Operator Overloading

1. 전역 함수로 정의하는 방법
2. 클래스의 멤버 함수로 정의하는 방법

전역 함수 정의

friend [리턴 타입] operator 오버로딩할연산자 (매개변수 목록)

friend 전역 함수

friend [리턴타입] 함수 이름 ( 매개변수 목록 )

friend를 사용하는 이유

⇒ friend 를 사용하게 되면 class 멤버 변수들(private 멤버 변수들)을 사용할 수 있게 된다. friend로 정의한 함수는 클래스의 멤버 함수(메소드) 가 아닌 전역 함수가 된다.

멤버 함수 정의

[리턴타입] 클래스이름::operator 오버로딩할연산자 (매개변수 목록)

<example>

1. 전역 함수를 이용한 Operator Overloading [ friend 사용 ]

#include <iostream>

#include <string>

**using** **namespace** std;

**class** Car {

**private**:

**int** ton ;

**int** kg ;

**int** speed ;

string name;

**public**:

Car(**int** t, **int** kg, **int** s, string n){ // constructor

ton = t;

**this**->kg = kg;

speed = s ;

name = n;

}

**friend** **int** **operator**+ (**const** Car& car1, **const** Car& car2){ // 전역 함수

**int** w = car1.ton \* 1000 + car1.kg + car2.ton \* 1000 + car2.kg;

**return** w ;

}

**void** speedUp(){

speed += 10;

}

**void** speedDown(){

**if**(speed >= 0 ){

speed -= 10;

}

}

**int** getWeight(){

**int** weight = ton \* 1000 + kg;

**return** weight;

}

**int** getSpeed(){

**return** speed;

}

};

**int** main(**void**){

Car \*car1 = **new** Car(1,400,60, "BMW");

Car \*car2 = **new** Car(1,600,40, "Benz");

**int** weight = \*car1+\*car2; // 1400 + 1600

cout << weight << endl;

**return** 0;

}

1. 멤버 함수를 이용한 Operator Overloading

#include <iostream>

#include <string>

**using** **namespace** std;

**class** Car {

**private**:

**int** ton ;

**int** kg ;

**int** speed ;

string name;

**public**:

Car(**int** t, **int** kg, **int** s, string n){ // constructor

ton = t;

**this**->kg = kg;

speed = s ;

name = n;

}

int operator+ (const Car& car);

**void** speedUp(){

speed += 10;

}

**void** speedDown(){

**if**(speed >= 0 ){

speed -= 10;

}

}

**int** getWeight(){

**int** weight = ton \* 1000 + kg;

**return** weight;

}

**int** getSpeed(){

**return** speed;

}

};

int Car::operator + (const Car &car){ // 멤버 함수

int w = getWeight() + car.ton \* 1000 + car.kg ;

return w;

}

**int** main(**void**){

Car \*car1 = **new** Car(1,400,60, "BMW");

Car \*car2 = **new** Car(1,600,40, "Benz");

**int** weight = \*car1+\*car2; // 1400 + 1600

cout << weight << endl;

**return** 0;

}

< 주의 사항 >

1. 전혀 새로운 연산자를 정의할 수 없습니다.

ex) 제곱을 나타내기 위한 \*\*라는 연산자를 새롭게 정의할 수 없습니다.

2. 기본 타입을 다루는 연산자의 의미는 재정의할 수 없으며, 따라서 오버로딩된 연산자의 피연산자 중 하나는 반드시 사용자 정의 타입이어야 합니다.

ex) 두 개의 double 형에 대한 덧셈 연산자(+)가 뺄셈을 수행하도록 오버로딩할 수 없습니다.

3. 오버로딩된 연산자는 기본 타입을 다루는 경우에 적용되는 피연산자의 수, 우선순위 및 그룹화를 준수해야 합니다.

ex) 나눗셈 연산자(/)는 이항 연산자이므로 단항 연산자로 오버로딩할 수 없습니다.

4. 오버로딩된 연산자는 디폴트 인수를 사용할 수 없습니다.