**2024 Spring OOP Assignment Report**

과제 번호 : 4

학번 : 20230024

이름 : 문요준

Povis ID : yojun313

**명예서약 (Honor Code)**

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

**1. 프로그램 개요**

**1-1. 개요**

이 프로그램은 몬스터와 싸우며 캐릭터를 강화하는 RPG(Role-Playing Game) 스타일의 게임이다. 사용자는 다양한 맵을 탐험하면서 몬스터를 처치하고, 경험치를 획득하여 캐릭터의 능력을 향상시킬 수 있다. 전투, 맵, 캐릭터, 상점 및 몬스터 등 여러 기능이 구현되어 있어 플레이어가 게임 세계에 몰입할 수 있도록 설계되었다. 전투 시스템은 플레이어와 몬스터 간의 상호 작용을 중심으로 구성되어 있으며, 플레이어는 전략적인 선택을 통해 전투를 진행해야 한다. 맵은 게임의 세계를 구성하는 핵심 요소로, 플레이어는 이동하면서 새로운 지역(칸)을 발견하고 다양한 타입을 지닌 몬스터와 조우하게 된다. 상점에서는 공격력이나 방어력, HP, MP, EXP 등을 구매해서 캐릭터의 전투 능력을 강화할 수 있다.

**1-2. 프로그램 실행 방법 및 사용 방법**

**[ 실행 방법 ]**

이 프로그램은 C++로 작성된 콘솔 기반 RPG 게임으로, 사용자는 텍스트 명령을 통해 게임을 조작한다. 프로그램 실행은 컴파일된 실행 파일을 통해 이루어지며, 컴파일 후 생성된 **.exe** 파일을 실행하면 된다. 게임이 시작되면 기본적으로 메인 메뉴가 표시된다. 여기에서 사용자는 게임을 시작하거나 게임을 종료할 수 있는 옵션을 선택할 수 있다.

**[ 사용 방법 ]**

**< 메인 메뉴 >**

- 메인 메뉴는 '게임 시작'과 '종료' 두 가지가 구현되어 있으며 사용하는 방법은 다음과 같다:

* 프로그램이 실행되면 사용자는 화면에 표시된 메뉴에서 '1'을 입력해 게임을 시작하거나 '2'를 입력해 게임을 종료할 수 있다.
* 사용자 입력은 표준 입력을 통해 받으며, 유효한 입력(1 또는 2)이 들어올 때까지 메뉴 선택을 반복한다.

- 맵에서의 동작은 다음과 같이 구성된다:

* **맵 이동**: 사용자는 맵에서 캐릭터를 움직일 수 있으며, 새로운 위치에서 무슨 일이 발생하는지에 따라 추가 선택을 할 수 있다.
* **현재 상태 보기**: 사용자가 현재 캐릭터의 상태를 확인할 수 있도록 상태 보기 옵션을 선택할 수 있다.
* **게임 종료**: 언제든지 사용자는 게임을 종료할 수 있는 옵션을 선택할 수 있다.

**< 전투 메뉴 >**

- 게임에서 몬스터와 전투를 할 때는 다음과 같은 옵션이 제공된다:

* **1. Melee attack**: 기본 공격을 실행한다.
* **2. Magic attack**: 단일 공격 스킬로 공격을 가한다.
* **3. Fire magic attack**: 불 속성의 마법 공격을 실행한다.
* **4. Grass magic attack**: 풀 속성의 마법 공격을 실행한다.
* **5. Water magic attack**: 물 속성의 마법 공격을 실행한다.

- 전투 중 사용자는 위 옵션 중 번호를 입력해 하나를 선택하여 몬스터에게 공격을 가할 수 있다. 각 공격 옵션 선택 시, 캐릭터의 상태(예: MP 부족)에 따라 실행 가능 여부가 결정된다.

**< 상점 메뉴 >**

- 상점을 방문하면 다음과 같은 메뉴 옵션이 제공된다:

* **1. Purchase attack**: 공격력을 구매한다.
* **2. Purchase defense**: 방어력을 구매한다.
* **3. Purchase EXP**: EXP를 구매한다.
* **4. Restore HP**: HP를 회복한다.
* **5. Restore MP**: MP를 회복한다.
* **6. Exit shop**: 상점에서 나간다.

각 항목을 번호를 입력하여 선택할 때는 사용자의 현재 골드와 아이템의 가격, 사용자의 상태 등을 고려하여 구매나 판매가 진행된다. 예를 들어, 골드가 충분하지 않다면 구매할 수 없으며, 판매 시에는 보유한 아이템의 유무가 확인된다.

**1-3. 프로그램 파일 설명**

**[ 소스 파일(cpp) ]**

1. **Battle.cpp**
   * 전투 시스템을 처리하는 클래스의 구현 파일이다. 몬스터와의 전투에서 사용자의 입력에 따라 캐릭터의 공격 유형을 결정하고, 전투 결과를 계산한다.
2. **Character.cpp**
   * 캐릭터의 속성 및 행동을 관리하는 클래스의 구현 파일이다. 캐릭터의 상태 변경, 레벨 업, 스킬 사용 등의 기능을 처리한다.
3. **main.cpp**
   * 게임의 주 실행 파일로, 메인 메뉴와 게임의 주요 흐름을 제어한다. 사용자의 입력을 받아 게임을 시작하거나 종료하는 등의 기본적인 흐름을 관리한다.
4. **Map.cpp**
   * 게임 맵을 관리하는 클래스의 구현 파일이다. 맵의 구조를 그리고, 캐릭터의 이동이나 위치에 따른 이벤트 처리 등을 담당한다.
5. **Monster.cpp**
   * 게임 내의 몬스터를 관리하는 클래스의 구현 파일이다. 몬스터의 생성, 공격 패턴, 상태 관리 등을 처리한다.
6. **Shop.cpp**
   * 상점 시스템을 관리하는 클래스의 구현 파일이다. 아이템의 구매, 판매, 상점에서의 행동 등을 처리한다.

**[ 헤더 파일(H) ]**

1. **Battle.h**
   * Battle 클래스를 정의하는 파일로, 전투 관련 메소드와 속성이 선언되어 있다.
2. **Character.h**
   * Character 클래스를 정의하는 파일로, 캐릭터의 속성과 캐릭터 관리를 위한 메소드들이 선언되어 있다.
3. **Map.h**
   * Map 클래스를 정의하는 파일로, 맵과 관련된 메소드 및 속성이 선언되어 있다. 맵의 구조 및 맵과 관련된 행동의 로직을 포함한다.
4. **Monster.h**
   * Monster 클래스를 정의하는 파일로, 게임 내의 몬스터들을 관리하기 위한 메소드와 속성이 선언되어 있다.
5. **Shop.h**
   * Shop 클래스를 정의하는 파일로, 상점 관련 메소드와 속성이 선언되어 있다.

**2. 프로그램의 구조 및 알고리즘**

**2-1. Class 설명**

**1. Class: Character**

개요: 외부에서 변수들에 쉽게 접근하지 못하게 대부분의 변수들을 Private안에서 정의했고 변수들이 필요한 상황에서 해당 변수를 반환하는 메서드를 따로 구현해 Public과 Private의 경계를 명확하게 했다.

**< 변수 >**

Private 멤버 변수

* int attack; - 캐릭터의 공격력.
* int defense; - 캐릭터의 방어력.
* int gold; - 캐릭터가 소유한 골드.
* int level; - 캐릭터의 레벨.
* int max\_hp; - 캐릭터의 최대 체력.
* int curr\_hp; - 캐릭터의 현재 체력.
* int max\_mp; - 캐릭터의 최대 마나.
* int curr\_mp; - 캐릭터의 현재 마나.
* int max\_exp; - 다음 레벨까지 필요한 경험치.
* int curr\_exp; - 캐릭터의 현재 경험치.

Public 멤버 변수

* vector<int> curr\_position; - 캐릭터의 현재 위치.
* vector<vector<int>> move\_history; - 캐릭터의 이동 경로 기록.

**< 메서드 >**

**1. 생성자 Character()**

* 기능: 캐릭터의 기본 속성을 초기화한다.
* 구현:
  + 공격력(attack)을 10으로 설정.
  + 방어력(defense)을 2로 설정.
  + 골드(gold)를 0으로 설정.
  + 레벨(level)을 1로 설정.
  + 최대 체력(max\_hp)과 현재 체력(curr\_hp)을 100으로 설정.
  + 최대 마나(max\_mp)와 현재 마나(curr\_mp)를 20으로 설정.
  + 다음 레벨까지 필요한 최대 경험치(max\_exp)를 100으로, 현재 경험치(curr\_exp)를 0으로 설정.
  + 현재 위치(curr\_position)와 이동 기록(move\_history)를 초기화한다.

**2. void show\_status() / void battle\_status() / void shop\_status()**

* 기능: 각 상황에서 캐릭터의 현재 상태를 출력한다.
* 구현:
  + 캐릭터의 레벨, 공격력, 방어력, 골드, 현재/최대 체력, 현재/최대 마나, 현재/최대 경험치를 출력한다.
  + shop\_status()는 매개변수 choiceNum에 따라 Shop에서 상품을 구매하여 값이 변화되었을 시 해당 번호의 출력을 '=>'을 표시해 다르게 하여 변화를 나타낸다.

**3. int change\_status()**

* 기능: 캐릭터의 상태(HP, MP, 골드)를 변경한다. shop 매개변수에 따라 상점에서의 변화를 처리하며, 상태 변경 후 변화된 값을 반환한다.
* 매개변수:
  + value: 변경할 값의 크기.
  + option: 변경할 상태를 지정 ('h'는 체력, 'm'은 마나, 'g'는 골드).
  + shop: 상점에서의 변경 여부를 나타내는 불리언 값. 상점에서 호출되었을 때를 제외하고는 change\_status()는 HP를 변화시키는 데에만 사용된다.
* 구현:
  + option 매개변수에 따라 다음과 같이 처리:
    - 'h' (체력):
      * curr\_hp에 value를 더하고, 결과가 max\_hp를 초과하지 않도록 조정한다.
      * 결과가 0 이하가 되면 curr\_hp를 0으로 설정한다.
    - 'm' (마나):
      * curr\_mp에 value를 더하고, 결과가 max\_mp를 초과하지 않도록 조정한다.
      * 결과가 0 이하가 되면 curr\_mp를 0으로 설정한다.
    - 'g' (골드):
      * gold에 value를 뺀다.
      * 결과가 0 이하가 되면 gold를 0으로 설정한다.
    - 'a' / 'd' / 'e' (공격력, 방어력, EXP):
      * 상점에서 구매했을 시 value값에 따라 추가한다.
  + shop이 true인 경우, 상점에서의 상품 구매를 수행할 수 있다.
  + 변경되기 전 상태 (체력, 마나 중 하나) 값을 반환한다. 이 값은 shop\_status()에서 변화를 나타내기 위해 쓰인다.

**4. void random\_act()**

* 기능: 게임 내에서 발생하는 랜덤 이벤트를 처리하며, 이벤트의 결과로 캐릭터의 체력이 손실되거나 회복되고, 골드가 증가하거나 아무 일도 일어나지 않을 수 있다.
* 구현:
  + 랜덤 이벤트를 결정하기 위한 옵션 배열을 생성: option = {"lost", "restore", "gold", "nothing"}.
  + rand() % 4를 사용하여 0에서 3 사이의 랜덤 인덱스를 생성하고, 이를 통해 이벤트를 선택한다.
  + 선택된 이벤트에 따라 다음과 같이 캐릭터의 상태에 변화를 준다:
    - "lost":
      * 캐릭터의 현재 체력(curr\_hp)에서 30%를 감소시킨다.
      * 새로운 체력 값이 계산되며, round() 함수를 사용하여 정수로 반올림된다.
      * 체력이 감소했다는 메시지 출력: "You lost HP..."
    - "restore":
      * 캐릭터의 현재 체력을 30% 회복시킨다.
      * 회복된 체력 값이 max\_hp를 초과하지 않도록 처리한다. 초과할 경우 curr\_hp는 max\_hp로 설정된다.
      * 체력이 회복되었다는 메시지 출력: "You restored HP"
    - "gold":
      * 캐릭터의 골드(gold)에 100을 추가한다.
      * 골드를 획득했다는 메시지 출력: "You found 100 gold!"
    - "nothing":
      * 아무런 변화가 없다.
      * 아무 일도 발생하지 않았다는 메시지 출력: "Nothing happened..."
  + 메서드 종료 후, 변경된 상태는 다른 메서드 또는 게임 로직에 의해 반영된다.

**5. bool level\_up(bool print\_option)**

* 기능: 캐릭터의 레벨을 증가시키고, 캐릭터의 능력치(공격력, 방어력, 최대 HP, 최대 MP)를 증가시킨다. 선택적으로 레벨 업 후의 캐릭터 상태를 출력할 수 있다.
* 매개변수:
  + print\_option: 레벨 업 정보를 출력할지 여부를 결정하는 불리언 값.
* 구현:
  + 캐릭터의 레벨(level)을 1 증가시킨다.
  + 공격력(attack)을 증가시킨다 (레벨마다 3씩 증가).
  + 방어력(defense)을 증가시킨다 (레벨마다 1씩 증가).
  + 최대 체력(max\_hp)을 증가시킨다 (레벨마다 50씩 증가).
  + 최대 마나(max\_mp)를 증가시킨다 (레벨 \* 100)
  + 캐릭터의 현재 체력과 마나를 각각 최대 체력과 최대 마나로 리셋한다.
  + 메서드가 실행을 완료하고 레벨 업이 성공적으로 이루어졌다면 true를 반환한다.

**6. bool judge\_level\_up()**

* 기능: 캐릭터의 경험치가 레벨업에 필요한 경험치를 초과했는지 판단한다.
* 구현:
  + 현재 경험치가 최대 경험치 이상인지 확인하고 결과를 반환한다.

**7. 공격 관련 메서드**

**-** Character 클래스에는 캐릭터가 몬스터와의 전투에서 사용할 수 있는 다양한 공격 옵션이 구현되어 있다.

1. **int attack\_basic(char type, int defense)**
   * 기능: 기본 공격을 수행하고, 그 결과로 발생하는 데미지를 계산하여 반환한다.
   * 구현: 기본 공격력 attack과 상대 방어력 defense를 고려하여 실제 데미지를 계산하고 반환한다. 결과값이 음수가 되지 않도록 최소값을 0으로 설정한다.
2. **int attack\_skill(char type, int defense)**
   * 기능: 스킬 공격을 수행한다.
   * 구현: attack\_basic과 유사하나, 위력이 더 강력하고 MP가 소모된다.
3. **int attack\_fire(char type, int defense)**
   * 기능: 불 속성의 마법 공격을 수행한다.
   * 구현: 불 속성 공격의 특성에 따른 추가 데미지 계산을 하며, type과 defense를 사용하여 계산된다.
4. **int attack\_grass(char type, int defense)**
   * 기능: 풀 속성의 마법 공격을 수행한다.
   * 구현: 풀 속성의 특성을 반영한 데미지 계산이 포함된다.
5. **int attack\_water(char type, int defense)**
   * 기능: 물 속성의 마법 공격을 수행한다.
   * 구현: 물 속성 공격의 특성에 따른 추가 데미지 계산이 이루어진다.

**8. void character\_reward(int reward)**

* 기능: 몬스터를 처치한 후 캐릭터에게 보상을 제공한다.
* 매개변수:
  + reward: 캐릭터에게 제공될 경험치의 양.
* 구현:
  + 캐릭터의 현재 경험치(curr\_exp)에 reward를 추가한다.
  + level\_up() 메서드를 통해judge\_level\_up() 메서드를 호출하여 현재 경험치가 레벨업에 필요한 경험치를 초과했는지 확인한다.
  + 레벨업 조건을 충족하면, level\_up() 메서드를 호출하여 캐릭터의 레벨업 처리를 수행한다. 매개변수에 true값을 대입해 레벨업 과정에서 발생한 변화를 출력한다.

**2. Class: Map**

개요: 멤버 변수 map\_info를 외부에서 접근하지 못하도록 Private안에 정의하고 나머지 메서드들은 Public에 구현했다.

**< 변수 >**

Private 멤버 변수

* std::vector<std::vector<char>> map\_info;
  + 2차원 벡터로, 맵의 각 칸에 대한 정보를 저장한다. 각 칸은 특정 문자로 표현되며, 이 문자들은 지형의 종류를 나타낸다 (예: 'E'는 비어있는 곳, 'M'은 몬스터가 있는 곳, 'S'는 상점이 있는 곳, 'R'은 랜덤 상황이 발생하는 곳).

**< 메서드 >**

**1. 생성자 Map()**

* 기능: 맵의 초기 상태를 설정한다.
* 구현:
  + 맵의 각 칸은 다양한 문자로 구성되며, 이는 지형의 종류를 나타낸다 (예: 'E'는 비어있는 곳, 'M'은 몬스터가 있는 곳).
  + map\_info 벡터는 8x8 그리드로 초기화되며, 각 칸에는 특정 문자가 할당된다.
  + 구현에서는 맵의 각 행을 초기화하는 방식으로, 각 행별로 다른 지형들이 설정된다.

**2. void draw\_map(Character\* myCharacter)**

* 기능: 맵을 시각적으로 그리며, 플레이어의 현재 위치와 방문한 타일을 표시한다.
* 구현:
  + 맵 전체를 그리기 전에 플레이어의 현재 위치(curr\_position)를 확인한다.
  + 유효하지 않은 위치(맵 바깥)일 경우, "Invalid position" 메시지를 출력하고 함수를 종료한다.
  + 맵의 상단에 경계선을 그린 후, 맵의 각 행을 순회하며 각 칸을 출력한다.
  + 플레이어의 현재 위치는 "You"로 표시되며, 방문한 타일은 공백으로 표시된다. 이는 플레이어의 move\_history를 확인하여 방문 여부를 결정한다.

**3. int map\_option\_select()**

* 기능: 맵 상에서 사용자에게 선택을 제공하는 메뉴를 관리한다.
* 구현:
  + ( 1. 방향 선택 | 2. Character 상태 보기 | 3. 게임 종료 ) 중 번호를 선택하게 하고 번호가 1, 2, 3 중 하나이면 그 번호를 반환하고 그렇지 않을 경우 다시 입력 받는다.

**4. char map\_move(Character\* myCharacter)**

* 기능: 사용자의 입력에 따라 캐릭터를 이동시키고, 새로운 위치의 칸 타입에 따라 해당 이벤트를 트리거한다.
* 구현:
  + 사용자로부터 이동할 방향을 입력받는다 (예: 'w', 'a', 's', 'd').
  + 입력받은 방향에 따라 캐릭터의 위치를 갱신한다.
  + 새로운 위치가 맵의 경계를 벗어나지 않는지 검사하고, 유효하면 캐릭터의 curr\_position을 업데이트하며, move\_history에 추가한다.
  + 새 위치의 칸에 따라 다른 이벤트를 트리거할 수 있다 (예: 'S'는 상점, 'M'은 몬스터).
  + 상점에 도착하면 'S'를, 몬스터가 있는 위치면 'M'을 반환하며, Shop을 제외하고 이미 방문한 위치이면 'N'을 반환한다.

**3. Class: Monster**

- 개요: Character Class와 마찬가지로 외부에서 변수들에 쉽게 접근하지 못하게 변수들을 Private안에서 정의했고 public 메서드 내에서만 쓰이는 메서드는 Private내에 구현했다. 변수들이 필요한 상황에서 해당 변수를 반환하는 메서드를 따로 구현해 Public과 Private의 경계를 명확하게 했다.

**< 변수 >**

Private 멤버 변수

* int difficulty; - 몬스터의 난이도를 나타내며, 몬스터의 위치에 따라 결정된다.
* int curr\_hp; - 몬스터의 현재 체력이다.
* int max\_hp; - 몬스터의 최대 체력이다.
* int attack; - 몬스터의 공격력이다.
* int reward; - 몬스터를 처치했을 때 획득할 수 있는 보상이다.
* int defense; - 몬스터의 방어력이다.
* char type; - 몬스터의 유형을 나타낸다 ('G'는 풀, 'W'는 물, 'F'는 불, 'B'는 노말).

**< 메서드 >**

**1. 생성자 Monster(vector<int> curr\_position, bool boss)**

* 기능: 몬스터 객체를 초기화하고, 위치와 보스 여부에 따라 몬스터의 특성을 설정한다.
* 구현:
  + 위치에 따른 난이도 계산: 난이도는 curr\_position의 x와 y 좌표 합으로 계산된다.
  + 기본 체력, 공격력, 방어력 설정: 난이도에 비례하여 max\_hp, attack, defense를 설정한다.
  + 보스 여부에 따라 추가 특성 설정: boss가 참이면, 고정된 높은 체력과 공격력, 방어력을 부여한다.
  + 몬스터 유형 결정: \_random\_type() 함수를 호출해 몬스터의 유형을 무작위로 설정한다.

**2. void battle\_status()**

* 기능: 전투 중 몬스터의 현재 상태를 콘솔에 출력한다.
* 구현:
  + 현재의 공격력, 방어력, 체력을 사용하여 상태 정보를 포맷팅하여 출력한다.

**3. void change\_status(int damage)**

* 기능: 몬스터가 받은 피해를 처리하여 현재 체력을 갱신한다.
* 구현:
  + 입력받은 damage 값을 현재 체력 curr\_hp에서 차감한다.

**4. int attack\_character(int defense)**

* 기능: 몬스터가 캐릭터를 공격할 때 계산된 피해량을 반환한다.
* 구현:
  + 몬스터의 공격력과 캐릭터의 방어력을 이용해 실제 피해량을 계산한다.
  + 계산 공식은 attack(monster) - defense(character)이며, 결과가 음수가 되지 않도록 0 이하인 경우 0을 반환한다.

**5. char \_random\_type()**

* 기능: 몬스터의 유형을 랜덤하게 결정한다.
* 구현:
  + 가능한 유형 배열(예: 'G', 'W', 'F', 'B')에서 랜덤하게 하나를 선택한다.
  + 선택된 유형을 반환한다.

**6. char monster\_type()**

* 기능: 몬스터의 유형을 반환한다.
* 구현:
  + 몬스터의 유형이 저장된 type 변수의 값을 반환한다.

**7. int monster\_defense()**

* 기능: 몬스터의 방어력을 반환한다.
* 구현:
  + 저장된 defense 변수의 값을 반환한다.

**8. int monster\_curr\_hp()**

* 기능: 몬스터의 현재 체력을 반환한다.
* 구현:
  + 저장된 curr\_hp 변수의 값을 반환한다.

**9. int monster\_reward()**

* 기능: 몬스터를 처치했을 때 받을 수 있는 보상을 반환한다.
* 구현:
  + 저장된 reward 변수의 값을 반환한다.

**4. Class: Shop**

**< 변수 >**

Private 멤버 변수

1. int plus\_attack;
   * 공격력 업그레이드를 구매할 수 있는 횟수를 나타낸다. 초기값은 1이며, 이는 한 번만 구매할 수 있음을 의미한다. 고객이 공격력 업그레이드를 구매하면 이 값을 감소시켜 재구매를 방지한다.
2. int plus\_defense;
   * 방어력 업그레이드를 구매할 수 있는 횟수를 나타낸다. 이 변수도 초기값은 1로 설정되어 있으며, 한 번 구매 후에는 더 이상 구매할 수 없다. 구매가 이루어지면 감소된다.
3. int plus\_exp;
   * 경험치 업그레이드를 구매할 수 있는 횟수를 나타낸다. 경험치 업그레이드도 한 번만 구매가 가능하며, 구매 시 해당 값이 감소한다.

**< 메서드 >**

**1. 생성자 Shop()**

* 기능: Shop 객체의 초기 상태를 설정한다.
* 구현:
  + 각 업그레이드 유형(공격력, 방어력, 경험치)의 초기 가용성을 설정한다. 공격력, 방어력, 경험치 모두 1로 초기화하여 한 번씩만 구매할 수 있다는 것을 의미한다.

**2. 메서드 void shop\_product(int choiceNum)**

* 기능: 상점에서 판매하는 상품 목록을 보여준다.
* 구현:
  + 각 상품 옵션 옆에 가격과 구매 가능 여부를 표시한다.
  + 선택된 상품(choiceNum에 해당하는 상품)은 "Sold Out"으로 상태 변경을 표시한다.
  + 각 상품의 가용성은 plus\_attack, plus\_defense, plus\_exp 변수를 확인하여 결정한다.
  + 상품 번호에 따라 해당 상품이 구매 가능한지 여부를 표시하며, 구매 불가능할 경우 "Sold Out"이라고 표시한다.

**3. 메서드 void shop\_choice(Character\* myCharacter)**

* 기능: 상점에서 사용자의 선택을 처리한다.
* 구현:
  + 사용자에게 현재 골드 상태와 함께 상품 목록을 보여준다.
  + 사용자의 입력을 받아 선택된 상품 번호에 따라 다음과 같이 처리한다:
    - 1-3번 (각각 공격력, 방어력, 경험치 업그레이드): 각 상품의 구매 조건(가용성과 필요 골드)을 검사하고, 조건을 충족하면 해당 업그레이드를 적용하고, 상품의 가용성을 감소시킨다.
    - 4-5번 (HP, MP 회복): 충분한 골드가 있을 경우 회복을 진행하고, 비용을 차감한다.
    - 6번 (상점 나가기): success 변수를 2로 바꿔 반복문을 탈출하고 상점을 종료한다.
  + 상품 구매가 성공적으로 이루어지면 success변수를 1로 바꿔 반복문을 탈출하고 구매 내역을 출력하고, 상품 목록을 새로 고친다.
  + 상품 구매가 성공적이지 않으면, 골드 부족 등의 메시지를 출력한다.
  + 상점 나가기 선택 시 또는 구매 후 자동으로 상점을 나가게 처리한다.

**5. Class: Battle**

- 개요: Private 멤버 변수에 자주 쓸 값(Character 방어력, Monster 타입/방어력)을 저장해 Public 메서드에서 사용하도록 구현했다.

**< 변수 >**

Private 멤버 변수

1. int character\_defense;
   * 이 변수는 전투 중인 캐릭터의 방어력을 저장한다. 캐릭터 객체로부터 초기화되며, 전투 중 몬스터의 공격을 계산할 때 사용된다.
2. int monster\_defense;
   * 몬스터의 방어력을 저장하는 변수다. 몬스터 객체에서 초기화되며, 캐릭터의 공격 피해량을 계산하는 데 필요하다.
3. int monster\_type;
   * 몬스터의 유형을 저장한다. 이 유형은 몬스터의 특성을 나타내며, 전투 중 특정 유형의 공격이 효과적인지 결정하는 데 사용될 수 있다. 예를 들어, 특정 유형의 몬스터에게 추가 피해를 주는 특수 공격을 적용할 때 이 변수를 참조한다.

**< 메서드 >**

**1. Battle(Character\* myCharacter, Monster\* myMonster)**

* 기능: Battle 인스턴스를 초기화하고, 전투에 필요한 캐릭터와 몬스터의 방어력 및 몬스터의 유형을 설정한다.
* 구현 세부사항:
  + character\_defense 멤버 변수에 myCharacter->character\_defense() 메서드 호출 결과를 저장하여 캐릭터의 방어력을 설정한다.
  + monster\_defense 멤버 변수에 myMonster->monster\_defense() 메서드 호출 결과를 저장하여 몬스터의 방어력을 설정한다.
  + monster\_type 멤버 변수에 myMonster->monster\_type() 메서드 호출 결과를 저장하여 몬스터의 유형을 설정한다.

**2. int \_choose\_action(Character\* myCharacter, Monster\* myMonster)**

* 기능: 전투 동안 사용자로부터 행동 선택을 입력받아 그 번호를 반환한다.
* 구현 세부사항:
  + 캐릭터와 몬스터의 현재 전투 상태(battle\_status)를 출력한다.
  + 사용자에게 가능한 전투 행동을 출력한다: 기본 공격, 일반 마법 공격, 화염/풀/물 마법 공격.
  + 입력 반복 루프 내에서, 유효한 행동 번호(1~5)를 사용자로부터 입력받는다. 유효한 번호가 입력될 때까지 계속 입력을 요청한다.
  + 입력된 행동 번호를 반환한다.

**3. char battle\_action(Character\* myCharacter, Monster\* myMonster)**

* 기능: 전투 시퀀스를 진행하고 전투 결과를 문자로 반환한다.
* 구현 세부사항:
  + 전투 시작을 알리는 메시지를 출력하고, \_choose\_action을 호출하여 사용자의 선택을 받는다.
  + 선택된 행동에 따라 캐릭터의 마나를 확인(마법 공격 시)하고, 부족할 경우 경고 메시지를 출력 후 다시 행동을 선택하게 한다.
  + 선택된 공격 유형에 따라 적절한 공격 메서드(attack\_basic, attack\_skill, attack\_fire, attack\_grass, attack\_water)를 호출하여 몬스터에게 적용할 피해를 계산한다.
  + 계산된 피해(CtoM\_damage)를 myMonster->change\_status()를 호출하여 몬스터에 적용하고, 공격 결과를 출력한다.
  + 몬스터의 반격: myMonster->attack\_character(character\_defense)를 호출하여 캐릭터에 적용할 피해를 계산하고, myCharacter->change\_status()를 사용하여 캐릭터에 적용한다.
  + 캐릭터 또는 몬스터의 체력이 0 이하로 떨어질 경우, 전투 결과('M' 또는 'C')를 반환한다. ('C': Character wins | 'M': Monster wins)
  + 몬스터가 패배했을 경우, 몬스터의 보상(monster\_reward)을 출력하고, 캐릭터에게 적용한다.

**2-2. 전체적 알고리즘 설명 - main().cpp**

**메인 함수 int main()**

* 랜덤 시드 초기화: srand(time(NULL));을 사용하여 랜덤 수 생성기를 초기화한다.
* 게임 내 객체 생성: Map, Character, Shop 객체를 생성한다. 이들은 게임을 진행하는 데 필요한 기본 요소들이다.
* 메인 메뉴 반복 처리:
  + 사용자로부터 메인 메뉴 선택을 받는다(option\_select() 호출).
  + 선택된 옵션에 따라 게임을 시작하거나 종료한다.

**메인 메뉴 옵션 선택 int option\_select()**

* 기능: 사용자에게 메인 메뉴에서 선택할 수 있는 옵션을 제공하고, 유효한 입력을 받는다.
* 구현 세부사항:
  + 사용자로부터 입력을 받아, 유효한 옵션(1: 게임 시작, 2: 종료)만을 받아들인다. 유효한 입력을 받으면 해당 값을 반환한다.

**게임 시작 (optionNum == 1)**

* 맵 상태 출력 및 옵션 선택: 게임 맵을 출력하고(myMap.draw\_map(&myCharacter);), 사용자의 입력에 따라 다음 행동을 선택하게 한다(myMap.map\_option\_select();).
* 맵 이동 및 이벤트 처리:
  + 이동 및 위치 기반 이벤트 처리: 사용자가 이동을 선택하면(map\_optionNum == 1), 새 위치의 유형에 따라 적절한 이벤트를 처리한다. 이벤트 유형에는 빈 공간, 랜덤 이벤트, 몬스터 또는 보스 전투, 상점 방문이 포함된다.
  + 전투 처리: 몬스터 또는 보스와의 전투가 시작되면 Battle 객체를 생성하고 전투를 진행한다(myBattle.battle\_action(&myCharacter, &myMonster);). 전투 결과에 따라 게임을 종료하거나 계속 진행한다.
  + 상점 처리: 상점을 방문하면 상점 메뉴를 출력하고 거래를 처리한다(myShop.shop\_choice(&myCharacter);).
* 캐릭터 상태 출력: 사용자가 캐릭터 상태 확인을 선택하면(map\_optionNum == 2), 캐릭터의 현재 상태를 출력한다(myCharacter.show\_status();).
* 게임 종료 선택: 사용자가 게임 종료를 선택하면 게임을 종료한다(return 0;).

**2-3. 변수 설명 - main().cpp**

**1. Map myMap;**

* 이 변수는 Map 클래스의 인스턴스로, 게임의 맵을 관리한다. 맵의 현재 상태를 표시하고, 플레이어의 이동을 처리하는 등의 기능을 수행한다.

**2. Character myCharacter;**

* 이 변수는 Character 클래스의 인스턴스로, 게임 내의 플레이어 캐릭터를 나타낸다. 캐릭터의 속성(체력, 공격력 등)과 상태(위치, 경험치 등)를 관리한다.

**3. Shop myShop;**

* 이 변수는 Shop 클래스의 인스턴스로, 게임 내의 상점 기능을 관리한다. 플레이어가 아이템을 구매하거나 판매할 수 있게 하는 등의 상점 관련 활동을 총괄한다.

**4. int optionNum;**

* 이 정수형 변수는 메인 메뉴에서 플레이어가 선택한 옵션을 저장한다. 이 값을 통해 게임이 시작되거나 종료될 수 있다.

**5. int map\_optionNum;**

* 이 정수형 변수는 맵에서 플레이어가 선택한 옵션을 저장한다. 이 값에 따라 플레이어는 맵에서 다양한 행동을 취할 수 있다(이동, 상태 확인, 게임 종료).

**6. char place;**

* 이 문자형 변수는 플레이어가 맵에서 이동한 후 도달한 위치의 유형을 나타낸다. 'E'는 빈 공간, 'M'은 몬스터가 있는 위치, 'R'은 랜덤 위치, 'B'는 보스 위치, 'S'는 상점 위치

**7. Monster myMonster;**

* 이 변수는 몬스터와의 전투에서 사용되는 Monster 클래스의 인스턴스이다. 몬스터의 유형, 위치, 체력 등을 관리하며 전투 로직에 중요한 역할을 한다.

**8. Battle myBattle;**

* 이 변수는 전투를 관리하는 Battle 클래스의 인스턴스이다. 캐릭터와 몬스터 간의 전투를 진행하며, 전투의 결과를 처리한다.

**3. 토론 및 개선**

**3-1. 배우거나 깨달은 내용**

1. **객체 지향 프로그래밍의 실제 적용:** Character, Map, Monster, Shop, Battle 등 여러 클래스의 인스턴스를 생성하고 관리하는 과정을 통해 객체 지향 프로그래밍의 원칙과 이점을 실제로 적용해 보았다. 각 클래스는 특정 기능과 데이터를 캡슐화하여 프로그램의 구조를 명확하게 하고, 코드 재사용성을 높이며, 유지보수를 용이하게 함을 깨달았다.
2. **에러 처리와 게임의 안정성:** 사용자 입력이나 게임 로직 처리 중 발생할 수 있는 예외 상황을 고려하는 것의 중요성을 이해했다. 예를 들어, 잘못된 입력이나 예상치 못한 게임 상태 변화를 적절히 처리하지 않으면 게임의 실행이 중단되는 경우가 있었다.
3. **모듈화와 코드 구성:** 각 클래스와 함수가 특정한 기능을 담당하도록 모듈화를 잘 해야 전체 코드의 가독성과 관리가 용이해진다는 점을 배웠다. 적절한 모듈화를 통해 프로그램의 복잡성을 관리하고, 개별 요소의 테스트와 디버깅을 간소화했다

**3-2. 개선 방향**

1. **세이브 및 로드 기능**: 사용자가 게임의 현재 상태를 저장하고 나중에 다시 불러올 수 있는 기능을 추가한다.
2. **퀘스트 및 업적 시스템**: 플레이어가 달성할 수 있는 다양한 목표와 업적을 설정하고, 이를 완료할 때마다 보상을 제공한다.