

C++프로그래밍 및 실습 Project: Mud game 보고서

학번: 213652

이름: 김지환

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 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
 - ④ 유저는 체력 20을 가지고 게임 시작
 - ⑤ 사용자가 이동할 때마다 사용자 체력 1씩 감소
 - ⑥ 처음 명령문을 입력 받을 때마다 HP 함께 출력
 - ⑦ HP가 0이 되면 "실패"를 출력하고 종료
 - ⑧ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력
 - {X}이/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)

3. 함수 계획

- ① 메인 함수: 사용자에게 값을 계속 입력받고, 그에 대한 함수 호출
- ② 지도와 현재 위치 출력 함수: displayMap()
- ③ 사용자 위치 체크 함수: checkXY()
- ④ 목적지에 도착 체크 함수: checkGoal()
- ⑤ 사용자의 HP 관리 및 현 위치의 아이템, 적, 포션 유무 확인 함수: checkState()
- ⑥ 사용자의 HP가 0 이하가 되었을 경우 프로그램 종료 함수: zeroHp()
- ⑦ 사용자가 맵을 벗어난 경우 다시 돌아가는 함수: goBack()

3. 설계 및 구현

1. 함수 설명

- 지도와 현재 위치 출력 함수

```
// 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;
}
```

1. 함수 스크린 샷(좌)
2. 입력
 - int map[][mapX]: 전체 지도
 - int user_x: 유저의 x좌표
 - int user_y: 유저의 y좌표
3. 반환값
 - 없음(void)
4. 결과
 - 전체 지도(아이템, 적, 포션, 목적지)를 출력
 - 사용자 위치를 출력
5. 설명
 - 이중 for문 이용하여 각 노드 출력
 - If문을 이용하여 탐색한 노드가 사용자의 위치 일 경우:
 - USER로 출력
 - 아닐 경우:
 - switch문 활용하여 case 분류하여 출력

(아무것도 X, 아이템, 적, 포션, 목적지)

- 사용자 위치 체크 함수

```
// 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;
}
```

1. 함수 스크린 샷(좌)

2. 입력

- int user_x: 유저의 x좌표

- int mapX: 맵의 x축 크기

- int user_y: 유저의 y좌표

- int mapY: 맵의 y축 크기

3. 반환값

- true: 이동하려는 곳이 유효한 좌표일 경우

- false: 이동하려는 곳이 유효하지 않은 좌표일 경우

4. 결과

- 사용자가 맵에서 벗어났는지 아닌지를 체크하여 true/false의 bool 값(checkFlag)을 반환한다.

5. 설명

- 반환할 값을 넣을 변수 checkFlag를 false값으로 초기화한다.

- if문을 활용하여 조건 $0 \leq user_x < mapX \wedge 0 \leq user_y < mapY$ 에 해당하는 조건문을 작성해준다.

- 조건을 만족할 경우 checkFlag = true로 반환, 만족하지 않을 경우 checkFlag= false로 반환

- 목적지에 도착 체크 함수

```
// 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. 함수 스크린 샷(좌)

2. 입력

- int map[] [mapX]: 전체 지도

- int user_x: 유저의 x좌표

- int user_y: 유저의 y좌표

3. 반환값

- true: 목적지에 도착했을 경우

- false: 목적지에 도착하지 않았을 경우

4. 결과

- 유저가 이동한 위치에 목적지(4)가 있을 경우 true를 반환, 아닐 경우 false를 반환

→ 유저의 위치가 목적지인지를 체크한다.

5. 설명

- if문을 활용 2차원 배열 map[user_y][user_x]에 4라는 값(목적지)가 들어있을 경우:

→ true를 반환

- 아닐 경우:

→ false를 반환

- 사용자의 HP 관리 및 현 위치의 아이템, 적, 포션 유무 확인 함수

```
//사용자의 HP 관리 및 현 위치의 아이템, 적, 포션 유무 확인 함수
void checkState(int map[][], int user_x, int user_y){
    //checkState 함수가 호출 되었다는 것은 올바른 위치로 사용자가 이동한 것으로 hp를 1 감소
    user_hp--;

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

1. 함수 스크린샷(좌)

2. 입력

- int map[][], mapX: 전체 지도

- int user_x: 유저의 x좌표

- int user_y: 유저의 y좌표

3. 반환값

- 없음(void)

4. 결과

- 함수가 호출될 경우 가장 먼저 유저의 hp를 1 감소시킴
- 사용자의 위치에 아이템이 있을 경우: 아이템이 있다는 메시지 출력
- 사용자의 위치에 적이 있을 경우: hp를 2 감소시키고 적이 있다는 메시지 출력
- 사용자의 위치에 포션이 있을 경우: hp를 2 증가시키고 포션이 있다는 메시지 출력

5. 설명

- 이 함수가 호출되었다는 것은 올바른 위치로 사용자가 이동한 것(메인함수 코드블록 참고)

→ 사용자가 이동을 했다면 hp를 1 감소시켜야 하는 기능 적용(user_hp--)

- 사용자의 위치에 아이템, 적, 포션이 있는지 체크하기 위해 if문과 else if문 활용

→ 아이템이 있을 경우: 메시지 출력

→ 적이 있을 경우: hp를 2 감소시킴, 메시지 출력(결과적으로 hp는 3 감소)

→ 포션이 있을 경우: hp를 2 증가시킴, 메시지 출력(결과적으로 hp는 1 증가)

- 사용자의 HP가 0 이하가 되었을 경우 프로그램 종료 함수

```
void zeroHp(int user_hp){
    //사용자의 hp가 0 이하가 되었을 경우 --> 프로그램 종료
    if(user_hp <= 0){
        cout << "HP가 0 이하가 되었습니다." << "실패했습니다." << endl;
        cout << "게임을 종료합니다." << endl;
        exit(0); //프로그램 종료
    }
}
```

1. 함수 스크린샷(좌)
2. 입력
 - int user_hp: 유저의 hp
3. 반환값
 - 없음(void)

4. 결과

- user_hp가 0 이하가 되었을 경우 종료 메시지 출력 후 프로그램 종료

5. 설명

- if문 활용하여 user_hp가 0 이하가 되었을 경우 종료 메시지 출력
- 함수에서 프로그램 종료를 위하여 exit(0)이라는 명령문 사용

- 사용자가 맵을 벗어난 경우 에러 메시지 출력 후 다시 돌아가는 함수

```
//사용자가 맵을 벗어난 경우 다시 돌아가는 함수
int goBack(int user_x, int mapX, int user_y, int mapY){
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if(inMap == false){
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        return 1; //goBack
    }
    else return 0; //0을 return --> 연산 결과 아무것도 하지 X
}
```

1. 함수 스크린샷(좌)
2. 입력
 - int user_x: 유저의 x좌표
 - int mapX: 맵의 x축 크기
 - int user_y: 유저의 y좌표
 - int mapY: 맵의 y축 크기

3. 반환값

- 0: 맵을 벗어나지 않은 경우
- 1: 맵을 벗어난 경우

4. 결과

- 맵을 벗어나지 않은 경우 0을 반환
- 맵을 벗어난 경우 맵을 벗어나서 돌아간다는 메시지 출력 후 1을 반환

5. 설명

- 다시 돌아갈 때의 연산을 위한 함수이다. 즉, 1이라는 값으로 연산을 하여 사용자가 다시 돌아가게 하는 것이다.
- 먼저 inMap이라는 bool 변수에 checkXY 함수를 이용하여 벗어났는지 안 벗어났는지에 대한 값을 초기화한다.
- inMap이 false일 경우 즉, 맵을 벗어난 경우 메시지 출력 후 1을 반환한다.(메인함수에서 연산 결과 → 다시 돌아가게 됨)
- inMap이 true일 경우 즉, 맵을 벗어나지 않은 경우 연산 결과 아무 영향을 끼치지 않게 하기 위해 0을 반환한다.

2. 기능 별 구현 사항(main 함수 내의 코드블록)

① 사용자에게 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받기

② 지도 밖으로 나가게 되면 에러 메시지 출력

⑤ 사용자가 이동할 때마다 사용자 체력 1씩 감소

⑧ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력

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

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

if (user_input == "상") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    int back = goBack(user_x, mapX, user_y, mapY);
    user_y += back;
    if(back == 0) {
        cout << "위로 한 칸 올라갑니다." << endl;
        displayMap(map, user_x, user_y);
        checkState(map,user_x,user_y);
    }
}
else if (user_input == "하") {
    // TODO: 아래로 한 칸 내려가기
    user_y += 1;
    int back = goBack(user_x, mapX, user_y, mapY);
    user_y -= back;
    if(back == 0) {
        cout << "위로 한 칸 내려갑니다." << endl;
        displayMap(map, user_x, user_y);
        checkState(map,user_x,user_y);
    }
}
else if (user_input == "좌") {
    // TODO: 왼쪽으로 이동하기
    user_x -= 1;
    int back = goBack(user_x, mapX, user_y, mapY);
    user_x += back;
    if(back == 0) {
        cout << "왼쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        checkState(map,user_x,user_y);
    }
}
else if (user_input == "우") {
    // TODO: 오른쪽으로 이동하기
    user_x += 1;
    int back = goBack(user_x, mapX, user_y, mapY);
    user_x -= back;
    if(back == 0) {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);
        checkState(map,user_x,user_y);
    }
}
else if (user_input == "지도") {
    // TODO: 지도 보여주기 함수 호출
    displayMap(map, user_x, user_y);
}
else if (user_input == "종료") {
    cout << "종료합니다.";
    break;
}
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

1. 코드블록 스크린샷(상)

2. 입력

- user_input: 사용자의 입력을 저장할 변수

- map: 맵(2차원 배열)

- user_hp: 유저의 hp

- user_x: 유저의 x좌표
- mapX: 맵의 x축 크기
- user_y: 유저의 y좌표
- mapY: 맵의 y축 크기
- back: 돌아갈지 말지에 대한 연산을 위한 변수
- goBack(): 사용자가 맵을 벗어난 경우 에러 메시지 출력 후 다시 돌아가는 함수
- displayMap(): 지도와 현재 위치 출력 함수
- checkState(): 사용자의 HP 관리 및 현 위치의 아이템, 적, 포션 유무 확인 함수

3. 결과

- 사용자의 입력에 따른 위치 이동
- 잘못된 위치일 경우 에러메시지 출력 후 다시 복귀
- 올바른 위치일 경우 이동메시지 출력 후 지도 출력
 - 지도 출력 후 hp 1 감소시킴
 - 해당 위치에 있는 것(아이템, 적, 포션) 출력 후 hp 관리
- 사용자의 입력이 위치 이동이 아닌 “지도”일 경우 지도 출력
- 사용자의 입력이 “종료”일 경우 프로그램 종료
- 사용자의 입력이 잘못되었을 경우 다시 입력하게 함

4. 설명

- if문과 else if문을 활용하여 각 case 분류{이동(상/하/좌/우), 지도, 종료, 잘못된 입력}
- 사용자가 위치 이동 관련된 것을 입력했을 경우 그에 따른 좌표 값을 1 증가시키거나 감소시킴
 - goBack() 함수를 이용해서 잘못된 위치일 경우 에러메시지 출력 후 다시 복귀(증가 → 감소, 감소 → 증가)
 - goBack() 함수의 반환값이 0일 경우 즉, 올바른 위치일 경우 이동 메시지 출력
 - displayMap() 함수를 이용해서 지도 출력
 - checkState() 함수를 이용해서 hp 관리 및 어떤 것을 만났을 경우에 대한 메시지 출력
- 사용자가 “지도”를 입력했을 경우 displayMap() 함수를 이용해서 지도 출력
- 사용자가 “종료”를 입력했을 경우 break문을 통해서 while(1)을 빠져나간 뒤 프로그램 종료
- 사용자가 잘못된 입력을 했을 경우 continue문을 통해서 다시 입력문으로 돌아감

③ 목적지에 도착하면 "성공"을 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;  
}
```

1. 코드블록 스크린 캡(좌)
2. 입력
 - finish: 목적지 도착 여부를 위한 변수
 - map: 맵(2차원 배열)
 - user_x: 유저의 x좌표
 - user_y: 유저의 y좌표
 - checkGoal(): 목적지에 도착 체크 함수

3. 결과

- 목적지에 도착했을 경우 성공 메시지를 출력한 뒤에 게임 종료

4. 설명

- checkGoal() 함수를 통해서 목적지 도착 여부 값은 반환 받은 뒤에 finish 변수에 대입
- if문 활용하여 finish가 true일 경우 성공 메시지 출력
 - 출력한 뒤에 break문을 통해서 while(1)을 빠져나간 뒤에 프로그램 종료

④ 유저는 체력 20을 가지고 게임 시작

```
int user_hp = 20; //유저의 hp, 초기값 = 20(함수에서의 사용을 위해 전역변수로 선언)
```

1. 코드블록 스크린 캡(상)

2. 입력

- user_hp: 유저의 hp

3. 결과

- 유저의 hp가 20으로 시작

4. 설명

- hp 관리 함수인 checkState()에서 사용하기 위해 전역변수로 선언

⑥ 처음 명령문을 입력 받을 때마다 HP 함께 출력

```
// 게임 시작  
while (1) { // 사용자에게 계속 입력받기 위해 무한 루프  
  
    // 사용자의 입력을 저장할 변수  
    string user_input = "";  
  
    cout << "현재 HP: " << user_hp << " ";  
    cout << "명령어를 입력하세요 (상,하,좌,우,지도,종료): ";  
    cin >> user_input;
```

1. 코드블록 스크린 캡(좌)
2. 입력
 - user_input: 사용자의 입력을 저장할 변수
 - user_hp: 유저의 hp
3. 결과
 - 명령어를 입력 메시지 이전에 현재 HP 표시

4. 설명

- 무한루프를 통해서 사용자의 입력을 계속 받게 되는데 cout 과 user_hp를 통해서 유저의 현재 HP를 먼저 출력하게 한 뒤에 입력하라는 메시지를 출력하도록 한다.

⑦ HP가 0이 되면 “실패”를 출력하고 종료

98

`zeroHp(user_hp);`

1. 코드블록 스크린 샷(상)

2. 입력

- zeroHp(): 사용자의 HP가 0 이하가 되었을 경우 프로그램 종료 함수
- user_hp: 유저의 hp

3. 결과

- 사용자의 hp가 0 이하가 되었을 경우 프로그램을 종료한다.

4. 설명

- 사용자가 이동하고 나서 zeroHp() 함수를 호출한다. (따라서 목적지로 이동했더라도 hp가 0이면 프로그램이 종료된다.)
- zeroHp() 함수는 유저의 hp가 0 이하가 되었을 경우 프로그램이 종료되도록 한다.

4. 테스트

1. 기능 별 테스트 결과:

① 사용자에게 “상”, “하”, “좌”, “우”, “지도”, “종료” 중 하나를 입력 받기

- 상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력

아이템 적 목적지			
아이템			
포션			적
현재 HP: 17	명령어를 입력하세요 (상,하,좌,우,지도,종료):	상 위로 한 칸 올라갑니다.	
		아이템	적
			목적지
아이템 USER			
포션			적
현재 HP: 15	명령어를 입력하세요 (상,하,좌,우,지도,종료):	좌 원쪽으로 이동합니다.	
		아이템	적
			목적지
아이템 USER 적 목적지			
포션			적
현재 HP: 14	명령어를 입력하세요 (상,하,좌,우,지도,종료):	우 오른쪽으로 이동합니다.	
		아이템	적
			목적지
아이템 USER 적 목적지			
포션			적

- “지도”를 입력하면 전체 지도와 함께 현재 위치를 출력

```
현재 HP: 20 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 지도
USER |아이템| 적 | 목적지 |
-----
아이템 |           |   적   |           |
-----|           |   적   |           |
-----|   적   | 포션 |           |
-----포션 |           |   적   |           |
```

- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청

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

② 지도 밖으로 나가게 되면 에러 메시지 출력

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

아이템이 있습니다
현재 HP: 19 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 좌
맵을 벗어났습니다. 다시 돌아갑니다.
현재 HP: 19 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료): 지도
|아이템| 적 | 목적지 |
-----
USER |           |   적   |           |
-----|           |   적   |           |
-----|   적   | 포션 |           |
-----포션 |           |   적   |           |
```

③ 목적지에 도착하면 "성공"을 출력하고 종료

```
현재 HP: 14 명령어를 입력하세요 (상,하,좌,우,지도,종료): 우
오른쪽으로 이동합니다.
|아이템| 적 | 목적지 |
-----|-----|-----|
아이템 |   |   | 적 | USER |
-----|-----|-----|
|   |   |   |   |
-----| 적 | 포션 |   |   |
-----|   |   |   |
포션 |   |   |   | 적 |
-----|-----|-----|
현재 HP: 13 명령어를 입력하세요 (상,하,좌,우,지도,종료): 상
위로 한 칸 올라갑니다.
|아이템| 적 | 목적지 |
-----|-----|-----|
아이템 |   |   | 적 |   |
-----|-----|-----|
|   |   |   |   |
-----| 적 | 포션 |   |   |
-----|   |   |   |
포션 |   |   |   | 적 |
-----|-----|-----|
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

④ 유저는 체력 20을 가지고 게임 시작

```
현재 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): ■
```

⑤ 사용자가 이동할 때마다 사용자 체력 1씩 감소

⑥ 처음 명령문을 입력 받을 때마다 HP 함께 출력

```
현재 HP: 20 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하
위로 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): 우
오른쪽으로 이동합니다.
|아이템| 적 | 목적지 |
-----|-----|-----|
아이템 | USER |   | 적 |   |
-----|-----|-----|
|   |   |   |   |
-----| 적 | 포션 |   |   |
-----|   |   |   |
포션 |   |   |   | 적 |
-----|-----|-----|
현재 HP: 18 명령어를 입력하세요 (상,하,좌,우,지도,종료): ■
```

⑤ + ⑥

⑦ HP가 0이 되면 “실패”를 출력하고 종료

```
현재 HP: 4 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하  
위로 한 칸 내려갑니다.  
|아이템| 적 | 목적지 |  
-----  
아이템 | | | |  
| | | USER | |  
-----  
| 적 | 포션 | | |  
-----  
포션 | | | | 적 |  
  
현재 HP: 3 명령어를 입력하세요 (상,하,좌,우,지도,종료): 상  
위로 한 칸 올라갑니다.  
|아이템| 적 | 목적지 |  
-----  
아이템 | | | USER | |  
| | | | |  
-----  
| 적 | 포션 | | |  
-----  
포션 | | | | 적 |  
  
적이 있습니다 HP가 2 줄어듭니다.  
HP가 0 이하가 되었습니다. 실패했습니다.  
게임을 종료합니다.
```

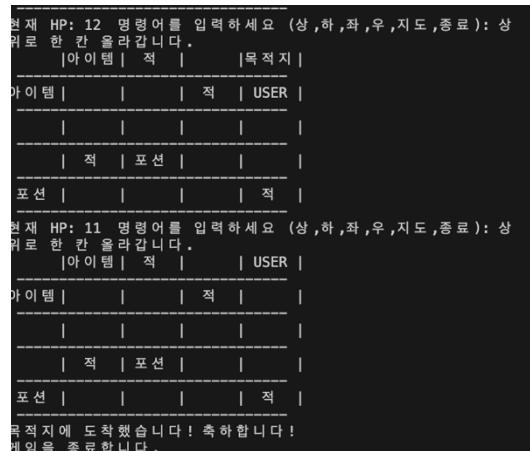
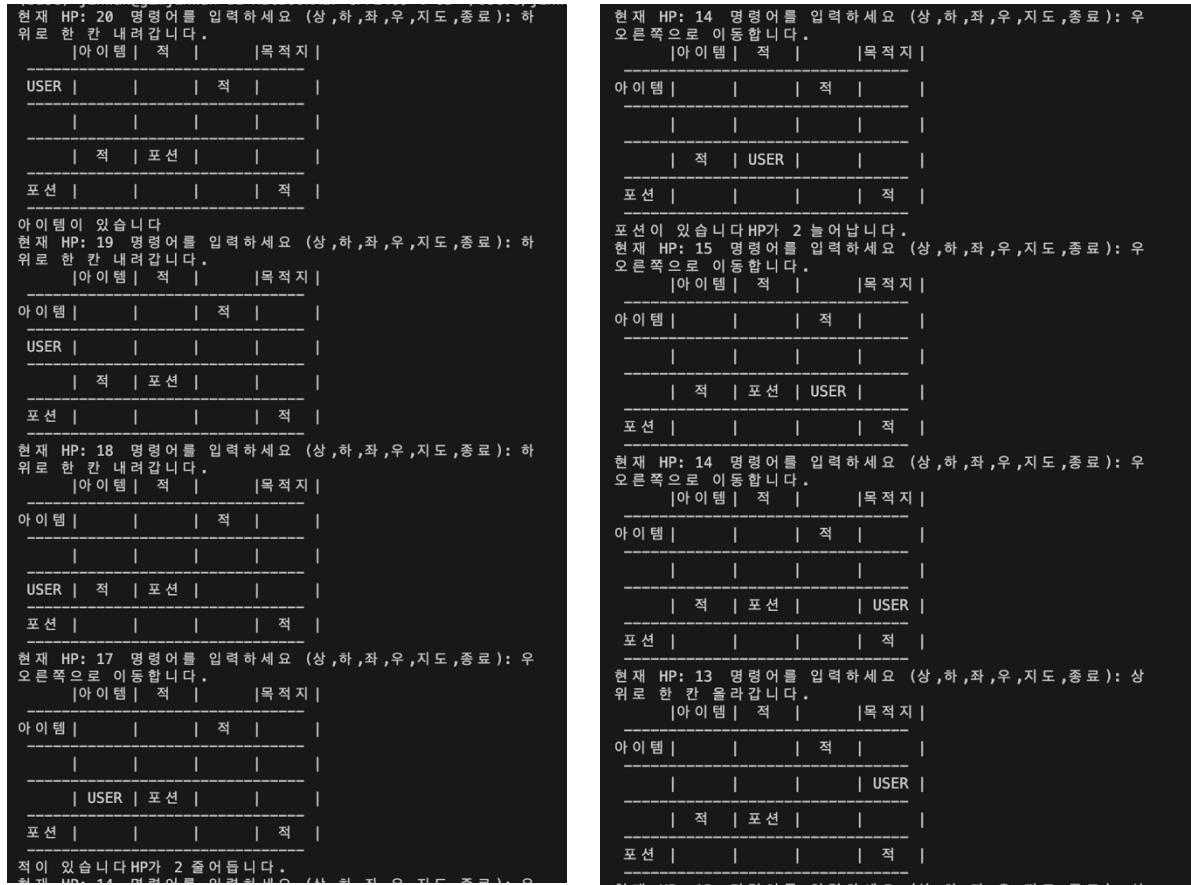
⑧ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력

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

```
현재 HP: 16 명령어를 입력하세요 (상,하,좌,우,지도,종료): 하  
위로 한 칸 내려갑니다.  
|아이템| 적 | 목적지 |  
-----  
아이템 | | | | 적 |  
| | | | |  
-----  
| 적 | USER | | |  
-----  
포션 | | | | 적 |  
  
포션이 있습니다 HP가 2 늘어납니다.
```

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

2. 최종 테스트 스크린샷



5. 결과 및 결론

1. 프로젝트 결과: mud game을 만들었고 새롭게 checkState(), zeroHp(), goBack()

함수를 구현했고 goBack() 함수를 통해 코드 최적화를 하였다.

2. 느낀 점: 함수와 코드 블록이 너무 많아서 보고서 작성 시간이 많이 걸렸고 기능

이 많다보니 기능 별 테스트 진행하는 데에도 시간이 많이 걸렸다.

그래도 여러 함수를 만들다 보니 함수화 하는 것에 적응이 꽤 된 것 같아서 코딩실력 향상에 도움이 되는 프로젝트였다.