1. 서론

1-1. 프로젝트 목적 및 배경
: 7주차까지의 학습 내용을 통합하여 실습형 프로젝트로 진행됩니다. 조건문, 반복문, 함수화 등을 중점적으로 활용하여 프로그래밍 기초 능력을 강화하는 것을 목표로 합니다.

1-2. 목표
: 간단한 Mud 게임 구현을 목표로 합니다.

2. 요구사항

2-1. 사용자 요구사항
: 유저가 아이템과 포션 그리고 적이 있는 맵에서 상하좌우로 이동하며 목적지에 도착하는 게임입니다.

2-2. 기능 계획

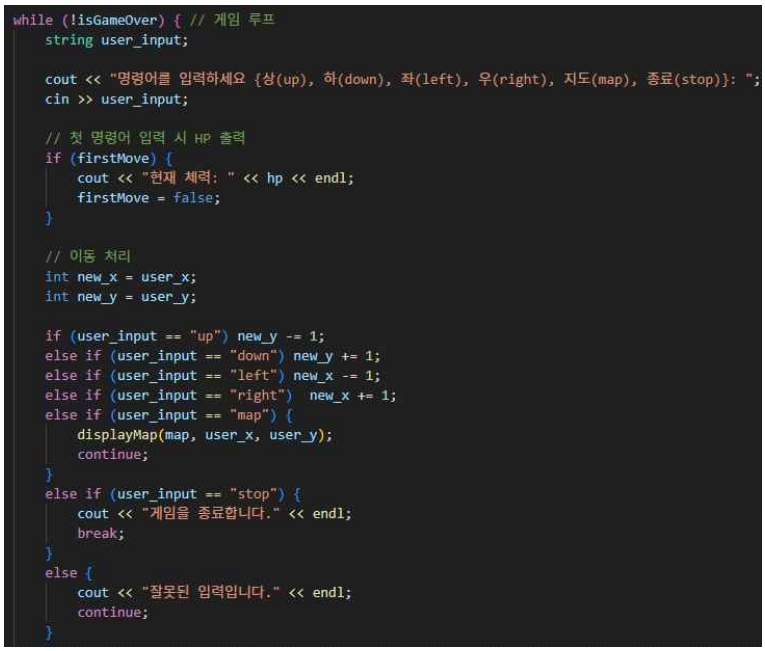
① 사용자에게 “상”, ”하”, ”좌”, ”우”, “지도”, “종료” 중 하나를 입력 받기
② 지도 밖으로 나가게 되면 에러 메시지 출력
③ 목적지에 도착하면 “성공”을 출력하고 종료
④ 유저는 체력 20을 가지고 게임 시작
⑤ 사용자가 이동할 때 마다 현재 맵을 출력하고, 사용자 체력 1씩 감소
⑥ 처음 명령문을 입력 받을 때 마다 HP 함께 출력
⑦ HP가 0이 되면 “실패”를 출력하고 종료
⑧ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력
⑨ 적을 만날 경우 HP가 2가 줄어들고 그에 대한 추가 메시지 출력
⑩ 포션을 만날 경우 HP가 2가 늘어나고 그에 대한 추가 메시지 출력

2-3. 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 맵과 사용자 위치 상태 확인 함수 : checkState()

3. 설계 및 구현

3-1. 기능 별 구현 사항:
①에 대한 구현
1. 스크린샷



```
while (!isGameOver) { // 게임 루프
    string user_input;

    cout << "명령어를 입력하세요 {상(up), 하(down), 좌(left), 우(right), 지도(map), 종료(stop)}: ";
    cin >> user_input;

    // 첫 명령어 입력 시 HP 출력
    if (firstMove) {
        cout << "현재 체력: " << hp << endl;
        firstMove = false;
    }

    // 이동 처리
    int new_x = user_x;
    int new_y = user_y;

    if (user_input == "up") new_y -= 1;
    else if (user_input == "down") new_y += 1;
    else if (user_input == "left") new_x -= 1;
    else if (user_input == "right")  new_x += 1;
    else if (user_input == "map") {
        displayMap(map, user_x, user_y);
        continue;
    }
    else if (user_input == "stop") {
        cout << "게임을 종료합니다." << endl;
        break;
    }
    else {
        cout << "잘못된 입력입니다." << endl;
        continue;
    }
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
    - int map[][] = 전체 지도 배열로, 게임 공간의 맵 정보를 담고 있습니다.
    - user_x = 사용자 위치의 x 좌표 값.
    - user_y = 사용자 위치의 y 좌표 값.
3. 반환값 (함수의 경우 작성)
    - 없음
4. 결과 (블록/함수가 종료된 결과)
    - 전체 지도를 출력 : 현재 맵의 상태를 콘솔에 표시합니다.
    - 사용자 위치를 출력 : 유저가 이동한 새로운 좌표 위치를 맵과 함께 출력합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
    - 사용자 입력 처리 : 사용자에게 이동 명령을 입력받습니다. 한글이 입력되지 않아 영어로 up, down, left, right, map, stop을 입력받으면 각 명령을 실행합니다.
    - 이동 처리 : 입력된 명령에 따라 사용자 좌표를 user_x와 user_y에 갱신합니다.
    - 맵 출력 명령 : map을 입력 시 displayMap() 함수를 통해 현재 지도를 화면에 표시합니다.
    - 게임 종료 명령 : stop을 입력 시 게임을 종료한다는 메시지를 출력하고 break로 while문을 탈출합니다.
    - 잘못된 입력 처리 : 앞서 정의한 입력과 다른 입력이 들어오면 오류 메시지를 출력하고 continue;를 통해 다시 한번 명령을 입력받습니다.

②에 대한 구현

1. 스크린샷

```
// 맵 경계 검증
if (checkXY(new_x, new_y)) {
    user_x = new_x;
    user_y = new_y;
    if (user_input == "up") cout << "위로 이동합니다." << endl;
    else if (user_input == "down") cout << "아래로 이동합니다." << endl;
    else if (user_input == "left") cout << "왼쪽으로 이동합니다." << endl;
    else if (user_input == "right")  cout << "오른쪽으로 이동합니다." << endl;

    // 이동 시 체력 1 감소
    hp -= 1;
    cout << "체력이 1 감소했습니다. 현재 체력: " << hp << endl;

    // 현재 위치 상태 확인 (checkState 함수 호출)
    checkState(map, user_x, user_y, hp, isGameOver);

} else {
    cout << "맵을 벗어났습니다. 다시 들아갑니다." << endl;
}
```

```
// 맵 경계 확인 함수
bool checkXY(int x, int y) {
    return (x >= 0 && x < mapX && y >= 0 && y < mapY);
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
    - new_x, new_y: 사용자가 이동하고자 하는 새로운 위치의 좌표 값.
    - user_x, user_y: 현재 사용자의 위치 좌표.
    - map: 전체 맵 정보 배열.
    - int mapX, int mapY: 맵의 가로(x)와 세로(y) 크기인 5입니다.
3. 반환값 (함수의 경우 작성)
    - bool : checkXY함수는 주어진 좌표가 맵의 경계 내에 있는지 여부를 반환합니다.
4. 결과 (블록/함수가 종료된 결과)
    - 유저 위치 갱신: 사용자의 좌표(user_x, user_y)가 업데이트됩니다.
    - 체력 감소: 이동 시 체력이 1 감소합니다.
    - 맵 경계 확인 결과 출력: 사용자가 맵을 벗어난 경우 경고 메시지를 출력합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
    - 맵 경계 검증: checkXY(new_x, new_y) 함수를 통해 사용자가 이동할 위치가 맵의 범위 내에 있는지 확인합니다.
    - 체력 감소: 사용자가 한 번 이동할 때마다 체력이 1씩 감소하며, 현재 체력을 출력합니다.
    - 위치 상태 확인: checkState() 함수를 호출하여 현재 맵 상태와 사용자의 위치, 체력, 게임 종료 여부를 확인합니다.
    - 맵 경계 벗어남 처리: 만약 사용자가 맵 경계를 벗어난 경우 메시지를 출력합니다.

③에 대한 구현

1. 스크린샷

```
// 목적지 도달 여부 체크
if (checkGoal(map, user_x, user_y)) {
    cout << "목적지에 도착했습니다! 축하합니다!" << endl;
    cout << "게임을 종료합니다." << endl;
    break;
}
```

```
// 목적지 도달 여부 확인 함수
bool checkGoal(int map[][mapX], int user_x, int user_y) {
    return (map[user_y][user_x] == 4);
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
   - new_x, new_y: 사용자가 이동하고자 하는 새로운 위치의 좌표 값.
   - user_x, user_y: 현재 사용자의 위치 좌표.
   - map: 전체 맵 정보 배열.
3. 반환값 (함수의 경우 작성)
   - bool : 해당 위치의 값이 4일 경우 true, 그 외에는 false를 반환합니다
4. 결과 (블록/함수가 종료된 결과)
   - 목적지 도달 메시지 출력: 사용자가 목표 위치에 도달했음을 알립니다.
   - 게임 종료 처리: 게임 종료 메시지를 출력한 후, break 명령을 통해 게임 루프를 종료합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
   - 목적지 도달 여부 확인 함수 checkGoal
   : 사용자의 현재 위치(user_x, user_y)에서 해당 맵 좌표의 값이 4인지 확인하고, 만약 맵의 해당 좌표 값이 4라면 true를 반환하여 목적지에 도달했음을 알립니다.
   - 목적지 도달 시 처리
   : 게임 루프 내에서 사용자가 목적지에 도달했는지 checkGoal 함수로 확인합니다. 도달한 경우에는 성공 메시지를 출력하고 break를 통해 게임 루프를 종료합니다.

④에 대한 구현

1. 스크린샷

```
const int initialHP = 20; // 초기 체력
```

```
int hp = initialHP; // 초기 체력 설정
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
   - initialHP: 사용자의 초기 체력 값으로 이 값은 20으로 고정됩니다.
   - hp: 현재 사용자의 체력을 나타내는 변수로, 처음에는 initialHP 값인 20으로 초기화됩니다
3. 반환값 (함수의 경우 작성)
   - 없음
4. 결과 (블록/함수가 종료된 결과)
   - hp 초기화 : 사용자의 체력이 initialHP값인 20으로 설정됩니다.

⑤에 대한 구현

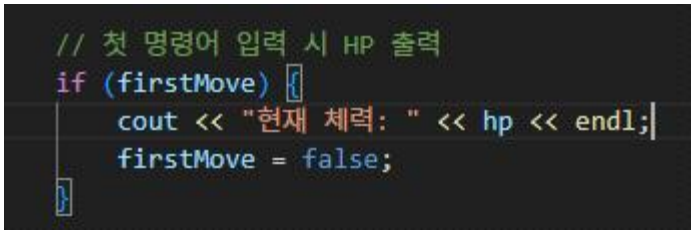
1. 스크린샷

```
// 맵과 사용자 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 << endl << " ------------------------------- " << endl;
    }
}
```

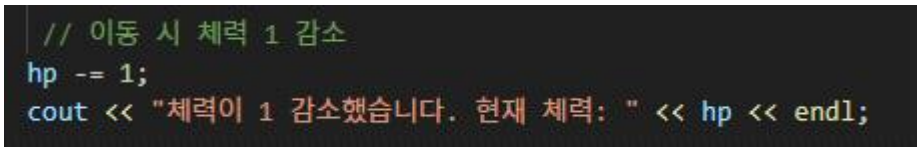
2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
    - map[][]: 맵의 정보를 담은 2차원 배열로, 각 칸에 다양한 객체가 저장됩니다.
    - user_x: 사용자의 현재 x 좌표 값입니다.
    - user_y: 사용자의 현재 y 좌표 값입니다.
3. 반환값 (함수의 경우 작성)
    - 반환값 없음: 이 함수는 현재 맵과 사용자 위치를 콘솔에 출력할 뿐, 별도의 반환값을 제공하지 않습니다.
4. 결과 (블록/함수가 종료된 결과)
    - 현재 맵의 상태를 콘솔에 출력합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
    - 맵 출력: 2중 for문을 이용한 맵을 시각적으로 출력.
    - 사용자 위치 표시: 현재 루프의 좌표(i, j)가 사용자 위치와 일치할 경우 "USER"를 표시합니다.
    - 맵 내 객체 확인 및 출력: 사용자 위치가 아닌 경우, switch문을 이용해 맵의 각 칸에 저장된 값에 따라 출력합니다.
    - 출력 마무리: 한 행이 출력된 후, 줄 구분선을 추가합니다.

⑥에 대한 구현

1. 스크린샷



```
// 첫 명령어 입력 시 HP 출력
if (firstMove) {
    cout << "현재 체력: " << hp << endl;
    firstMove = false;
}
```

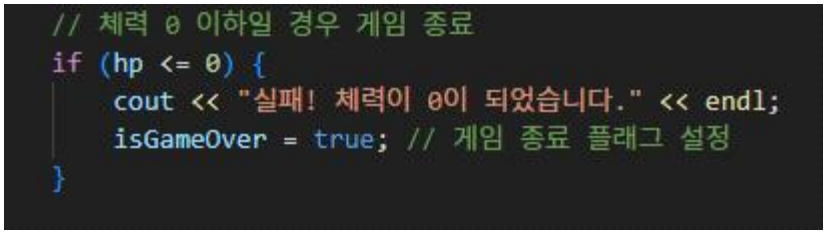


```
// 이동 시 체력 1 감소
hp -= 1;
cout << "체력이 1 감소했습니다. 현재 체력: " << hp << endl;
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
   - firstMove: 사용자가 첫 명령을 입력했는지를 확인하는 변수. 처음에는 true로 설정됩니다.
   - hp: 사용자의 현재 체력 값.
3. 반환값 (함수의 경우 작성)
   - 없음
4. 결과 (블록/함수가 종료된 결과)
   - 현재 체력 출력: 사용자의 현재 체력이 출력됩니다.
   - firstMove 값 변경: 첫 명령어가 입력되면 firstMove가 false로 변경되어 이후 이 코드가 다시 실행되지 않도록 합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
   - 첫 명령어 입력 여부 확인: firstMove가 true일 경우, 사용자의 체력을 출력하는 코드가 실행됩니다.
   - 현재 체력 출력: cout << "현재 체력: " << hp << endl;를 통해 현재 체력 값이 표시됩니다.
   - firstMove 상태 변경: 첫 번째 명령어 입력 이후 firstMove = false로 설정되어 이후의 명령어 입력 시 더 이상 if문이 실행되지 않습니다. 추후에 hp 출력은 이동 시 hp를 감소시키고, 체력이 출력되게 했습니다.

⑦에 대한 구현

1. 스크린샷

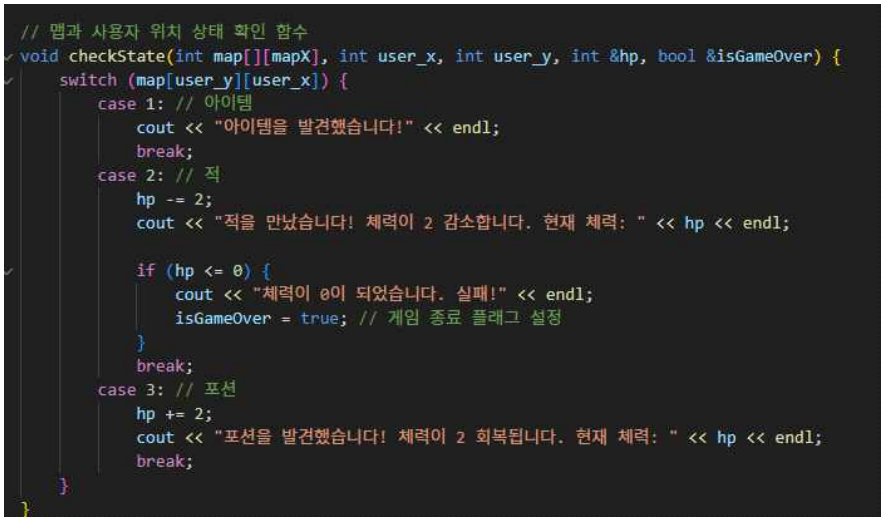


```
// 체력 0 이하일 경우 게임 종료
if (hp <= 0) {
    cout << "실패! 체력이 0이 되었습니다." << endl;
    isGameOver = true; // 게임 종료 플래그 설정
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
    - hp: 사용자의 현재 체력 값.
    - isGameOver: 게임 종료 여부를 나타내는 불리언 변수. 체력이 0 이하가 되면 true로 설정됩니다.
3. 반환값 (함수의 경우 작성)
    - 없음
4. 결과 (블록/함수가 종료된 결과)
    - 실패 메시지 출력: 체력이 0 이하가 되면 실패 메시지를 출력합니다.
    - 게임 종료 플래그 설정: isGameOver가 true로 설정되어 게임 루프가 종료됩니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
    - 체력 확인: if (hp <= 0) 조건을 통해 사용자의 체력이 0 이하인지 확인합니다.
    - 게임 루프 종료: isGameOver가 true로 설정되면 게임 루프가 종료되고, 더 이상 게임이 진행되지 않습니다.

⑧, ⑨, ⑩에 대한 구현

1. 스크린샷



```
// 맵과 사용자 위치 상태 확인 함수
void checkState(int map[][mapX], int user_x, int user_y, int &hp, bool &isGameOver) {
    switch (map[user_y][user_x]) {
        case 1: // 아이템
            cout << "아이템을 발견했습니다!" << endl;
            break;
        case 2: // 적
            hp -= 2;
            cout << "적을 만났습니다! 체력이 2 감소합니다. 현재 체력: " << hp << endl;

            if (hp <= 0) {
                cout << "체력이 0이 되었습니다. 실패!" << endl;
                isGameOver = true; // 게임 종료 플래그 설정
            }
            break;
        case 3: // 포션
            hp += 2;
            cout << "포션을 발견했습니다! 체력이 2 회복됩니다. 현재 체력: " << hp << endl;
            break;
    }
}
```

2. 입력 (블록/함수에 입력되는 변수, 값들과 설명)
    - map[][]: 맵의 정보를 담고 있는 2차원 배열
    - user_x: 사용자의 현재 x 좌표.
    - user_y: 사용자의 현재 y 좌표.
    - hp: 사용자의 체력 값.
    - isGameOver: 게임 종료 여부를 나타내는 플래그.
3. 반환값 (함수의 경우 작성)
    - 반환값 없음: 이 함수는 사용자의 현재 위치 상태에 따라 적절한 처리를 수행하며, 반환값은 없습니다.
4. 결과 (블록/함수가 종료된 결과)
    - 아이템 발견 시: 아이템 발견 메시지가 출력됩니다.
    - 적을 만났을 때: 체력이 2 감소하고, 이에 대한 메시지를 출력합니다. 만일 적을 만나 체력이 0이하가 되었을 때는 게임이 종료되도록 했습니다.
    - 포션을 발견할 때: 체력이 2 증가하고, 이에 대한 메시지를 출력합니다.
5. 설명 (코드 내 작동 순서, 내용 등 추가 설명)
    - 사용자가 위치한 칸의 상태 확인: map[user_y][user_x]의 값에 따라 switch문을 이용해 수행합니다. 아이템은 1, 적은 2, 포션은 3으로 처리합니다.
    - 아이템 발견 처리: 아이템이 1로 처리되기에 사용자가 위치한 칸이 1일 경우 아이템 발견 메시지를 출력합니다.
    - 적 발견 처리 : 적이 2로 처리되기에 사용자가 위치한 칸이 2일 경우 hp가 2 감소하고, 이에 대한 메시지를 출력합니다. 만일 적을 만나 체력이 0이하가 되면 게임이 종료되도록 했습니다.
    - 포션 발견 처리: 포션은 3으로 처리되기에 사용자가 위치한 칸이 3일 경우 체력이 2 증가하고 이에 대한 메시지를 출력합니다.

4. 테스트

①에 대한 결과

```
명령어를 입력하세요 {상(up), 하(down), 좌(left), 우(right), 지도(map), 종료(stop)}:
```

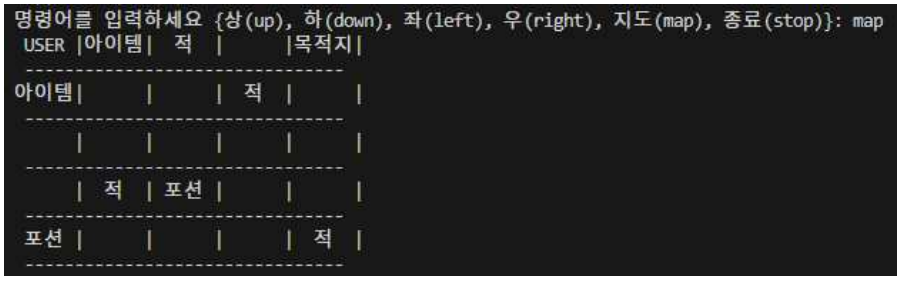
상, 하, 좌, 우, 지도, 종료 입력

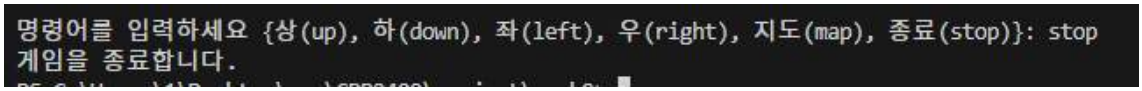
```
명령어를 입력하세요 {상(up), 하(down), 좌(left), 우(right), 지도(map), 종료(stop)}: up
위로 이동합니다.
체력이 1 감소했습니다. 현재 체력: 18
 USER |아이템|  적  |     |목적지|
 ------------------------------
아이템|     |     |  적  |     |
 ------------------------------
     |     |     |     |     |
 ------------------------------
     |  적  | 포션 |     |     |
 ------------------------------
 포션 |     |     |     |  적  |
 ------------------------------
```

```
명령어를 입력하세요 {상(up), 하(down), 좌(left), 우(right), 지도(map), 종료(stop)}: down
현재 체력: 20
아래로 이동합니다.
체력이 1 감소했습니다. 현재 체력: 19
아이템을 발견했습니다!
     |아이템|  적  |     |목적지|
 ------------------------------
 USER |     |     |  적  |     |
 ------------------------------
     |     |     |     |     |
 ------------------------------
     |  적  | 포션 |     |     |
 ------------------------------
 포션 |     |     |     |  적  |
 ------------------------------
```

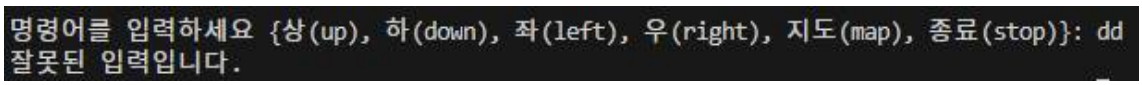
```
명령어를 입력하세요 {상(up), 하(down), 좌(left), 우(right), 지도(map), 종료(stop)}: left
왼쪽으로 이동합니다.
체력이 1 감소했습니다. 현재 체력: 16
 USER |아이템|  적  |     |목적지|
 ------------------------------
아이템|     |     |  적  |     |
 ------------------------------
     |     |     |     |     |
 ------------------------------
     |  적  | 포션 |     |     |
 ------------------------------
 포션 |     |     |     |  적  |
 ------------------------------
```

```
명령어를 입력하세요 {상(up), 하(down), 좌(left), 우(right), 지도(map), 종료(stop)}: right
오른쪽으로 이동합니다.
체력이 1 감소했습니다. 현재 체력: 17
아이템을 발견했습니다!
     | USER |  적  |     |목적지|
 ------------------------------
아이템|     |     |  적  |     |
 ------------------------------
     |     |     |     |     |
 ------------------------------
     |  적  | 포션 |     |     |
 ------------------------------
 포션 |     |     |     |  적  |
 ------------------------------
```



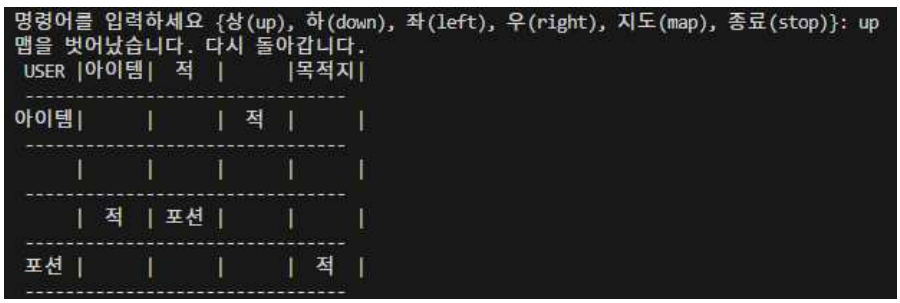


잘못된 입력 시



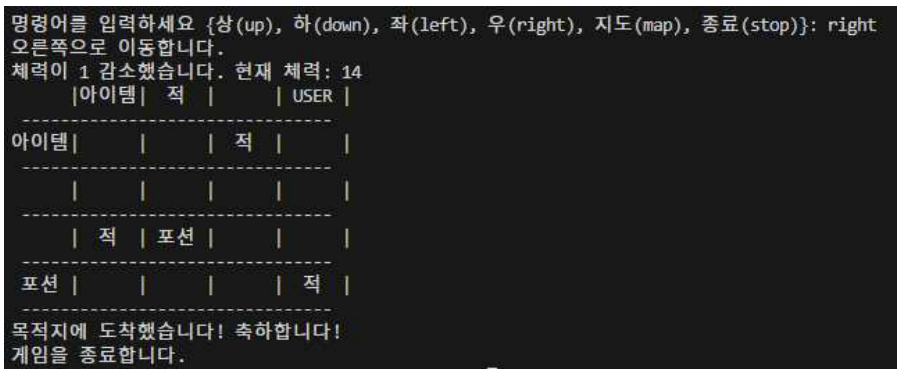
② 에 대한 결과

지도 밖으로 나가게 되면 에러 메시지 출력



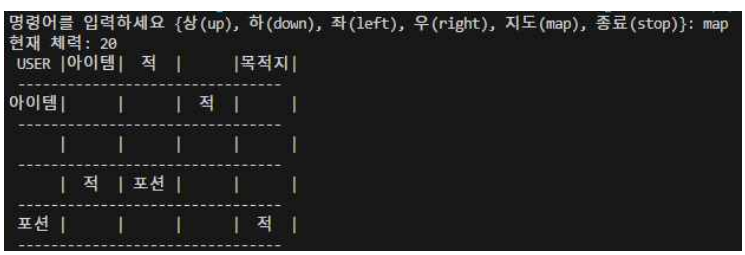
③ 에 대한 결과

목적지에 도착하면 “성공”을 출력하고 종료



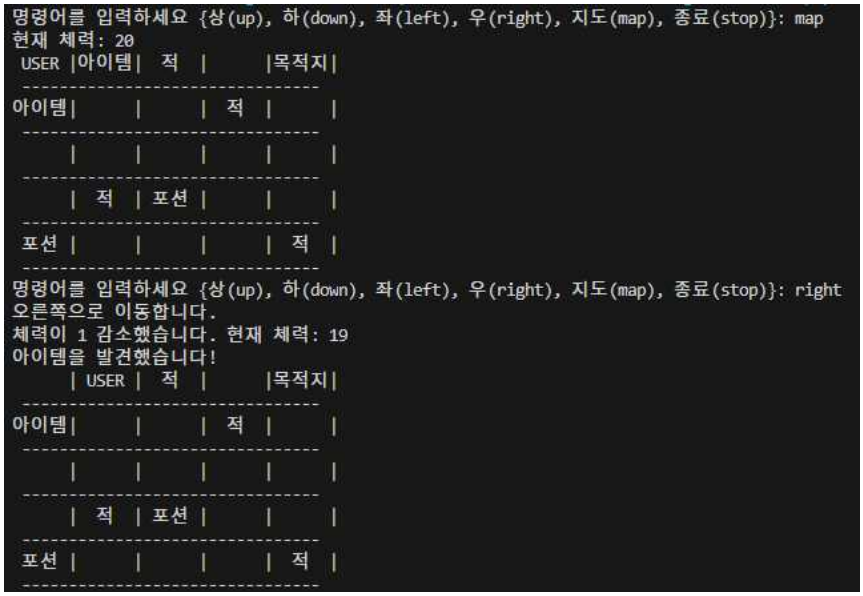
④ 에 대한 결과

유저는 체력 20을 가지고 게임 시작



⑤ 에 대한 결과

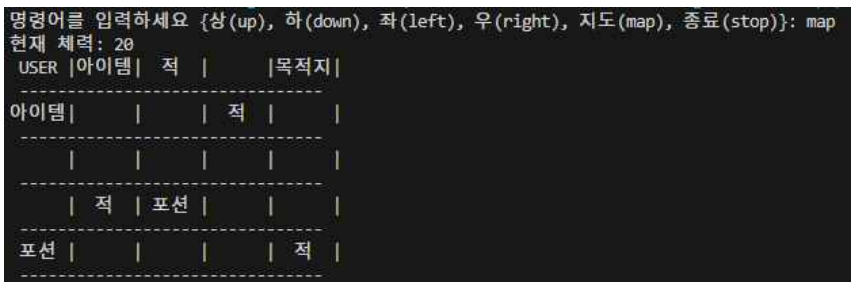
사용자가 이동할 때 마다 현재 맵을 출력하고, 사용자 체력 1씩 감소



시작 후 오른쪽 이동

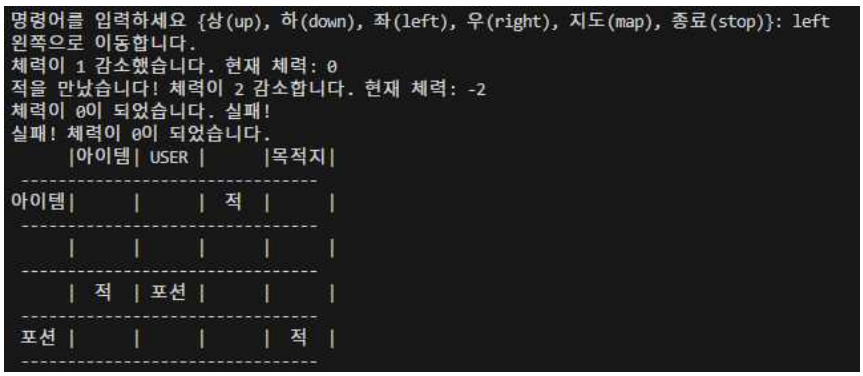
⑥ 에 대한 결과

처음 명령문을 입력 받을 때 마다 HP 함께 출력



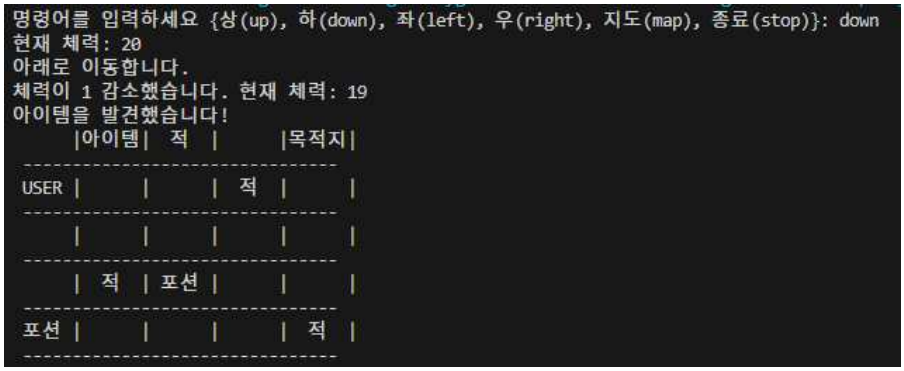
⑦ 에 대한 결과

HP가 0이 되면 “실패”를 출력하고 종료

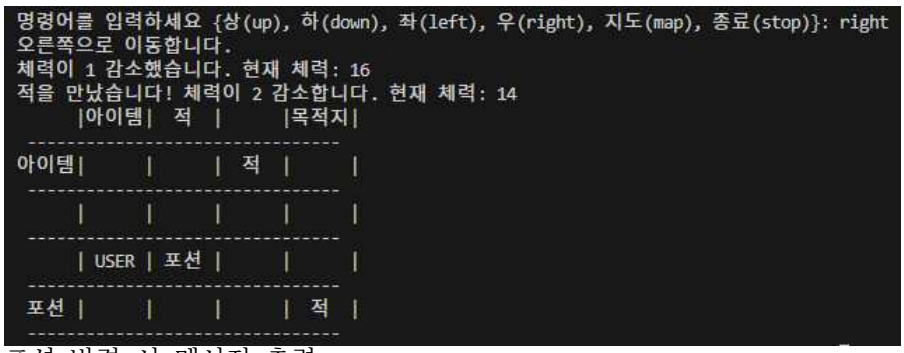


⑧ 에 대한 결과

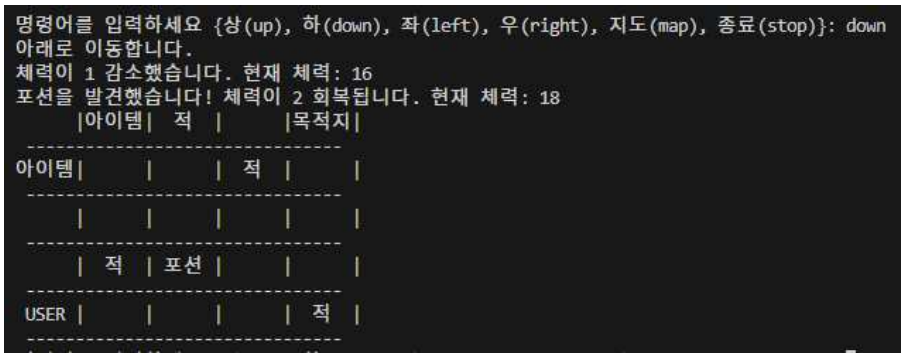
아이템 발견 시 메시지 출력



적 발견 시 메시지 출력

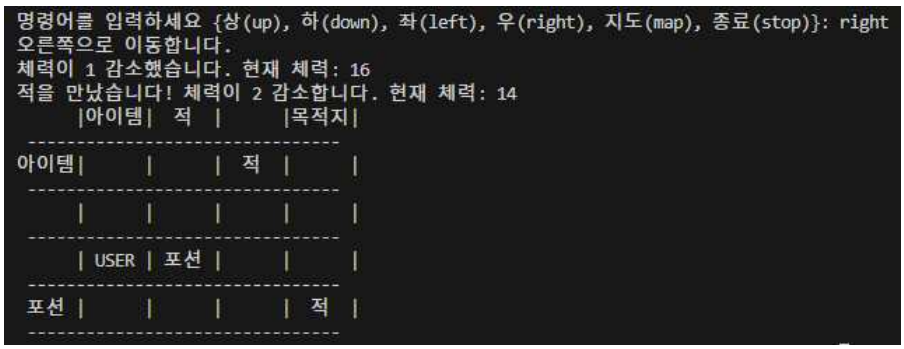


포션 발견 시 메시지 출력

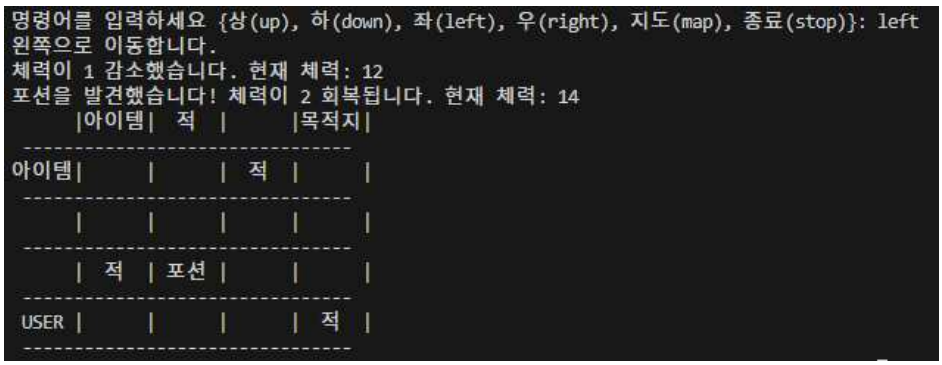


⑨ 에 대한 결과

적을 만날 경우 HP가 2가 줄어들고 그에 대한 추가 메시지 출력



⑩ 에 대한 결과
포션을 만날 경우 HP가 2가 늘어나고 그에 대한 추가 메시지 출력



5. 결과 및 결론

5-1. 프로젝트 결과
: 텍스트 기반 게임으로서 사용자의 명령에 따라 맵 위에서 상, 하, 좌, 우로 이동하며, 아이템과 적, 포션을 만나고 목적지에 도달하는 방식으로 구성됩니다. 체력을 도입함으로써 체력 관리를 주의하면서 목적지에 도달하는 것이 승리 조건인 게임입니다. 조건문과 함수화를 적극 활용하여 제작되었습니다.

5-2. 느낀 점: 개인적으로는 과거에 초등학교에서 노트에 적어서 했던 게임들이 생각났던 실습이었습니다. C++ 수업 중 배웠던 내용들을 골고루 이용한 것 같습니다. 조건문을 통해 입력받은 내용에 대한 명령을 수행하도록 하였고, switch문을 사용하여 아이템, 적, 포션에 도달했을 때 각 수행해야 할 명령이 진행되도록 했습니다. 또한, 함수화를 이용하여 메인 함수가 간소화됨을 알 수 있었습니다. 아이템이나 적의 위치를 랜덤하게 배치하거나 적과의 전투 시 체력 감소를 확률적으로 조정하는 등 개선할 수 있는 아이디어도 떠올랐습니다. 개인적으로는 게임 개발이 꿈이라서 재미있었습니다.