2021-1 C++프로그래밍 실습과제 09

(1) 각 문제에 대한 분석과 및 해결 방법

9.1. 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] : Monster 클래스를 확장하여 Smombi 클래스를 만들고 move 함수를 오버라이드 하여 4방향 대각중 한 방향으로 이동하게 Smombi 클래스를 구현하였다. 기존의 Jiangshi 클래스를 상속하고 일정 move 횟수마다 방향 전환이 이루어지도록 int moveCount, int actionPeroid 멤버 변수를 추가하여 해당 내용을 구현하였다. KGhost의 순간 이동과, Siangshi의 방향 전환의 특성을 가진 BlinkSiangshi를 Siangshi를 상속하여 구현하였다.

(2) 자신이 구현한 주요 코드

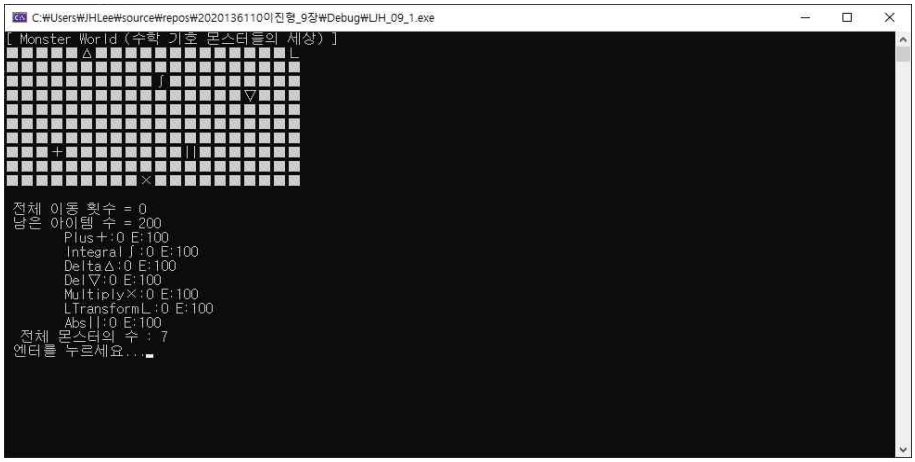
9.1. 7절의 Monster World 3 프로그램을 다음과 같이 확장하라.

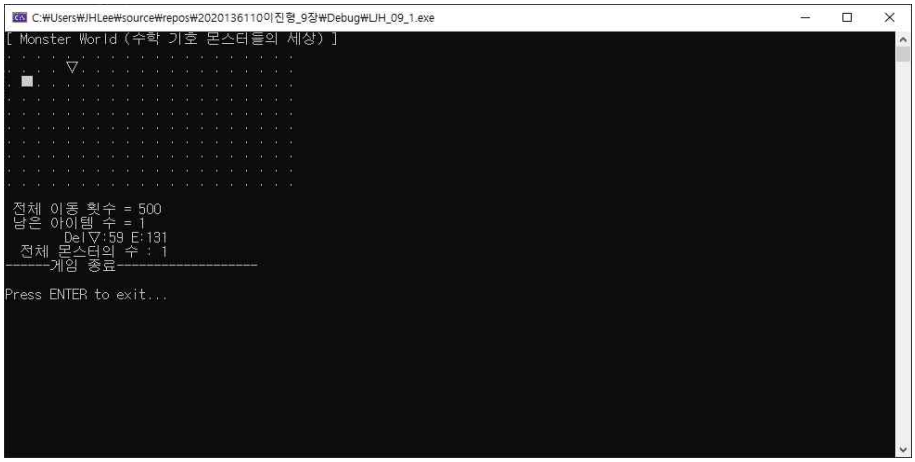
<VariousMonsters.h>

```
#pragma once
#include "Monster.h"
 ~ ... ~
class Smombi : public Monster {
public:
  Smombi(string n ="스몸비", string i ="▣", int x =0, int y =0)
    : Monster(n, i, x, y) {}
  virtual ~Smombi() { cout <<" Smombi"; }
  virtual void move(int ** map, int maxx, int maxy) {
    x += (rand() % 2) *2 -1;
    y += (rand() % 2) *2 -1;
    clip(maxx, maxy);
    eat(map);
  }
};
class Siangshi : public Jiangshi {
protected:
  int moveCount =0;
  int actionPeroid;
public:
  Siangshi(string n ="샹시", string i ="♠", int x =0, int y =0, bool bH =true, int aP
=50)
    : Jiangshi(n, i, x, y, bH), actionPeroid(aP) {}
  virtual ~Siangshi() { cout <<" Siangshi"; }
  virtual void moveCountAction(int ** map, int maxx, int maxy) {
    bHori =!bHori;
  }
  virtual void move(int ** map, int maxx, int maxy) {
    if (++moveCount % (actionPeroid +1) ==0) {
      moveCountAction(map, maxx, maxy);
    }
     Jiangshi::move(map, maxx, maxy);
  }
};
class BlinkSiangshi : public Siangshi {
public:
  BlinkSiangshi(string n ="점멸샹시", string i ="♤", int x =0, int y =0, bool bH =true,
int aP =50)
    : Siangshi(n, i, x, y, bH, aP) {}
  virtual ~BlinkSiangshi() { cout <<" BlinkSiangshi"; }
  virtual void moveCountAction(int ** map, int maxx, int maxy) {
    Siangshi::moveCountAction(map, maxx, maxy);
    x = rand() % maxx;
    x = rand() % maxy;
  }
};
```

(3) 다양한 입력에 대한 테스트 결과

9.1. 7절의 Monster World 3 프로그램을 다음과 같이 확장하라.





(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

9.1. 7절의 Monster World 3 프로그램을 다음과 같이 확장하라.

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

<VariousMonsters.h>
```
#pragma once
#include "Monster.h"
 ~ ... ~
class Smombi : public Monster {
public:
  Smombi(string n ="스몸비", string i ="▣", int x =0, int y =0)
    : Monster(n, i, x, y) {}
  virtual ~Smombi() { cout <<" Smombi"; }
  virtual void move(int ** map, int maxx, int maxy) {
    x += (rand() % 2) *2 -1;
    y += (rand() % 2) *2 -1;
    clip(maxx, maxy);
    eat(map);
  }
};
```

대각선으로만 움직일 수 있는 Smombi를 구현하기 위해서 Monster를 상속하고, move 함수를 오버라이드해서 Smombi의 대각 이동 로직을 구현하였다. 랜덤 대각선 이동을 위해서 (rand() % 2) * 2 – 1의 계산식으로 –1과 1이 x와 y에 더해질 수 있도록 하였다.

(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.

<VariousMonsters.h>
```
#pragma once
#include "Monster.h"
 ~ ... ~
class Siangshi : public Jiangshi {
protected:
  int moveCount =0;
  int actionPeroid;
public:
  Siangshi(string n ="샹시", string i ="♠", int x =0, int y =0, bool bH =true, int aP =50)
    : Jiangshi(n, i, x, y, bH), actionPeroid(aP) {}
  virtual ~Siangshi() { cout <<" Siangshi"; }
  virtual void moveCountAction(int ** map, int maxx, int maxy) {
    bHori =!bHori;
  }
  virtual void move(int ** map, int maxx, int maxy) {
    if (++moveCount % (actionPeroid +1) ==0) {
      moveCountAction(map, maxx, maxy);
    }
     Jiangshi::move(map, maxx, maxy);
  }
};
 ~ ... ~
```

Jiangshi를 상속하고 move를 오버라이드하여 actionPeroid마다 bHori를 반전시키는 Siangshi 클래스를 구현하였다. 이후의 BlinkSiangshi에서 moveCountAction 부분을 재활용하는 것을 염두에 두고 Siangshi 클래스를 설계할 때 함수를 분리하였다.

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

<VariousMonsters.h>
```
#pragma once
#include "Monster.h"
 ~ ... ~
class BlinkSiangshi : public Siangshi {
public:
  BlinkSiangshi(string n ="정멸샹시", string i ="♤", int x =0, int y =0, bool bH =true, int
aP =50)
    : Siangshi(n, i, x, y, bH, aP) {}
  virtual ~BlinkSiangshi() { cout <<" BlinkSiangshi"; }
  virtual void moveCountAction(int ** map, int maxx, int maxy) {
    Siangshi::moveCountAction(map, maxx, maxy);
    x = rand() % maxx;
    x = rand() % maxy;
  }
};
```

KGhost의 랜덤 이동 특성과 Siangshi의 특성을 결합한 BlinkSiangshi 클래스를 구현하였다. moveCountAction을 상속하여 Siangshi의 moveCountAction 로직을 그대로 이용할 수 있게끔 하였다.

(5) 이번 과제에 대한 느낀점

이번 과제에서는 간단한 구현을 위해서 단순한 상속 구조를 활용했지만, Strategy 디자인 패턴을 이용하여 최상위 행동 인터페이스 MoveAction와 순수 가상 함수 void runMoveAction을 두고 DefaultMoveAction BlinkMoveAction, DiagonalMoveAction, HorizontalMoveAction, VerticalMoveAction 등을 만들어 사용하면 BlinkSiangshi 같은 클래스를 만들 때 조금 더 재사용성이 뛰

어났을 것 같다는 생각이 들었다.