정리노트 #4

13조(소재현, 전대원)

프렌드 : 친구로 지정되면 나의 클래스가 아님에도, 외부 함수임에도 나의 클래스의 모든 멤버를 접근할 수 있는 권한이 부여된다. friend 키워드를 사용해 함수에 지정한다.

프렌드 함수가 되는 3 가지 : 클래스 외부에 선언된 전역 함수, 다른 클래스의 특정 멤버 함수, 모든 멤버 함수

friend bool '외부함수'(), friend bool 클래스::멤버(), friend 클래스

연산자 중복: +를 숫자와 물체에 적용, 중복 사용, +기호를 숫자가 아닌 곳에도 사용
정수 더하기, 문자열 합치기, 색 섞기, 배열 합치기
c++에 본래 있는 연산자만 중복 가능, 피 연산자 타입이 다른 새로운 연산 정의
연산자는 함수 형태로 구현 - 연산자 함수
클래스와 관계를 가짐
피연산자의 개수를 바꿀 수 없고 우선 순위 변경 안됨, 모든 연산자가 중복되는 건 아님.

연산자 함수 구현 방법 2가지 : 클래스의 멤버 함수, 외부 함수->프렌드 함수
리턴 타입 operator 연산자(매개변수리스트);

외부 함수로 구현하는 방법

```
Color operator + (Color op1, Color op2);
 bool operator ==(Color op1, Color op2);
class Color{
friend Color operator + (Color op1, Color op2);
friend Color operator == (Color op1, Color op2);
}

class power{
int kick; int punch;
public:
power operator+(power op2);};
power power::operator+(power op2){
power tmp;
tmp.kick = this-> kick +op2.kick;
tmp.punch = this->punch+op2.punch;
return tmp;}
```

예제 7-4

```
#include <iostream>
using namespace std;

class power {
        int kick;
        int punch;
public:
        power(int kick = 0, int punch = 0) {
                this->kick = kick; this->punch = punch;
        }
        void show();
        power operator + (power op2);
};

void power::show() {
        cout << "kick=" << kick << ',' << "punch=" << punch << endl;
}
power power::operator+ (power op2) {
        power tmp;
        tmp.kick = this->kick + op2.kick;
        tmp.punch = this->punch + op2.punch;
        return tmp;
}

int main() {
        power a(3, 5), b(4, 6), c;
        c = a + b;
        a.show();
        b.show();
        c.show();
}
```

a+b -> 컴파일러에 의한 변환 a.+(b)

올바른 것

```
power& return_this(){
return *this}                주소의 내용물
power return_this(){
return *this}                복사한 것을 넘김 복사 생성자
power& return_this(){
return this}
power* return_this(){
return this}                 내용물을 가르킴
```

power return_this(){
return this}
답은 1,2,4

단항 연산자 : 피연산자가 하나 뿐인 연산자 ex) a++(후위 연산자) ++a(전위 연산자)
b=a++; a=1 b=0, b=++a; a=1 b=1;
전위 연산자는 매개 변수 없음 ++a -> 컴파일러 -> a.++() power& return *this;
후위 연산자는 매개 변수 존재 a++ -> 컴파일러 -> a.++(변수) power power::operator++(int x)
power tme = *this // 증가 이전 객체 저장
return tmp; // 증가 이전 리턴

2+a의 경우 b=2+a -> b=+(2,a);로 변환됨
power operator+ (int op1, power op2)
이 경우 프렌드 선언을 해줘야 private 속성인 킥과 펀치에 접근할 수 있음
-> a.+(2)가 아니라 외부 함수인 +(2,a)이기 때문 -> 매개변수가 2개일 경우 매개변수가 많다고 오류가 뜸

단항 연산자 ++ 프렌드 작성
++a -> 컴파일러 -> ++(a) power& operator++ (power& op){ return op;}
a++ -> 컴파일러 -> ++(a,0) power operator++ (power op, int x){
power tmp = op; return tmp;}

<<연산자
복사 생성자를 쓰거나(power tmp; return tmp;), 참조 리턴을 하거나.(power& power::operator <<(int x) return *this;)
a(1,2) a<<3<<5<<6; 결과 15,16