

214958 김휘승

서론

1. 프로젝트 목적 및 배경: 7 주차까지 배운 내용에 대한 실습을 위해 진행
2. 목표: 간단한 Mud 게임 구현

요구사항

1. 사용자 요구사항

- ① 유저가 상하좌우로만 이동하며 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

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()

4. 추가기능 요구사항

- ① 유저는 체력 20 을 가지고 게임 시작
- ② 사용자가 이동할 때 마다 사용자 체력 1 씩 감소
- ③ 처음 명령문을 입력 받을 때 마다 HP 함께 출력
- ④ HP 가 0 이 되면 "실패"를 출력하고 종료
- ⑤ 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력

설계 및 구현

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

상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력

```

if (user_input == "up") {
    correctmove = nextpoint( 0, -1, map); //이동하는 함수, 유효한 이동이면 true를 correctmove에 반환
    if(correctmove)
        checkEvent(map); //유효한 이동이면 event가 있는지 확인&발생
}
else if (user_input == "down") {
    correctmove = nextpoint( 0, 1, map); // 위와 등
    if(correctmove)
        checkEvent(map); // 위와 등
}
else if (user_input == "left") {
    correctmove = nextpoint( -1, 0, map); // 위와 등
    if(correctmove)
        checkEvent(map); // 위와 등
}
else if (user_input == "right") {
    correctmove = nextpoint( 1, 0, map); // 위와 등
    if(correctmove)
        checkEvent(map); // 위와 등
}
}

```

입력 : user_input = 유저의 명령, nextpoint(dx, dy, map) = 이동함수(이동할 x 좌표, y 좌표, 지도)

결과 : 사용자의 위치를 이동

반환값 : nextpoint()가 유효한 명령이면 true를 아니면 false를 return

설명 : 유저의 명령에 따라 다른 이동명령이 실행, 이동명령이 유효한 위치로 이동하는 명령이면 추가로 이벤트를 확인&실행

```
명령어를 입력하세요 (up,down,left,right,map,exit): down
|아이템|적| |목적지|
-----
USER | | |적| |
-----
| | | | |
-----
|적|포션| | |
-----
포션| | |적| |
```

```
현재 HP: 19 명령어를 입력하세요 (up,down,left,right,map,exit): right
|아이템| 적 | |목적지|
-----
아이템| USER | | 적 | |
-----
| | | | |
-----
| 적 | 포션 | | |
-----
포션 | | | | 적 |
```

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

```
void displayMap(int map[][mapX]) {
    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[][] = 전체지도

mapX = 지도의 x 축

mapY = 지도의 y 축

user_x = 유저의 x 좌표

user_y = 유저의 y 좌표

popState = 지도 내 특징

결과 : 전체 지도를 출력

사용자 위치를 출력

설명 : 2 차원 배열에 있는 맵을 출력

출력하다가 사용자 위치와
동일한 좌표를 발견할 경우
사용자 정보를 출력

```
// 위의 입력외 다른 입력시 다시 입력받음
else {
    cout << "잘못된 입력입니다." << endl;
    continue;
}
```

결과 : 잘못된 입력이면 입력을 다시 받음

설명 : 사용자 입력이 잘못되면 잘못된
입력임을 알리고 사용자 입력을 받기위한
루프로 다시 돌아간다.

2. 지도 밖으로 나가게 되면 에러 메시지 출력

```
bool checkXY(int user_x, int user_y) {
    bool checkFlag = false;
    if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY) { //유효한 좌표이면 true를 반환
        checkFlag = true;
    }
    return checkFlag;
}
```

```
bool nextpoint(int dx, int dy, int map[][mapX]) {

    // 이동하려는 좌표가 유효한 좌표이면 실행
    if (checkXY((user_x + dx), (user_y + dy))) {
        user_x = user_x + dx; // 이동
        user_y = user_y + dy; // 이동
        displayMap(map); // 이동 후 지도 표시
        cout << "이동했습니다." << endl;
        health -= 1; // 유효한 이동 후 체력을 1 감소
        return true; // 유효한 이동이면 true를 반환
    }
    // 이동하려는 좌표가 유효하지 않은 좌표이면 실행
    else {
        cout << "맵을 벗어났습니다. 다시 입력해주세요." << endl;
        return false; // 유효하지 않은 이동이면 false를 반환
    }
}
```

```
USER |      |      | 적 |      |
-----
      |      |      |      |      |
-----
      |  적  | 포션 |      |      |
-----
포션 |      |      |      |  적  |
-----
현재 HP: 17 명령어를 입력하세요 (up,down,left,right,map,exit): left
맵을 벗어났습니다. 다시 입력해주세요.
현재 HP: 17 명령어를 입력하세요 (up,down,left,right,map,exit):
```

입력 : user_x, user_y = 유저의 좌표, map[], mapX, mapY = 전체지도, 지도의 x 좌표, 지도의 y 좌표

dx, dy = 이동할 유저의 좌표, health = 유저의 체력

반환값 : checkXY 는 유효한 좌표이면 true 를 아니면 false 를 반환

nextpoint 는 이동하려는 좌표가 유효한 좌표면 true 를 아니면 false 를 반환

결과 : 유저의 위치를 이동시키고 맵을 벗어나면 이를 알리고 다시 입력 받음

설명 : checkXY 는 좌표의 유효성을 검사하는 함수이며, 입력 받은 좌표가 양수이며, 지도 내의 좌표이면 true 를, 입력 받은 좌표가 음수이거나 지도 내의 좌표가 아니라면 false 를 반환한다.

반환된 false 값은 사용자의 입력을 받는 무한루프를 벗어나지 못하므로 다시 입력을 받는 루프의 처음으로 돌아간다. nextpoint의 반환값은 나중에 event 처리에 사용된다.

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

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

| | 아이템 | 적 | | USER |
|-----|-----|----|---|------|
| 아이템 | | | 적 | |
| | | | | |
| | 적 | 포션 | | |
| 포션 | | | | 적 |

이동했습니다.

목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
PS C:\CPP2409>

반환된 값은 finish 에 저장되고, finish 가 true 이면 도착함을 알리고 사용자 입력을 받는 무한 루프를 break; 그 후 return 0 를 통해 종료된다.

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

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

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

    // 중복 이벤트를 막기 위한 변수
    bool correctmove = false;
    cout << "명령어를 입력하세요 (up,down,left,right,map,exit): ";
    cin >> user_input;
```

```
명령어를 입력하세요 (up,down,left,right,map,exit):
```

입력 : user_input = 사용자의 입력, correctmove = event 처리를 위한 변수

결과 : 유저로부터 입력을 받음

설명 : while(1)은 무한루프로 사용자에게 계속 값을 요구하고 입력받은 값은 user_input 변수에 저장된다. 초기문제에서는 상,하,좌,우,지도,종료였지만, 컴파일의 문제상 한글의 '상'과 실제 입력받은 '상'이 동일하지 않아 아래의 if 문의 조건을 만족할 수 없어 영어로 교체하였다.

2. 지도와 현재 위치 출력 함수: displayMap()

사용자 위치 체크 함수: checkXY()

목적지에 도착 체크 함수: checkGoal()

위의 설명 참조

1. 추가 기능 요구사항

1. 유저는 체력 20 을 가지고 게임 시작
2. 사용자가 이동할 때 마다 사용자 체력 1 씩 감소
3. 처음 명령문을 입력 받을 때 마다 HP 함께 출력

```
const int mapX = 5; // 지도의 x크기
const int mapY = 5; // 지도의 y크기
int health = 20; // 플레이어의 체력 초기값 20
```

```
//남은 체력을 표시해줌
cout<<"현재 HP: "<<health<<" ";
```

```
// 이동할 때 마다 (dx,dy) 만큼 이동
bool nextpoint(int dx, int dy, int map[][mapX]) {
    // 이동하려는 좌표가 유효한 좌표이면 실행
    if (checkXY((user_x + dx),(user_y + dy))) {
        user_x = user_x + dx; // 이동
        user_y = user_y + dy; // 이동
        displayMap(map); // 이동 후 지도 표시
        cout << "이동했습니다." << endl;
        health -= 1; // 유효한 이동 후 체력을 1 감소
        return true; // 유효한 이동이면 true를 반환
    }
    // 이동하려는 좌표가 유효하지 않은 좌표이면 실행
    else {
        cout << "맵을 벗어났습니다. 다시 입력해주세요." << endl;
        return false; // 유효하지 않은 이동이면 false를 반환
    }
}
```

입력 : health = 유저의 체력

반환값 : nextpoint()에서 이동할 때 마다 health 가 1 씩 줄어든다.

결과 : 초기 유저의 체력은 20 으로 설정, 유저가 이동할 때 마다 1 씩 감소, 처음 명령문을 받을 때 마다 유저의 체력을 표시

설명 : int health = 20 은 메인 함수 밖에서 정의된 초기값이다.

Cout << health 는 사용자로부터 입력을 계속 입력 받는 무한루프의 마지막에 있으며 유저가 반복문을 벗어나지 못하면(종료하거나, 도착하거나, 체력이 0) 남은 체력을 알리고 반복문의 맨 처음으로 돌아간다.

nextpoint()는 사용자의 위치를 이동시키는 함수로, 이동하려는 좌표가 유효할 때만 이동하고 체력을 1 감소한다. 그러므로 유효하지 않는 좌표를 입력하면 이동하지 않고, 체력도 줄지 않는다.

4. HP 가 0 이 되면 "실패"를 출력하고 종료

```
//체력이 0이하면 게임을 종료
if(health<=0){
    cout << "HP가 0 이하가 되었습니다. 실패했습니다." << endl;
    break;
}
```

```
현재 HP: 1 명령어를 입력하세요 (up,down,left,right,map,exit): left
|아이템| 적 | |목적지|
-----
아이템| USER | | 적 | |
-----
| | | | |
-----
| 적 | 포션 | | |
-----
포션 | | | | 적 |
```

```
이동했습니다.
HP가 0 이하가 되었습니다. 실패했습니다.
PS C:\CPP2409>
```

입력 : health = 유저의 체력

결과 : 체력이 0 이되면 실패를 출력하고 종료된다.

설명 : 사용자가 이동 또는 이벤트를 통해 health 가 0 이하가 되면 이를 알리고 무한루프를 종료하는 break 가 발생, 그 후 메인 함수의 return 0;를 통해 종료된다.

무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력

- 예) {X}이/가 있습니다.
- 적을 만날 경우 HP 가 2 가 줄어들고 그에 대한 추가 메시지 출력
- 포션을 만날 경우 HP 가 2 가 늘어나고 그에 대한 추가 메시지 출력
- (적이나 포션 등은 사라지지 않음을 전제)

```

// 이벤트를 확인&처리하는 함수
void checkEvent(int map[][mapX]) {
    // 현재 자신의 위치에 이벤트가 있는지 확인
    int event = map[user_y][user_x];

    // 각 이벤트 처리 (1,2,3이 아니면 아무것도 처리하지 x)
    switch(event){
        case 1:
            cout<< "아이템이 있습니다."<<endl;
            break;
        case 2:
            cout<< "적이 있습니다. HP가 2 줄어듭니다."<<endl;
            health -=2; // 적을 만나면 체력 2 감소
            break;
        case 3:
            cout<< "포션이 있습니다. HP가 2 늘어납니다."<<endl;
            health +=2; // 포션을 만나면 체력 2 증가
        }
    }

    if (user_input == "up") {
        correctmove = nextpoint( 0, -1, map); //이동하는 함수, 유효한 이동이면 true를 correctmove에 반환
        if(correctmove)
            checkEvent(map); //유효한 이동이면 event가 있는지 확인&발생
    }
}

```

입력 : map[][] = 전체 지도, event = 맵의 특정 좌표에서의 이벤트, correctmove = checkEvent 함수를 발생시키는 조건 (bool)

반환값 : nextpoint()가 유효한 좌표로 이동하면 true 를 아니면 false 를 반환

결과 : 유효한 좌표로 이동한 후 각 좌표에서의 이벤트를 처리한다.

설명 : correctmove 는 nextpoint()의 반환 값이다. 즉 사용자의 입력을 통해 사용자의 좌표가 유효한 좌표로 이동했을 때만 correctmove 가 true 가 되고 checkEvent 함수가 실행된다.

checkEvent 는 유저가 있는 좌표의 특정 이벤트를 처리하며, switch 문으로 처리하였다.

correctmove 변수를 사용한 이유는 이벤트의 중복 발생을 방지하기 위해 *ex) 지도의 맨 왼쪽 포션 이벤트에서 왼쪽으로 이동(유효하지 않는 이동)을 할 때 체력이 다시 회복됨* nextpoint()가 유효할 때만 event 처리를 하도록 코드를 구성하였다.

이동했습니다.
아이템이 있습니다.
현재 HP: 19 명령어를 입력하세요 (up,down,left,right,map,exit):

명령어를 입력하세요 (up,down,left,right,map,exit): down

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

현재 HP: 17 명령어를 입력하세요 (up,down,left,right,map,exit): right
|아이템| 적 | |목적지|

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

이동했습니다.
적이 있습니다. HP가 2 줄어듭니다.
현재 HP: 14 명령어를 입력하세요 (up,down,left,right,map,exit): right

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

이동했습니다.
포션이 있습니다. HP가 2 늘어납니다.
현재 HP: 15 명령어를 입력하세요 (up,down,left,right,map,exit):

기능 별 테스트

상/하/좌/우 입력시 해당 방향으로 이동 후 지도 출력

```
명령어를 입력하세요 (up,down,left,right,map,exit): down
|아이템| 적 | |목적지|
-----
USER | | | 적 | |
-----
| | | | |
-----
| 적 | 포션 | | |
-----
포션 | | | | 적 |
-----
```

```
현재 HP: 19 명령어를 입력하세요 (up,down,left,right,map,exit): right
|아이템| 적 | |목적지|
-----
아이템| USER | | 적 | |
-----
| | | | |
-----
| 적 | 포션 | | |
-----
포션 | | | | 적 |
-----
```

“지도”를 입력하면 전체 지도와 함께 현재 위치를 출력

```
현재 HP: 18 명령어를 입력하세요 (up,down,left,right,map,exit): map
|아이템| 적 | |목적지|
-----
아이템| USER | | 적 | |
-----
| | | | |
-----
| 적 | 포션 | | |
-----
포션 | | | | 적 |
-----
```

이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청

```
현재 HP: 18 명령어를 입력하세요 (up,down,left,right,map,exit): side
잘못된 입력입니다.
명령어를 입력하세요 (up,down,left,right,map,exit):
```

지도 밖으로 나가게 되면 에러 메시지 출력

```
-----
USER | | | 적 | |
-----
| | | | |
-----
| 적 | 포션 | | |
-----
포션 | | | | 적 |
-----
현재 HP: 17 명령어를 입력하세요 (up,down,left,right,map,exit): left
맵을 벗어났습니다. 다시 입력해주세요.
현재 HP: 17 명령어를 입력하세요 (up,down,left,right,map,exit):
```

목적지에 도착하면 "성공"을 출력하고 종료

```
      |아이템| 적 |      | USER
-----
아이템|      |      | 적 |
-----
      |      |      |      |
-----
      |  적  | 포션 |      |
-----
포션 |      |      |      |  적
-----
이동했습니다.
```

```
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
PS C:\CPP2409>
```

메인 함수: 사용자에게 값을 계속 입력받고, 그에 대한 함수 호출

```
명령어를 입력하세요 (up,down,left,right,map,exit):
```

유저는 체력 20 을 가지고 게임 시작

사용자가 이동할 때 마다 사용자 체력 1 씩 감소

처음 명령문을 입력 받을 때 마다 HP 함께 출력

```
USER |      |      |  적 |      |
-----
      |      |      |      |      |
-----
      |  적  | 포션 |      |      |
-----
포션 |      |      |      |  적 |
-----
이동했습니다.
아이템이 있습니다.
현재 HP: 19 명령어를 입력하세요 (up,down,left,right,map,exit): down
```

HP 가 0 이 되면 "실패"를 출력하고 종료

```
현재 HP: 1 명령어를 입력하세요 (up,down,left,right,map,exit): left
|아이템|  적  |      |목적지|
-----
아이템| USER |      |  적  |      |
-----
      |      |      |      |      |
-----
      |  적  | 포션 |      |      |
-----
포션  |      |      |      |  적  |
-----

이동했습니다.
HP가 0 이하가 되었습니다. 실패했습니다.
PS C:\CPP2409>
```

무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력

- 예) {X}이/가 있습니다.
- 적을 만날 경우 HP 가 2 가 줄어든고 그에 대한 추가 메시지 출력
- 포션을 만날 경우 HP 가 2 가 늘어나고 그에 대한 추가 메시지 출력
- (적이나 포션 등은 사라지지 않음을 전제)

```
이동했습니다.
아이템이 있습니다.
현재 HP: 19 명령어를 입력하세요 (up,down,left,right,map,exit):

명령어를 입력하세요 (up,down,left,right,map,exit): down
|아이템|  적  |      |목적지|
-----
USER  |      |      |  적  |      |
-----
      |      |      |      |      |
-----
      |  적  | 포션 |      |      |
-----
포션  |      |      |      |  적  |
-----

현재 HP: 17 명령어를 입력하세요 (up,down,left,right,map,exit): right
|아이템|  적  |      |목적지|
-----
아이템|      |      |  적  |      |
-----
      |      |      |      |      |
-----
      | USER | 포션 |      |      |
-----
포션  |      |      |      |  적  |
-----

이동했습니다.
적이 있습니다. HP가 2 줄어듭니다.
현재 HP: 14 명령어를 입력하세요 (up,down,left,right,map,exit): right
```

```

-----
아이템 |      |      | 적 |      |
-----
      |      |      |      |      |
-----
      | 적 | USER |      |      |
-----
포션 |      |      |      | 적 |
-----
이동했습니다.
포션이 있습니다. HP가 2 늘어납니다.
현재 HP: 15 명령어를 입력하세요 (up,down,left,right,map,exit):

```

최종 테스트 스크린샷

```

-----
아이템 |      |      | 적 |      |
-----
      |      |      |      |      |
-----
      | 적 | 포션 |      |      |
-----
포션 |      |      |      | 적 |
-----
이동했습니다.
목적지에 도착했습니다! 축하합니다!
게임을 종료합니다.
PS C:\CPP2409>

```

결과 및 결론

프로젝트 결과: 주어진 mud game 코드를 좀 더 효율적으로 작성하였다.

느낀점 : 함수화를 통해 main 함수 내에서 코드를 깔끔하게 정리할 수 있었고 이는 가독성 측면 뿐 아니라 실제로 코딩하면서 오류를 잡는데 핵심적인 역할을 했습니다.

예를 들어 왼쪽 끝에 있는 포션 위치에 있을 때 왼쪽으로 이동 혹은 아래로 이동처럼 유효하지 않는 이동을 할 때 계속 피가 차는 게임 입장에서는 치명적인 버그가 있었는데 이를 해결할 때 기존의 방식보다 함수화를 거쳐 함수를 호출하는 조건만 수정하여 쉽게 고칠 수 있었습니다.

한줄평

프로그램을 짤 때 적절한 함수화가 왜 필요한지 느낄 수 있던 프로젝트였습니다.