Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №5**

на тему:

**“ Перевантаження операторів”**

**Виконала:**

ст. гр. КІ-15

Аркавенко Л.С.

**Перевірив:**

асист. каф. ЕОМ

Козак Н. Б.

Львів – 2020

**Мета роботи:** познайомитися із перевантаженням операторів.

**Теоретична частина:**

Кожному оператору мова С++ ставить у відповідність ім'я функції, що складається з ключового слова operator, власне оператору та аргументів відповідних типів:

**тип operator символОператору (списокПараметрів){**

**//тіло методу}**

Щоб використовувати операцію над об'єктами класів, ця операція повинна бути перевантажена, але є два виключення.

Операції присвоювання (=)і взяття адреси (&)створюються в класі автоматично за замовчуванням, тому їх можна використовувати без явного перевантаження. За замовчуванням операція присвоювання зводиться до побітового копіювання даних-елементів класу. Проте таке побітове копіювання небезпечне для класів з елементами, що вказують на динамічно виділені області пам'яті, масиви, рядки, оскільки в цьому випадку відбувається копіювання не даних (глибоке копіювання), а лише вказівників на дані (поверхневе копіювання).

Для таких класів слід явно перевантажувати операцію присвоюванняі здійснювати у ній глибоке копіювання. Операція адресації також може бути використана з об'єктами будь-яких класів без перевантаження.Вона просто повертає адресу об'єкта в пам'яті. Але операцію адресації можна також і перевантажувати.Перевантаження операцій підпорядковується наступним правилам:

• При перевантаженні зберігаються кількість аргументів, пріоритети операцій та правила асоціації, що використовуються у стандартних типах даних;

• Для стандартних типів даних операції не підлягають перевизначенню;

• Перевантажена функція-оператор не може мати параметрів по замовчуванню, не успадковується та неможе бути визначеною як static;

• Функція-оператор може бути визначена трьома способами –методкласу, дружня функція або звичайна функція.

В останніх двох випадках вона повинна приймати хоча б один аргумент, що має тип класу, вказівника або посилання на клас.При перевантаженні операцій ( ), [], -> та= функція перевантаження операції може бути оголошеналишеяк методкласу. Для інших операцій функції перевантаження операцій можуть не бути методами класу.

**Індивідуальне завдання:**

Розширити функціональність розроблених у 4 лабораторній роботі класів за допомогою операторів (+, -, <, >), що задані варіантом та оператора присвоювання. Конкретні функції операторів реалізувати на власний розсуд (крім оператора присвоювання). Організувати виведення та введення даних за допомогою класів-потоків сin, cout та перевантажених операторів вводу/виводу. Написати програму, яка демонструє роботу з об'єктами цього класу.

**Хід роботи:**

**Код програми:**

#include <iostream>

#include <conio.h>

using namespace std;

class CBox {

private:

double m\_dblLength = 10, m\_dblWіdth = 10, m\_dblHeight = 10;

public:

void Print() {

cout << "Length = " << m\_dblLength << endl << "Width = " << m\_dblWіdth << endl << "Height = " << m\_dblHeight << endl;

}

void Record() {

double L, W, H;

cout << "Enter Length = ";

cin >> L;

cout << "Enter Width = ";

cin >> W; cout << "Enter Height = ";

cin >> H;

if (L > 0.0) {

if (L <= 50.0)

m\_dblLength = L;

}

if (W > 0) {

if (W <= 50.0)

m\_dblWіdth = W;

}

if (H > 0)

{

if (H <= 50.0)

m\_dblHeight = H;

}

}

friend std::ostream& operator<< (std::ostream &out, const CBox &x);

friend std::istream& operator>> (std::istream &input, CBox &x);

double Volume(bool a) {

double boxV;

boxV = m\_dblLength \* m\_dblWіdth \* m\_dblHeight;

if (a == true) cout << "volume of boxes = " << boxV << endl;

return boxV;

}

double Surface\_area(bool a) {

double boxS;

boxS = (m\_dblLength \* m\_dblWіdth + m\_dblLength \* m\_dblHeight + m\_dblWіdth \* m\_dblHeight) \* 2;

if (a == true)

cout << "Box surface area" << boxS << endl;

return boxS;

}

double girth\_L\_W(bool a) {

double girth;

girth = (m\_dblLength + m\_dblWіdth) \* 2;

if (a == true)

cout << "Box cover LW = " << girth << endl;

return girth;

}

double girth\_L\_H(bool a) {

double girth;

girth = (m\_dblLength + m\_dblHeight) \* 2;

if (a == true)

cout << "Box cover LH = " << girth << endl;

return girth;

}

double girth\_W\_H(bool a) {

double girth;

girth = (m\_dblWіdth + m\_dblHeight) \* 2;

if (a == true)

cout << "Box cover WH = " << girth << endl;

return girth;

}

CBox operator = (const CBox& x) {

this->m\_dblHeight = x.m\_dblHeight;

this->m\_dblLength = x.m\_dblLength;

this->m\_dblWіdth = x.m\_dblWіdth;

return \*this;

}

const CBox operator + (CBox& x) {

CBox a;

a.m\_dblHeight = this->m\_dblHeight + x.m\_dblHeight;

a.m\_dblLength = this->m\_dblLength + x.m\_dblLength;

a.m\_dblWіdth = this->m\_dblWіdth + x.m\_dblWіdth;

return \*new CBox(a);

}

const CBox operator - (CBox& x) {

if (this->m\_dblHeight > x.m\_dblHeight && this->m\_dblLength > x.m\_dblLength && this->m\_dblWіdth > x.m\_dblWіdth) {

CBox a;

a.m\_dblHeight = this->m\_dblHeight - x.m\_dblHeight;

a.m\_dblLength = this->m\_dblLength - x.m\_dblLength;

a.m\_dblWіdth = this->m\_dblWіdth - x.m\_dblWіdth;

return \*new CBox(a);

}

else {

cout << "Subtraction error!!!" << endl;

system("pause"); }

}

bool operator > (CBox& x) {

if (this->m\_dblHeight \*this -> m\_dblLength\*this->m\_dblWіdth > x.m\_dblHeight\*x.m\_dblLength\*x.m\_dblWіdth && x.m\_dblHeight)

return true;

else

return false;

}

bool operator < (CBox& x) {

if (this->m\_dblHeight \*this -> m\_dblLength\*this->m\_dblWіdth < x.m\_dblHeight\*x.m\_dblLength \*x.m\_dblWіdth && x.m\_dblHeight)

return true;

else

return false;

}

};

std::ostream & operator<<(std::ostream & out, const CBox & x)

{

out << "Point(" << x.m\_dblHeight << ", " << x.m\_dblLength << ", " << x.m\_dblWіdth << ")";

return out;

}

std::istream & operator >>(std::istream & input, CBox & x)

{

cout << "Enter:" << endl;

input >> x.m\_dblHeight;

input >> x.m\_dblHeight;

input >> x.m\_dblWіdth;

return input;

}

int main() {

CBox a, b, c;

cout << "Enter the size of the box A" << endl;

a.Record();

cout << "Enter the size of the box B" << endl;

b.Record();

cout << "Demonstration of the operator of the addition (C=A+B)" << endl;

c = a + b;

c.Print();

cout << "Demonstration of subtraction operator (C=A-B)" << endl;

c = a - b;

c.Print();

cout << "Demonstration of the operator > (A>B)" << endl;

if (a > b) cout << "(a > b) = true" << endl;

else cout << "(a > b) = false" << endl;

cout << "Demonstration of the operator > (A<B)" << endl;

if (a < b) cout << "(a < b) = true" << endl;

else cout << "(a < b) = false" << endl;

cout << "Demonstration of the assignment operator = (C=A; C.Print())" << endl;

c = a;

c.Print();

cout << "Demonstration of the operator of the addition (<<)" << endl;

std::cout << a;

std::cout << b;

std::cout << c;

cout << endl;

a.Print();

b.Print();

c.Print();

cout << "Demonstration of the operator of the addition (>>)" << endl;

std::cin >> a;

std::cin >> b;

std::cin>> c;

std::cout <<"You entered: "<< a<<" , "<<b<< " , " <<c<<" . ";

\_getch();

return 0;

}

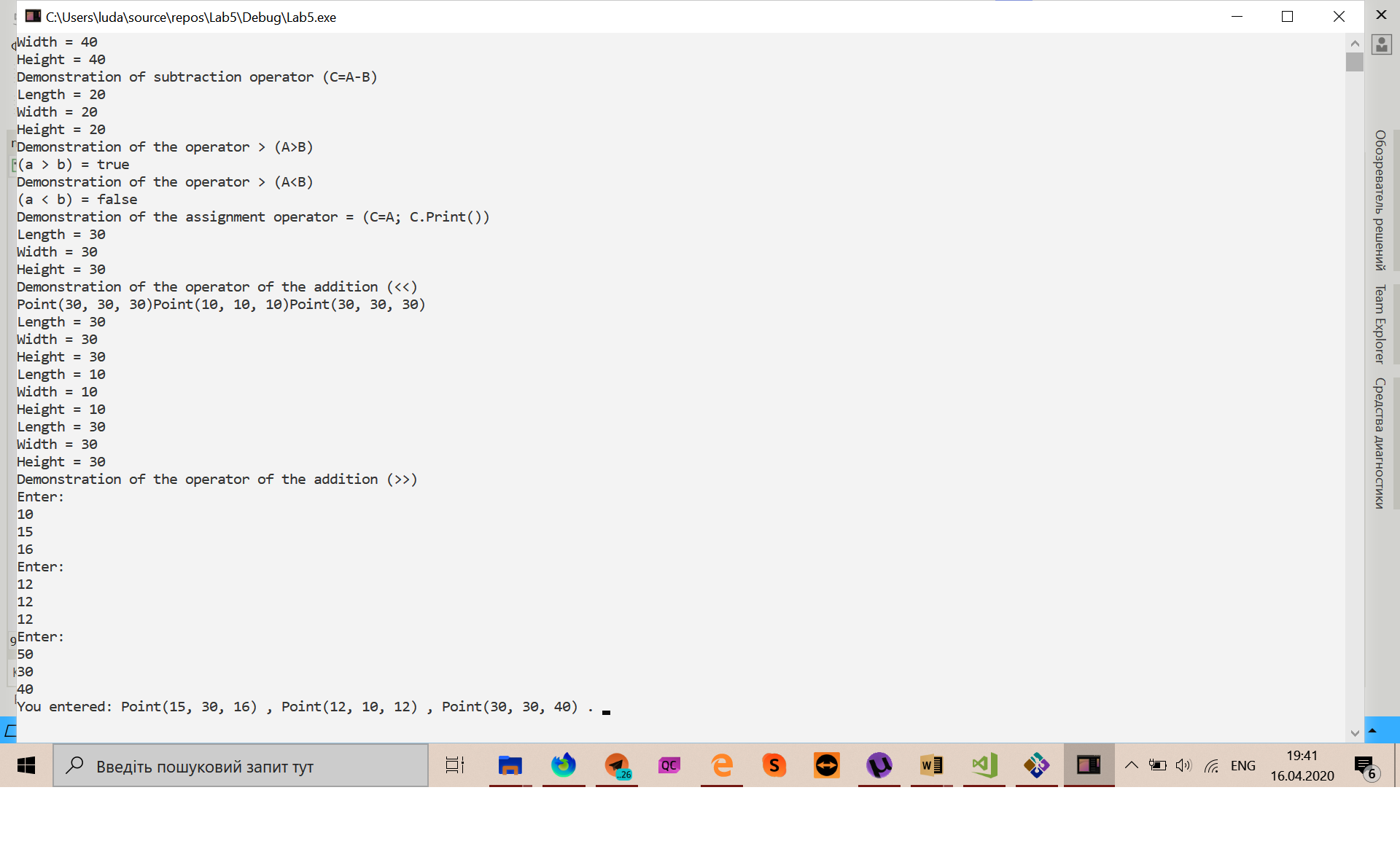


Рис 1. Програма та консольний вивід до завдання.

**Висновок:** під час виконання лабораторної роботи я познайомилась з поняттям перевантаження операторів, засвоїла основні їх властивості.