

C++ 프로그래밍 및 실습

던전에서 살아남기

최종 보고서

제출일자: 2024/12/21

제출자명: 이재륜

제출자학번: 213235

1. 프로젝트 목표

1) 배경 및 필요성

현재 수업 중에 만든 MUD 게임은 텍스트 위주로 구성되어 있어 시각적 재미가 부족하고 캐릭터나 아이템이 단순한 텍스트로만 표시됩니다. 이러한 요소는 게임의 몰입도를 떨어뜨리고 플레이어가 상상에 의존하여 즐기기에 한계가 있습니다.

따라서 이 프로젝트에서는 ASCII 기반의 아트를 사용하여 캐릭터, 아이템, 몬스터 등을 시각적으로 표현하여 게임에 생동감을 더하여 플레이어가 더 직관적으로 게임에 몰입할 수 있도록 합니다. 또한 다양한 스토리 선택지와 전투 및 아이템 시스템을 통해 기존의 MUD 게임보다 즐길거리가 풍부한 경험을 제공하여 사용자가 게임으로 하여금 재미를 느낄 수 있게 합니다.

2) 프로젝트 목표

ASCII 아트 기반의 시각적 요소와 선택형 스토리 및 난이도 설정 시스템을 통해 플레이어가 몰입감 넘치는 모험을 경험하도록 하는 텍스트 기반 게임을 만드는 것을 목표로 합니다.

3) 차별점

기존 MUD 게임의 단순한 텍스트 기반에서 벗어나 ASCII 아트를 활용하여 시각적 흥미를 추가합니다. 더불어 플레이어가 던전 난이도를 선택할 수 있으며 난이도에 따라 몬스터의 강약이 변화하여 각기 다른 도전의 재미를 제공합니다.

2. 기능 계획

1) 기능 1: 스토리 진행 및 선택 시스템

- 텍스트와 ASCII 아트로 제공되는 스토리를 따라가며 플레이어가 선택지를 통해

모험을 진행하는 기능

(1) 세부 기능 1 (상황 묘사 및 선택지 제공)

- 미궁 속에서 발생하는 다양한 상황을 텍스트와 ASCII 아트를 통해 묘사하고 플레이어가 선택할 수 있는 여러 선택지를 제공합니다.

(2) 세부 기능 2 (선택에 따른 결과 제공)

- 플레이어가 선택한 옵션에 따라 스토리 전개가 달라지며 전투 또는 아이템 발견 등의 이벤트가 발생합니다.

2) 기능 2: 던전 난이도 선택 시스템

- 게임 시작 시 플레이어가 던전의 난이도를 선택할 수 있으며 선택한 난이도에 따라 등장하는 몬스터의 강약이 달라지는 기능

(1) 세부 기능 1 (난이도별 몬스터 설정)

- 쉬움, 보통, 어려움의 난이도를 선택할 수 있으며 난이도가 높아질수록 몬스터의 체력과 공격력이 증가합니다.

(2) 세부 기능 2 (선택에 따른 결과 제공)

- 플레이어가 선택한 옵션에 따라 스토리 전개가 달라지며 전투 또는 아이템 발견 등의 이벤트가 발생합니다.

3) 기능 3: 전투 시스템

- 플레이어와 적이 차례대로 공격하는 텍스트 기반 전투 기능

(1) 세부 기능 1 (플레이어와 적의 공격 및 체력 관리)

- ASCII 아트를 사용해 전투 중 캐릭터와 무기를 출력하고 플레이어와 적의 체력 및 공격력을 관리합니다.

(2) 세부 기능 2 (던전 난이도에 따른 전투 밸런스 조정)

- 선택한 던전 난이도에 따라 전투의 밸런스가 조정되어 플레이어가 도전을 즐길 수 있도록 설계합니다.

4) 기능 4: 아이템 및 보상 시스템

- 플레이어가 미궁에서 아이템을 발견하거나 전투에서 승리하면 보상을 획득하는 기능

(1) 랜덤 아이템 드랍

- 전투 승리 후 무작위로 회복 아이템 장비 등 다양한 아이템을 드랍하여 게임의 재미를 높입니다.

(2) 아이템 사용 및 효과 적용

- ASCII 아트로 아이템을 표현하고 플레이어가 아이템을 사용하여 체력을 회복하거나 전투 중 공격력을 높일 수 있도록 하는 등의 기능을 합니다.

3. 진척사항

1) 기능 구현

(1) 메인 입출력 함수 (2주차 내용)

- 입출력
 - 입력: 플레이어의 선택 (0, 1, 2, 3, 4, 5, 9 중 하나)
 - 0: 게임 종료
 - 1, 2, 3: 각각 첫 번째, 두 번째, 세 번째 길을 선택
 - 4, 5: 각각 히든 던전, 보스의 방 선택 (특정 조건 달성시 던전이 열려 선택 가능)

- 9 :가방 열기 선택
- **출력:**
 - 게임 스토리 및 ASCII 아트 출력
 - 플레이어 선택에 따른 결과 메시지
 - 잘못된 입력 시 경고 메시지 출력
 - 0 입력 시 종료 메시지 출력

- 설명

- 이 코드는 사용자와의 상호작용을 통해 텍스트 기반 게임을 진행합니다.
- 초기 스토리와 선택지를 보여주고, 사용자 입력에 따라 특정 이벤트(몬스터와의 전투, 보물 발견 등)를 실행합니다.
- while 루프를 통해 게임은 종료될 때까지 계속 진행되며, 각 선택에 따라 함수 (ProcessChoice(), ViewBag() 등)가 호출됩니다.
- 유효하지 않은 입력에 대해서는 오류 메시지를 출력하고, 사용자에게 재입력을 요청합니다.
- 특정 조건을 충족할 경우, 히든 던전과 보스 방이 잠금 해제됩니다.

- 적용된 배운 내용

- **반복문:**
 - while 루프를 사용해 게임이 종료될 때까지 사용자 입력을 지속적으로 처리합니다.
- **조건문:**
 - if-else를 활용해 입력값에 따라 게임 진행 방향을 결정합니다.
 - 유효성 검사(예: `choice >= 1 && choice <= 3`)로 올바른 입력인지 확인합니다.
- **예외 처리:**

- try-catch 블록을 사용해 stoi 함수로 입력값 변환 시 발생할 수 있는 예외 (invalid_argument, out_of_range)를 처리합니다.

- 함수 호출:

- DisplayIntro()로 게임 초기 스토리 출력.
- ProcessChoice()로 선택한 길에 따른 이벤트 처리.
- ViewBag()으로 가방 열기 및 아이템 사용 구현.

- 랜덤성:

- srand(time(0))으로 게임 이벤트의 무작위성을 보장.

- 가독성 향상:

- 주요 작업을 함수로 분리해 코드의 구조적 가독성을 높이고 유지보수를 용이하게 함.

- 코드 스크린샷

```
// 메인 함수
int main() {
    int choice;
    srand(time(0));
    DisplayIntro();

    while (true) {
        PresentChoices();
        cout << "선택을 입력하세요 (0-5, 9): ";
        cin >> choice;

        if (choice == 0) {
            DisplayDieArt();
            cout << "게임을 종료합니다. 철수는 당신이 끝까지 지켜봐주지 않아 고독사했습니다." << endl;
            break;
        }

        if (choice == 9) {
            ViewBag();
            continue;
        }

        cout << "현재 물리친 몬스터 수: " << monster_defeat_count << "\n";

        if (choice >= 1 && choice <= 3 || (choice == 4 && hidden_dungeon_unlocked) || (choice == 5 && amumu_boss_unlocked)) {
            ProcessChoice(choice);
        } else {
            cout << "잘못된 선택입니다. 다시 선택해 주세요.\n";
        }
    }

    return 0;
}
```

(2) 플레이어의 선택에 따른 결과를 처리하는 함수 (3주차 내용)

- 입출력

- **입력:** 플레이어의 선택값 (choice)과 랜덤 이벤트 결정값 (event).
 - choice: 1 (첫 번째 길), 2 (두 번째 길), 3 (세 번째 길).
 - event: 랜덤으로 결정되는 이벤트 번호 (0, 1, 2, 3 중 하나).
- **출력:**
 - 랜덤 이벤트에 따라 적의 ASCII 아트 출력 및 전투 메시지.
 - 보물상자 발견 시 보물상자 ASCII 아트와 아이템 획득 메시지.
 - 길에 따라 다른 적 또는 보물 이벤트 출력.

- 설명

- 사용자가 선택한 길에 따라 무작위로 이벤트(몬스터와 전투, 보물상자 발견)가 발생합니다.
- 각 길마다 이벤트 발생 확률이 다르며 세 번째 길에서는 고블린 전사, 궁수, 마법사와의 전투 또는 보물상자 발견이 동일 확률(25%)로 일어납니다.
- 이벤트 발생 시 ASCII 아트를 출력하고 보물상자가 발견되면 dropItem() 함수를 호출합니다.
- 특정 조건 달성시 히든 던전 및 보스 던전의 방이 열리도록 함.
- 전투시스템을 위하여 몬스터들의 체력과 공격력을 Battle함수로 묶어 전달함.

- 적용된 배운 내용

- **랜덤 처리:** rand()와 % 연산을 활용해 이벤트를 무작위로 선택.
- **조건문:** if-else를 사용해 선택된 길과 이벤트를 처리.
- **함수 호출:** 이벤트에 따른 ASCII 아트 출력하는 함수나 보물상자 아이템 드랍 처리를 하는 함수 등을 함수 이름만으로 간단히 호출함.

- 코드 스크린샷

```

// 플레이어의 선택에 따른 결과를 처리하는 함수
void ProcessChoice(int choice) {
    int event = rand() % 4; // 0, 1, 2, 3 중 랜덤 이벤트 선택 -> 선택지가 4개인 3번째 길을 고려하여 수정
    if (choice == 1) {
        // 첫 번째 길: 미노타우르스, 멧돼지, 보물상자
        if (event == 0) {
            DisplayMinotaurArt();
            cout << "도끼를 든 미노타우르스가 돌진해옵니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("미노타우르스", 50, 7);
        } else if (event == 1) {
            DisplayTreasureArt();
            cout << "보물상자를 발견했습니다.\n";
            DropItem(1); // 첫 번째 길 아이템 드랍
        } else {
            DisplayBoarArt();
            cout << "야생 멧돼지가 돌진해옵니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("야생 멧돼지", 20, 5);
        }
    } else if (choice == 2) {
        // 두 번째 길: 해골전사, 마녀, 보물상자
        if (event == 0) {
            DisplayWitchArt();
            cout << "마녀가 당신을 노려봅니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("마녀", 11, 10);
        } else if (event == 1) {
            DisplayTreasureArt();
            cout << "보물상자를 발견했습니다.\n";
            DropItem(2); // 두 번째 길 아이템 드랍
        } else {
            DisplaySkeletonWarriorArt();
            cout << "해골전사가 나타났습니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("해골 전사", 15, 7);
        }
    } else if (choice == 3) {
        // 세 번째 길: 고블린 전사, 궁수, 마법사, 보물상자
        if (event == 0) {
            DisplayGoblinWarriorArt();
            cout << "고블린 전사가 나타났습니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("고블린 전사", 20, 5);
        } else if (event == 1) {
            DisplayGoblinArcherArt();
            cout << "고블린 궁수가 당신을 겨누고 있습니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("고블린 궁수", 15, 3);
        } else if (event == 2) {
            DisplayGoblinMageArt();
            cout << "고블린 마법사가 주문을 외우고 있습니다!\n";
            cout << "전투를 시작합니다\n";
            Battle("고블린 마법사", 11, 7);
        } else {
            DisplayTreasureArt();
            cout << "보물상자를 발견했습니다.\n";
            DropItem(3); // 세 번째 길 아이템 드랍
        }
    } else if (choice == 4 && hidden_dungeon_unlocked) {
        EnterHiddenDungeon();
    } else if (choice == 5 && amumu_boss_unlocked) {
        EnterAmumuBossRoom();
    }
}
}

```



```
// 플레이어에게 선택지를 보여주는 함수
void PresentChoices() {
    cout << "\n어떻게 할까요?\n";
    cout << "1. 첫번째 길로 간다\n";
    cout << "2. 두번째 길로 간다\n";
    cout << "3. 세번째 길로 간다\n";
    if (hidden_dungeon_unlocked) {
        cout << "4. 히든 던전: 대악마 아트록스의 방\n";
    }
    if (amumu_boss_unlocked) {
        cout << "5. 슬픈 미라 아무무의 방\n";
    }
    cout << "0. 게임종료\n";
    cout << "9. 가방을 열다\n";
}
}
```

(3) 보물상자를 발견했을 때 아이템을 랜덤으로 드랍하는 함수 (5주차 내용)

- 입출력

- **입력:** 함수 `dropItem()`이 호출될 때 전달된 path 값 (1, 2, 또는 3).
 - `path == 1`: 첫 번째 길
 - `path == 2`: 두 번째 길
 - `path == 3`: 세 번째 길
- **출력:** 보물상자에서 랜덤으로 선택된 아이템 이름이 출력됩니다.

- 설명

- 각 길에서 발견할 수 있는 고유한 보물상자 아이템을 랜덤으로 드랍합니다.
- 길에 따라 드랍 아이템 목록이 다르며, 목록 내에서 무작위로 선택됩니다.

- 적용된 배운 내용

- **배열:** 각 길의 고유 아이템 리스트를 배열로 저장.
- **랜덤 처리:** `rand() % 배열 크기`로 무작위 아이템 선택.
- **조건문:** `if-else`를 사용해 선택된 길에 따라 적절한 배열 사용.

- 코드 스크린샷

```
// 보물상자를 발견했을 때 아이템을 랜덤으로 드랍하는 함수
void DropItem(int path) {
    int item_index;
    string item;

    if (path == 1) {
        string items[] = {"피물은 도끼", "가죽갑옷", "빨간포션"};
        item_index = rand() % 3;
        item = items[item_index];
    } else if (path == 2) {
        string items[] = {"날카로운 철검", "빨간포션", "천갑옷"};
        item_index = rand() % 3;
        item = items[item_index];
    } else {
        string items[] = {"야만의 몽둥이", "빨간포션"};
        item_index = rand() % 2;
        item = items[item_index];
    }
    if (item == "피물은 도끼") {
        DisplayAxeArt();
    } else if (item == "가죽갑옷") {
        DisplayArmor2Art();
    } else if (item == "빨간포션") {
        DisplayPotionArt();
    } else if (item == "날카로운 철검") {
        DisplaySwordArt();
    } else if (item == "천갑옷") {
        DisplayArmorArt();
    } else if (item == "야만의 몽둥이") {
        DisplayMaceArt();
    }
    cout << "보물상자에서 " << item << "을(를) 얻었습니다!\n";
    player_bag.push_back(item);
}
}
```

(4) ASCII 아트 출력 및 스토리 출력 함축 함수 (2주차, 3주차 내용)

- 입출력

- 입력:
 - 별도의 사용자 입력은 없음. 함수 호출 시 고정된 ASCII 아트와 스토리를 출력.
- 출력:

- 특정 이벤트(예: 철수가 잠든 모습, 몬스터, 보물상자 등)에 따른 ASCII 아트와 메시지를 출력.
- 스토리 소개를 통해 게임의 배경 설명과 플레이어의 몰입도 증가.

- 설명

- 각 함수는 고유한 ASCII 아트를 출력하며, 게임 스토리와 이벤트를 시각적으로 표현.
- 게임의 서사적 요소와 시각적 몰입감을 제공하기 위해 활용됨.
 - displayAsciiArt: 철수가 책상에서 잠든 모습을 ASCII 아트로 표현.
 - displayMinotaurArt, displayBoarArt, displaySkeletonWarriorArt 등: 적의 모습을 ASCII 아트로 출력.
 - displayTreasureArt: 보물상자 발견 시 아트 출력.
 - displayDieArt: 게임 종료 시 사망 메시지를 출력.
 - displayIntro: 게임 시작 시 배경 스토리를 출력.
 - DisplayAxeArt, DisplayArmorArt, DisplayArmor2Art, DisplayPotionArt, DisplaySwordArt, DisplayMaceArt : 보물상자에서 나온 아이템을 출력
 - DisplayAtroxArt DisplayAmumuArt : 히든 던전 보스 아트룩스 출력, 최종 던전 보스 아무무 출력
 - DisplayGameClear() : 게임 클리어 출력
 - DisplayDarkinSwordArt() : 다르킨의 검 출력

- 적용된 배운 내용

- 함수:
 - 각 ASCII 아트를 별도의 함수로 분리하여 유지보수성과 가독성을 향상.
 - 필요할 때마다 호출 가능.
- 출력 제어:

- cout을 활용한 텍스트 출력.
- 줄바꿈(endl)과 공백 등을 활용해 아트를 정확한 위치에 배치.
- 코드 구조화:
 - 여러 적, 이벤트, 스토리를 각각의 함수로 분리하여 코드 중복 제거.
 - 스토리텔링 요소를 효율적으로 구현.

- 코드 스크린샷

```
// 던전에서 잠든 철수의 ASCII 아트를 출력하는 함수
void displayAsciiArt() {
    cout << "      zZZ...      " << endl;
    cout << "    ,-'.....-' " << endl;
    cout << "  , '          ' , " << endl;
    cout << " /              \\ " << endl;
    cout << " | ~~~~ ~~~~ | " << endl;
    cout << " | - - - - | " << endl;
    cout << " | ' _ _ _ ' | " << endl;
    cout << " \\          / " << endl;
    cout << " ' ,          ' " << endl;
    cout << "    '-.....-' " << endl;
}

// 미노타우르스의 ASCII 아트를 출력하는 함수
void displayMinotaurArt() {
    cout << "      --      " << endl;
    cout << "    /  \\~~~/  \\ " << endl;
    cout << "  ,----(    )----, " << endl;
    cout << " /      \\ _ ^ /      \\ " << endl;
    cout << " |          | |          | " << endl;
    cout << " \\      __/  \\__  / " << endl;
    cout << " '---/      \\---' " << endl;
    cout << "    | AXE!!! | " << endl;
    cout << "    \\_____/ " << endl;
}
```

```
// 옛돼지의 ASCII 아트를 출력하는 함수
void displayBoarArt() {
    cout << "      /\      /\      " << endl;
    cout << "    //\\ (o o) //\\ " << endl;
    cout << "    ||  \\^_//  ||  " << endl;
    cout << "    \\\\  '-'  //  " << endl;
    cout << "    '-.....-'      " << endl;
}

```

```
// 보물상자의 ASCII 아트를 출력하는 함수
void displayTreasureArt() {
    cout << "      _____      " << endl;
    cout << "      /           \\      " << endl;
    cout << "      / $$$$$$$$ \\      " << endl;
    cout << "      | $$$$$$$$ |      " << endl;
    cout << "      | $$$$$$$$ |      " << endl;
    cout << "      \\_____/[T]____/      " << endl;
}

```

```
// 해골전사의 ASCII 아트를 출력하는 함수
void displaySkeletonWarriorArt() {
    cout << "      .-.      " << endl;
    cout << "      (o.o)      " << endl;
    cout << "      |=|      " << endl;
    cout << "      _|_      " << endl;
    cout << "      //.=|.\\\\      " << endl;
    cout << "      // .|=|.\\\\      " << endl;
    cout << "      \\\\ .|=|. //      " << endl;
    cout << "      \\\\(_=_)//      " << endl;
    cout << "      (:| |: )      " << endl;
    cout << "      || ||      " << endl;
    cout << "      () ()      " << endl;
}

```

```
// 마녀의 ASCII 아트를 출력하는 함수
void displayWitchArt() {
    cout << "      _____      " << endl;
    cout << "      .-.'-----'-.      " << endl;
    cout << "      / .--.      \\      " << endl;
    cout << "      / _.'      '-._\\      " << endl;
    cout << "      | ( o      o _ ) |      " << endl;
    cout << "      \\  '._--._.' /      " << endl;
    cout << "      '-.      .-'      " << endl;
    cout << "      '-----'      " << endl;
}

```

```
// 고블린 전사의 ASCII 아트를 출력하는 함수
void displayGoblinWarriorArt() {
    cout << "      .----.      " << endl;
    cout << "      / o o \\      " << endl;
    cout << "      | ^ |      " << endl;
    cout << "      | \\_/_/ |      " << endl;
    cout << "      \\_____/      " << endl;
    cout << "      //  \\\\      " << endl;
    cout << "      ||  ||      " << endl;
}

```

```
// 고블린 궁수의 ASCII 아트를 출력하는 함수
void displayGoblinArcherArt() {
    cout << "      .----.      " << endl;
    cout << "      / o o \\      " << endl;
    cout << "      | ^ |      " << endl;
    cout << "      | \\_/_/ |      " << endl;
    cout << "      \\--| |--/      " << endl;
    cout << "      | |      " << endl;
}

```

```
// 고블린 마법사의 ASCII 아트를 출력하는 함수
void DisplayGoblinMageArt() {
    cout << "      ,----, " << endl;
    cout << "    /  o o  \\" << endl;
    cout << "    |    ^    | " << endl;
    cout << "    |  \\___/  | " << endl;
    cout << "    \\_--==--_/ " << endl;
    cout << "    /        \\" << endl;
}
```

```
// 사망 메시지 출력 함수
void DisplayDieArt() {
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
}
```

```
// 소개 스토리를 출력하는 함수
void DisplayIntro() {
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "[]" << endl;
    cout << "zzz..." << endl;
    cout << " ,-'.....-' " << endl;
    cout << " ,-' " << endl;
    cout << " /        \\" << endl;
    cout << " |  ~~~~  | " << endl;
    cout << " |  -    -  | " << endl;
    cout << " |  '.....'  | " << endl;
    cout << " |  \\        /  | " << endl;
    cout << " |  ,-'.....-'  | " << endl;
    cout << " |  ,-'.....-'  | " << endl;
    cout << "한밤중, 코딩에 집중하던 철수는 졸음에 못 이겨 그만 책상에 고꾸라져 잠이 들고 말았습니다.\n";
    cout << "얼마나 시간이 흘렀을까, 깨달아 보니 낮선 돌바닥 위, 주변은 차갑고 축축한 공기로 가득 찼습니다.\n";
    cout << "눈앞엔 고대 유적 같은 돌기둥들이 우뚝 서 있고, 어둠 속에서 흐릿한 불빛이 깜박입니다.\n";
    cout << "'여긴... 어디지?' 철수는 아찔한 느낌에 사로잡혔습니다. 뒤를 돌아보니 더는 돌아갈 길이 없었습니다.\n";
    cout << "갑자기 멀리서 낮게 울리는 소리-점승의 숨소리 같은 것이 들려옵니다.\n";
    cout << "'살아남으려면 뭔가를 찾아야 해...' 철수는 두근거리는 가슴을 안고 주변을 살피기 시작했습니다.\n";
    cout << "손을 뻗어 벽을 더듬는 순간, 날카로운 금속의 감촉이 느껴졌습니다.\n";
    cout << "철수는 작은 칼을 발견하고 움켜쥐었습니다. 이제, 이 던전에서 생존을 위한 첫 발걸음을 내디딤니다.\n\n";
    cout << "앞을 보니 세 가지 길이 있습니다.\n\n";
}
```

```
void DisplayAxeArt() {
```

```
cout << "
```

[illegible]

}//피묻은 도끼

```
cout << "
```

[illegible]

```
}//천갑옷
```

```
cout << "|||||";
```

[illegible]

}//가죽갑옷

```
cout << "
```

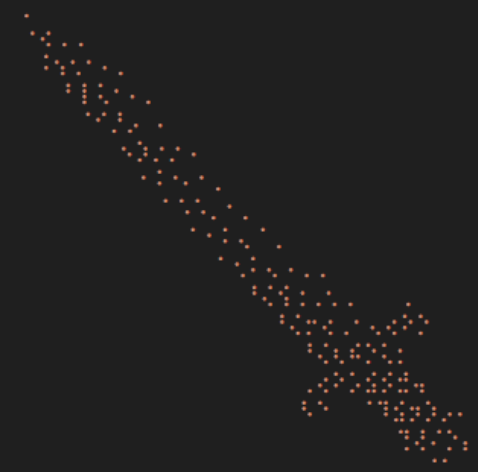
[illegible]

} //빨간 포션

```

void DisplaySwordArt() {
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
} //날카로운 철검

```



```

void DisplayMaceArt() {
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
    cout << "                                     " << endl;
} //야만의 몽둥이

```



(5) 전투시스템 관련 함수

- 입출력 및 설명

입력:

- 별도의 사용자 입력은 없음. 함수 호출 시, 전투 및 관련 이벤트에 따른 ASCII 아트와 스토리가 출력됨.
- 플레이어의 행동 선택에 따라 (싸운다/도망친다) 전투 결과와 체력 변화를 출력함.

출력:

- Battle: 전투 함수 호출 시, 플레이어와 몬스터의 현재 체력 및 상태를 출력하고, 사용자가 선택한 행동에 따른 결과를 출력함.
- 전투가 종료되면 승패 결과를 출력함. (플레이어 승리 또는 몬스터 승리)
- monster_defeat_count: 몬스터를 물리친 횟수에 따라 숨겨진 던전을 해제하는 이벤트를 출력함.

함수:

- Battle: 플레이어와 몬스터 간의 전투 로직을 처리하는 함수. 전투의 각 턴마다 플레이어와 몬스터의 체력, 공격력, 행동을 결정하고 출력함.
- Action: 플레이어가 선택한 행동(싸운다, 도망친다)에 따라 결과가 다르게 처리되며, 전투가 계속될지, 종료될지를 결정함.
- monster_defeat_count: 플레이어가 물리친 몬스터 수를 계산하여 특정 이벤트(히든 던전 또는 보스 등장)를 트리거함.

출력 제어:

- monster_health, player_health의 변화에 따라 실시간으로 출력되는 상태 메시지가 게임 진행 상황을 알려줌.

- 적용된 배운 내용

코드 구조화:

- 여러 몬스터 및 이벤트(보물상자 발견, 도망치는 상황 등)를 각기 다른 함수로 분

리하여 코드의 중복을 줄이고, 가독성을 높임.

- 각 이벤트가 독립적으로 처리되므로, 새로운 몬스터나 아이템을 추가하는 것이 용이함.

- 코드 스크린샷

```
// 아트록스의 정보
const int ATROX_HEALTH = 100;
const int ATROX_ATTACK = 0;

// 아무무의 정보
const int AMUMU_HEALTH = 150;
const int AMUMU_ATTACK = 0;

// 히든 던전 관련 변수
bool hidden_dungeon_unlocked = false;
bool amumu_boss_unlocked = false;
int monster_defeat_count = 0; // 물리친 몬스터 수

// 플레이어(철수)의 체력과 공격력
int player_health = 100;
int player_attack = 10;
```

```

// 플레이어와 몬스터의 전투 함수
void Battle(string monster_name, int monster_health, int monster_attack) {
    cout << monster_name << "과(와) 전투를 시작합니다!\n";

    while (monster_health > 0 && player_health > 0) {
        cout << "\n철수 체력: " << player_health << " / " << monster_name << " 체력: " << monster_health << "\n";
        cout << "1. 싸운다\n2. 도망친다\n";
        cout << "행동을 선택하세요 (1-2): ";
        int action;
        cin >> action;

        if (action == 1) {
            cout << "철수가 " << monster_name << "을(를) 공격합니다! " << monster_name << " 체력 -" << player_attack << "\n";
            monster_health -= player_attack;

            if (monster_health > 0) {
                cout << monster_name << "이(가) 철수를 공격합니다! 철수 체력 -" << monster_attack << "\n";
                player_health -= monster_attack;
            }
        } else if (action == 2) {
            player_health -= 5;
            cout << "도망치던 도중, " << monster_name << "이(가) 날린 공격에 스쳐 체력이 5 줄었습니다.\n";
            cout << "현재 체력: " << player_health << "\n";
            return;
        } else {
            cout << "잘못된 선택입니다. 다시 입력해주세요.\n";
        }
    }

    if (player_health <= 0) {
        cout << "철수가 쓰러졌습니다. 게임 종료.\n";
        exit(0);
    } else if (monster_health <= 0) {
        cout << monster_name << "을(를) 물리쳤습니다!\n";
        monster_defeat_count++; // 물리친 몬스터 수 증가
        cout << "현재 물리친 몬스터 수: " << monster_defeat_count << "\n";

        if (monster_defeat_count >= 4 && !hidden_dungeon_unlocked) {
            hidden_dungeon_unlocked = true;
            cout << "\n히든 던전이 열렸습니다. 대악마 아트록스의 방!\n";
        }

        if (monster_name == "대악마 아트록스") {
            cout << "축하합니다! 대악마 아트록스를 물리쳤습니다!\n";
            cout << "아트록스가 사용했던 '다르킨의 검'을 획득했습니다!\n";
            player_bag.push_back("다르킨의 검");
            amumu_boss_unlocked = true; // 아무무 보스 방 잠금 해제
            cout << "슬픈 미라 아무무의 방이 열렸습니다!\n";
        } else if (monster_name == "슬픈 미라 아무무") {
            cout << "축하합니다! 슬픈 미라 아무무를 물리쳤습니다!\n";
            cout << "철수는 이 던전을 탈출했습니다. 게임 클리어!\n";
            exit(0); // 게임 종료
        }
    }
}
}

```

```
// 히든 던전의 전투 함수
void EnterHiddenDungeon() {
    cout << "\n대악마 아트록스의 방에 들어갑니다...\n";
    DisplayAtroxArt();
    Battle("대악마 아트록스", ATROX_HEALTH, ATROX_ATTACK);
}

// 슬픈 미라 아무무의 방 진입 함수
void EnterAmumuBossRoom() {
    cout << "\n슬픈 미라 아무무의 방에 들어갑니다...\n";
    DisplayAmumuArt();
    Battle("슬픈 미라 아무무", AMUMU_HEALTH, AMUMU_ATTACK);
}
```

(6) 아이템을 넣는 가방 관련 함수

입출력 및 설명

- 입력:

함수 ViewBag이 호출하는 숫자를 입력함.

함수는 player_bag 벡터에 담긴 아이템들을 순차적으로 출력함.

- 출력:

ViewBag 함수가 호출되면, 플레이어의 가방에 아이템이 있는지 확인 후 출력됨.

```
// 가방 내용을 출력하는 함수
void ViewBag() {
    if (player_bag.empty()) {
        cout << "가방이 비어 있습니다.\n";
    } else {
        cout << "철수의 가방 내용:\n";
        for (int i = 0; i < player_bag.size(); i++) {
            cout << i + 1 << ". " << player_bag[i] << "\n";
        }
    }
}
```

```
// 철수의 가방  
vector<string> player_bag;
```

- 적용된 배운 내용

vector<string> - 문자열을 여러 개 저장하는 동적 배열을 활용하여 가방을 생성함.

반복문 및 배열을 활용하여 가방에 아이템을 넣고 관리함.

(7) 아이템 사용 관련 함수 (6,7주차 내용)

입출력

입력

- 사용자는 가방에 있는 아이템을 선택하기 위해 숫자를 입력해야 합니다.
- 입력값은 문자열로 받으며, stoi를 사용해 정수로 변환합니다.
- "다르킨의 검"을 선택하면 추가적으로 'y' 또는 'n'을 입력해야 합니다.
- 유효하지 않은 입력값에 대해서는 예외 처리로 잘못된 선택 메시지가 출력됩니다.

출력

- 아이템 사용 시 효과 메시지가 출력됩니다.
 - 예: "빨간포션을 사용했습니다. 체력이 20 증가했습니다. 현재 체력: 120"
- "다르킨의 검" 사용 시 추가적인 경고 메시지와 선택 결과가 출력됩니다.
 - 예: "다르킨의 검을 사용할 경우 체력이 크게 줄어들 수 있습니다. 그래도 사용하시겠습니까? (y/n)"
- 체력이 0 이하로 떨어지면 "게임 종료" 메시지가 출력됩니다.
- 잘못된 선택을 했을 때, "잘못된 선택입니다. 다시 시도해주세요." 메시지가 출력됩니다.

2. 설명

기능 요약

- 사용자가 선택한 아이템의 고유 효과를 적용합니다.
 - 아이템 효과는 체력 증가, 공격력 증가 등으로 나뉩니다.
 - "다르킨의 검"은 고위험 고보상 아이템으로, 공격력이 크게 증가하는 대신 체력이 감소합니다.
- 선택한 아이템은 사용 후 가방에서 제거됩니다.
- 입력값이 유효하지 않거나, 아이템을 사용할 수 없는 경우 에러 메시지를 출력하고 함수를 종료합니다.

주요 로직

입력 처리:

- 문자열로 입력받아 stoi를 통해 정수로 변환합니다.
- 변환 중 에러가 발생하면 예외 처리로 잘못된 입력 메시지를 출력합니다.

아이템 효과 적용:

- 아이템 이름에 따라 고유 효과를 적용합니다.
- 예: "빨간포션" 사용 시 체력 +20, "피묻은 도끼" 사용 시 공격력 +5.

"다르킨의 검" 특별 처리:

- 사용 전에 추가 확인 메시지 출력.
- 'y' 또는 'Y' 입력 시 공격력 증가(+20), 체력 감소(-50).
- 체력이 0 이하로 떨어지면 게임 종료.

아이템 제거:

- 효과 적용 후, 해당 아이템은 player_bag에서 제거됩니다.

예외 처리:

- `invalid_argument` 예외: 입력값이 숫자가 아닌 경우.
- `out_of_range` 예외: 입력값이 가방 아이템 개수 범위를 벗어나는 경우.

3. 적용된 배운 내용

문자열과 정수 변환

- `stoi`를 사용해 문자열 입력을 정수로 변환.
- 변환 중 발생할 수 있는 예외(`invalid_argument`, `out_of_range`) 처리.

조건문

- `if-else` 구조로 아이템 이름에 따라 효과를 적용.
- "다르킨의 검" 선택 시 추가 조건 확인(y/n 입력).

반복문

- `for`를 사용해 가방에 있는 아이템 리스트를 출력.

벡터 활용

- `vector<string>`으로 가방 구현.
- `erase`로 사용한 아이템 제거.

예외 처리

- `try-catch` 블록으로 입력값에 대한 예외 처리.
- 잘못된 입력값에 대해 명확한 에러 메시지를 출력.

사용자 선택 처리

- 숫자로 변환된 입력값이 유효한 범위인지 확인.
- 유효하지 않으면 에러 메시지를 출력하고 함수를 종료.

게임 종료 처리

- 체력이 0 이하로 떨어지면 `exit(0)`으로 게임 종료.

- 사용자가 "다르킨의 검" 사용 여부를 결정할 수 있도록 선택권 부여.

-코드 스크린샷

```
// 가방 내용을 출력하고 아이템을 사용할 수 있는 옵션을 제공하는 함수
void ViewBag() {
    if (player_bag.empty()) {
        cout << "가방이 비어 있습니다.\n";
    } else {
        cout << "철수의 가방 내용:\n";
        for (int i = 0; i < player_bag.size(); i++) {
            cout << i + 1 << ". " << player_bag[i] << "\n";
        }
        cout << "\n아이템을 사용하시겠습니까? (y/n): ";
        char use_choice;
        cin >> use_choice;
        if (use_choice == 'y' || use_choice == 'Y') {
            UseItem();
        }
    }
}
```

```

// 아이템을 사용하는 함수
void UseItem() {
    if (player_bag.empty()) {
        cout << "가방에 사용할 아이템이 없습니다.\n";
        return;
    }

    cout << "사용할 아이템을 선택하세요:\n";
    for (int i = 0; i < player_bag.size(); i++) {
        cout << i + 1 << ". " << player_bag[i] << "\n";
    }
    cout << "0. 취소\n";
    cout << "선택을 입력하세요: ";

    string item_choice_str;
    cin >> item_choice_str; // 입력을 문자열로 받음

    // 숫자로 변환
    try {
        int item_choice = stoi(item_choice_str); // 문자열을 정수로 변환
        if (item_choice == 0) {
            cout << "아이템 사용을 취소했습니다.\n";
            return;
        }

        if (item_choice < 1 || item_choice > player_bag.size()) {
            cout << "잘못된 선택입니다. 다시 시도해주세요.\n";
            return;
        }

        string selected_item = player_bag[item_choice - 1];

        if (selected_item == "빨간포션") {
            player_health += 20;
            cout << "빨간포션을 사용했습니다. 체력이 20 증가했습니다. 현재 체력: " << player_health << "\n";
        }
        else if (selected_item == "피물은 도끼") {
            player_attack += 5;
            cout << "피물은 도끼를 사용했습니다. 공격력이 5 증가했습니다. 현재 공격력: " << player_attack << "\n";
        }
        else if (selected_item == "날카로운 철검") {
            player_attack += 7;
            cout << "날카로운 철검을 사용했습니다. 공격력이 7 증가했습니다. 현재 공격력: " << player_attack << "\n";
        }
        else if (selected_item == "야만의 몽둥이") {
            player_attack += 6;
            cout << "야만이 몽둥이를 사용했습니다. 공격력이 6 증가했습니다. 현재 공격력: " << player_attack << "\n";
        }
        else if (selected_item == "천갑옷") {
            player_health += 25;
            cout << "천갑옷을 사용했습니다. 체력이 25 증가했습니다. 현재 체력: " << player_health << "\n";
        }
        else if (selected_item == "가죽갑옷") {
            player_health += 30;
            cout << "가죽갑옷을 사용했습니다. 체력이 30 증가했습니다. 현재 체력: " << player_health << "\n";
        }
        else if (selected_item == "다르킨의 검") {
            // 사용 여부 확인
            cout << "다르킨의 검을 사용할 경우 체력이 크게 줄어들수있습니다. 그래도 사용하시겠습니까? (y/n): ";
            char confirm;
            cin >> confirm;

            if (confirm == 'y' || confirm == 'Y') {
                player_attack += 20;
                player_health -= 50;
                cout << "다르킨의 검을 사용했습니다. 공격력이 20 증가하고 체력이 50 감소했습니다.\n";
                cout << "현재 공격력: " << player_attack << ", 현재 체력: " << player_health << "\n";

                if (player_health <= 0) {
                    cout << "다르킨의 검을 사용한 후 체력이 모두 소진되었습니다. 게임 종료.\n";
                    exit(0);
                }
            }
            else {
                cout << "다르킨의 검 사용을 취소했습니다.\n";
                return;
            }
        }
        else {
            cout << "이 아이템은 사용할 수 없습니다.\n";
            return;
        }

        // 사용한 아이템을 가방에서 제거
        player_bag.erase(player_bag.begin() + (item_choice - 1));
    } catch (invalid_argument& e) {
        cout << "잘못된 선택입니다. 다시 시도해주세요.\n";
        return;
    } catch (out_of_range& e) {
        cout << "잘못된 선택입니다. 다시 시도해주세요.\n";
        return;
    }
}

```

(8) 레벨업 관련 함수 (추가사항)

1. 입출력

입력:

- 경험치는 Battle 함수에서 전투 후 랜덤으로 획득되며, 사용자 입력이 직접적으로 필요하지 않습니다.

출력:

- 플레이어가 레벨업할 경우 축하 메시지 출력:
 - "축하합니다! 레벨 X로 올라셨습니다!"
 - 공격력 및 체력 증가 내용 출력:
 - "공격력이 X 증가하여 Y가 되었고, 체력이 Z 증가하여 W가 되었습니다."
- 현재 경험치와 다음 레벨까지의 경험치 상태를 출력:
 - "경험치 X을(를) 획득했습니다! 현재 경험치: Y / Z"

2. 설명

- **CheckLevelUp 함수:**
 - 플레이어의 현재 경험치(player_exp)가 다음 레벨까지 필요한 경험치(next_level_exp) 이상인지 확인.
 - 조건을 만족하면 경험치를 소모(player_exp -= next_level_exp)하고 레벨을 상승(player_level++).
 - 레벨업 시 다음 레벨까지 필요한 경험치 증가(next_level_exp += 50) 및 보상 제공:
 - 공격력(player_attack)과 체력(player_health) 증가.
 - 레벨업에 성공할 때마다 관련 메시지를 출력합니다.
- **Battle 함수와의 연계:**

- 전투 후 몬스터를 처치하면 무작위 경험치(rand() % 20)를 획득.
- 획득한 경험치를 플레이어 경험치(player_exp)에 추가 후 CheckLevelUp 함수 호출.
- 몬스터를 4마리 이상 처치하면 히든 던전 잠금 해제.

적용된 배운 내용

- **반복문:**
 - while 루프를 사용해 경험치가 다음 레벨 경험치 이상인 동안 레벨업을 반복적으로 처리.
- **조건문:**
 - 경험치와 다음 레벨 경험치를 비교(player_exp >= next_level_exp)하여 레벨업 여부를 판단.
- **함수 분리:**
 - 레벨업 로직을 CheckLevelUp 함수로 분리해 재사용성과 가독성을 높임.
 - Battle 함수와 CheckLevelUp 함수 간의 연계를 통해 경험치 획득 및 레벨업 처리를 구조적으로 구현.
- **랜덤성:**
 - rand()를 사용해 전투에서 획득할 경험치 값을 무작위로 생성.
- **상태 업데이트:**
 - 경험치 감소, 레벨 증가, 공격력/체력 증가 등 상태 변화를 변수에 반영.
- **게임 디자인:**
 - 경험치와 레벨업 보상 시스템을 도입해 게임 진행에서 성장감을 제공.
 - 플레이어의 성장이 히든 던전 잠금 해제 등 추가 이벤트와 연결되도록 설계.

-코드 스크린샷

```
// 플레이어 레벨링 관련 변수
int player_level = 1;
int player_exp = 0;
int next_level_exp = 100; // 다음 레벨까지 필요한 경험치
```

```
// 레벨업 체크 함수
void CheckLevelUp() {
    while (player_exp >= next_level_exp) {
        player_exp -= next_level_exp; // 초과된 경험치를 다음 레벨로 넘김
        player_level++;
        next_level_exp += 50; // 레벨업 시 다음 레벨까지 필요한 경험치 증가
        player_attack += 5; // 레벨업 보상: 공격력 증가
        player_health += 20; // 레벨업 보상: 체력 증가
        cout << "\n축하합니다! 레벨 " << player_level << "로 올라셨습니다!\n";
        cout << "공격력이 5 증가하여 " << player_attack << "이(가) 되었고, 체력이 20 증가하여 " << player_health << "이(가) 되었습니다.\n";
    }
}
```

```
// 경험치 획득
int exp_gained = rand() % 20; //랜덤 경험치
player_exp += exp_gained;
cout << "경험치 " << exp_gained << "을(를) 획득했습니다! 현재 경험치: " << player_exp << " / " << next_level_exp << "\n";

// 레벨업 확인
CheckLevelUp();
```

4. 테스트 결과

(1) 메인 입출력 함수

- 설명

- ASCII 아트와 소개 스토리 출력.
- 사용자 입력을 받아 선택을 처리하거나 종료.
- 유효하지 않은 입력값에 대한 오류 처리.
- 전체적인 흐름을 한눈에 보여줌
- 코드의 본체

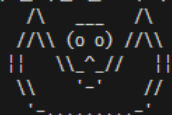
- 테스트 결과 스크린샷



앞을 보니 세 가지 길이 있습니다.

1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
0. 게임종료
9. 가방을 열다

현재 물리친 몬스터 수: 0



어떻게 할까요?

1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
4. 히든 던전: 대악마 아트록스의 방
0. 게임종료
9. 가방을 열다

선택을 입력하세요 (0-5): 2

현재 물리친 몬스터 수: 8

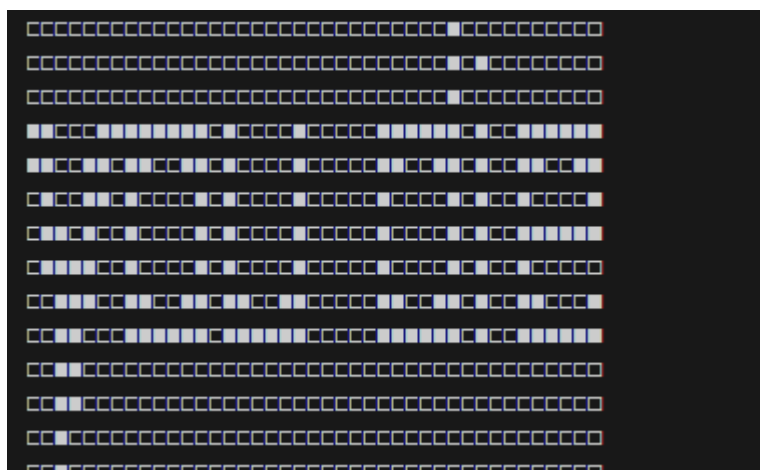
```

      /-----\
      / $$$$$$$$ \
      | $$$$$$$$$$$$ |
      | $$$$$$$$$$$$ |
      | $$$$$$$$$$$$ |
      \-----/ [T]
  
```

보물상자를 발견했습니다.



보물상자에서 날카로운 철검을(를) 얻었습니다!



(2) 플레이어의 선택에 따른 결과를 처리하는 함수

- 설명

- 플레이어의 선택에 따라 랜덤 이벤트(몬스터, 보물상자)를 처리.

- 테스트 결과 스크린샷

어떻게 할까요?

1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
0. 게임종료

선택을 입력하세요 (0-3): 1

```
  ^      ^
 //\    (o o) //\
 ||   \.^_//   ||
  \    ^-^    //
   ^-.....^
```

야생 멧돼지가 돌진해옵니다!
전투를 시작합니다

어떻게 할까요?

1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
0. 게임종료

선택을 입력하세요 (0-3): 2

```
  /-----\
 / $$$$$$$$$$ \
 | $$$$$$$$$$ |
 | $$$$$$$$$$ |
 \_____[T]____/
```

보물상자를 발견했습니다.
보물상자에서 천갑옷을(를) 얻었습니다!

어떻게 할까요?

1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
0. 게임종료

선택을 입력하세요 (0-3): 3

```
  .----.
 /  o o  \
 |    ^   |
 |  \___/  |
  \_____/
  //   \ \
  ||    ||
```

고블린 전사가 나타났습니다!
전투를 시작합니다

어떻게 할까요?

1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
0. 게임종료

선택을 입력하세요 (0-5): 3
현재 물리친 몬스터 수: 3

```

      -----
     /          \
    / $$$$$$$$  \
   | $$$$$$$$  |
   | $$$$$$$$  |
   \_____ [T] ____/

```

보물상자를 발견했습니다.



보물상자에서 야만의 몽둥이(를) 얻었습니다!

선택을 입력하세요 (0-5): 1
현재 물리친 몬스터 수: 4

```

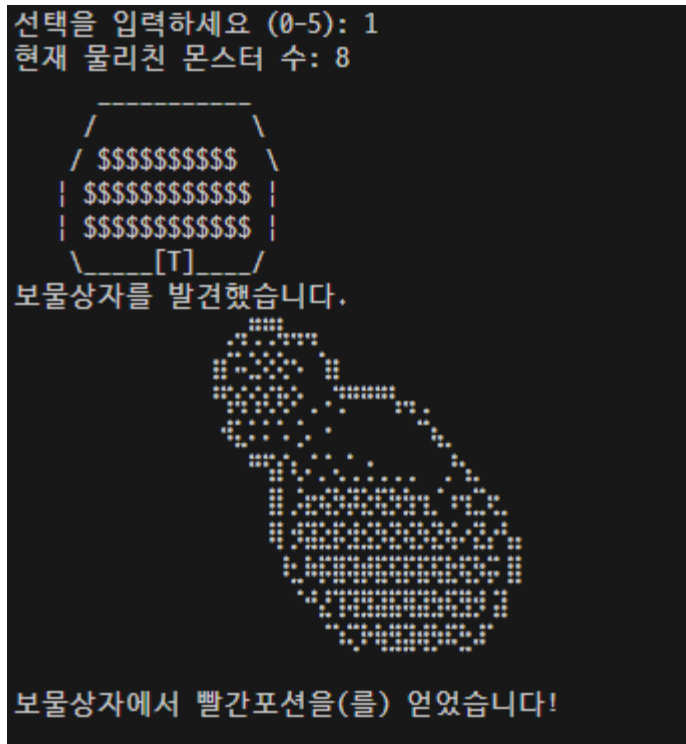
      -----
     /          \
    / $$$$$$$$  \
   | $$$$$$$$  |
   | $$$$$$$$  |
   \_____ [T] ____/

```

보물상자를 발견했습니다.



보물상자에서 피문은 도끼(를) 얻었습니다!



(4) ASCII 아트 출력 및 스토리 출력 함축 함수

- 설명

- 각 상황에 맞는 ASCII 아트 및 스토리 출력.

- 테스트 결과 스크린샷

Feynman diagram illustrating the decay of a Z boson into two photons, which then decay into fermion-antifermion pairs. The diagram shows a central Z boson line (dashed) splitting into two photon lines (wavy), which then decay into fermion-antifermion pairs (solid lines). The fermion pairs are labeled with 'zll...' and 'zz...'.

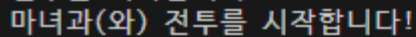
앞을 보니 세 가지 길이 있습니다.



어떻게 할까요?

© 2006 The Authors
Journal compilation © 2006 Blackwell Publishing Ltd

현재 물리친 몬스터 수: 0



마녀이(가) 철수를 공격합니다! 철수 체력 -10

현재 물리친 몬스터 수: 1

선택을 입력하세요 (0-5):

(5) 전투 시스템 함수 (4,5주차 내용)

- 설명

- 각 상황에 맞는 ASCII 아트 및 스토리 출력(결과 출력을 편하게 하기 위해 던전보스의 공격력은 임시로 0으로 설정하였음)

- 테스트 결과 스크린샷

```
보물상자에서 날카로운 철검을(를) 얻었습니다!

어떻게 할까요?
1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
0. 게임종료
9. 가방을 열다
선택을 입력하세요 (0-5): 1
현재 물리친 몬스터 수: 3
  /\      /\
 /  \  (o o) /  \
 ||   \  ^  //   ||
  \    '-'    //
   '-.....-'

야생 멧돼지가 돌진해옵니다!
전투를 시작합니다
야생 멧돼지와(와) 전투를 시작합니다!

철수 체력: 73 / 야생 멧돼지 체력: 20
1. 싸운다
2. 도망친다
행동을 선택하세요 (1-2): 1
철수가 야생 멧돼지를(를) 공격합니다! 야생 멧돼지 체력 -10
야생 멧돼지이(가) 철수를 공격합니다! 철수 체력 -5

철수 체력: 68 / 야생 멧돼지 체력: 10
1. 싸운다
2. 도망친다
행동을 선택하세요 (1-2): 1
철수가 야생 멧돼지를(를) 공격합니다! 야생 멧돼지 체력 -10
야생 멧돼지를(를) 물리쳤습니다!
현재 물리친 몬스터 수: 4

히든 던전이 열렸습니다: 대악마 아트록스의 방!

어떻게 할까요?
1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
4. 히든 던전: 대악마 아트록스의 방
0. 게임종료
9. 가방을 열다
선택을 입력하세요 (0-5): █
```

대악마 아트록스의 방에 들어갑니다...



대악마 아트록스가 당신을 노려봅니다. 그의 존재만으로도 공기가 무겁습니다!
대악마 아트록스와(와) 전투를 시작합니다!

철수 체력: 68 / 대악마 아트록스 체력: 100

1. 싸운다
2. 도망친다

행동을 선택하세요 (1-2): 1

철수가 대악마 아트록스를(를) 공격합니다! 대악마 아트록스 체력 -10

대악마 아트록스이(가) 철수를 공격합니다! 철수 체력 -0

철수 체력: 68 / 대악마 아트록스 체력: 10

1. 싸운다

2. 도망친다

행동을 선택하세요 (1-2): 1

철수가 대악마 아트록스를(를) 공격합니다! 대악마 아트록스 체력 -10

대악마 아트록스를(를) 물리쳤습니다!

현재 물리친 몬스터 수: 5

축하합니다! 대악마 아트록스를 물리쳤습니다!

아트록스가 사용했던 '다르킨의 검'을 획득했습니다!

슬픈 미라 아무무의 방이 열렸습니다!

어떻게 할까요?

1. 첫번째 길로 간다

2. 두번째 길로 간다

3. 세번째 길로 간다

4. 히든 던전: 대악마 아트록스의 방

5. 슬픈 미라 아무무의 방

0. 게임종료

9. 가방을 열다

선택을 입력하세요 (0-5): █

슬픈 미라 아무무의 방에 들어갑니다...



슬픈 미라 아무무가 당신을 바라보며 흐느낍니다...

슬픈 미라 아무무와(와) 전투를 시작합니다!

철수 체력: 68 / 슬픈 미라 아무무 체력: 150

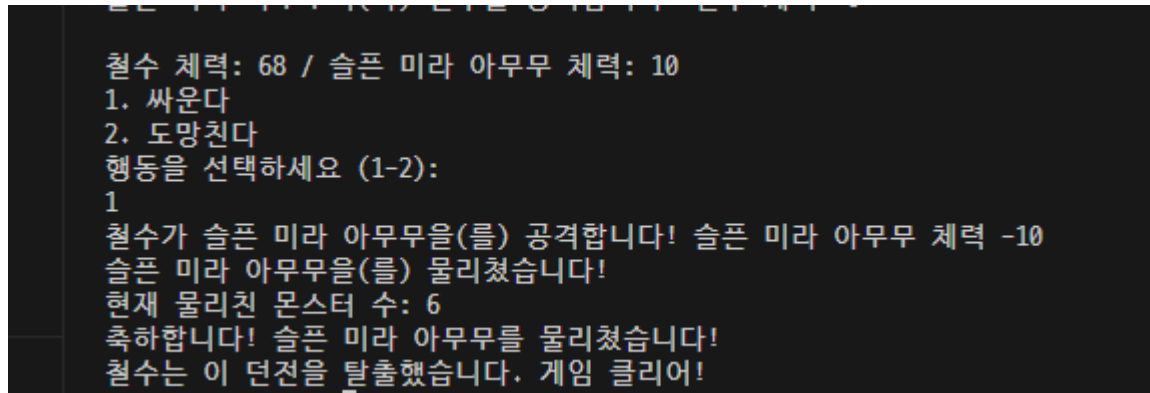
1. 싸운다

2. 도망친다

행동을 선택하세요 (1-2): 1

철수가 슬픈 미라 아무무를(를) 공격합니다! 슬픈 미라 아무무 체력 -10

슬픈 미라 아무무가(가) 철수를 공격합니다! 철수 체력 -0

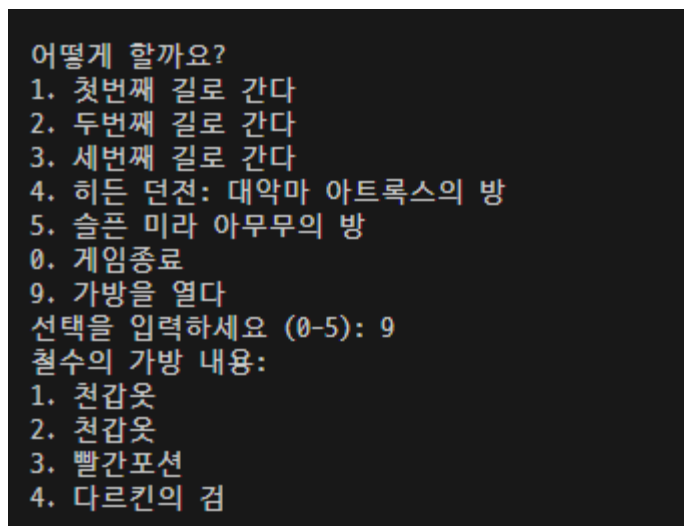


(6) 가방 함수 (4,5주차 내용)

- 설명

- 가방을 열면 얻은 아이템들의 리스트를 출력해줌.

- 테스트 결과 스크린샷



(7) 아이템 사용 관련 함수 (6,7주차 내용)

- 설명

- 가방을 열고 아이템을 사용할 수 있음

- 테스트 결과 스크린샷

```

선택을 입력하세요 (0-5, 9): 9
철수의 가방 내용:
1. 천갑옷
2. 피문은 도끼
3. 날카로운 철검
4. 천갑옷

아이템을 사용하시겠습니까? (y/n):

```

```

아이템을 사용하시겠습니까? (y/n): y
사용할 아이템을 선택하세요:
1. 천갑옷
2. 피문은 도끼
3. 날카로운 철검
4. 천갑옷
0. 취소
선택을 입력하세요:

```

```

선택을 입력하세요: 1
천갑옷을 사용했습니다. 체력이 25 증가했습니다. 현재 체력: 69

어떻게 할까요?
1. 첫번째 길로 간다
2. 두번째 길로 간다
3. 세번째 길로 간다
4. 히든 던전: 대악마 아트록스의 방
0. 게임종료
9. 가방을 열다
선택을 입력하세요 (0-5, 9):

```

(8) 레벨업 관련 함수 (추가 내용)

- 설명

- 몬스터 처치시 적당한 랜덤 경험치를 얻음. 100경험치를 얻으면 레벨업하고 그에 따라 공격력 및 체력이 증가함. 레벨업 한 후에는 경험치 통이 늘어나고 또 그걸 채우면 레벨업함.

- 테스트 결과 스크린샷

```

2. 도강전사
행동을 선택하세요 (1-2): 1
철수가 고블린 전사을(를) 공격합니다! 고블린 전사 체력 -10
고블린 전사을(를) 물리쳤습니다!
현재 물리친 몬스터 수: 5
경험치 9을(를) 획득했습니다! 현재 경험치: 64 / 100

```

```
현재 골라진 몬스터 수: 8
경험치 18을(를) 획득했습니다! 현재 경험치: 104 / 100

축하합니다! 레벨 2로 올라셨습니다!
공격력이 5 증가하여 15이(가) 되었고, 체력이 20 증가하여 87이(가) 되었습니다.
```

```
문제   출력   디버그 콘솔   터미널   포트   주석
경험치 9을(를) 획득했습니다! 현재 경험치: 151 / 150

축하합니다! 레벨 3로 올라셨습니다!
공격력이 5 증가하여 20이(가) 되었고, 체력이 20 증가하여 117이(가) 되었습니다.
```

5. 계획 대비 변경 사항

1) 기능2의 세부 기능1

- 이전 : 쉬움,보통,어려움으로 단순히 난이도를 유저가 선택하게끔하려고함
- 이후 : 길1,길2,길3로 나눠서 길마다 등장 몬스터나, 보물상자에서 나오는 아이템이 다르도록 조정함
- 사유 : 난이도를 알려줘서 플레이하면 재미가 없을 것 같아서이다.

2) 기능3의 개발 일정 조정

- 이전 : 기능2 이후 기능3을 개발하려고함
- 이후 : 기능2와 3을 같이 개발하려고함
- 사유 : 난이도별 몬스터의 체력과 공격력을 조절하는건 전투 시스템을 구축하면서 생각해봐야할 문제이기 때문이다. 던전에 있는 길마다의 전투 밸런스를 조정한다거나 하는 등 함께 고려해야할 사항이 많기 때문에 다음 진척보고서까지 함께 고려하여 한번에 개발을 진행하려고한다.

3) 기능4의 세부기능2 개발 일정 조정

- 이전 : 랜덤 아이템 드랍을 6주차부터 개발하려고했음
- 이후 : 미리 개발 완료를 하였음
- 사유 : 스토리 진행부분인 2,3주차 부분에서 아이템을 드랍하는 내용을 만드는 도중에 어차피 나중에 랜덤으로 드랍할건데 지금 구현하면 편할 것 같아서 미리 했다.

4) 레벨업 기능 추가

- 이전 : 플레이어가 경험치를 획득하더라도 레벨업 시스템을 구현하지 않을 계획이었음.
- 이후 : 경험치 시스템을 추가하고, 플레이어의 누적 경험치에 따라 레벨업하도록 변경함.
- 사유 : 플레이어가 전투를 통해 성장하고 보상을 받는 요소를 추가하면 더 몰입감 있는 게임 플레이를 제공할 수 있을 것이라 판단했기 때문임.

5) 몬스터 체력 및 게임 내 출력문이나 함수 리팩토링

- 이전 : 처음에 구현했던 함수나 출력문 그대로 있음
- 이후 : 체력, 출력문을 상황이나 적절한 난이도를 고려해 변경하고 함수를 더 효율적으로 리팩토링함.
- 사유 : 게임을 구현한 뒤 마지막 일주일간 검증을 거쳐서 게임의 난이도를 고려하여 몬스터의 체력을 조정하였고 상황에 맞는 출력문을 조정해주었으며 함수들을 효율적으로 리팩토링하였다. 특히 DropItem함수를 통해 코드 가독성을 높였다.

6. 느낀점

이번 프로젝트를 통해 텍스트 기반의 게임을 보다 생동감 있게 개선하며 다양한 프로그래밍 경험을 쌓을 수 있었습니다. ASCII 아트를 활용해 게임을 시각적으로 표현하는 작업은 기존의 단순 텍스트 기반 게임에 비해 훨씬 많은 노력을 요구했지만, 그만큼 완성된

게임이 더 몰입감 있고 재미있다는 것을 느꼈습니다. 특히 ASCII 아트를 통해 몬스터, 아이템, 보물상자 등을 표현하면서 게임에 생동감을 부여하는 과정이 매우 흥미로웠습니다. 텍스트 기반 환경에서도 플레이어에게 시각적 피드백을 제공할 수 있다는 점이 큰 강점이었습니다.

게임 설계 초기에는 단순히 난이도 선택만 제공하려고 했으나, 이를 더 흥미롭게 만들기 위해 길 별로 등장하는 몬스터와 아이템이 달라지는 시스템으로 변경하였습니다. 이 과정을 통해 게임 디자인에서 플레이어의 선택이 얼마나 중요한지 깨달았습니다. 난이도를 플레이어가 직관적으로 느끼게 하면서도 게임이 지나치게 어려워지거나 너무 쉬워지지 않도록 밸런스를 맞추는 작업이 생각보다 어렵고 많은 테스트가 필요하다는 점도 배웠습니다.

또한, 전투 시스템을 구현하면서 게임의 핵심 메커니즘인 플레이어와 몬스터의 상호작용을 설계하는 데 많은 노력을 기울였습니다. 전투 로직은 단순한 계산 이상의 요소가 필요했는데, 이를테면 플레이어가 사용하는 아이템과 던전의 난이도가 전투에 영향을 미치도록 설계하는 작업이 특히 까다로웠습니다. 처음 구현했던 코드와 이후 리팩토링을 통해 개선된 코드를 비교하며, 효율적이고 유지보수하기 쉬운 코드가 무엇인지 고민하게 되었습니다.

경험치와 레벨업 시스템을 추가하면서 게임의 성장 요소를 도입한 것도 매우 뜻깊은 작업이었습니다. 플레이어가 전투를 통해 성취감을 느끼고, 성장하는 과정을 게임에 녹여내면서 더 큰 몰입감을 줄 수 있었습니다. 레벨업 시스템을 구현하며 경험치 증가와 레벨업 조건을 함수로 분리하고, 이를 다양한 게임 상황에 연계하는 작업은 코드의 구조와 설계를 깊이 고민하게 했습니다.

이 프로젝트를 통해 다양한 프로그래밍 개념을 실질적으로 적용할 수 있었습니다. 조건문, 반복문, 예외 처리, 벡터 및 동적 배열 활용, 함수 분리 등 수업에서 배운 내용을 체계적으로 활용해 프로젝트를 완성할 수 있었던 점이 매우 뿌듯했습니다. 특히, 프로젝트 초반에는 단순히 작동하는 코드를 작성하는 데 초점을 맞췄지만, 후반으로 갈수록 가독성과 유지보수성을 고려한 코드를 작성하려고 노력했습니다. 이는 이후 더 큰 프로젝트를 진행하거나 다른 사람과 협업할 때 큰 도움이 될 것이라 생각합니다.

GitHub라는 원격 저장소를 활용한 점도 큰 장점이었다고 생각합니다. 프로젝트 파일을 원격 저장소에 업로드해 백업을 유지할 수 있었으며, 작업 중에 로컬 데이터가 손실되더

라도 안전하게 복원할 수 있다는 점이 마음에 들었습니다. 또한, GitHub의 커밋 로그와 변경 내역을 통해 프로젝트의 발전 과정을 시각적으로 확인할 수 있어 작업 진행 상황을 쉽게 파악할 수 있었습니다.

마지막으로, 이번 프로젝트는 프로그래밍 기술뿐만 아니라 문제 해결 능력, 창의적 사고, 시간 관리 등 여러 면에서 저를 성장시켜 주었습니다. 게임을 설계하고 개발하며 예상치 못한 문제를 해결하고, 더 나은 결과를 위해 계속 고민하며 개선해 나가는 과정이 매우 값진 경험이었습니다. 이러한 과정에서 얻은 경험과 교훈은 앞으로의 팀플에서 큰 도움이 될 것이라고 생각합니다.