

C++프로그래밍실습

보고서

제출자명: 강예준

제출자학번: 234091

프로그램 실행 방법

(main.cpp), (main.exe), (save_load.cpp), (user.cpp), (user.h) 파일 다운로드

1201 추가

(enemy.cpp), (enemy.h), (game_logic.cpp), (game_logic.h), (save_load.h)

한 폴더에 위 파일들 넣기 (save파일 생성)

첫 번째 방법: vsCode에서 main.cpp 실행

두 번째 방법:

1. cmd실행

2. main.exe파일 붙여넣기

3. 엔터키 누르고 실행

명령어 목록

up 플레이어를 위로 이동.

down 플레이어를 아래로 이동.

left 플레이어를 왼쪽으로 이동.

right 플레이어를 오른쪽으로 이동.

map 현재 맵 상태 출력.

save 게임 상태 저장.

exit 게임 종료 및 저장.

inventory 인벤토리 확인. // 오류 발생 미완성

~~use <아이템> 특정 아이템 사용 (예: use 포션).~~

1201 추가

아이템 사용 yes/no

사용 할 아이템 armor/potion

기능 별 구현 상황

```
void saveGame(const string& user_id, const vector<vector<int>>& map, int user_x, int user_y, const User& user, bool weapon, int armor) {
    ofstream save_file(user_id + "_game_save.txt");

    if (save_file.is_open()) {
        // 맵 저장
        for (const auto& row : map) {
            for (int cell : row) {
                save_file << cell << " ";
            }
            save_file << endl;
        }
        // 플레이어 상태 저장
        save_file << user_x << " " << user_y << endl;
        save_file << user.GetHP() << endl;
        save_file << weapon << " " << armor << endl;

        save_file.close();
    }
}
```

```
// 맵과 사용자 상태를 저장
void saveGame(const string& user_id, const vector<vector<int>>& map, int user_x, int user_y, int user_hp, bool weapon, int armor) {
    string filename = user_id + "_game_save.txt";

    ofstream save_file(filename);

    if (save_file.is_open()) {
        // 맵 데이터 저장
        for (const auto& row : map) {
            for (int cell : row) {
                save_file << cell << " ";
            }
            save_file << endl;
        }

        // 사용자 위치
        save_file << user_x << " " << user_y << endl;
        // 사용자 체력
        save_file << user_hp << endl;
        // 무기 갑옷 상태
        save_file << weapon << " " << armor << endl;

        save_file.close();
        cout << "게임 상태가 '" << filename << "' 파일에 저장되었습니다." << endl;
    }
    else {
        cout << "파일 저장에 실패했습니다!" << endl;
    }
}
```

● 게임 저장 (11/17)

입력:

- user_id (string): 사용자 ID, 저장 파일 이름에 사용
- user (User): 플레이어 객체, 체력 등 상태 정보
- weapon (bool): 플레이어의 무기 보유 여부
- armor (int): 플레이어가 보유한 갑옷의 수량

설명:

user_id를 기반으로 파일명을 생성하고, 현재 게임 상태를 해당 파일에 저장.

파일에 저장되는 정보:

1. 맵 상태(각 타일의 값).
2. 플레이어의 X, Y 좌표.
3. 플레이어의 체력(HP).

4. 무기 상태 및 갑옷 수량.

파일 저장이 완료되면 사용자에게 저장 완료 메시지 출력.

```
bool loadGame(const string& user_id, vector<vector<int>>& map, int& user_x, int& user_y, User& user, bool& weapon, int& armor) {
    ifstream load_file(user_id + "_game_save.txt");

    if (!load_file.is_open()) return false;

    // 맵 로드
    for (auto& row : map) {
        for (int& cell : row) {
            load_file >> cell;
        }
    }
    // 플레이어 상태 로드
    load_file >> user_x >> user_y;
    int hp;
    load_file >> hp;
    user.IncreaseHP(hp - user.GetHP()); // HP 초기화
    load_file >> weapon >> armor;

    load_file.close();
    return true;
}
```

● 게임 로드 (11/17)

입력:

user_id (string): 사용자 ID, 로드할 파일 이름에 사용됨.

user (User&): 플레이어 객체, 불러온 체력 상태를 저장.

weapon (bool&): 불러온 무기 상태를 저장할 참조 변수.

armor (int&): 불러온 갑옷 수량을 저장할 참조 변수.

반환값:

bool: 파일 로드 성공 여부.

true: 저장된 파일이 정상적으로 로드됨.

false: 파일을 찾을 수 없거나 읽기 실패.

설명:

user_id를 기반으로 파일명을 생성하고, 해당 파일이 존재하면 게임 상태를 읽어 변수에 저장.

로드되는 정보:

맵 상태.

플레이어의 X, Y 좌표.

플레이어의 체력(HP).

무기 상태 및 갑옷 수량.

파일이 없거나 읽기에 실패하면 false를 반환하며, 사용자에게 메시지 출력.

인벤토리 (11/17) 미완성

```
else if (user_input == "inventory") {
    user.DisplayInventory(); // 인벤토리 출력
} else if (user_input.rfind("use", 0) == 0) { // "use 포션"과 같은 명령
    if (user_input.length() > 4) {
        string item = user_input.substr(4); // 명령어 뒤의 아이템 이름 추출
        user.UseItem(item);
    } else {
        cout << "사용할 아이템의 이름을 입력해주세요! (예: use 포션)" << endl;
    }
}
```

```
//아이템 사용
void User::UseItem(const string& item) {
    if (inventory[item] > 0) {
        inventory[item]--;
        cout << item << "을(를) 사용했습니다. 남은 수량: " << inventory[item] << endl;

        if (item == "포션") {
            IncreaseHP(5);
            cout << "포션 사용으로 HP +5. 현재 HP: " << GetHP() << endl;
        }
    } else {
        cout << item << "이(가) 인벤토리에 없습니다!" << endl;
    }
}

// 인벤토리 표시
void User::DisplayInventory() const {
    cout << "=== 인벤토리 ===" << endl;
    for (const auto& pair : inventory) {
        cout << pair.first << ": " << pair.second << endl;
    }
    cout << "======" << endl;
}
```

플레이어가 획득한 아이템을 인벤토리에 저장하고, 필요할 때 확인할 수 있도록 하여 관리 요소를 추가

// 인벤토리 추가에는 성공했으나 포션이나 아이템 사용시 오류 발생

```
명령어를 입력하세요 (up,down,left,right,map,save,exit): use 포션
사용할 아이템의 이름을 입력해주세요! (예: use 포션)
명령어를 입력하세요 (up,down,left,right,map,save,exit): 잘못된 입력입니다.
```

인벤토리 (1201)

```
//아이템 추가
void User::AddItem(const string& item) {
    inventory[item]++;
    cout << item << "을(를) 획득했습니다! 현재 수량: " << inventory[item] << endl;
}

//아이템 사용
void User::UseItem(const string& item) {
    if (inventory[item] > 0) {
        inventory[item]--;
        if (item == "armor") {
            ActivateArmor(); // 갑옷 활성화
        } else if (item == "potion") {
            IncreaseHP(3); // 포션 사용 시 체력 회복
            cout << "포션 사용으로 HP +3. 현재 HP: " << GetHP() << endl;
        }
    } else {
        cout << item << "이(가) 인벤토리에 없습니다!" << endl;
    }
}

// 인벤토리 표시
void User::DisplayInventory() const {
    cout << "=== 인벤토리 ===" << endl;
    for (const auto& pair : inventory) {
        cout << pair.first << ": " << pair.second << endl;
    }
    cout << "===== " << endl;
}
```

입력

item (string): 인벤토리에 추가하거나 사용할 아이템 이름.

User::AddItem 함수:

- item: 추가할 아이템 이름 (예: "potion", "armor")

User::UseItem 함수:

- item: 사용할 아이템 이름 (예: "potion", "armor")

User::DisplayInventory 함수

반환값

AddItem: 반환값 없음

- item이 인벤토리에 추가되고 메시지가 출력됨

UseItem:

- 아이템 사용 성공 시 효과 적용(예: 체력 증가, 갑옷 착용 활성화)
- 아이템 수량이 0일 경우 메시지 출력

설명

1. 아이템 추가 (AddItem)

- 지정된 아이템이 인벤토리에 추가됩니다.
- 추가된 아이템의 현재 수량이 출력됩니다.

2. 아이템 사용 (UseItem)

- 사용 가능한 아이템이 인벤토리에서 제거되며, 아이템의 효과가 즉시 적용됩니다.

3. 인벤토리 확인 (DisplayInventory)

- 플레이어의 현재 아이템 목록과 수량을 출력합니다.

적 다양화 (12/01)

```
Enemy goblin("Goblin", 10, 5, 2);
Enemy orc("Orc", 20, 8, 5);
Enemy dragon("Dragon", 50, 15, 10);

Enemy::Enemy(const string& name, int hp, int attack, int defense)
    : name(name), hp(hp), attack(attack), defense(defense) {}
Enemy CreateRandomEnemy() {
    int random = rand() % 3; // 0, 1, 2 중 랜덤 선택
    if (random == 0) return goblin;
    else if (random == 1) return orc;
    else return dragon;
```

입력

CreateRandomEnemy 함수

- 호출 시 랜덤 적 객체 생성

Enemy 클래스

- name (string): 적 이름 (예: "Goblin", "Orc", "Dragon")
- hp (int): 적의 체력
- attack (int): 적의 공격력
- defense (int): 적의 방어력

반환값

CreateRandomEnemy 함수:

- 생성된 적 객체

설명

- CreateRandomEnemy 함수로 전투 시 랜덤한 적이 생성됩니다.
- 적의 종류와 능력치는 고유하며, 플레이어는 적의 특성을 고려해 전략을 선택해야 합니다.

사용자 스킬 (12/01)

```
// 스킬 사용
if (user.GetCharacterType() == "warrior") {
    cout << "전사의 스킬 '베기'를 사용했습니다!" << endl;
    currentEnemy.TakeDamage(15); // 전사 스킬 데미지
} else if (user.GetCharacterType() == "mage") {
    cout << "마법사의 스킬 '라이트닝 볼트'를 사용했습니다!" << endl;
    currentEnemy.TakeDamage(20); // 마법사 스킬 데미지
}
cout << currentEnemy.GetName() << "의 남은 HP: " << currentEnemy.GetHP() << endl;
```

입력

checkState 함수

- user (User&): 플레이어 객체
- action (string): 플레이어가 선택한 행동

User 클래스

- GetCharacterType(): 플레이어의 직업 반환 (예: "warrior", "mage")

반환값

플레이어의 스킬 효과가 적에게 적용됨

적의 체력이 감소하며, 남은 체력이 출력됨

설명

플레이어는 전투 중 스킬을 사용할 수 있습니다

스킬은 플레이어의 직업에 따라 다르게 동작합니다:

- 전사: "베기"로 적에게 15 피해
- 마법사: "라이트닝 볼트"로 적에게 20 피해

공격 및 도망 (12/01)

```
if (action == "attack") {
    // 기본 공격
    int playerAttack = 10; // 플레이어 기본 공격력
    currentEnemy.TakeDamage(playerAttack);
    cout << currentEnemy.GetName() << "에게 " << playerAttack << " 피해를 입혔습니다!" << endl;
    cout << currentEnemy.GetName() << "의 남은 HP: " << currentEnemy.GetHP() << endl;
} else if (action == "run") {
    // 도망
    cout << "전투에서 도망쳤습니다!" << endl;
    return; // 전투 종료
}
```

입력

checkState 함수

- action (string): 플레이어의 행동 선택 (예: attack, run)

Enemy 클래스

- TakeDamage(int): 적의 체력을 감소

반환값

attack:

- 적의 체력이 감소하며 남은 체력이 출력됨

run:

- 전투 종료 메시지 출력

설명

공격 :

- 플레이어의 기본 공격으로 적에게 피해를 입힙니다

도망 :

- 전투를 종료하고 전투 루프에서 탈출합니다

스킬 데미지 수정 및 방어력 무시 (12/15)

```
if (action == "attack") {
    // 기본 공격
    int playerAttack = 20; // 플레이어 기본 공격력
    currentEnemy.TakeDamage(playerAttack);
    cout << currentEnemy.GetName() << "에게 " << playerAttack << " 피해를 입혔습니다!" << endl;
    cout << currentEnemy.GetName() << "의 남은 HP: " << currentEnemy.GetHP() << endl;
} else if (action == "skill") {
    // 스킬 사용
    if (user.GetCharacterType() == "warrior") {
        cout << "전사의 스킬 '베기'를 사용했습니다!" << endl;
        currentEnemy.TakeDamage(30, true); // 전사 스킬 데미지
    } else if (user.GetCharacterType() == "mage") {
        cout << "마법사의 스킬 '라이트닝 볼트'를 사용했습니다!" << endl;
        currentEnemy.TakeDamage(20, true); // 마법사 스킬 데미지
    }
    cout << currentEnemy.GetName() << "의 남은 HP: " << currentEnemy.GetHP() << endl;
}
```

입력:

- damage (int): 스킬에 의해 입혀질 고정 데미지 값
- ignoreDefense (bool): 방어력을 무시할지 여부 (기본값: false)

반환값:

- void: 반환값 없음. 적의 체력(HP) 상태를 갱신함

설명:

- 스킬 사용 시 적의 방어력을 무시하고 고정적인 데미지를 입히도록 수정함
- 방어력 무시 로직 추가:
- ignoreDefense가 true인 경우 방어력을 고려하지 않고 데미지를 그대로 적용
- 일반 공격과 스킬 구분:

- 일반 공격은 방어력을 적용하며, 스킬은 고정 데미지를 입힌다

갑옷 기능 오류 수정 (12/15)

```
if (currentEnemy.IsAlive()) {
    if (user.IsArmorActive()) {
        cout << currentEnemy.GetName() << "의 공격이 갑옷에 막혔습니다!" << endl;
        user.DeactivateArmor(); // 아머 비활성화
    } else {
        int enemyDamage = currentEnemy.GetAttack();
        user.DecreaseHP(enemyDamage);
        cout << currentEnemy.GetName() << "이(가) 공격합니다! 플레이어가 " << enemyDamage << " 피해를 입었습니다." << endl;
        cout << "현재 HP: " << user.GetHP() << endl;
    }
}
```

입력:

- map (vector<vector<int>>&): 게임 맵 상태.
- user_x, user_y (int&): 플레이어의 현재 좌표.
- user (User&): 플레이어 객체, 체력(HP) 및 갑옷 상태를 관리.
- armor_active (bool&): 갑옷이 활성화되어 있는지 확인하는 상태 변수.
- currentEnemy (Enemy&): 전투 중인 적 객체.

출력:

- 갑옷을 사용했을 때 적의 공격을 방어하고, 갑옷이 해제되었음을 출력.
- 갑옷이 없는 상태에서 적의 공격을 받으면 HP 감소 메시지 출력.

실행 결과


```
사용자 ID를 입력하세요 : 1111
새로운 게임을 시작합니다.
명령어를 입력하세요 (up,down,left,right,map,save,exit):
```

새로운 유저 접속

```
명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit): inventory
=== 인벤토리 ===
갑옷 : 1
=====
```

인벤토리 시스템

```
명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit): save
게임 상태가 저장되었습니다.
```

 1111_game_save.txt

SAVE 시스템

```

사용자 ID를 입력하세요 : 1111
저장된 게임을 불러옵니다.
명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit): map
|아이템|적|목적지|
-----
USER |   |   |   |   |
-----
|   |   |   |   |
-----
|적|포션|   |   |
-----
포션|   |   |   |적|
-----

```

LOAD 시스템

```

명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit): use 포션
사용할 아이템의 이름을 입력해주세요! (예: use 포션)
명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit): 잘못된 입력입니다.
명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit): use포션
>이(가) 인벤토리에 없습니다!
명령어를 입력하세요 (up,down,left,right,map,inventory,save,exit):

```

USE 시스템 오류

++SAVE시 아이템 사라짐 이슈

실행 결과(1201)

```

=== 인벤토리 ===
armor: 1
potion: 1
=====
아이템을 사용하시겠습니까? (yes/no): yes
사용할 아이템 이름을 입력하세요 (armor/potion): armor
갑옷을 착용했습니다! 적의 공격을 방어할 수 있습니다.

```

인벤토리 아이템 사용 여부, 사용할 아이템

```

Dragon과 (와) 전투가 시작되었습니다!
적 정보 - HP: 50, 공격력: 15, 방어력: 10

```

몬스터 다양화

```

행동을 선택하세요 (attack, skill, run):

```

적과 만났을 때 선택

```

행동을 선택하세요 (attack, skill, run): skill
Dragon의 남은 HP: 50
Dragon이 (가) 공격합니다! 플레이어가 15 피해를 입었습니다.
현재 HP: 13

```

스킬 오류

```

행동을 선택하세요 (attack, skill, run): run
전투에서 도망쳤습니다!

```

도망가기

```
행동을 선택하세요 (attack, skill, run): attack
Dragon이(가) 방어했습니다!
Dragon에게 10 피해를 입혔습니다!
Dragon의 남은 HP: 50
Dragon이(가) 공격합니다! 플레이어가 15 피해를 입었습니다.
현재 HP: 13
```