//Please like and comment

**//rectangleType.cpp**

#include "rectangleType.h"

rectangleType::rectangleType():length(0),width(0)  
{  
}

rectangleType::rectangleType(double l, double w):length(l),width(w)  
{  
}

rectangleType::~rectangleType()  
{  
}

double rectangleType::getLength() const  
{  
   return length;  
}

double rectangleType::getWidth() const  
{  
   return width;  
}

void rectangleType::setDimension(double l, double w)  
{  
   length = l;  
   width = w;  
}

rectangleType rectangleType::operator+(const rectangleType & obj)  
{  
   rectangleType temp(length + obj.length, width + obj.width);  
   return temp;  
}

rectangleType rectangleType::operator-(const rectangleType & obj)  
{  
   rectangleType temp(length - obj.length, width - obj.width);  
   return temp;  
}

rectangleType rectangleType::operator\*(const rectangleType & obj)  
{  
   rectangleType temp(length \* obj.length, width \* obj.width);  
   return temp;  
}

rectangleType rectangleType::operator++()  
{    
   ++length;  
   ++width;  
   return rectangleType(length,width);  
}

rectangleType rectangleType::operator++(int)  
{  
   rectangleType temp(length++, width++);  
   return temp;  
}

rectangleType rectangleType::operator--()  
{  
   --length;  
   --width;  
   return rectangleType(length, width);  
}

rectangleType rectangleType::operator--(int)  
{  
   rectangleType temp(length--, width--);  
   return temp;  
}

bool rectangleType::operator==(const rectangleType & a)  
{  
   return area() == a.area();  
}

bool rectangleType::operator!=(const rectangleType & a)  
{  
   return area() != a.area();  
}

bool rectangleType::operator<=(const rectangleType & a)  
{  
   return area() <= a.area();  
}

bool rectangleType::operator<(const rectangleType & a)  
{  
   return area() < a.area();  
}

bool rectangleType::operator>=(const rectangleType & a)  
{  
   return area() >= a.area();  
}

bool rectangleType::operator>(const rectangleType & a)  
{  
   return area() > a.area();  
}

double rectangleType::area() const  
{  
   return length\*width;  
}  
ostream & operator << (ostream &out, const rectangleType &obj)  
{  
   return out;  
}  
istream & operator >> (istream &in, rectangleType &obj)  
{  
   return in;  
}

**//rectangleType.h**

#pragma once  
#include<iostream>  
using namespace std;  
class rectangleType  
{  
public:  
   rectangleType();  
   rectangleType(double l, double w);  
   ~rectangleType();  
   double getLength() const;  
   double getWidth() const;  
   void setDimension(double l, double w);  
   rectangleType operator+(const rectangleType& obj);  
   rectangleType operator-(const rectangleType& obj);  
   rectangleType operator\*(const rectangleType& obj);  
   rectangleType operator ++();  
   rectangleType operator ++(int);

   rectangleType operator --();  
   rectangleType operator --(int);  
   bool operator==(const rectangleType& a);  
   bool operator!=(const rectangleType& a);  
   bool operator<=(const rectangleType& a);  
   bool operator<(const rectangleType& a);  
   bool operator>=(const rectangleType& a);  
   bool operator>(const rectangleType& a);  
   double area() const;  
   friend ostream & operator << (ostream &out, const rectangleType &obj);  
   friend istream & operator >> (istream &in, rectangleType &obj);

private:  
   double length;  
   double width;

};

**//boxType.cpp**

#include "boxType.h"

boxType::boxType():height(0)  
{  
}

boxType::boxType(double l, double w, double h):height(h),rectangleType(l,w)  
{  
}

boxType::~boxType()  
{  
}

double boxType::getHeight() const  
{  
   return height;  
}

void boxType::setDimension(double l, double w, double h)  
{  
   height = h;  
   rectangleType::setDimension(l, w);    
}

double boxType::volume() const  
{  
   return area()\*height;  
}

boxType boxType::operator+(const boxType & obj)  
{

   boxType temp(getLength() + obj.getLength(), getWidth() + obj.getWidth(), obj.height + getHeight());  
   return temp;  
}

boxType boxType::operator-(const boxType & obj)  
{  
   boxType temp(getLength() - obj.getLength(), getWidth() - obj.getWidth(), getHeight() - obj.height);  
   return temp;  
}

boxType boxType::operator\*(const boxType & obj)  
{  
   boxType temp(getLength() \* obj.getLength(), getWidth() \* obj.getWidth(), obj.height \* getHeight());  
   return temp;  
}

boxType boxType::operator++()  
{  
   rectangleType::operator++();  
   ++height;  
   return boxType(getLength(), getWidth(),height);  
}

boxType boxType::operator++(int)  
{  
   boxType temp;

   rectangleType::operator++(0);    
   temp.height = height++;  
   return temp;  
}

boxType boxType::operator--()  
{  
   rectangleType::operator--();  
   --height;  
   return boxType(getLength(), getWidth(), height);  
}

boxType boxType::operator--(int)  
{  
   boxType temp;  
   rectangleType::operator--(0);  
   temp.height = height--;  
   return temp;  
}

bool boxType::operator==(const boxType & a)  
{  
   return volume() == a.volume();  
}

bool boxType::operator!=(const boxType & a)  
{  
   return volume() != a.volume();  
}

bool boxType::operator<=(const boxType & a)  
{  
   return volume() <= a.volume();  
}

bool boxType::operator<(const boxType & a)  
{  
   return volume() < a.volume();  
}

bool boxType::operator>=(const boxType & a)  
{  
   return volume() >= a.volume();  
}

bool boxType::operator>(const boxType & a)  
{  
   return volume() > a.volume();  
}  
ostream & operator << (ostream &out, const boxType &obj)  
{  
   cout << "Length: " << obj.getLength() << " Width: " << obj.getWidth() << " Height: " << obj.getHeight();  
   return out;  
}  
istream & operator >> (istream &in, boxType &obj)  
{  
   return in;  
}

**//boxType.h**

#pragma once  
#include "rectangleType.h"  
#include<iostream>  
using namespace std;  
class boxType :  
   public rectangleType  
{  
public:  
   boxType();  
   boxType(double l, double w, double h);  
   ~boxType();  
   double getHeight() const;  
   void setDimension(double l, double w, double h);  
   double volume() const;  
   boxType operator+(const boxType& obj);  
   boxType operator-(const boxType& obj);  
   boxType operator\*(const boxType& obj);  
   boxType operator++();  
   boxType operator++(int);  
   boxType operator--();  
   boxType operator--(int);  
   bool operator==(const boxType& a);  
   bool operator!=(const boxType& a);  
   bool operator<=(const boxType& a);  
   bool operator<(const boxType& a);  
   bool operator>=(const boxType& a);  
   bool operator>(const boxType& a);  
   friend ostream & operator << (ostream &out, const boxType &obj);  
   friend istream & operator >> (istream &in, boxType &obj);  
private:  
   double height;  
};

**//Main.cpp**

#include<iostream>  
using namespace std;  
#include "boxType.h"  
int main()  
{  
   boxType box1(12, 9, 6);  
   boxType box2(8, 7, 5);  
   boxType box3, box4, box5;  
   cout << "box1: " << box1 << endl;  
   cout << "box2: " << box2 << endl;  
   box3 = box1 + box2;  
   cout << "box3: " << box3 << endl;  
   box4 = box1 - box2;  
   cout << "box4: " << box4 << endl;  
   box5 = box1 \* box2;  
   cout << "box5: " << box5 << endl;  
   if (box1 > box2)  
       cout << "Volume of box1 is greater than the volume of box2." << endl;  
   else  
       cout << "Volume of box1 is less than or equal to the volume of box2." << endl;  
   box1++;  
   cout << "After increment the length, width, and height of "  
       << "box1 by one unit, \nbox1: "

       << box1 << endl;  
   box3 = ++box2;  
   cout << "New dimension of box2: " << box2 << endl;  
   cout << "New dimension of box3: " << box3 << endl;  
   return 0;

}