P1.S9

작성일시	@2024년 10월 19일 오전 7:03
강의 번호	C++ 언리얼
유형	강의
복습	☐
날짜	@2024년 10월 11일

TextRPG#4

전방선언

플레이어가 몬스터를 타겟으로 가지고 있다고 해보자

직접적으로 가지고 있다면 main에서 만든 플레이어의 크기는 얼마인가? → 몬스터가 정해지지 않았다면 알수가 없음 → 그럼 플레이어.h 에서 #include로 몬스터를 가져와야함 → 헤더파일 사이에 의존성이 생기면 위험

반면에 몬스터를 포인터로 가져오면 플레이어의 크기가 바로 정해질 수 있다

```
#pragma once

class Monster
{
public:

    void KillMe();

public:
    int _monsterId; // +0
    int _hp; // +4
    int _defence; // +8
    // ...
};
```

```
#include "Monster.h"

void Monster::KillMe()
{
    _hp = 0;
}
```

```
#pragma once

//#include "Monster.h"

// 위에서 전방 선언 해주던가 클래스 안에서 해주던가 둘중 하나
//class Monster;

// class는 설계도
class Player
{
public:

    void KillMonster();

    void KillMonster2();

public:
    int _hp;
    int _attack;

    //Monster _target;
    // 포인터로 들고 있으면 고정크기여서 당장은 전방선언으로 통과시킬수있디
    // 하지만 실질적으로 그녀석의 기능을 사용하고 싶으면 include 필요 ->
    class Monster* _target2;

    // Player _target3; 이 코드는 말이 되는 코드일까?
    // 플레이어 안에 플레이어 무한 반복이 될 것임
    Player* _target3;
};
```

```
#include "Player.h"
#include "Monster.h"

void Player::KillMonster()
{
    // 다른애의 함수를 호출해줄때도 include 필요
    _target2->KillMe();

    // 다른애의 구성성분 사용할때 include 필요
    // _target2->_hp = 0;
    // [ 주소 ] -> [ [monsterId][hp][defence] ]
    //(*_target2)._hp = 0;
}

void Player::KillMonster2()
{
    _target2->_hp = 0;
}
```

```
#include <iostream>
using namespace std;
#include "Player.h"


// 오늘의 주제 : 전방선언


int main()
{
    // 핵심 질문 : Player는 몇 바이트?
    // int 2개 = 2 * 4 = 8 바이트 + sizeof(Player*) = 12바이트

    // Player [ hp ] [ attack ] [ 주소 ]
    //
    // sizeof(Player*) = 4 or 8

    // sizeof(Monster*) = 4 or 8
```

```
    Player p1; // 지역변수 (Stack)

    Player* p2 = new Player(); // 동적할당 (Heap)

    p1._target3 = p2;

    return 0;
}
```