력	결과	설명
	new_x, new_y: 사용자 입력에 의해 변형된 x, y 좌표값	유효성 검사를 통과하지 못한 경우 에러 메시지를 출력하고 다시 입력을 받습니다.	checkXY 함수를 호출한 후 유효성 검사를 통과하지 못한 경우 에러 메시지를 출력하고 다시 입력을 받습니다.

3			
	입력	결과	설명
	map: 현재의 보드판 user_x, user_y: 입력에 따라 반영된 좌표	만일 목표 지점에 도달했다면 축하 메시지를 출력하고 종료합니다.	checkGoal 함수를 통해 현재 위치가 목표 위치인지 검사한 후 목적지에 도달했다면 축하 메시지를 출력하고 while문을 탈출하여 프로그램을 종료합니다.

4

```cpp
#include <iostream>
#include <string>
using namespace std;

const int mapX = 5;
const int mapY = 5;
int hp = 20; // 체력 20으로 시작
```

입력	결과	설명
-	유저는 체력 20을 게임 시작마다 가지게 됩니다.	HP 20으로 시작해서 추후 HP가 감소하거나 증가하는 상호작용의 결과로 HP가 0이 되면 게임을 종료합니다.

5

```cpp
// 좌표 유효성 체크
if (!checkXY(new_x, mapX, new_y, mapY)) {
    cout << "맵을 벗어났습니다. 다시 돌아갑니다.\n";
    continue;
}

// 유효성 체크 후 유저 좌표에 반영하며 HP 감소
user_x = new_x;
user_y = new_y;
hp -= 1;  // 이동에 따른 HP 감소 반영
checkState(map, hp, user_x, user_y); // 이동한 위치의 효과에 따른 HP 반영 함수 호
```

입력	결과	설명
new_x, new_y: 사용자 입력에 의해 변형된 x, y 좌표값	이동한 뒤 hp를 1 감소시킵니다.	좌표 유효성 체크를 통과한 후 실제 좌표에 사용자의 명령으로 이동한 좌표를 대입해서 이동한 뒤 hp를 1 감소시킵니다.

6

```
while (1) {
    //사용자에게 계속 입력받으며 루프
    cout << "HP: " << hp << "\n명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
    string user_input = "";
    cin >> user_input;// 사용자의 입력 저장
```

입력	결과	설명
user_input: 사용자의 입력을 저장	처음 명령문을 입력받는 메시지와 함께 HP를 출력합니다.	사용자의 입력을 받기 전에 HP를 알려주며 선택에 긴장감을 부여합니다.

7

```
// HP 0일 시 실패 메세지 출력 후 게임 종료
if (hp <= 0) {
    cout << "HP가 0이 되어 사망했습니다.\n";
    break;
}
```

입력	결과	설명
	hp가 0이라면 실패 메시지를 출력하고 종료합니다.	hp가 0이 되면 break로 while문을 탈출하여 함수를 종료합니다.

8

```
// 유효성 체크 후 유저 좌표에 반영하며 HP 감소
user_x = new_x;
user_y = new_y;
hp -= 1;  // 이동에 따른 HP 감소 반영
checkState(map, hp, user_x, user_y); // 이동한 위치의 효과에 따른 HP 반영 함수 호출
```

입력	결과	설명
user_x, user_y: 입력에 따라 반영된 좌표	이동에 따른 hp 감소 후 이동한 위치 좌표에 따라 다르게 작동하는 checkState 함수를 호출합니다.	매개변수로 현재 위치와 hp를 전달하여 현재 위치의 기물에 따라 hp를 변화시킵니다.

9

```
// 현 위치의 기물 상황 체크하고, 각 경우별로 HP 갱신 함수
void checkState(int map[][mapX], int& hp, int x, int y) {
    switch (map[y][x]) {
    case 1: cout << "쓸모 없는 아이템을 발견했습니다! 아무 일도 일어나지 않습니다!.\n"; break
    case 2:
        cout << "적과 조우했습니다! 위협을 받아 HP가 2 감소합니다.\n";
        hp -= 2;
        break;
    case 3:
        cout << "진귀한 포션을 발견했습니다. 포션을 복용해 HP가 2 증가합니다.\n";
        hp += 2;
        break;
    }
}
```

입력	결과	설명
map[][mapX]: 현재 보드 hp: 현재 hp x, y : 현재 좌표	현재 좌표에 배치된 기물에 따라 다르게 상호작용합니다. hp를 증가 혹은 감소시킬 수 있습니다.	아이템 발견: 메시지를 출력하고 hp는 변화가 없습니다. 적과 조우: 메시지를 출력하고 hp가 2 감소합니다. 포셔 발견: 메시지를 출력하고 hp가 2 증가합니다.

10

```
while (1) {
    //사용자에게 계속 입력받으며 루프
    cout << "HP: " << hp << "\n명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
    string user_input = "";
    cin >> user_input;// 사용자의 입력 저장

    int new_x = user_x, new_y = user_y; // new_x, new_y에 기존 좌표 저장

    if (user_input == "상") new_y -= 1; // 위로 한 칸 이동
    else if (user_input == "하") new_y += 1; // 아래로 한 칸 이동
    else if (user_input == "좌") new_x -= 1; // 왼쪽으로 한 칸 이동
    else if (user_input == "우") new_x += 1;// 오른쪽으로 한 칸 이동
    else if (user_input == "지도") {// 지도 펼치기
        displayMap(map, user_x, user_y);
        continue;
    }
    else if (user_input == "종료") break;
    else {
        cout << "잘못된 입력입니다.\n";
        continue;
    }

    // 좌표 유효성 체크
    if (!checkXY(new_x, mapX, new_y, mapY)) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다.\n";
        continue;
    }

    // 유효성 체크 후 유저 좌표에 반영하며 HP 감소
    user_x = new_x;
    user_y = new_y;
    hp -= 1;  // 이동에 따른 HP 감소 반영
    checkState(map, hp, user_x, user_y); // 이동한 위치의 효과에 따른 HP 반영 함수 호출

    // HP 0일 시 실패 메세지 출력 후 게임 종료
    if (hp <= 0) {
        cout << "HP가 0이 되어 사망했습니다.\n";
        break;
    }

    // 지도 보여주기 함수 호출
    displayMap(map, user_x, user_y);

    // 목적지 도달했는데 체크 함수 호출
    if (checkGoal(map, user_x, user_y)) {
        cout << " 생존을 축하합니다! 게임을 종료합니다.\n";
        break;
    }
}
return 0;
```

입력	결과	설명
map: 현재의 보드판 user_x, user_y: 입력에 따라 반영된 좌표 new_x, new_y: 사용자 입력에 의해 변형된 x, y 좌표값 hp: 20부터 시작해 0이 되면 게임을 종료하는 지표	if문 안마다 쓰였던 bool inMap함수와 checkXY 함수를 if문 밖으로 꺼내며 한번만 쓸 수 있게 바꾸었습니다.	if 문 안에 있던 유효성 체크와 실제 좌표 이동 및 돌아감을 if문 밖으로 꺼내고, 가상 좌표인 new_x, new_y를 통해 유효성을 체크한 후 반영하여 반복되는 코드를 줄였습니다.

11

```cpp
// 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

    while (1) {
        //사용자에게 계속 입력받으며 루프
        cout << "HP: " << hp << "\n명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
        string user_input = "";
        cin >> user_input;// 사용자의 입력 저장

        int new_x = user_x, new_y = user_y; // new_x, new_y에 기존 좌표 저장

        if (user_input == "상") new_y -= 1; // 위로 한 칸 이동
        else if (user_input == "하") new_y += 1; // 아래로 한 칸 이동
        else if (user_input == "좌") new_x -= 1; // 왼쪽으로 한 칸 이동
        else if (user_input == "우") new_x += 1;// 오른쪽으로 한 칸 이동
        else if (user_input == "지도") {// 지도 펼치기
            displayMap(map, user_x, user_y);
            continue;
        }
        else if (user_input == "종료") break;
        else {
            cout << "잘못된 입력입니다.\n";
            continue;
        }

        // 좌표 유효성 체크
        if (!checkXY(new_x, mapX, new_y, mapY)) {
            cout << "맵을 벗어났습니다. 다시 돌아갑니다.\n";
            continue;
        }

        // 유효성 체크 후 유저 좌표에 반영하며 HP 감소
        user_x = new_x;
        user_y = new_y;
        hp -= 1;  // 이동에 따른 HP 감소 반영
        checkState(map, hp, user_x, user_y); // 이동한 위치의 효과에 따른 HP 반영 함수 호출

        // HP 0일 시 실패 메세지 출력 후 게임 종료
        if (hp <= 0) {
            cout << "HP가 0이 되어 사망했습니다.\n";
            break;
        }

        // 지도 보여주기 함수 호출
        displayMap(map, user_x, user_y);

        // 목적지 도달했는데 체크 함수 호출
        if (checkGoal(map, user_x, user_y)) {
            cout << " 생존을 축하합니다! 게임을 종료합니다.\n";
            break;
        }
    }
    return 0;
}
```

입력	결과	설명
map: 현재의 보드판 user_x, user_y: 사용자 입력에 따라 반영된 좌표 new_x, new_y: 사용자 입력에 의해 변형된 x, y 좌표값 hp: 20부터 시작해 0이 되면 게임을 종료하는 지표	hp가 0이 되거나 목표 지점에 도달할 때, 혹은 사용자가 종료를 입력할 때까지 루프합니다.	while문을 반복하며 사용자의 입력에 따라 hp와 위치를 변화시키며 hp가 0이 되거나 목표 지점에 도달할 때까지 루프합니다. 루프를 탈출하면 return 0;를 만나 종료합니다.

12

```
// 지도와 사용자의 현 위치 출력
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 |";
            }
            else {
                switch (map[i][j]) {
                case 0: cout << "      |"; break;
                case 1: cout << "아이템|"; break;
                case 2: cout << "  적  |"; break;
                case 3: cout << " 포션 |"; break;
                case 4: cout << "목적지|"; break;
                }
            }
        }
        cout << "\n--------------------------------\n";
    }
}
```

입력	결과	설명
map: 현재의 보드판 user_x, user_y: 사용자 입력에 따라 반영된 좌표	map 배열을 순회하며 사용자의 좌표에 도달하면 해당 칸을 우선적으로 사용자에게 배정합니다. 아닐 경우 map의 번호에 맞게 지도를 출력합니다.	위 함수에서 map배열 자체를 변형하지 않고, 출력만 하므로 아이템, 적, 포션, 목적지는 보존됩니다.

13

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int x, int maxX, int y, int maxY) {
    bool checkFlag = false;
    if (x >= 0 && x < maxX && y >= 0 && y < maxY) {
        checkFlag = true;
    }
    return checkFlag;

}
```

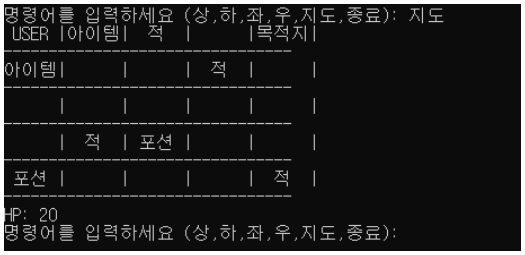
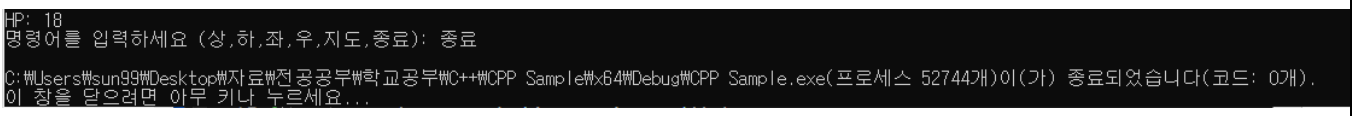
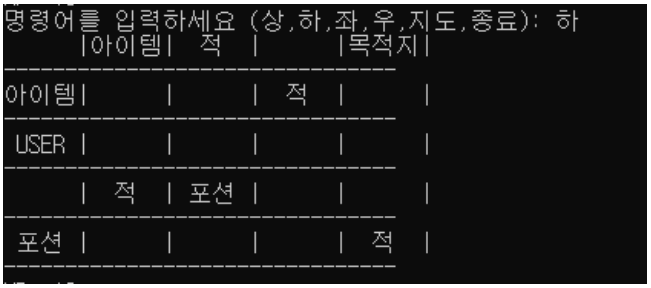
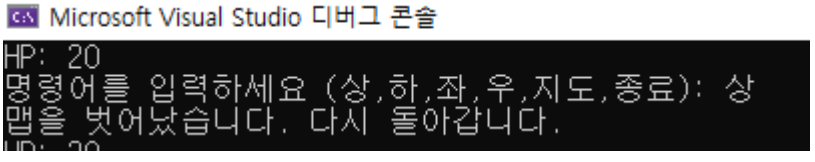
입력	결과	설명
maxX, maxY: 한계 좌표, 이 게임에서는 mapX, mapY를 대입합니다. int x, int y: new_x, new_y를 받습니다. checkFlag: 좌표의 유효성 여부로 리턴되는 변수	checkFlag 를 반환합니다. 좌표가 유효하면 true, 유효하지 않으면 false입니다.	x, y가 모두 보드판 내에 있다면, 즉 new_x, new_y 가 유효 좌표라면 checkFlag는 True가 되며 마지막에 checkFlag를 반환합니다.

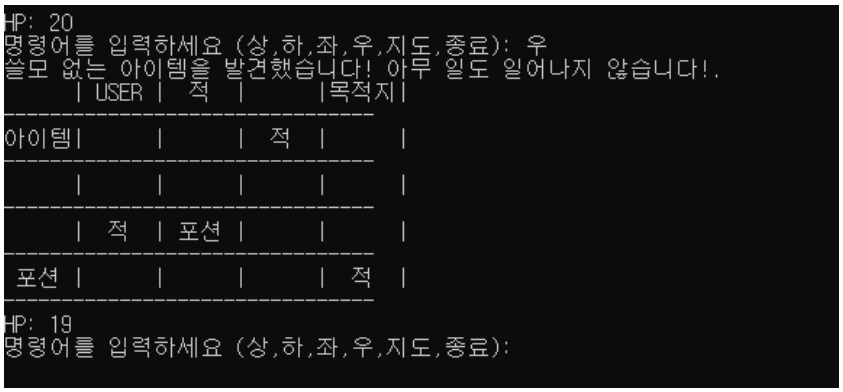
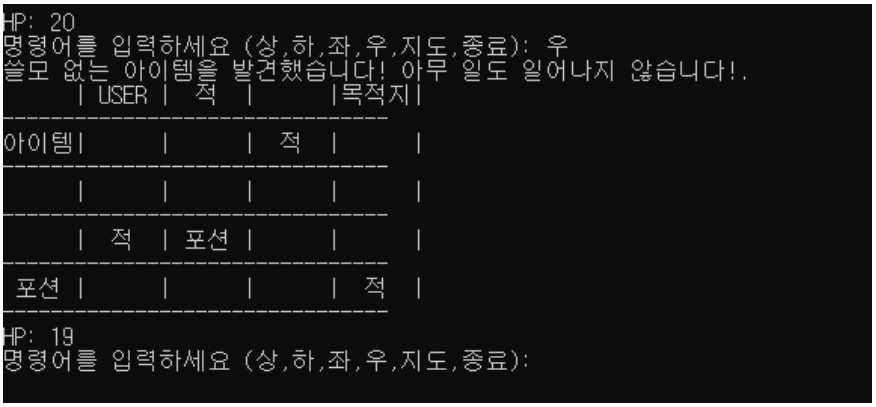
14

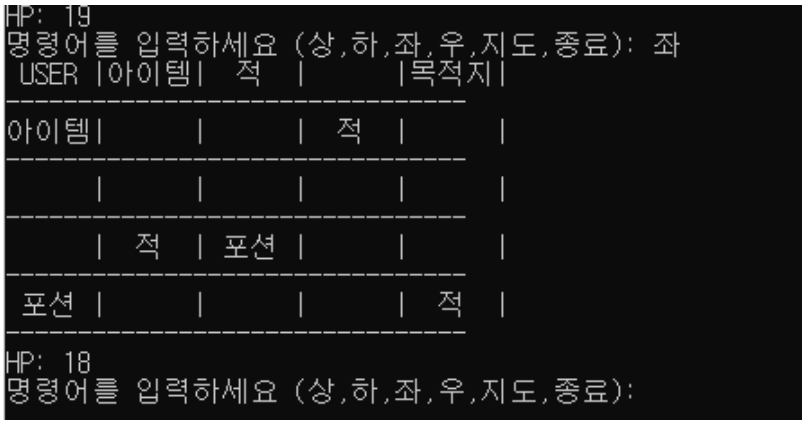
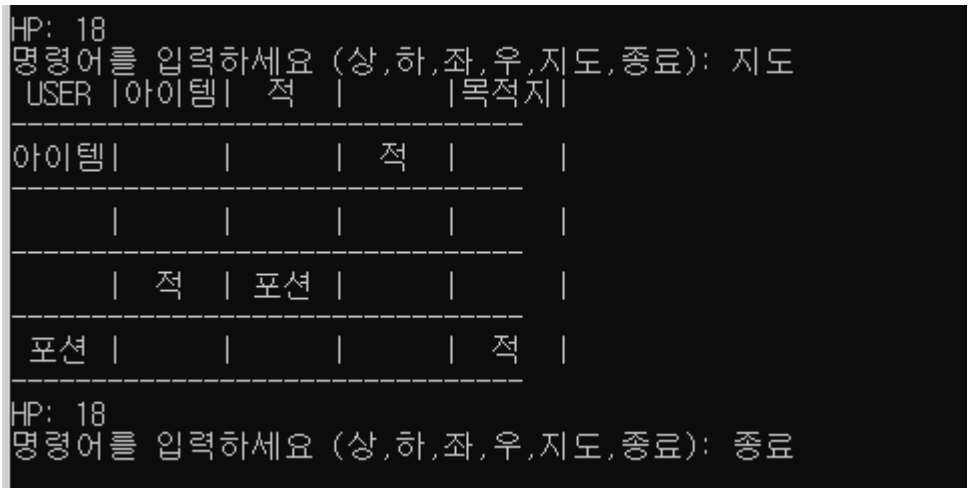
```
// 유저의 위치가 목적지인지 체크하는 함수
bool checkGoal(int map[][mapX], int x, int y) {
    // 목적지 도착하면
    if (map[y][x] == 4) {
        return true;
    }
    return false;
}
```

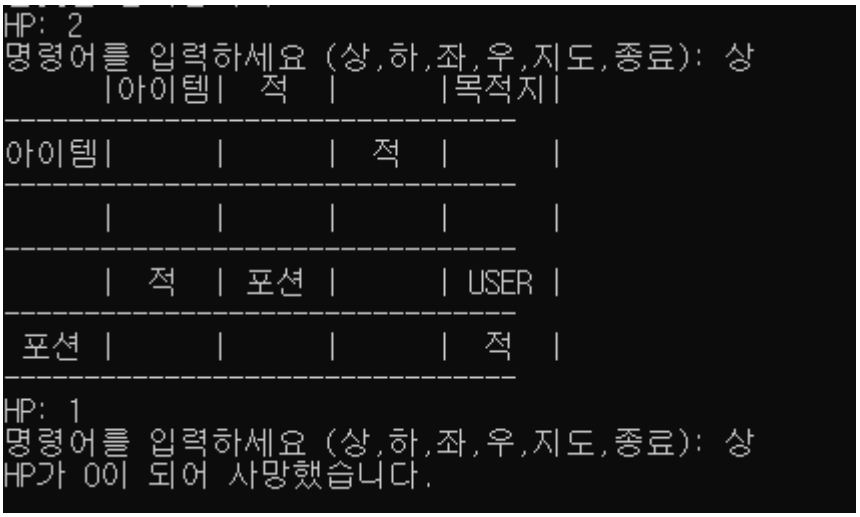
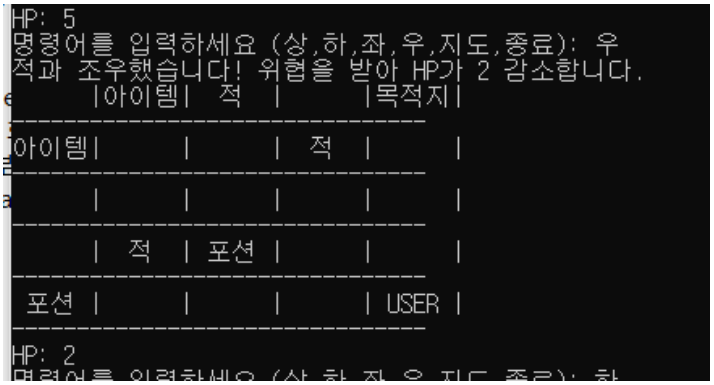
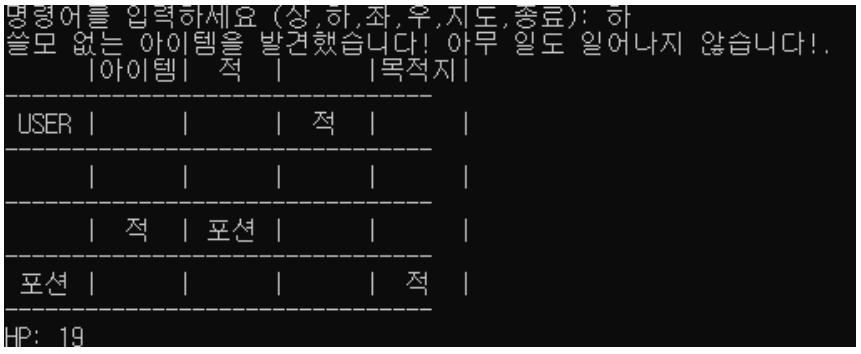
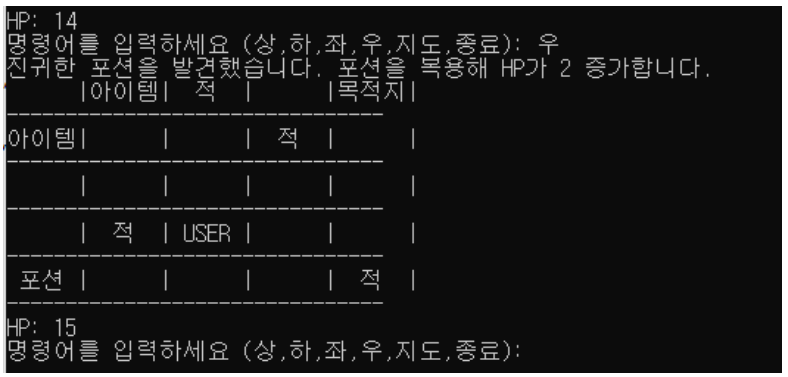
입력	결과	설명
map: 현재의 보드판 int x, int y: user_x, user_y 를 받습니다.	만약 4(목적지) 가 현재 위치의 기물이면 true를 반환하며 아니면 false를 반환합니다.	메인 함수의 마지막에 호출되어서 게임 지속 여부를 확인합니다. 만약 목적지에 도달해서 현재위치의 기물이 4(목적지)라면 true를 반환하며 메인 함수의 마지막에서 메시지를 출력하고 게임을 종료합니다.

IV 테스트

	코드블록 스크린샷
1	   
2	

3	 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우 쓸모 없는 아이템을 발견했습니다! 아무 일도 일어나지 않습니다!. | USER | 적 | |목적지| 아이템| | 적 | | 적 | 포션 | 포션 | | 적 | HP: 19 명령어를 입력하세요 (상,하,좌,우,지도,종료):
4	 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우 쓸모 없는 아이템을 발견했습니다! 아무 일도 일어나지 않습니다!. | USER | 적 | |목적지| 아이템| | 적 | | 적 | 포션 | 포션 | | 적 | HP: 19 명령어를 입력하세요 (상,하,좌,우,지도,종료):

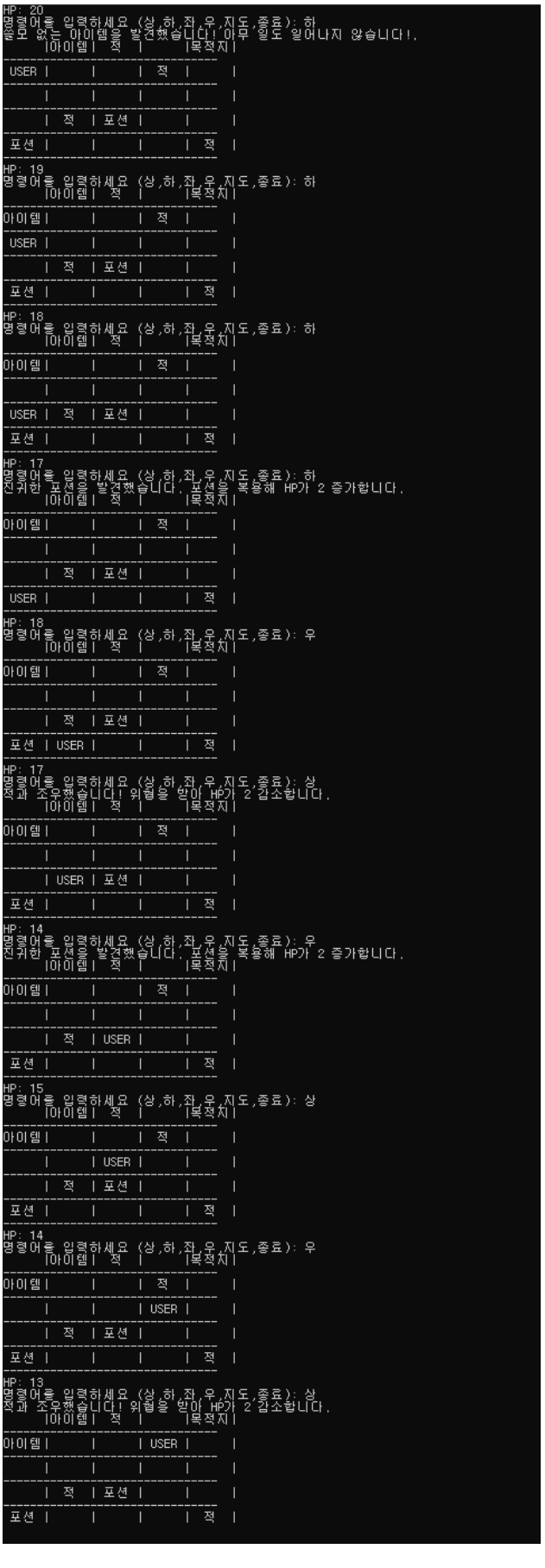
5	 HP: 19 명령어를 입력하세요 (상,하,좌,우,지도,종료): 좌 USER |아이템| 적 | |목적지| 아이템| | | 적 | | | | | | | | 적 | 포션 | | | 포션 | | | | 적 | HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료):
6	 HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): 지도 USER |아이템| 적 | |목적지| 아이템| | | 적 | | | | | | | | 적 | 포션 | | | 포션 | | | | 적 | HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): 종료

7	 HP: 2 명령어를 입력하세요 (상,하,좌,우,지도,종료): 상 |아이템| 적 | |목적지| 아이템| | 적 | | | | | | | | 적 | 포션 | | USER | 포션 | | | 적 | HP: 1 명령어를 입력하세요 (상,하,좌,우,지도,종료): 상 HP가 0이 되어 사망했습니다.
8	 HP: 5 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우 적과 조우했습니다! 위협을 받아 HP가 2 감소합니다. |아이템| 적 | |목적지| 아이템| | 적 | | | | | | | | 적 | 포션 | | | 포션 | | | USER | HP: 2  명령어를 입력하세요 (상,하,좌,우,지도,종료): 하 쓸모 없는 아이템을 발견했습니다! 아무 일도 일어나지 않습니다!. |아이템| 적 | |목적지| USER | | 적 | | | | | | | | 적 | 포션 | | | 포션 | | | 적 | HP: 19  HP: 14 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우 진귀한 포션을 발견했습니다. 포션을 복용해 HP가 2 증가합니다. |아이템| 적 | |목적지| 아이템| | 적 | | | | | | | | 적 | USER | | | 포션 | | | 적 | HP: 15 명령어를 입력하세요 (상,하,좌,우,지도,종료):

9

전체 동작 스크린샷


HP: 20
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
맵을 벗어났습니다. 다시 돌아갑니다.
HP: 20
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
쓸모 없는 아이템을 발견했습니다! 아무 일도 일어나지
|아이템| 적 |목적지|
USER | | 적 |
| 적 | 포션 |
포션 | | 적 |
HP: 19
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
|아이템| 적 |목적지|
아이템| | 적 |
USER | 적 | 포션 |
포션 | | 적 |
HP: 18
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
|아이템| 적 |목적지|
아이템| | 적 |
USER | 적 | 포션 |
포션 | | 적 |
HP: 17
명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
적과 조우했습니다! 위협을 받아 HP가 2 감소합니다.
|아이템| 적 |목적지|
아이템| | 적 |
USER | 포션 |
포션 | | 적 |
HP: 14
명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
진귀한 포션을 발견했습니다. 포션을 복용해 HP가 2 증
|아이템| 적 |목적지|
아이템| | 적 |
적 | USER |
포션 | | 적 |
HP: 15
명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
|아이템| 적 |목적지|
아이템| | 적 |
적 | 포션 | USER |
포션 | | 적 |
HP: 14
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
|아이템| 적 |목적지|
아이템| | 적 |
USER |
적 | 포션 |
포션 | | 적 |
HP: 13
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
적과 조우했습니다! 위협을 받아 HP가 2 감소합니다.
|아이템| 적 |목적지|
아이템| | USER |
적 | 포션 |
포션 | | 적 |
HP: 10
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
|아이템| 적 | USER |목적지|


HP: 20
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
쓸모 없는 아이템을 발견했습니다! 아무 일도 일어나지 않습니다!.
|아이템| 적 |목적지|
USER | | 적 |
| 적 | 포션 |
포션 | | 적 |
HP: 19
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
|아이템| 적 |목적지|
아이템| | 적 |
USER |
| 적 | 포션 |
포션 | | 적 |
HP: 18
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
|아이템| 적 |목적지|
아이템| | 적 |
USER | 적 | 포션 |
포션 | | 적 |
HP: 17
명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
진귀한 포션을 발견했습니다. 포션을 복용해 HP가 2 증가합니다.
|아이템| 적 |목적지|
아이템| | 적 |
| 적 | 포션 |
USER | | 적 |
HP: 18
명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
|아이템| 적 |목적지|
아이템| | 적 |
| 적 | 포션 |
포션 | USER | 적 |
HP: 17
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
적과 조우했습니다! 위협을 받아 HP가 2 감소합니다.
|아이템| 적 |목적지|
아이템| | 적 |
USER | 포션 |
포션 | | 적 |
HP: 14
명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
진귀한 포션을 발견했습니다. 포션을 복용해 HP가 2 증가합니다.
|아이템| 적 |목적지|
아이템| | 적 |
| 적 | USER |
포션 | | 적 |
HP: 15
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
|아이템| 적 |목적지|
아이템| | 적 |
USER |
| 적 | 포션 |
포션 | | 적 |
HP: 14
명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
|아이템| 적 |목적지|
아이템| | 적 |
USER |
| 적 | 포션 |
포션 | | 적 |
HP: 13
명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
적과 조우했습니다! 위협을 받아 HP가 2 감소합니다.
|아이템| 적 |목적지|
아이템| | USER |
| 적 | 포션 |
포션 | | 적 |

v 결과 및 결론

1. 프로젝트 결과

텍스트로 진행하는 MUD게임을 만들었습니다. 처음에는 막연했는데 베이스 코드를 읽어보며 하나씩 시도하며 차근차근 진행했습니다. checkXY와 dispalyMap, checkGoal, checkState 함수, new_x, new_y를 통해서 main 함수의 복잡도를 줄이는 것에 집중했습니다.

시스템 상으로는 한 칸 이동할 때 HP를 1 감소시키며, 적을 만나면 2 감소, 포션을 만나면 2 증가지만 사용자 입장에서는 그것들이 정산되어 보이기에 마치 3 감소, 1 증가처럼 보인다는 점이 아쉬웠습니다. 그렇다고 hp 출력을 더 하자니 게임 내 분량에 비해 부자연스러울 정도로 많은 느낌이었습니다. 게임 내에 컨텐츠를 더 추가한다면 이동에 따른 hp의 감소와 이벤트 발생에 따른 hp감소가 충분한 간격을 두고 자연스럽게 발생할 수 있도록 처리할 수 있겠다는 생각을 했습니다.

2. 느낀 점

어떤 걸 함수화해야 할지 고민할 때 전체의 구조와 효율을 생각하는 부분이 어려웠습니다. 또한 네이밍과 주석에도 신경 쓰지 않으면 길을 잃기 쉽다는 걸 느꼈습니다. 게임은 화려한 그래픽과 장황한 스토리에서도 즐거움을 줄 수 있지만, 간단한 조작과 볼륨이 적은 컨텐츠라도 충분히 즐거울 수 있다는 점을 알았고, 그 점이 굉장히 매력적이었습니다. 그래서 기말 프로젝트에도 그런 컨셉의 게임을 만들어보고 싶습니다.