R E P O R T
[MUD GAME 만들기]

• 과 목 명	: C++프로그래밍및실습
• 담 당 교 수	: 김 미 수
• 제 출 일	: 2023.10.27.(금)
• 학 과	: 통 계 학 과
• 학 번	: 2 0 3 1 8 5
• 성 명	: 김 지 웅

<목차>

1. 서론
    1. 프로젝트 목적 및 배경
    2. 목표

2. 요구사항
    1. 사용자 요구사항
    2. 기능 구현 계획
    3. 함수 구현 계획

3. 설계 및 구현
    1. 기능별 구현사항
    2. 함수별 구현사항

4. 테스트
    1. 기능별 테스트 결과
    2. 최종 테스트 결과

5. 결과 및 결론
    1. 프로젝트 결과
    2. 느낀 점

1. 서론

1. 프로젝트 목적 및 배경

이 프로젝트는 C++ 언어를 활용하여 Multi-User Dungeon (MUD) 게임을 개발하는 것을 목표로 한다. MUD는 다중 플레이어가 함께 참여할 수 있는 텍스트 기반의 온라인 게임이다.

유저는 상하좌우로 이동하며 게임 내에 존재하는 여러 조건들을 이겨내고 목적지에 도착하면 승리하게 된다.

우리는 8주라는 수업 기간동안 이 게임을 개발하기 위한 내용들을 배워왔다. C++프로그램 언어를 복습하고 활용하는 데에 있어 수행 능력을 기르고 프로그래밍 및 게임 개발 능력 또한 향상시키기 위해 프로젝트를 진행한다.

2. 목표

프로젝트의 궁극적인 목표는 MUD 게임을 구현이다. 이와 더불어 사용자가 텍스트 기반 환경에서 상호 작용하며 게임을 즐길 수 있도록 하는 것도 주된 목표이다.

위의 목표를 달성하기 위해서는 기본적인 게임 요소와 기능을 구현하여 완성도 높은 게임을 제작하는 것이 중요하다. 게임을 만드는 과정에서 프로그래밍 언어를 다방면으로 이해하고 활용하는 것을 부가적인 목표로 설정한다.

2. 요구사항

1. 사용자 요구사항

사용자 관점에서 게임에 대한 기대와 특별한 기능에 대한 요구를 정의한다.

따라서 다음과 같이 사용자 요구사항을 작성할 수 있다.

- 유저는 상/하/좌/우/지도/종료 중 하나를 선택하기
- 상 : 위쪽으로 한 칸 이동하기
- 하 : 아래쪽으로 한 칸 이동하기
- 좌 : 왼쪽으로 한 칸 이동하기
- 우 : 오른쪽으로 한 칸 이동하기
- 지도 : 현재 위치 확인하기
- 종료 : 게임 종료하기

2. 기능 구현 계획

기능 구현 계획은 실제로 구현할 내용들에 대해 하나하나 상세하게 작성한다. 이는 프로그램의 메인 함수에서 표현할 것들로 플레이어의 이동, 맵 관리, 성공 및 실패 조건 등을 정의한 부분들이 필요하다.

따라서 다음과 같이 기능 요구사항을 작성할 수 있다.

- 플레이어는 HP 20으로 시작한다.
- 맵은 5x5 사이즈로 설정한다.
- 상/하/좌/우/지도/종료 중 하나를 입력 받는다.
- 맵 범위를 넘어 나가게 되면 에러로 처리하고 다시 입력 받는다.
- 플레이어가 목적지에 도달하면 "성공"을 출력하고 종료한다.

- 플레이어는 한 칸씩 이동할 때 HP가 1씩 감소한다.

- 명령문을 입력 받을 때마다 현재 HP를 같이 출력한다.

- HP가 0이 되면 "실패"를 출력하고 종료한다.

- 아이템을 만났을 때 "아이템을 만났습니다."를 출력한다.

- 포션을 만났을 때 "포션을 만났습니다."를 출력하고 HP를 2를 회복한다.

- 적을 만났을 때 "적을 만났습니다."를 출력하고 HP를 2를 감소한다.

3. 함수 구현 계획

   함수 구현 계획은 메인 함수에서 기능을 표현하기 위해 필요한 함수들을 어떻게 구현할 것인가에 대해 자세하게 작성한다.

   - 메인 함수 : 유저에게 값을 입력 받아, 그에 대한 기능을 표현한다. 예를 들어 플레이어가 "상"을 입력할 경우,

   - displayMap() 함수 :

   - checkXY() 함수 :

   - checkGoal() 함수:

   - checkState() 함수 :

3. 설계 및 구현

1. 기능별 구현 사항

- 맵 사이즈인 5x5인 배열로 설정하기 위한 상수 선언

```
//맵 사이즈 설정
const int mapX = 5;
const int mapY = 5;
```

- 유저의 기본 HP를 설정하는 변수 선언

```
//유저 체력 설정
int hp = 20;
```

- int map[][]를 활용하여 2차원 배열의 맵을 만들고

  게임이 성공하거나 실패할 경우를 제외하고 계속하기 위한

  무한루프를 설정한다.

  또한 플레이어로부터 값을 입력 받기 위해 메시지 출력한다.

```
// 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} };
    // x가 세로를 y가 가로를 의미하므로 배열 순서에 유의


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

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

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

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

```cpp
//사용자에게 입력받은 값에 따른 행동 정리(if문 활용)
if (user_input == "상") {
    // 위로 한 칸 올라가기
    user_y -= 1;
    //2차원 배열의 시작이 좌측 상단에서 시작하므로 -1를 해야 한칸 위로 올라감.
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y += 1; //이미 위로 올라간 상태이므로 맵을 벗어났을 경우 다시 원상복구해야 함.
    }
    else {
        hp -= 1;
        cout << "위로 한 칸 올라갑니다." << endl;
        displayMap(map, user_x, user_y);
    }
}
```

```cpp
else if (user_input == "하") {
    // TODO: 아래로 한 칸 내려가기
    user_y += 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);
    if (inMap == false) {
        cout << "맵을 벗어났습니다. 다시 돌아갑니다." << endl;
        user_y -= 1;
    }
    else {
        hp -= 1;
        cout << "아래로 한 칸 내려갑니다." << endl;
        displayMap(map, user_x, user_y);
    }
}
```

```cpp
else if (user_input == "좌") {
    // TODO: 왼쪽으로 이동하기
    user_x -= 1;
    bool inMap = checkXY(user_x, mapX, user_y, mapY);

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

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

⇨ 상/하/좌/우를 입력 받을 경우 유저의 x, y 좌표 값을 +- 1 씩하여 위치를 조정한다. 또한 유저의 위치가 맵을 벗어날 경우 checkXY 함수를 활용하여 거짓일 경우 다시 입력을 받고 참일 경우 유저의 위치를 이동시킨다.

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

⇨ "지도"를 입력한 경우 displayMap 함수를 활용하여 지도와 현재 유저의 위치를 출력한다.

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

⇨ "종료"를 입력한 경우 break 를 사용하여 프로그램을 종료시킨다.

```
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

⇨ 설정한 명령어 이외의 명령을 받을 경우 에러 메시지 출력과 다시 명령을 받기 위해 continue 를 사용한다.

```
if (hp == 0) {
    cout << "HP가 0이 되었습니다. " << "실패하였습니다." << endl;
    cout << "게임을 종료합니다.";
    break;
}
```

⇨ 유저의 HP 가 0 이 되는 경우 게임 조건을 만족시키지 못했으므로 실패를 출력하고 프로그램을 종료시킨다.

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

⇨ 유저가 목적지에 도달했을 경우 유저의 좌표와 목적지의 좌표가 같은지를 확인하는 checkGoal 함수를 활용하여 진위 여부를 확인하고 축하 메시지와 함께 게임을 종료한다.

```
checkState(map, user_x, user_y);
```

checkState 함수를 불러옴으로써 아이템/적/포션 중 하나를 마주할 경우 게임에서 요구하는 기능과 메시지를 출력한다.

2. 함수별 구현 사항

- displayMap() 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;
    }
}
```

- 입력

int map[][] = 전체 지도

user_x = 유저의 x 값

user_y = 유저의 y 값

- 반환 : 없음.

- 결과 : 게임에서 설명한 전체 지도와 유저의 위치를 출력한다.

- 설명

2차원 배열의 맵을 출력하는 과정에서 배열에 있는 값에 따라 아이템/포션/적/목적지를 설정하여 전체 맵을 구성함. 이 기능을 구현하는데에 있어서 switch문을 활용하여 동일한 좌표 값에 따라 case를 구분 지어 코드를 작성함.

- checkXY() 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;
}
```

- 입력

user_x = 유저가 입력하는 x 좌표

mapX = 맵의 x 좌표 크기

user_y = 유저가 입력하는 y 좌표

mapY = 맵의 y 좌표 크기

- 반환 : checkFlag의 참/거짓 여부

- 결과 : 유저가 이동하려는 좌표가 유효한지를 확인하여 출력

- 설명

유저가 입력하려는 x, y 좌표 값을 최솟값인 0과 최댓값이 mapX, mapY와 비교하여 범위 내에 있을 경우 참을 반환하는 함수이다.

- checkGoal() 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;
}
```

- 입력

int map[][] = 전체 지도

user_x = 유저의 x 좌표 값

user_y = 유저의 y 좌표 값

- 반환 : 현재 유저의 좌표가 참인지 거짓인지 반환

- 결과 : 유저의 현재 위치가 목적지와 같은지를 확인

- 설명

bool를 사용하여 유저의 현재 위치가 목적지의 좌표인 4와 같은지를 확인하고 같다면 true를 같지 않다면 false를 반환하는 함수이다.

- checkState() 함수

```
// 아이템/포션/적을 만났을 때 해당 효과와 메세제 출력하는 함수
bool checkState(int map[][mapX], int user_x, int user_y) {
    bool isItem = (map[user_y][user_x] == 1);
    bool isPotion = (map[user_y][user_x] == 3);
    bool isEnemy = (map[user_y][user_x] == 2);

    // 아이템, 포션, 적 여부를 출력
    if (isItem == true) {
        cout << "아이템이 있습니다." << endl;
    }
    else if (isPotion == true) {
        hp += 2;
        cout << "포션이 있습니다. HP가 2 회복됩니다." << endl;
    }
    else if (isEnemy == true) {
        hp -= 2;
        cout << "적이 있습니다. HP가 2 줄어듭니다." << endl;
    }
    else {
        false;
    }

    // 아이템, 포션, 적 중 하나라도 있는지 여부를 반환
    return (isItem || isPotion || isEnemy);
}
```

- 입력

int map[][] = 전체 지도

user_x = 유저의 x 좌표 값

user_y = 유저의 y 좌표 값

- 반환 : isItem / isPotion / isEnemy 중 해당 되는 하나를 반환

- 결과 : 아이템/포션/적 중 하나를 마주할 경우 해당 조건에 맞는 메시지와 행동을 출력

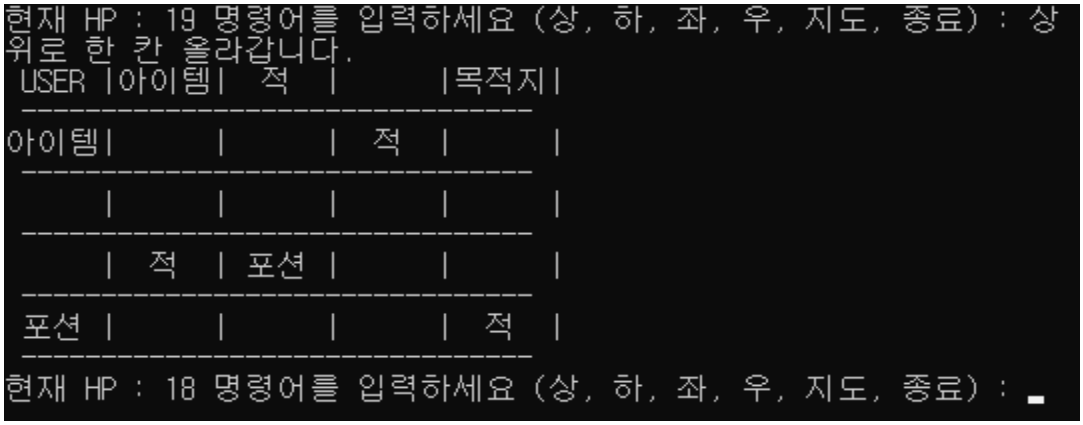
- 설명

isItem / isPotion / isEnemy를 bool를 사용하여 참/거짓으로 나타내고 이 값들 중 하나라도 참일 경우 그 값을 반환하는 함수이다.

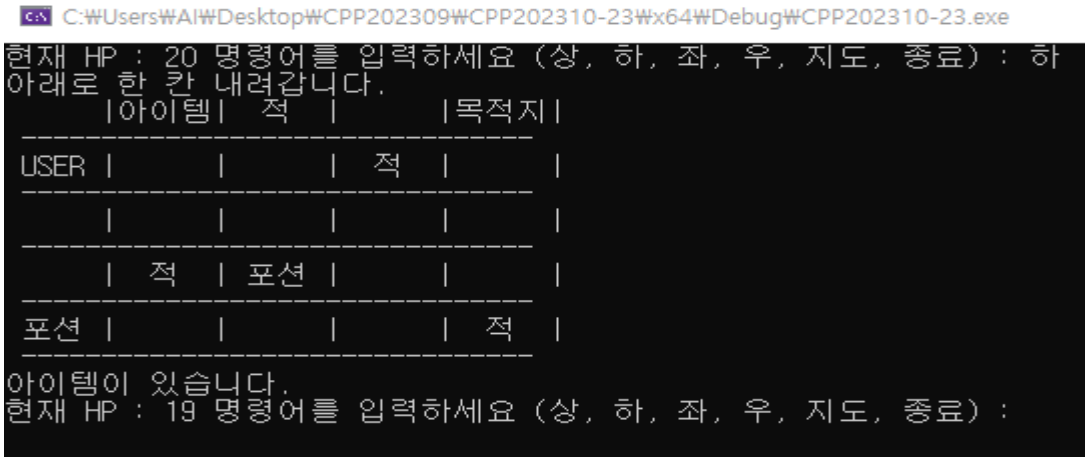
4. 테스트

1. 기능별 테스트

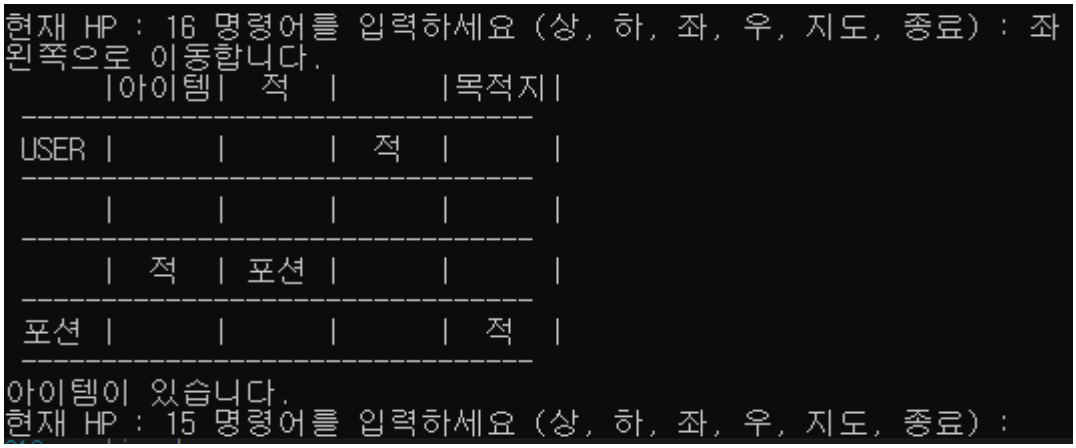
* "상"을 입력하며 유저가 한 칸 올라가며 HP가 1 줄었음을 출력



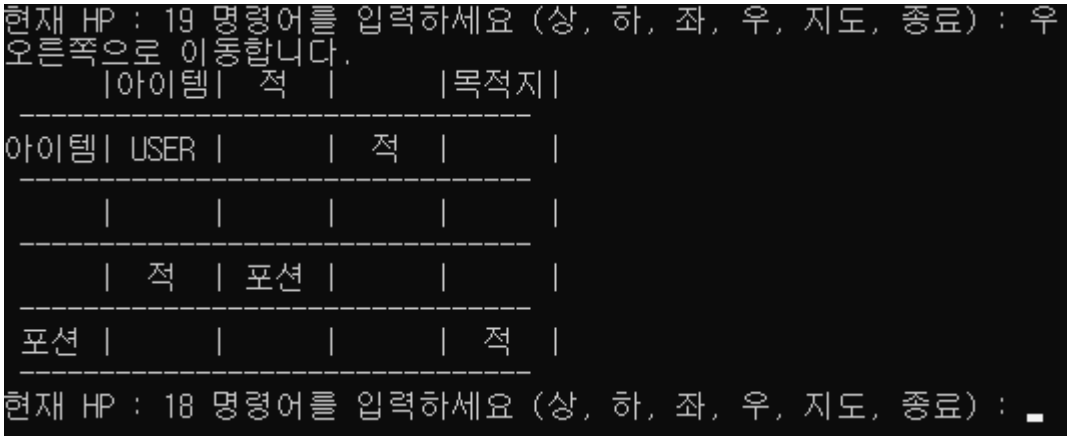
* "하"를 입력하면 유저가 한 칸 내려가며 HP가 1 줄었음을 출력



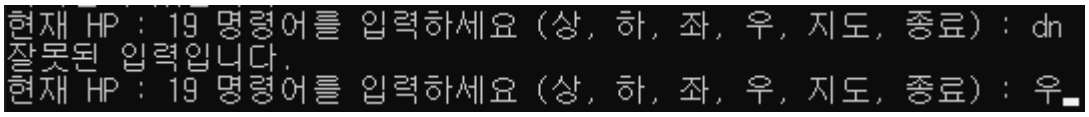
* "좌"를 입력하면 유저가 왼쪽으로 한 칸 이동하며 HP가 1 줄었음을 출력



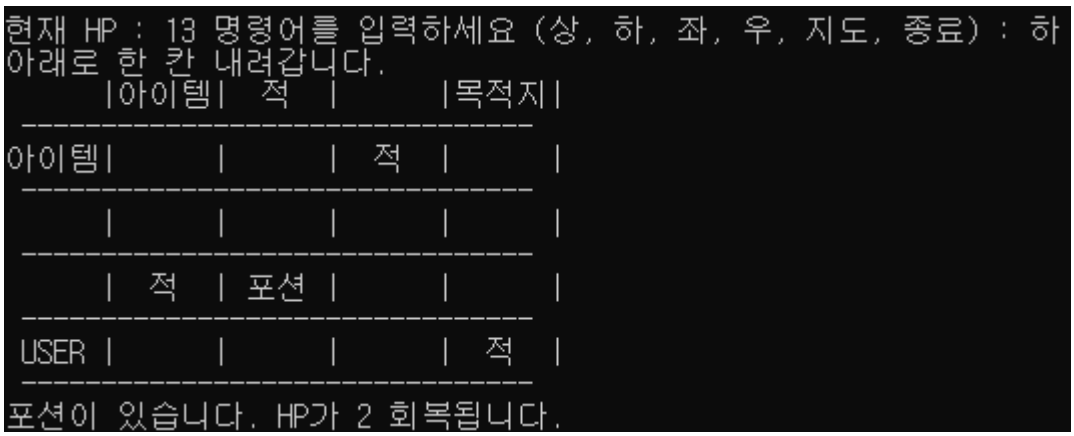
* "우"를 입력하면 유저가 오른쪽으로 한 칸 이동하며 HP가 1 줄었음을 출력



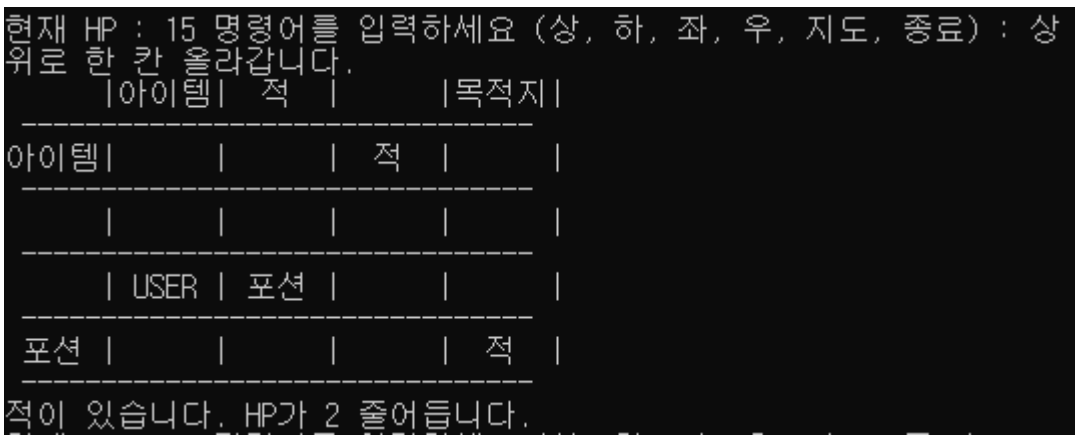
* 선택지 외의 명령을 입력했을 때는 에러가 발생했음을 출력



* "포션"을 마주하였을 경우 유저가 HP의 2 회복되었음을 출력



* "적"을 마주하였을 경우 유저의 HP가 2줄었음을 출력



* "지도"를 입력하면 전체 지도와 현재 유저의 위치를 함께 출력

```
현재 HP : 12 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료) : 지도
     |아이템|  적  |      |목적지|
 ------------------------------
아이템|      |      |  적  |      |
 ------------------------------
     |      |      |      |      |
 ------------------------------
     | USER | 포션 |      |      |
 ------------------------------
 포션 |      |      |      |  적  |
 ------------------------------
```

* "종료"를 입력하면 게임을 종료함을 출력

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

* 유저가 맵 밖으로 나갈 경우 에러 메시지와 함께 명령어를 다시 입력 받음

```
현재 HP : 18 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료) : 좌
맵을 벗어났습니다. 다시 돌아갑니다.
현재 HP : 18 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료) :
```

2. 최종 테스트

* 유저가 목적지에 도달하였을 경우 축하 메시지와 함께 게임을 종료함

```
현재 HP : 3 명령어를 입력하세요 (상, 하, 좌, 우, 지도, 종료) : 우
오른쪽으로 이동합니다.
     |아이템|  적  |      | USER |
 ------------------------------
아이템|      |      |  적  |      |
 ------------------------------
     |      |      |      |      |
 ------------------------------
     |  적  | 포션 |      |      |
 ------------------------------
 포션 |      |      |      |  적  |
 ------------------------------
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
```

5. 결과 및 결론

1) 프로젝트 결과

프로젝트를 통해 c++언어를 사용하여 MUD 게임을 만들어냈다.

게임은 유저가 상하좌우로 이동하며 주어진 HP 내에서 목적지에 도달하는 것을 목표로 한다. 세부적인 조건으로 적을 마주할 경우 HP가 2 감소하며 포션을 마주할 경우 HP가 2 회복하며 한 칸씩 이동할 때마다 HP가 1씩 감소하고 HP가 0이 되면 게임을 실패한다.

아쉽게도 이 게임에는 버그가 존재한다.

유저의 좌표가 맵 지도의 2차원 배열을 넘어갔을 때, "맵을 벗어났습니다."라는 문구와 함께 유저의 원래 위치로 돌아가게 된다. 이때, 해당 자리에 포션이나 적이 있을 경우 계속해서 HP를 회복하거나 감소시킬 수 있다.

이 버그를 해결하기 위해서는 맵에서 벗어났을 경우 아이템의 적용을 받지 않게 하거나 한 번 아이템을 사용한 경우 아이템이 사라지게끔 하여 버그를 고칠 수 있다.

2) 느낀 점

이 프로젝트보다 먼저 진행했던 tic tac toe 프로젝트를 경험해봄으로써 한층 더 간결하게 프로젝트를 진행할 수 있었던 것 같다. C++ 언어의 활용이나 함수 이용 측면에서도 훨씬 이해력이 높아졌으며 앞으로 진행할 개인 프로젝트보다 완성도 높게 만들고 싶다는 생각이 들었던 프로젝트이다.