Mud_gaem구현 보고서

소프트웨어공학과/213958/윤상권

목차

1.서론

1) 프로젝트 목적 및 배경

7주차까지 배운 내용을 토대로 간단한 mud게임 구현하는 실습을 위해 진행하였다.

2) 목표

상하좌우로 이동하며 무사히 목적지에 도착하는 간단한 mud게임 만들기이다.

2.요구 사항

1) 사용자 요구사항

상하좌우로 이동하며 무사히 목적지에 도착한다.

2) 기능 계획

(1). 사용자에게 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받는다.

- 상/하/좌/우 입력시 해당 방향으로 이동 후 지도를 출력한다.

- "지도"를 입력하면 전체 지도와 함께 현재 위치를 출력한다.

- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청을 한다.

(2). 지도 밖으로 나가게 되면 에러 메시지를 출력한다.

(3). 목적지에 도착하면 "성공"을 출력하고 종료한다.

추가 기능 요구사항

(4). 유저는 체력 20을 가지고 게임을 시작한다.

(5). 사용자가 이동할 때 마다 사용자 체력을 1씩 감소 시킨다.

(6). 처음 명령문을 입력 받을 때 마다 HP 함께 출력한다.

(7). HP가 0이 되면 "실패"를 출력하고 종료시킨다.

(8). 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력한다.

3) 함수 계획

(1). 메인 함수: 사용자에게 값을 계속 입력 받고, 그에 대한 함수 호출을 한다.

(2). displaMap() : 지도와 현재 위치를 출력하는 함수이다.

(3). checkXY() : 사용자 위치 유효 범위를 체크하는 함수이다.

(4). checkGoal() : 목적지에 도착했는지 체크하는 함수이다.

(5). moving() : 사용자의 입력에 따라 위치를 움직이는 함수이다.

(6). checkState() : 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력하고 효과를 적용시키는 함수이다.

3. 설계 및 구현

1) 기능 별 구현사항

(1) main() :

- 함수 스크린샷

```
13  // 메인 함수
14  int main()
15  {
16      // 0은 빈 공간, 1은 아이템, 2는 적, 3은 포션, 4는 목적지
17      int map[mapY][mapX] = {{0, 1, 2, 0, 4},
18                             {1, 0, 0, 2, 0},
19                             {0, 0, 0, 0, 0},
20                             {0, 2, 3, 0, 0},
21                             {3, 0, 0, 0, 2}};
22      // 유저의 위치를 저장할 변수
23      int user_x = 0; // 가로 번호
24      int user_y = 0; // 세로 번호
25      int user_hp = 20; // 사용자의 체력
26      // 게임 시작
27      while (1)
28      { // 사용자에게 계속 입력받기 위해 무한 루프
29
30          // 사용자의 입력을 저장할 변수
31          string user_input = "";
32          cout << "명령어를 입력하세요 (w(up),s(down),a(left),d(right),map,exit): ";
33          cin >> user_input;
34          // 종료 여부를 체크한다.
35          if (user_input == "exit" || user_hp <= 0) {
36              user_hp == 0 ? cout << "체력이 0이 되어 종료합니다. 실패했습니다." << endl : cout << "종료합니다." << endl;
37              break;
38          }
39          // moving함수를 호출하고 그 반환 값이 true이면 다음행동을 진행하고 false면 다시 입력을 받는다.
40          if (!moving(user_input, user_hp, map, user_x, user_y))
41          {
42              cout << "잘못된 입력입니다. 다시 돌아갑니다." << endl;
43              continue;
44          }
45          displayMap(map, user_x, user_y);
46          checkState(map, user_hp, user_x, user_y);
47          // 목적지에 도달했는지 체크
48          bool finish = checkGoal(map, user_x, user_y);
49          if (finish == true)
50          {
51              cout << "목적지에 도착했습니다! 축하합니다!" << endl;
52              cout << "게임을 종료합니다." << endl;
53              break;
54          }
55      }
56      return 0;
57  }
```

- 입력

map[][] : 전체 지도

mapY, mapX : 지도의 x, y 최대 범위 (integer 값을 가진다)

user_x, user_y : user의 현재 좌표 (integer 값을 가진다)

user_hp : user의 체력(HP) (integer 값을 가진다)

user_input : user가 입력한 동작할 행동 (string 값을 가진다)

finish : 목적지에 도착했는지 체크하는 변수 (bool 값을 가진다.)

- 반환 값

Integer로 함수 마지막에 정상 종료되면 0을 반환 값으로 시스템에 보낸다.

- 결과

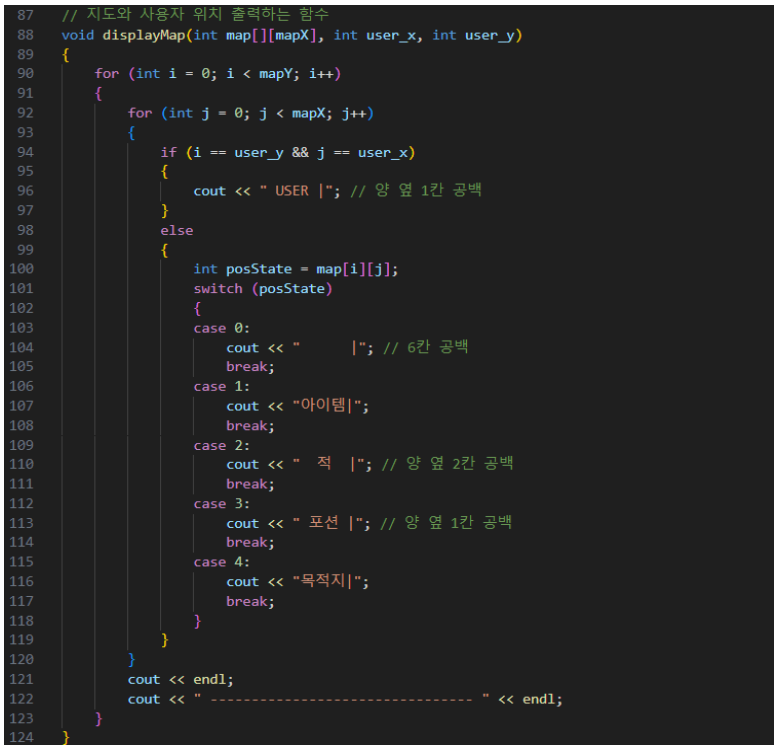
최초 사용자로부터 입력을 받아 동작을 수행한다. 이후에 목적지에 도착을 하였거나, exit를 입력 혹은 hp가 0이 되면 종료된다. 동작 기능에 따라 함수를 호출하고 실행하고 결과를 출력한다.

- 설명

처음 user의 위치는 (0,0)에서 시작한다. 기본 체력은 20으로 설정된다. 그 다음에 할 동작을 string으로 user_input 변수를 통해 입력 받는다. 만약 입력이 "exit"이거나 현재 체력이 0이라면 게임을 종료 시킨다. 그 다음으로 moving함수를 통해서 사용자가 입력한 값을 처리한다. 만약 사용자가 입력한 값이 올바르지 않다면 false의 반환 값을 받아 다시, 입력을 받고 정상적인 명령이면 true를 반환 값으로 받아 행동을 실행 시킨다. 변화된 맵을 출력 시키고, 현재 위치에 특별한 아이템이나 동작을 추가로 요구하는지 검사한다. 만약 목적지에 도착하였다면 축하한다는 메시지와 함께 게임을 종료시킨다. 아니라면 위 과정을 반복하며 게임이 진행되게 한다.

(2) displaMap():

- 함수 스크린샷



```
// 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

map[][] : 게임의 전체 지도

user_x, user_y : user의 현재 위치

posState : map의 길과 아이템, 적, 목적지를 출력한다.

- 반환값

없음

- 결과

게임의 전체 지도와 USER의 현제 위치가 출력된다.

- 설명

2중 반복문을 통해 맵을 출력한다. 2차원 배열로 저장된 map을 출력 할 때, 0은 길로써 공백으로 출력하고, 1은 "아이템"으로 2는 "적"으로 3은 "포션"으로 4는 "목적지"로 치환하여 출력한다.

(3) checkXY()

- 함수 스크린샷

```
125  // 이동하려는 곳이 유효한 좌표인지 체크하는 함수
126  bool checkXY(int user_x, int mapX, int user_y, int mapY)
127  {
128      bool checkFlag = false;
129      if (user_x >= 0 && user_x < mapX && user_y >= 0 && user_y < mapY)
130      {
131          checkFlag = true;
132      }
133      return checkFlag;
134  }
```

- 입력

user_x, user_y : 유저의 현재 위치

mapX, mapY : map의 크기

checkFlag : 유저가 이동하려는 곳이 유효한 곳인지 체크하는 변수

- 반환값

Bool 연산자 fasle, true가 반환된다.

- 결과

연산 결과에 따라 true or false가 반환된다.

- 설명

유저가 이동하려는 곳이 0부터 mapX, mapY까지의 배열의 유효 인덱스를 검사해서 인덱스 범위 안이라면 true를 반환하고 그 밖의 범위라면 false를 반환한다.

(4). checkGoal() :

- 함수 스크린샷

```
136  // 유저의 위치가 목적지인지 체크하는 함수
137  bool checkGoal(int map[][mapX], int user_x, int user_y)
138  {
139      // 목적지 도착하면
140      if (map[user_y][user_x] == 4)
141      {
142          return true;
143      }
144      return false;
145  }
146
```

- 입력

map[][] : 전체 지도

user_x, user_y: 유저의 현재 위치

- 반환값

bool연산자로 true or false가 반환된다.

- 결과

유저의 위치가 목적지라면 true를 반환하고 아니라면 false를 반환한다.

- 설명

map의 목적지는 4로 표현이 되어 저장되어있다. 만약 user의 위치가 map의 위치로 보았을 때 4가 저장되어 있는 위치라면 목적지에 도착한 것이므로 true를 반환시키고 아니라면 false를 반환 시킨다.

(5). moving() :

- 함수 스크린샷

```cpp
// 유저의 위치를 옮기는 함수
bool moving(string user_input, int &user_hp, int map[][mapX], int &user_x, int &user_y)
{
    int index = 0;
    int buf_x, buf_y; // 임시로 움질일 위치를 저장
    int directions[5][2] = {
        // wasd에 따른 상대적 위치 정보
        {0, 0},  // 움직이지 않음.
        {-1, 0}, // 상 (위로 이동)
        {1, 0},  // 하 (아래로 이동)
        {0, -1}, // 좌 (왼쪽으로 이동)
        {0, 1}   // 우 (오른쪽으로 이동)
    };
    if (user_input == "w")
        index = 1;
    else if (user_input == "s")
        index = 2;
    else if (user_input == "a")
        index = 3;
    else if (user_input == "d")
        index = 4;
    else if (user_input == "map")
        index = 0;
    else
    {
        return false;
    }

    buf_x = user_x + directions[index][1];
    buf_y = user_y + directions[index][0];
    if (!checkXY(buf_x, mapX, buf_y, mapY)){
        cout << "지도의 범위를 벗어났습니다" << endl;
        return false; }
        // buf_x, buf_y에 값이 true라면 hp를 깍고 실제 위치로 적용한다.
    else
    {
        user_hp--;
        user_x = buf_x;
        user_y = buf_y;
        if (index == 0)
            user_hp++; // map이나 exit과 같이 제자리 움직임의 경우 hp를 다시 회복시켜줌.
        return true;
    }
    cout << "debug 0" << endl; // 조건문의 예외가 발생함.
}
```

- 입력

user_input : 유저가 입력한 다음에 진행할 행동

user_hp : 현재 유저의 체력

map[][] : 전체 지도

user_x, user_y : 유저의 현재 위치

index : 유저의 이동에 관한 정보를 저장하는 변수

buf_x, buf_y : 움직일 위치가 올바른 값인지 확인을 위해 임시로 유저의 위치 정보를 저장하는 변수

directions[][] : 유저의 움직임에 대한 상대적 위치 정보를 저장한 배열

- 반환값

Bool 연산자로 true or false의 값을 가진다.

- 결과

user_input의 값이 올바른 값이면 유저의 위치를 이동 시키고, hp를 1만큼 줄이고 true를 반환 한다. user_input 값이 올바른 값이 아니라면 유저의 위치를 이동시키지 않고 false를 반환한다.

- 설명

유저의 이동(w(up), s(down), a(left), d(right))에 따라 상대적인 위치 주소를 저장해둔 배열 directions[][]의 값을 선택하여 기존 위치에 더하는 방식으로 유저의 위치를 변화 시킨다. 이때 buf_x, buf_y에 변화된 위치를 임시로 저장 해두고 checkXY() 함수를 통해 입력 값의 유효를 검사한다. 반환 값이 올바른 값인 true라면 user의 위치를 저장하고 hp를 1만큼 줄이고 true를 반환한다. 이에 반해 반환 값이 false라면 fasle를 반환한다. 만약 처음에 user_input이 "map"이라면 단순히 지도를 출력해주면 되기에 유저를 움직이지 않고 hp도 줄이지 않고 true를 리턴해준다.

(6). checkState() :

- 함수 스크린샷

```
59  void checkState(int map[][mapX], int &user_hp, int user_x, int user_y)
60  {
61      string state = "";
62      int posState = map[user_y][user_x];
63      switch (posState)
64      {
65      case 1:
66          state = "아이템";
67          break;
68      case 2:
69          state = "적";
70          cout << "적을 만나 HP가 2 감소합니다." << endl;
71          user_hp = user_hp-2;
72          break;
73      case 3:
74          state = "포션";
75          cout << "포션 효과로 HP가 2 치유됩니다." << endl;
76          user_hp = user_hp +2;
77          break;
78      default :
79          break;
80      }
81      cout << "HP : " << user_hp << "\t";
82      if (state != "") cout << state << " 이/가 있습니다." <<endl;
83      else cout << endl;
84  }
85
```

- 입력

map[][] : 게임의 전체 지도

user_hp : 유저의 현재 체력

user_x, user_y : 유저의 현재 위치

state : user의 위치가 길에 있는지, 적이나 아이템, 포션이 함께 있는지 저장하는 변수

posState : 유저의 위치에 있는 map배열의 정보를 저장하는 변수

- 반환값

없음

- 결과

유저 위치에 map상에 어떤 아이템인지에 따라 효과를 적용시키고, 무엇이 있는지 출력한다.

- 설명

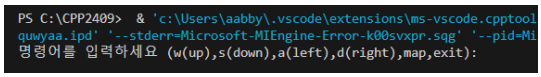
map상에 integer정보로 0은 길이고, 1은 아이템이며 2는 적이고 3은 포션이다. 현재 위

치에 무엇이 있는지에 따라 효과를 적용시킨다. 유저 위치에 map상에 적이 있다면 hp를 2만큼 추가로 줄이고, 포션이 있다면 hp를 2만큼 늘린다. 그리고 위치에 따라 아이템이나 적, 포션 등이 있음을 알리며 효과를 적용했다는 메시지를 출력 한다.

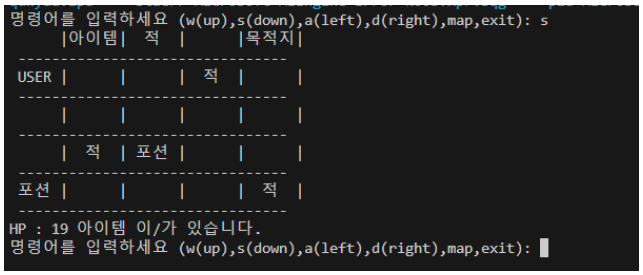
4. 테스트

1) 기능 별 테스트 결과:

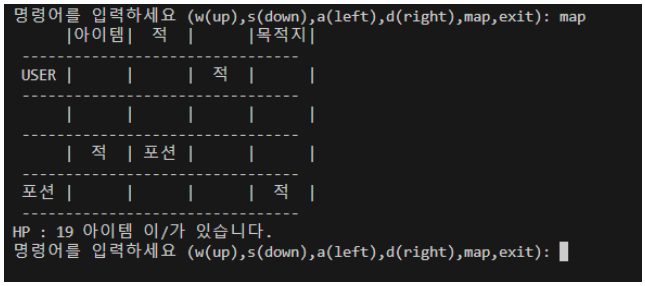
(1). 사용자에게 "상", "하", "좌", "우", "지도", "종료" 중 하나를 입력 받는다.



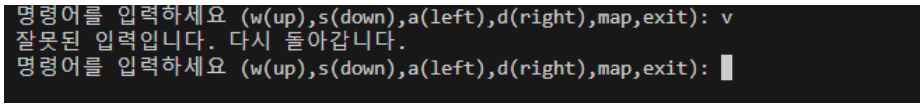
- 상/하/좌/우 입력시 해당 방향으로 이동 후 지도를 출력한다.



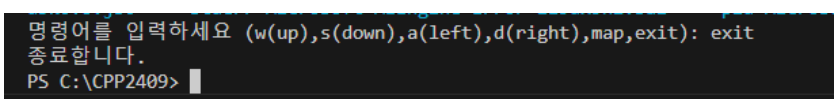
- "지도"를 입력하면 전체 지도와 함께 현재 위치를 출력한다.



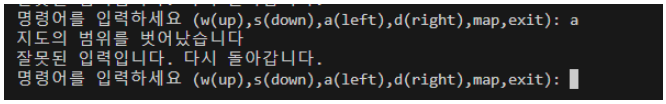
- 이 중 다른 것을 입력하면 에러 메시지 출력 후 재 입력 요청을 한다.



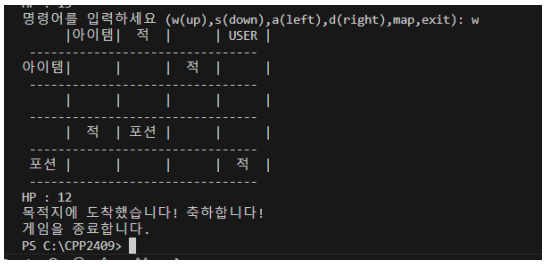
-exit 입력 시 게임이 종료된다.



(2). 지도 밖으로 나가게 되면 에러 메시지를 출력한다.

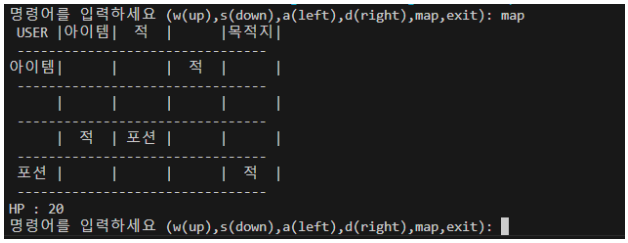


(3). 목적지에 도착하면 "성공"을 출력하고 종료한다.



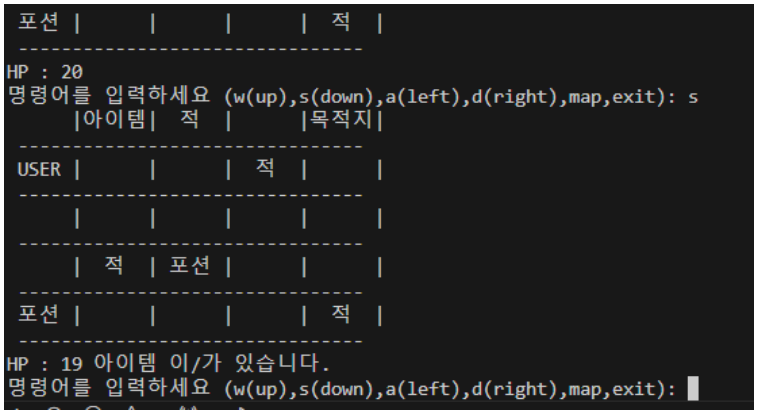
추가 기능 요구사항

(4). 유저는 체력 20을 가지고 게임을 시작한다.



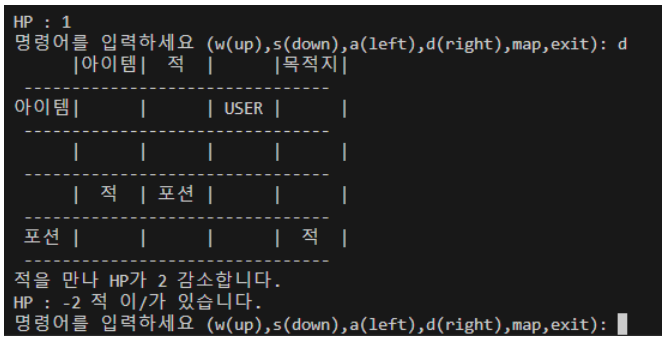
(5). 사용자가 이동할 때 마다 사용자 체력을 1씩 감소 시킨다.

(6). 처음 명령문을 입력 받을 때 마다 HP 함께 출력한다.



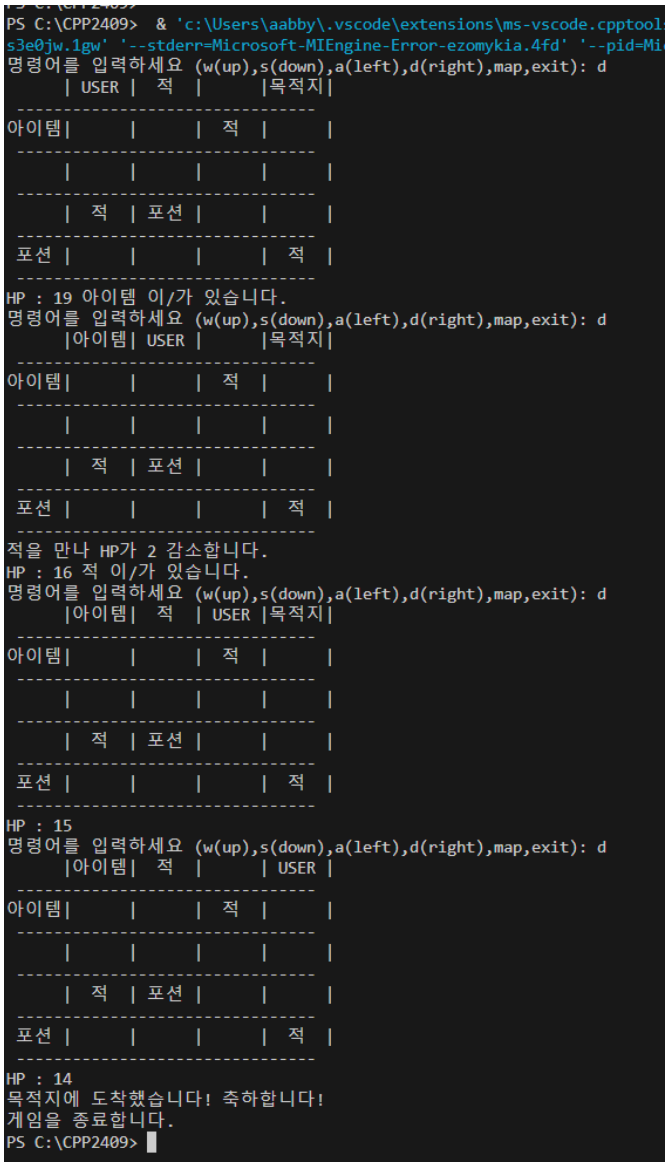
(7). HP가 0이 되면 "실패"를 출력하고 종료시킨다.

(8). 무기/갑옷, 포션, 적을 만났을 때 그에 대한 메시지를 출력한다.



체력 1에서 이동시 -1, 적을 만나 -2가 되어 종료된다.

2)최종 테스트 스크린샷



5. 결과 및 결론

1) 프로젝트 결과

요구사항에 맞추어 mud게임을 완성할 수 있었다.

2) 느낀점

함수를 만들 때, 기능 요구 중 어느 범위 까지를 함수로 만들지에 대해 고민을 많이 하였다. 기능과 UI를 분리해서 함수를 만드는게 좋다고 어디선가 들어본 적이 있었다. 코드의 가독성과 논리적인 오류가 생기지 않도록 하고, 유저의 입력값의 유효범위를 체크하면서 프로그램이 오류나지 않게 하는 것 등 많은 요소들을 고려하며 코드를 짜는 것은 매우 힘든 일임을 다시 한번 느낄 수 있었다.