**Chapter 19 Notes**

**Alex Yang**

**(5/17/2021)**

1. operator= overloading

*#include <iostream>*

*using namespace std;*

*class A{*

*public:*

*A(int a, int b){*

*this->a = a;*

*this->b = b;*

*}*

*int a, b;*

*A& operator=(const A& x){*

*this->a = x.a;*

*this->b = x.b;*

*return \*this;*

*}*

*};*

*int main(void){*

*A aa(123, 456);*

*A bb(789, 1011);*

*A cc(1112, 1314);*

*cout<<"bb.a = " <<bb.a<<"\n";*

*cout<<"bb.b = " <<bb.b<<"\n";*

*cout<<"cc.a = " <<cc.a<<"\n";*

*cout<<"cc.b = " <<cc.b<<"\n";*

*bb = aa; //bb.operator=(aa)*

*cout<<"bb.a = " <<bb.a<<"\n";*

*cout<<"bb.b = " <<bb.b<<"\n";*

*cc = (bb = aa); //(bb=aa)'s return value is bb; it is equivalent to cc=bb;*

*cout<<"cc.a = " <<cc.a<<"\n";*

*cout<<"cc.b = " <<cc.b<<"\n";*

*return 0;*

*}*

1. Why does the following example seem like inheriting operator= from base ?

*#include <iostream>*

*class Base {*

*public:*

*Base(){*

*std::cout << "Constructor base" << std::endl;*

*}*

*~Base(){*

*std::cout << "Destructor base" << std::endl;*

*}*

*Base& operator=(const Base& a){*

*std::cout << "Assignment base" << std::endl;*

*}*

*};*

*class Derived : public Base{*

*public:*

*};*

*int main (void) {*

*Derived p;*

*Derived p2;*

*p2 = p; //Actually, p2 an p can’t be objects of the base*

*return 0;*

*}*

1. Pointer of base class and derived class

*#include <iostream>*

*using namespace std;*

*class A{*

*public:*

*int a, b;*

*};*

*class B: public A{*

*public:a*

*int x, y;*

*};*

*int main(void){*

*A a, \*ptr\_a;*

*a.a = 123; a.b = 456;*

*ptr\_a = &a;*

*cout<<ptr\_a->a<<"\n";*

*B b, \*ptr\_b;*

*b.x=789; b.y=1011;*

*ptr\_a = &b;*

*cout<<ptr\_a->a<<"\n";*

*A \*ptr\_any;*

*ptr\_any = static\_cast<A\*>(ptr\_b);*

*return 0;*

*}*

1. “uses a” relationship in C++

*#include <iostream>*

*using std::cout;*

*class Other{*

*public:*

*void func(){cout << "Here is class Other!" << "\n"; }*

*};*

*class Another{*

*public:*

*void test(Other &o) { o.func();};*

*};*

*int main(void){*

*Other o;*

*Another a;*

*a.test(o);*

*return 0;*

*}*

1. “knows a” relationship in C++ - two class instantiate each other

*#include <iostream>*

*class B;*

*class A{*

*public:*

*B \*b;*

//B b; //Error

*int ia;*

*};*

*class B{*

*public:*

*A a;*

*int ib;*

*};*

*int main(void){*

*A a;*

*B b;*

*return 0;*

*}*

//----------------------------------------------------

*class Eagle{*

*class Goose \*food;*

*};*

*class Goose{*

*class Eagle \*predator;*

*};*