2021-1 C++프로그래밍 실습과제 09

학번	2020136129	이름	최수연

(1) 각 문제에 대한 분석과 및 해결 방법

2. 9.7절의 Monster World 3 프로그램을 다음과 같이 확장하라.

(0) 교재 9.7절의 코드를 사용해도 되지만 가능하면 지난 과제까지 자신이 구현했던 몬스터 월드를 확장하는 것이 더 좋을 것이다.

(1) 대각선으로만 움직일 수 있는 스몸비(Smombi) 클래스를 추가하라.

(2) 강시의 동작이 너무 제한적이어서 아이템을 먹는데 다소 불리하다. 일정한 시간이 되면 움직이는 방향(가로세로)을 바꿀 수 있는 수퍼 강시, 샹시(Siangshi) 클래스를 추가하라. 단 이 클래스는 반드시 기존의 강시(Jiangshi) 클래스를 상속해서 구현해야 한다.

(3) 다른 몬스터 클래스를 상속해 자신만의 몬스터 클래스를 만들어라.

(4) main()함수에서 추가된 클래스의 객체들을 생성하여 몬스터 월드에 추가하고 테스트하라.

[문제분석 및 해결방법]

(1) 플랭크몬스터(Plank) 클래스를 생성하여 대각선으로만 움직일 수 있도록 하였다. move()함수에서 swith 함수를 사용해 0~3까지의 숫자를 랜덤으로 돌려서 각 숫자마다 대각선 4방향을 하나씩 지정하였다.

(2) 기존의 강시(Jiangshi)를 대체한 크런치몬스터(Crunch)를 부모 클래스로 하여, 기구를 사용하는 기구크런치몬스터(Erunch) 자식 클래스를 생성하였다. 부모 클래스와 움직임이 다르므로 move()함수를 재정의 해주었다. 정적 변수 num을 하나 추가하여 0으로 초기화하고 move가 한번 실행될 때마다 1씩 더해지도록 한다. move()함수 중간에는 switch 함수를 통해 본 정적 변수 num을 50으로 나누었을 때 나오는 숫자에 따라 bHori가 true 혹은 false로 나뉜다. 이렇게 나눔으로써 50회 이동할 때마다 상하가 죄우로, 또는 좌우가 상하로 바뀌도록 한다. 만약 bHori가 참일 때, x가 좌우로 이동하도록 하고, 거짓일 때, y가 상하로 이동하도록 한다.

(3) 문제 (1)에서 만든 플랭크몬스터(Plank) 클래스를 부모 클래스로 하여, 변형플랭크몬스터(Trank, Transformation Plank) 자식 클래스를 생성하였다. 본 클래스는 한 방향의 대각선으로 움직이다가 일정 기간이 지나면 다른 방향의 대각선으로 바뀌어 움직이는 플랭크의 변형몬스터이므로 move()함수를 재정의 해주었다. 문제(2)와 유사하게 정적 변수 num을 선언하고 move() 함수가 한번 실행될 때마다 1씩 더해지게 하였다. switch 함수를 통해 num을 50으로 나누었을 때, 50회마다 번갈아가면서 대각선 방향이 바뀌도록 설정하였다. 이를 통해 현재 num에 해당하는 대각선이 정해지면, 그 대각선상에서 위로 또는 아래로 움직이느냐에 관한 것은 앞의 switch함수 내에 switch 함수를 추가하여 랜덤으로 2개의 숫자를 돌려 선택하는 것으로 하였다.

(4) 문제 (1), (2), (3)에서 생성한 클래스의 객체들을 MonsterWorldGame.cpp 파일에 추가하였다. 각각 플랭크몬스터, 케이블크런치, 사이드플랭크라는 이름으로 객체를 추가하였다.

(2) 자신이 구현한 주요 코드

VariousMonsters.h

```
class Crunch : public Monster {
protected: bool bHori;
...};
```

상하 또는 좌우로만 움직일 수 있는 크런치몬스터(대륙강시)
기구크런치몬스터 Erunch를 위해 protected로 변경하여, 자식 클래스가 사용할 수 있도록 함.

```
class Plank : public Monster {
public:
    Plank(string n = "플랭크몬스터", string i = "♨", int x = 0, int y = 0)
        : Monster(n, i, x, y) {}
    ~Plank() { cout << "Plank"; }

    void move(int** map, int maxx, int maxy) {
        switch (rand() % 4) {
        case 0: x++; y--; break;
        case 1: x++; y++; break;
        case 2: x--; y++; break;
        case 3: x--; y--; break;
        }
        clip(maxx, maxy);
        eat(map);
    }
};
```

Plank는 대각선으로만 움직일 수 있는 몬스터(과제 1번, 스몸비) 클래스이다.
다음 클래스는 Monster 클래스의 자식 클래스이며, 코드는 다음과 같다.

```
class Erunch : public Crunch {
public:
    Erunch(string n = "기구크런치몬스터", string i = "↔", int x = 0, int y = 0, bool bH = true)
        : Crunch(n, i, x, y, bH) { }
    ~Erunch() { cout << "Equipment Crunch"; }
    void move(int** map, int maxx, int maxy) {
        static int num = 0;
        int dir = rand() % 2;
        int jump = rand() % 2 + 1;
        switch (num / 50) {
        case 0: case 2: case 4: case 6: case 8: bHori = true; break;
        case 1: case 3: case 5: case 7: case 9: bHori = false; break;
        }
        if (bHori) x += ((dir == 0) ? -jump : jump);
        else y += ((dir == 0) ? -jump : jump);
        clip(maxx, maxy);
        eat(map);
        num++;
    }
```

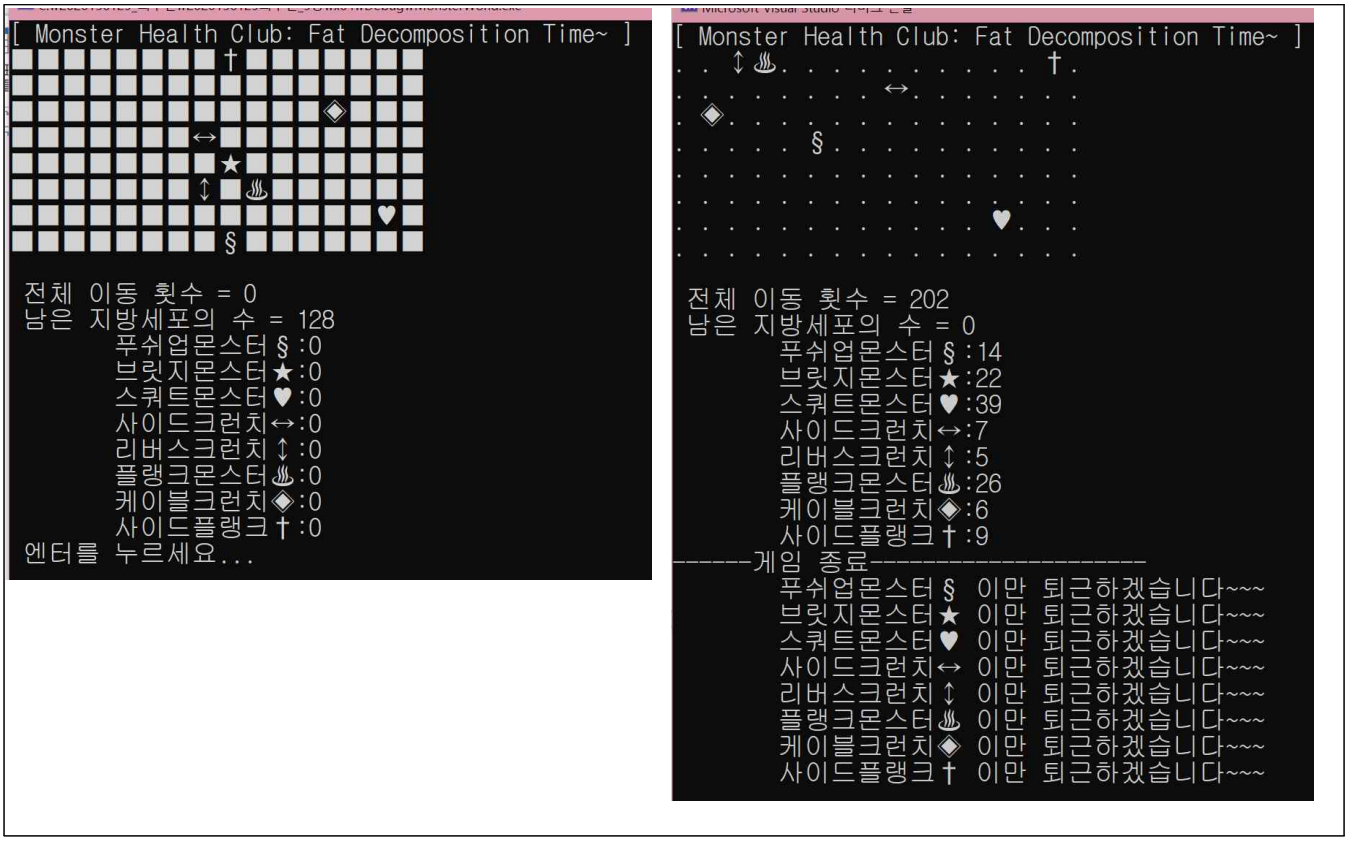
Erunch는 일정시간이 지나면 상하 또는 좌우가 바뀌는 몬스터(과제 2번, 샹시) 클래스로, Crunch의 자식클래스이다. Erunch 클래스 코드는 다음과 같다.

};
class Trank : public Plank {
public:
Trank(string n = "변형플랭크몬스터", string i = "†", int x = 0, int y = 0)
: Plank(n, i, x, y) { }
~Trank() { cout << "Transformation Plank"; }
 	
void move(int** map, int maxx, int maxy) {
static int num = 0;
switch (num / 50) {
case 0: case 2: case 4: case 6: case 8: {
switch (rand() % 2) {
case 0: x++; y++; break;
case 1: x--; y--; break;
}
} break;
case 1: case 3: case 5: case 7: case 9: {
switch (rand() % 2) {
case 0: x++; y--; break;
case 1: x--; y++; break;
}
} break;
}
clip(maxx, maxy);
eat(map);
num++;
}
};
	Trank는 일정시간이 지나면 왼쪽 또는 오른쪽 대각선으로 바뀌는 몬스터(과제 3번) 클래스로, Plank 클래스의 자식 클래스이다. 코드는 다음과 같다.

MonsterWorld.h	
((Pushup*)pMon[0])->move(world.Data(), xMax, yMax); ((Bridge*)pMon[1])->move(world.Data(), xMax, yMax); ((Squat*)pMon[2])->move(world.Data(), xMax, yMax); ((Crunch*)pMon[3])->move(world.Data(), xMax, yMax); ((Crunch*)pMon[4])->move(world.Data(), xMax, yMax); ((Plank*)pMon[5])->move(world.Data(), xMax, yMax); ((Erunch*)pMon[6])->move(world.Data(), xMax, yMax); ((Trank*)pMon[7])->move(world.Data(), xMax, yMax);	다음 코드는 MonsterWorld 클래스의 play()함수의 for문이다. 다음 코드를 통해 각 클래스명을 다음과 같이 변경하였음을 확인할 수 있다.
MonsterWorldGame.cpp	
game.add(new Pushup("푸쉬업몬스터", "§", rand() % w, rand() % h)); game.add(new Bridge("브릿지몬스터", "★", rand() % w, rand() % h)); game.add(new Squat("스쿼트몬스터", "♥", rand() % w, rand() % h)); game.add(new Crunch("사이드크런치", "↔", rand() % w, rand() % h, true)); game.add(new Crunch("리버스크런치", "↕", rand() % w, rand() % h,	Plank와 Erunch, Trank를 추가하였다. Erunch는 Crunch의 자식클래스, Trank는 Plank의 자식클래스이다.

```
false));
game.add(new Plank("플랭크몬스터", "♨", rand() % w, rand() % h));
game.add(new Erunch("케이블크런치", "◈", rand() % w, rand() % h,
true));
game.add(new Trank("사이드플랭크", "†", rand() % w, rand() % h));
```

(3) 다양한 입력에 대한 테스트 결과



(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

.

(5) 이번 과제에 대한 느낀점

이번 과제에서는 새로운 자식 클래스의 move를 만들어내는 것이 매우 흥미로웠다. 또한 새로운 상속 클래스의 이름을 만들어내는 과정에서 부모와 자식클래스를 분류하는 것이 재밌었다. 사실 어떤 주제를 잡고 이름을 바꿔야할지 고민을 많이 했는데, 요즘 취미인 운동을 넣어서 해보니까 상속 과정에서 운동을 분류해보면서 다양한 운동 자세가 있다는 것을 알 수 있었다. 코딩뿐만 아니라 운동 이론도 같이 학습할 수 있어 일석이조였다.

(6) 궁금한 점이나 건의사항

지난 과제에서 했던 에너지를 포함한 코드를 사용하여 이번 과제와 합치면 오류가 발생합니다. 지난 과제에서 교수님께서 에너지가 0이 된 경우 마지막 배열을 에너지 0인 배열에 옮기고, 배열을 하나씩 줄이라고 하셨

습니다. 그런데 그 코드를 사용하니까 pMon 배열이 각각 이미 지정된 상태여서 그런 건지, 에너지가 0이 된 몬스터가 생기는 시점부터, 마지막 배열에 지정된 클래스의 move가 에너지 0이 된 배열에 지정된 클래스의 move로 바뀌어 실행됩니다. 물론 본 과제에서는 에너지를 추가하라는 문제는 없었지만, 만약 에너지 모드를 추가한다고 하면 이 부분은 어떻게 해결해야하는지 궁금합니다.