Rule of three

#include <iostream>

using namespace std;

class SimpleExample{

public:

int \*data; //pointer

//constructor

SimpleExample(int value=0){

//memory allocation using new keyword

data=new int;

//assign value

\*data=value;

}

~SimpleExample(){

//deallocate memory

delete data;

}

//copy constructor

SimpleExample(const SimpleExample& other){

//allocate memory

data=new int;

\*data=\*(other.data);

}

//assignment overloading

SimpleExample& operator=(const SimpleExample& other){

if(this!=&other){

delete data;//free the memory space to avoid memory leaks

//allocate memory

data=new int;

\*data=\*(other.data);

}

return \*this;

}

//print data

void print(){

cout<<\*data<<endl;

}

};

int main() {

//create object

SimpleExample obj1(10), obj2;

obj2=obj1;

cout<<"Before updating data!!"<<endl;

obj1.print(); //10

obj2.print(); //10

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*(obj2.data)=11;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cout<<"After updating obj2 data!!"<<endl;

obj1.print(); //10

obj2.print(); //11

cout << "Finishhh!!";

return 0;

}