mud_game 만들기

빅데이터 융합학과 222400 김서연

1. 서론

1. 프로젝트 목적 및 배경:
   수업시간에 지금까지 배운 내용들과 함수를 이용해서 프로젝트 진행
2. 목표 :
   간단한 Mud game 만들기

2. 요구사항

1. 사용자 요구사항:
   1) 사용자는 “상”,“하”,“좌”,“우”,“지도”,“종료” 중 하나 입력하기

2. 기능 계획 :
   1) 사용자는 체력 20을 가지고 게임 시작
   2) 사용자에게 “상”,“하”,“좌”,“우”,“지도”,“종료” 중 하나 입력받기
      - 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
   3) 지도 밖으로 나가게 되면 에러 메시지 출력
   4) 목적지에 도착하면 성공을 출력하고 종료
   5) 사용자가 이동할 때마다 사용자 체력 1씩 감소
   6) 아이템, 포션, 적을 만났을 때 그에 대한 메시지를 출력
      - ex) {x}가 있습니다.
      - 적을 만날 경우 HP 2가 줄어들고 그에 대한 추가 메시지 출력
        ex) 적이 있습니다. HP가 2 줄어듭니다.
      - 포션을 만날 경우 HP가 2 늘어나고 그에 대한 추가 메시지 출력
        ex) 포션이 있습니다. HP가 2 늘어납니다.
      - 적이나 포션 등은 사라지지 않음을 전제
   7) 처음 명령문을 입력 받을 때마다 HP 함께 출력
   8) HP가 0이 되면 실패를 출력하고 종료

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) HP 체크 함수 : chectState()

3. 설계 및 구현

< 기능 구현 >

1) 사용자는 체력을 20가지고 시작

<스크린샷>

```
int HP = 20; // 유저는 체력 20을 가지고 게임 시작
```

<설명>

- HP를 20이라고 설정

2) 사용자에게 “상”,“하”,“좌”,“우”,“지도”,“종류” 중 하나 입력받기

+3) 지도 밖으로 나가게 되면 에러 메시지 출력

1. “상”을 입력했을 때

<함수 스크린샷>

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

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

//사용자가 상을 입력했을 때
if (user_input == "상") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    HP -= 1; //움직일 때마다 HP 1씩 감소

    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y += 1;
        // 맵을 벗어날 땐 다시 이동하기 때문에 1을 다시 더해줌
        HP += 1;
    }
    else {
        cout << "위로 한 칸 올라갑니다." << endl;
        displayMap(map, user_x, user_y);

    }
}
```

<입력>

- user_input : 사용자 입력 값
- HP : 20 (위에 정의함)
- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도
- bool inMap = checkXY(user_x, mapX, user_y, mapY)

<결과>
- 상 입력시 위로 한 칸 이동하기
- 맵을 벗어날 경우 에러 메시지 출력후 재 입력 요청하기
- 지도와 현재 위치 출력
- 출력 후 while문 초반으로 이동

<설명>
- 2차원 배열을 이용해 상을 입력하면 y에 -1을 더해준다.
- checkXY함수를 이용해 사용자 위치를 체크한다.
- if문을 이용해 맵을 벗어나면(inMap==false) 다시 돌아간다.
  그렇지 않을 경우에는 위로 올라간다.
- displayMap 함수를 이용해 지도와 현재 위치 출력한다.

2. “하”를 입력했을 때

<함수 스크린샷>

```
//사용자가 하를 입력했을 때
else if (user_input == "하") {
    // 아래로 한 칸 내려가기
    user_y += 1;
    HP -= 1;

    //checkXY를 이용해 유효한 좌표인지 확인하고, 맞으면 내려가고 아니면 다시 돌아가기
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y -= 1;
        HP += 1;
    }
    else {
        cout << "위로 한 칸 내려갑니다." << endl;
        displayMap(map, user_x, user_y);

    }
}
```

<입력>
- user_input : 사용자 입력 값
- HP : 20 (위에 정의함)
- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도
-bool inMap = checkXY(user_x, mapX, user_y, mapY)

<결과>
- 하 입력시 아래로 한 칸 이동하기
- 맵을 벗어날 경우 에러 메시지 출력후 재 입력 요청하기
- 지도와 현재 위치 출력
- 출력 후 while문 초반으로 이동

<설명>

- 2차원 배열을 이용해 하를 입력하면 y에 1을 더해준다.
- checkXY함수를 이용해 사용자 위치를 체크한다.
- if문을 이용해 맵을 벗어나면 (inMap==false) 다시 돌아간다.
  그렇지 않을 경우에는 아래로 내려간다.
- displayMap 함수를 이용해 지도와 현재 위치를 출력한다.

3. “좌”를 입력했을 때

<함수 스크린샷>

```
    //사용자가 좌를 입력했을 때
    else if (user_input == "좌") {
        // 왼쪽으로 이동하기
        // user_x는 가로축 배열이기 때문에 1을 빼주면 왼쪽으로 이동
        user_x -= 1;
        HP -= 1;

        bool inMap = checkXY(user_x, mapX, user_y, mapY);

        if (inMap == false) {
            cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
            user_x += 1;
            HP += 1;
        }
        else {
            cout << "왼쪽으로 이동합니다." << endl;
            displayMap(map, user_x, user_y);

        }
    }
```

<입력>

- user_input : 사용자 입력 값
- HP : 20 (위에 정의함)
- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도
-bool inMap = checkXY(user_x, mapX, user_y, mapY)

<결과>

- 좌 입력시 왼쪽으로 한 칸 이동하기
- 맵을 벗어날 경우 에러 메시지 출력후 재 입력 요청하기
- 지도와 현재 위치 출력
- 출력 후 while문 초반으로 이동

<설명>

- 2차원 배열을 이용해 좌를 입력하면 x에 -1을 더해준다.
- checkXY함수를 이용해 사용자 위치를 체크한다.
- if문을 이용해 맵을 벗어나면(inMap==false) 다시 돌아간다.
  그렇지 않을 경우에는 왼쪽으로 이동한다.
- displayMap 함수를 이용해 지도와 현재 위치 출력한다.

4. “우”를 입력했을 때

<함수 스크린샷>

```
//사용자가 우를 입력했을 때
else if (user_input == "우") {
    // 오른쪽으로 이동하기
    user_x += 1;
    HP -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_x -= 1;
        HP += 1;
    }
    else {
        cout << "오른쪽으로 이동합니다." << endl;
        displayMap(map, user_x, user_y);

    }
}
```

<입력>

- user_input : 사용자 입력 값
- HP : 20 (위에 정의함)
- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도
- bool inMap = checkXY(user_x, mapX, user_y, mapY)

<결과>

- 우 입력시 오른쪽으로 한 칸 이동하기
- 맵을 벗어날 경우 다시 돌아가기
- 지도와 현재 위치 출력
- 출력 후 while문 초반으로 이동

<설명>
- 2차원 배열을 이용해 우를 입력하면 x에 1을 더해준다.
- checkXY함수를 이용해 사용자 위치를 체크한다.
- if문을 이용해 맵을 벗어나면(inMap==false) 다시 돌아간다.
  그렇지 않을 경우에는 오른쪽으로 이동한다.
- displayMap 함수를 이용해 지도와 현재 위치 출력한다.

5. “지도”를 입력했을 때

<함수 스크린샷>

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

<입력>
- -user_input : 사용자 입력 값
- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도

<결과>
- 지도와 현재 위치 출력

<설명>
- 만약 사용자가 지도를 입력하면 displayMap 함수를 이용해 지도와 현재 위치를 출력한다.

6. “종료”를 입력했을 때

<함수 스크린샷>

```
//사용자가 종료를 입력했을 때
else if (user_input == "종료") {
    cout << "종료합니다.";
    break;
}
```

<입력>
- -user_input : 사용자 입력 값

<결과>
- 종료 하기

<설명>
- break 사용해서 프로그램 종료한다.

7. 잘못 입력했을 때
<함수 스크린샷>

```
//잘못 입력했을 때
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

<입력>
-없음

<결과>
- 잘못 입력했다고 출력

<설명>
- 다른 걸 입력하면 잘못 입력했다고 출력한다.
- continue 이용해서 다시 처음으로 돌아간다.

4) 목적지에 도착하면 성공을 출력하고 종료
<함수 스크린샷>

```
// 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>
- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도

<결과>
- 목적지에 도착하면 게임 종료

<설명>
- 목적지에 도착하면 break를 이용해서 게임 종료한다.

5) 사용자가 이동할 때마다 사용자 체력 1씩 감소

<함수 스크린샷>

- 상을 입력했을 때의 예시 화면

```
if (user_input == "상") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    HP -= 1; //움직일 때마다 HP 1씩 감소

    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y += 1;
        // 맵을 벗어날 땐 다시 이동하기 때문에 1을 다시 더해줌
        HP += 1;
    }
    else {
        cout << "위로 한 칸 올라갑니다." << endl;
        displayMap(map, user_x, user_y);

    }
}
```

<입력>

- HP의 초기값 =20

<결과>

- 사용자가 이동할 때 마다 체력 1씩 감소

<설명>

- 상,하,좌,우로 이동 할 때마다 HP에 -1을 해준다.
- 맵을 벗어날 땐 다시 이동해야 되기 때문에 1을 더해준다.

6) 아이템, 포션, 적을 만났을 때 그에 대한 메시지를 출력

<함수 스크린샷>

```
// 사용자의 위치 정보 체크
checkState(map, user_x, user_y);
```

<입력>

- user_y= 유저 y값
- user_x= 유저 x값
- map = 지도

<결과>
- 사용자가 이동할 때 마다 체력 1씩 감소

<설명>
- 상,하,좌,우로 이동 할 때마다 HP에 -1을 해준다.
- 맵을 벗어날 땐 다시 이동해야 되기 때문에 1을 더해준다.

7) 처음 명령문을 입력받을 때마다 HP 함께 출력
<함수 스크린샷>

```
cout << "현재 HP: " << HP << "     명령어를 입력하세요 (상,하,좌,우,지도,종료): ";
```

<입력>
- HP =20

<결과>
- 처음 명령문을 입력 받을 때마다 HP 출력

<설명>
-cout를 이용해 현재 HP 출력한다.

8) HP가 0이 되면 실패를 출력하고 종료
<함수 스크린샷>

```
// 만약 HP가 0이면 종료
if (HP == 0) {
    cout << "HP가 0 이하가 되었습니다. 실패했습니다.";
    cout << "게임을 종료합니다.";
    break;
}
```

<입력>
- HP =20

<결과>
- HP가 0이 되면 종료

<설명>
-HP가 0이 되면 break를 이용해 프로그램 종료한다.

<기능 별 함수 구현>

1. 지도와 현재 위치 출력 함수 : displayMap()

<함수 스크린샷>

```
// 지도와 사용자 위치 출력하는 함수
void displayMap(int map[][mapX], int user_x, int user_y) {
    //i가 세로 번호
    for (int i = 0; i < mapY; i++) {
        //j는 가로 번호
        for (int j = 0; j < mapX; j++) {
            //i와 user_y가 같고, j와 user_x가 같으면 user 출력
            if (i == user_y && j == user_x) {
                cout << " USER |"; // 양 옆 1칸 공백
            }
            //다르면 지도 출력 (0은 빈공간, 1은 아이템, 2는 적 , 3은 포션, 4는 목적지)
            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>

- int map[][] = 전체 지도
- user_x = 유저 x 값
- user_y = 유저 y값

<반환값>

-없음

<결과>

- 전체 지도 출력
- 사용자 위치 출력

<설명>

- 사용자 위치와 동일한 좌표를 발견할 경우 사용자를 출력한다.
- 다를 경우에는 2차원 배열에 있는 지도를 출력한다.
- 0은 빈공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지를 출력한다.

2. 사용자 위치 체크 함수 : checkXY()

<함수 스크린샷>

```
// 이동하려는 곳이 유효한 좌표인지 체크하는 함수
bool checkXY(int user_x, int mapX, int user_y, int mapY) {
    bool checkFlag = false;
    //만약 user_x,user_y가 0보다 크거나 같고, mapX,mapY보다 작으면 유효한 좌표이기 때문에 true
    if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) {
        checkFlag = true;
    }
    return checkFlag;
}
```

<입력>

- user_x = 유저 x 값
- user_y = 유저 y값

<반환값>

- checkFlage

<결과>

- 좌표가 유효한 값인지 확인

<설명>

- bool 논리 연산자를 이용해 checkFlag 초기값을 False로 설정한다.
- if 조건문을 이용해서 유효한 좌표면 true를 반환한다.

3) 목적지에 도착 체크 함수 : checkGoal()

<함수 스크린샷>

```
// 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>

- user_x = 유저 x 값
- user_y = 유저 y값
- int map[][] = 전체 지도

<반환값>

- True, False

<결과>

- 유저의 위치가 목적지인지 확인

<설명>

- 만약 유저의 위치가 4이면 True 반환
- 4가 아니면 False 반환

4) HP 체크 함수 : chectState()

<함수 스크린샷>

```
// 아이템, 포션,적을 만났을 때 그에 대한 메시지 출력
void checkState(int map[][mapX], int user_x, int user_y) {

    switch (map[user_y][user_x]) {
    case 1:
        cout << "아이템이 있습니다" << endl;
        break;
    case 2:
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
        HP -= 2;
        break;
    case 3:
        cout << "포션이 있습니다. HP가 2 늘어납니다." << endl;

        HP += 2;
        break;
    }
}
```

<입력>
- user_x = 유저 x 값
- user_y = 유저 y값
- int map[][] = 전체 지도

<반환값>
- 없다.

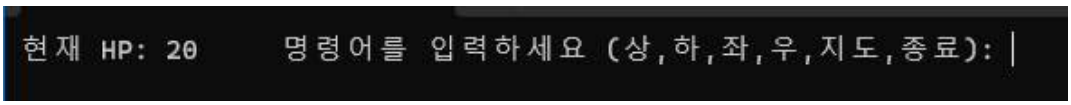
<결과>
- 아이템, 포션, 적을 만났을 때 그에 대한 메시지와 HP 출력

<설명>
- switch문 이용해서 1일 경우 “아이템이 있습니다.” 출력한다.
- 2일 경우에는 “적이 있습니다. HP가 2 줄어듭니다.” 출력하고 HP에서 2 줄인다.
- 3일 경우에는 “포션이 있습니다. HP가 2 늘어납니다.” 출력하고 HP에 2 더한다.

4. 테스트

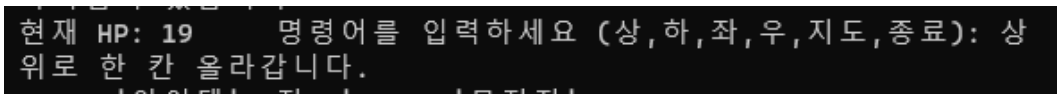
1. 기능별 테스트

1) 사용자는 체력 20을 가지고 게임 시작

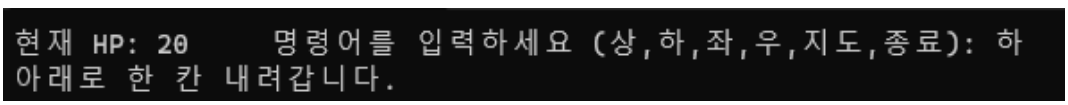


2) 사용자에게 “상”,“하”,“좌”,“우”,“지도”,“종류” 중 하나 입력받기

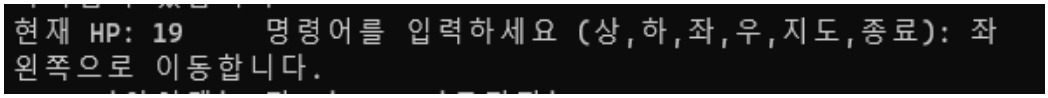
1. “상” 을 입력했을 때



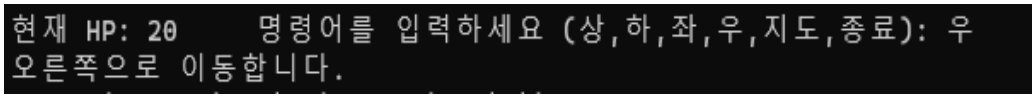
2. “하”를 입력했을 때



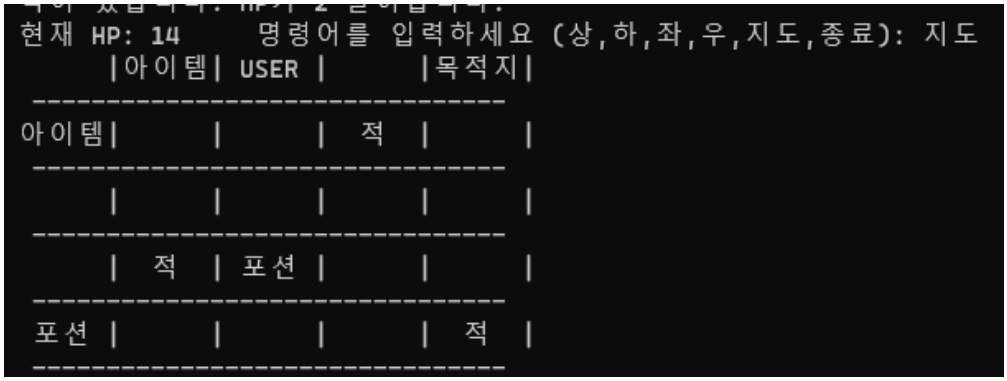
3. “좌”를 입력했을 때



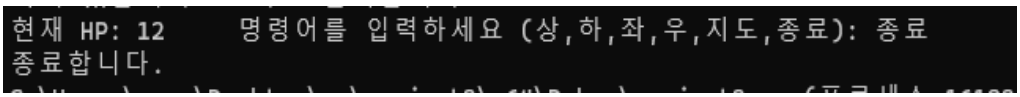
4. “우”를 입력했을 때



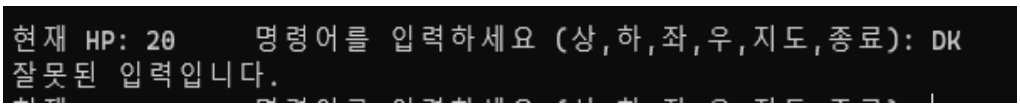
5. “지도”를 입력했을 때



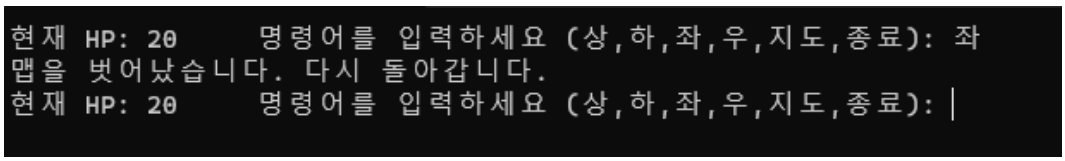
6. “종료”를 입력했을 때



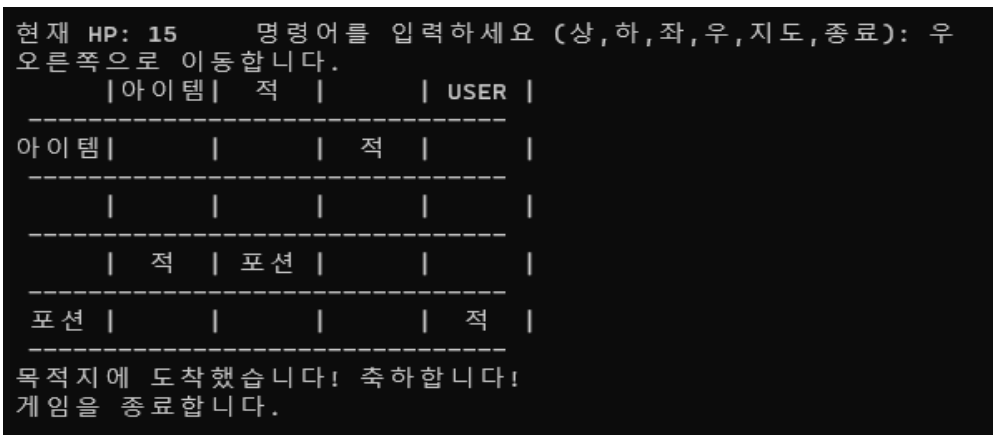
7. 잘못 입력했을 때



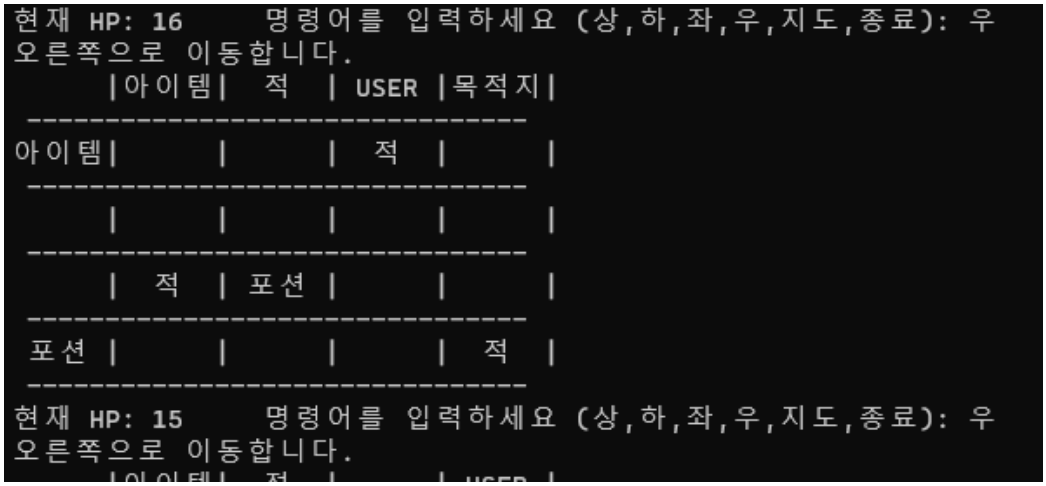
3) 지도 밖으로 나가게 되면 에러 메시지 출력



4) 목적지에 도착하면 성공을 출력하고 종료

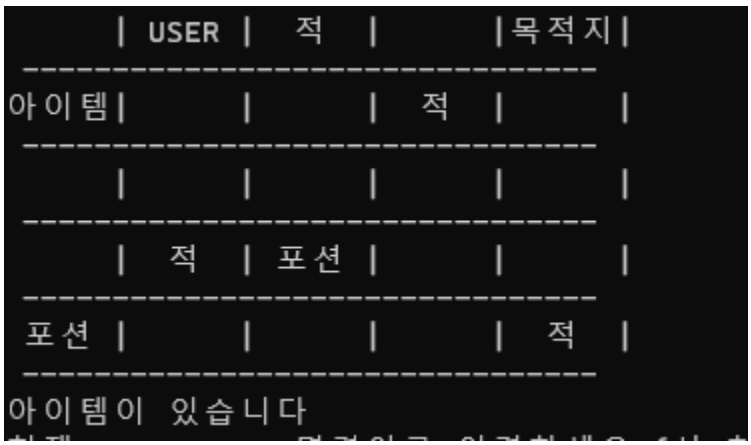


5) 사용자가 이동할 때마다 사용자 체력 1씩 감소

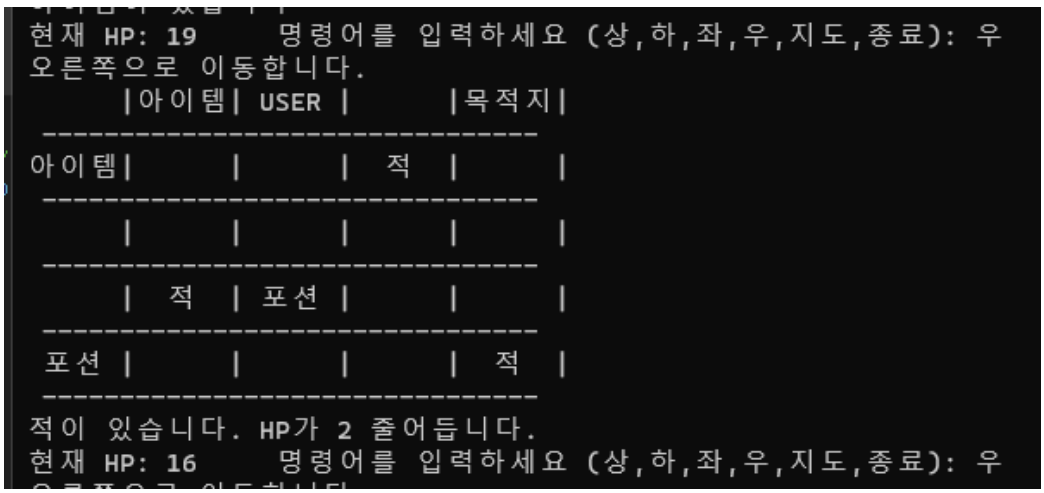


6) 아이템, 포션, 적을 만났을 때 그에 대한 메시지를 출력

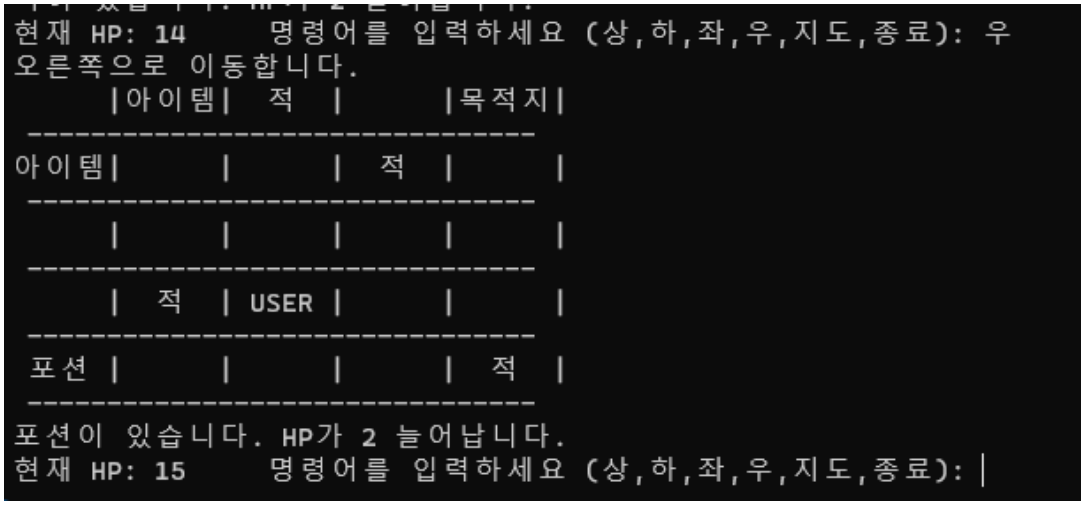
1. 아이템 만났을 때



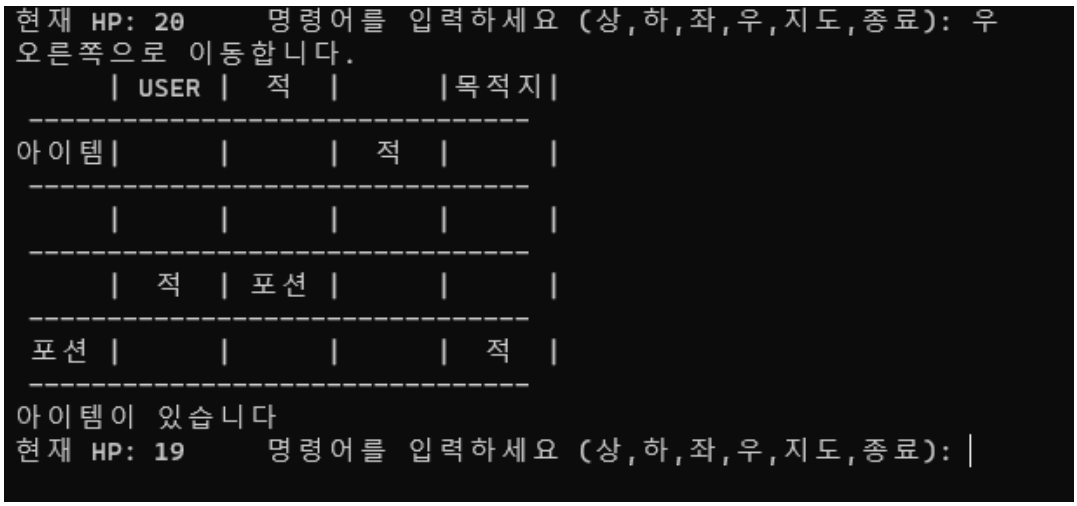
2. 적 만났을 때



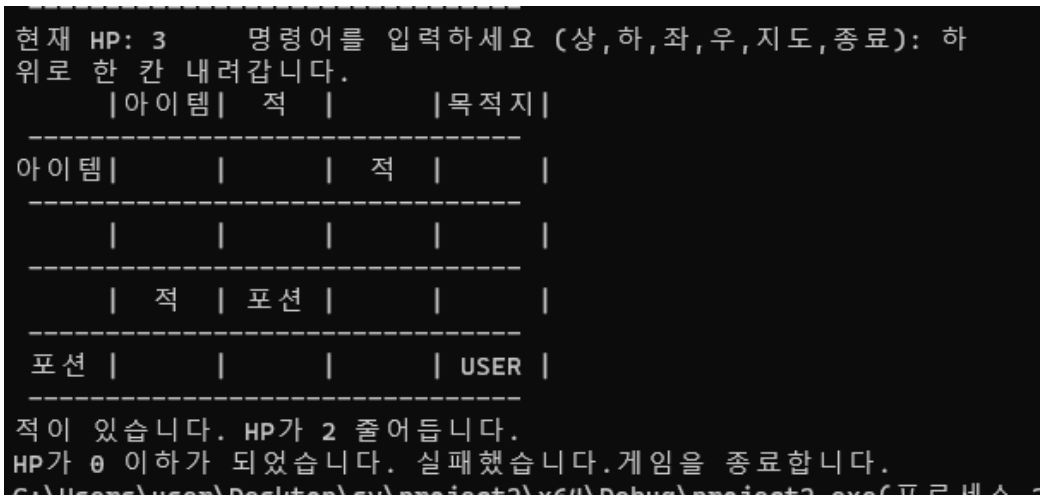
3. 포션 만났을 때



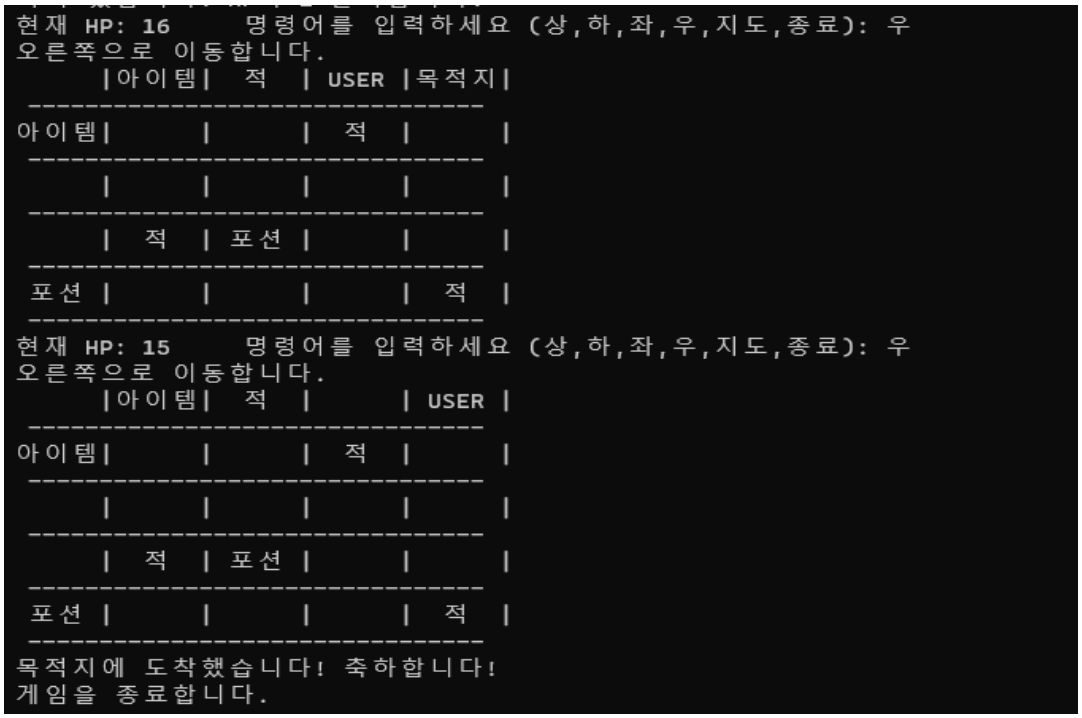
7) 처음 명령문을 입력 받을 때마다 HP 함께 출력



8) HP가 0이 되면 실패를 출력하고 종료



2. 최종 테스트 스크린 샷



5. 결과 및 결론

1. 프로젝트 결과 :

   mud game을 만들어서 게임을 할 수 있다.

2. 느낀점 :

   금요일까지 시험이 있었기 때문에 프로젝트 할 수 있는 시간이 많이 부족했다. 그래서 보고서 작성과 코드 이해를 촉박하게 했다는 게 아쉬운 부분인 거 같다. 그래도 결과물을 보니 다 해냈다는 점이 뿌듯했다. 너무 어려운 내용이었지만 코드를 하나하나씩 이해를 해보니 앞으로 어떤 식으로 접근해야 할지 조금 알게 된 거 같다. 우리가 수업시간에 함수에 대해 배웠는데 함수가 가장 어려웠고, 이걸 어떻게 써야 하는지도 잘 몰랐다. 하지만 직접 만들어 보니 함수를 이용해서 코드를 짜는 게 재사용하기도 좋고, 편하다는 것을 알게 되었다. 또한 보고서 작성하면서 기능별로 테스트 할 때 내가 몰랐던 오류도 찾을 수 있었고, 어떤 걸 만들고 싶은 지 바로 보여서 왜 보고서의 필요성에 대해서도 알게 되었다. 저번 프로젝트에 이어서 이번 프로젝트도 어렵긴 했지만 조금씩 알아가고 있는 내 모습을 보니 앞으로도 더 열심히 해보고 싶다.