Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №5

з дисципліни: “Програмування, ч.2 (ООП)”

на тему: “ перевантаження операторів”

Варіант 3

Виконав: ст.гр. КІ-15

Блищак Р.С.

Прийняв:

Козак Н.Б.

Львів

2020

Мета: : познайомитися із перевантаженням операторів.

Теоретичні відомості :

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 присвоювання і 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 не може бути визначеною як static;

• Функція-оператор може бути визначена трьома способами – метод класу, 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 оголошена лише як метод класу. Для інших операцій функції перевантаження операцій можуть не бути методами класу. \

Завдання (Варіант 3) -

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.

Оператори - \*, + , -, >

Код для вирішення задачі даного варіанту –

#include <iostream>

#include <String>

#include <cmath>

using namespace std;

void clear() {

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

void empty() {

cout << endl;

}

class MyVector {

public:

void xyz(int x, int y, int z) {

coord[0] = x;

coord[1] = y;

coord[2] = z;

}

void x(int ix) {

coord[0] = ix;

}

void y(int iy) {

coord[1] = iy;

}

void z(int iz) {

coord[2] = iz;

}

float rx() {

return coord[0];

}

float ry() {

return coord[1];

}

float rz() {

return coord[2];

}

float coord[3] = { 0,0,0 };

MyVector(int x, int y, int z) {

coord[0] = x;

coord[1] = y;

coord[2] = z;

}

MyVector() {

coord[0] = 0;

coord[1] = 0;

coord[2] = 0;

}

friend float operator - (MyVector left, MyVector right) {

return (pow(left.rx()-right.rx(),2)+pow(left.ry() - right.ry(),2)+pow(left.rz() - right.rz(),2));

}

friend float operator + (MyVector left, MyVector right) {

return (pow(left.rx() + right.rx(), 2) + pow(left.ry() + right.ry(), 2) + pow(left.rz() + right.rz(), 2));

}

friend float operator \* (MyVector left, MyVector right) {

return left.rx() \* right.rx() + left.ry() \* right.ry() + left.rz() \* right.rz();

}

MyVector\* operator = (MyVector v1){

this->coord[0] = v1.coord[0];

this->coord[1] = v1.coord[1];

this->coord[2] = v1.coord[2];

return this;

}

};

class CQuad {

public:

CQuad(MyVector v1, MyVector v2, MyVector v3, MyVector v4) {

vector1 = v1;

vector2 = v2;

vector3 = v3;

vector4 = v4;

a = sqrt(v1-v2);

b = sqrt(v2-v3);

c = sqrt(v3-v4);

d = sqrt(v4-v1);

vA.xyz(v2.rx() - v1.rx(), v2.ry() - v1.ry(), v2.rz() - v1.rz());

vB.xyz(v2.rx() - v3.rx(), v2.ry() - v3.ry(), v2.rz() - v3.rz());

vC.xyz(v4.rx() - v3.rx(), v4.ry() - v3.ry(), v4.rz() - v3.rz());

vD.xyz(v4.rx() - v1.rx(), v4.ry() - v1.ry(), v4.rz() - v1.rz());

s1 = vA\*vD;

s2 = vB\*vC;

al = (s1) / (a \* d);

bt = (s2) / (b \* c);

sinal = sqrt(1 - pow(al, 2));

sinbt = sqrt(1 - pow(bt, 2));

p = (a + b + c + d) / 2;

t = (1 + (al \* bt - sinal \* sinbt)) / 2;

sized = sqrt((p - a) \* (p - b) \* (p - c) \* (p - d) - a \* b \* c \* d \* t);

}

float size() {

return sized;

}

bool square() {

if (al == 0 && bt == 0) {

return true;

}

else return false;

}

float perimetr() {

return a + b + c + d;

}

friend bool operator > (CQuad left, CQuad right) {

if (left.size() > right.size() && left.perimetr() > right.perimetr()) {

return true;

}

else return false;

}

private:

MyVector vector1, vector2, vector3, vector4, vA, vB, vC, vD;

float a, b, c, d, d1, d2, p, t, al, bt, s1, s2, sinal, sinbt, sized = 0;

};

int main() {

MyVector vA, vB, vC, vD;

int x, y, z = 0;

cout << "Enter a x parameter to A point - " << endl;

cin >> x;

cout << "Enter a y parameter to A point - " << endl;

cin >> y;

cout << "Enter a z parameter to A point - " << endl;

cin >> z;

vA.xyz(x,y,z);

clear();

x, y, z = 0;

cout << "Enter a x parameter to B point - " << endl;

cin >> x;

cout << "Enter a y parameter to B point - " << endl;

cin >> y;

cout << "Enter a z parameter to B point - " << endl;

cin >> z;

vB.xyz(x, y, z);

clear();

x, y, z = 0;

cout << "Enter a x parameter to C point - " << endl;

cin >> x;

cout << "Enter a y parameter to C point - " << endl;

cin >> y;

cout << "Enter a z parameter to C point - " << endl;

cin >> z;

vC.xyz(x, y, z);

clear();

x, y, z = 0;

cout << "Enter a x parameter to D point - " << endl;

cin >> x;

cout << "Enter a y parameter to D point - " << endl;

cin >> y;

cout << "Enter a z parameter to D point - " << endl;

cin >> z;

vD.xyz(x, y, z);

clear();

CQuad myQuad(vA,vB,vC,vD);

cout << "Okay, first quad has been created and completed";

x, y, z = 0;

cout << "Second square - " << endl;

cout << "Enter a x parameter to A point - " << endl;

cin >> x;

cout << "Enter a y parameter to A point - " << endl;

cin >> y;

cout << "Enter a z parameter to A point - " << endl;

cin >> z;

vA.xyz(x, y, z);

clear();

x, y, z = 0;

cout << "Enter a x parameter to B point - " << endl;

cin >> x;

cout << "Enter a y parameter to B point - " << endl;

cin >> y;

cout << "Enter a z parameter to B point - " << endl;

cin >> z;

vB.xyz(x, y, z);

clear();

x, y, z = 0;

cout << "Enter a x parameter to C point - " << endl;

cin >> x;

cout << "Enter a y parameter to C point - " << endl;

cin >> y;

cout << "Enter a z parameter to C point - " << endl;

cin >> z;

vC.xyz(x, y, z);

clear();

x, y, z = 0;

cout << "Enter a x parameter to D point - " << endl;

cin >> x;

cout << "Enter a y parameter to D point - " << endl;

cin >> y;

cout << "Enter a z parameter to D point - " << endl;

cin >> z;

vD.xyz(x, y, z);

clear();

CQuad myQuad1(vA, vB, vC, vD);

cout << "Okay, second quad has been created and completed" << endl;

if (myQuad>myQuad1) {

cout << "First quad is bigger than the second one" << endl;

}

}

Скріншот виконання програми –



Висновок : в даній лабораторній роботі я опрацював роботу з классами, об’єктами классів, застосуванням їх і використанням геттерів і сеттерів.