Міністерство освіти і науки

Національний університет “Львівська політехніка”

**Кафедра ЕОМ**



**Звіт**

з лабораторної роботи № 5

# з дисципліни: “ Програмування, частина 2 (Об'єктно-орієнтоване програмування”

на тему: “Перевантаження операторів ”

**виконав:**

**студент групи кі-15**

**Височанський Сергій**

**Прийняв:**

**Козак Н.Б.**

Львів 2020

Мета: познайомитися із перевантаженням операторів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Перевантаження операторів 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 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; 78 • 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:

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.

Варіант 5: (), ==, !=, ++

Виконання роботи:

#include <math.h>

#include <iostream>

#include <string>

class Triangle

{

private:

double max;

double r;

double p;

double pp;

private:

double a;

double b;

double c;

public:

Triangle(int \*\*arr);

~Triangle(){};

double right();

double type();

double perimetr();

double square();

double operator()(int i)

{

return(r+=i);

}

bool operator==(const Triangle& other)

{

return this->a==other.a&&this->b==other.b&&this->c==other.c;

}

bool operator!=(const Triangle& other)

{

return !(this->a == other.a && this->b == other.b&& this->c==other.c);

}

Triangle& operator++(int i)

{

a++;

b++;

c++;

return\*this;

}

friend std::ostream& operator<< (std::ostream &out, const Triangle &triangle);

friend std::istream& operator>> (std::istream &in, Triangle &triangle);

};

Triangle::Triangle(int \*\*arr) {

a = sqrt(powl((arr[1][0] - arr[0][0]), 2) + powl((arr[1][1] - arr[0][1]), 2) + powl((arr[1][2] - arr[0][2]), 2));

b = sqrt(powl((arr[2][0] - arr[1][0]), 2) + powl((arr[2][1] - arr[1][1]), 2) + powl((arr[2][2] - arr[1][2]), 2));

c = sqrt(powl((arr[2][0] - arr[0][0]), 2) + powl((arr[2][1] - arr[0][1]), 2) + powl((arr[2][2] - arr[0][2]), 2));

p = a+b+c;

pp=((a+b+c)/2);

}

std::ostream& operator<< (std::ostream &out, const Triangle &triangle)

{

out <<"Довжина а:" <<triangle.a<<" "<<"Довжина b: "<<triangle.b<<" Довжина c:"<<triangle.c;

return out;

}

std::istream& operator>> (std::istream &in, Triangle &triangle)

{

in >> triangle.a;

in >> triangle.b;

in >> triangle.c;

return in;

}

double Triangle::perimetr()

{

return (p);

}

double Triangle::right() {

max=a;

if ( max < b)

b = max;

else if (max < c)

c = max;

return ( max );

}

double::Triangle::type(){

if(max==sqrt(pow(a,2)+pow(b,2))||max==sqrt( pow(b,2)+pow(c,2))||max==sqrt(pow(a,2)+pow(c,2)))

return (1);

else

return(0);

}

double Triangle::square() {

return (sqrt(pp\*(pp-a)\*(pp-b)\*(pp-c))); }

using namespace std;

int main() {

setlocale(LC\_ALL, "Ukrainian");

int \*\*arr = new int \*[3];

for (int i = 0; i < 3; i++)

{

arr[i] = new int[3];

if (i == 0)

cout << "Введiть координати 1 сторони: ";

else if (i == 1)

cout << "Введiть координати 2 сторони: ";

else if (i == 2)

cout << "Введiть координати 3 сторони: "; cin>> arr[i][0] >> arr[i][1] >> arr[i][2];

}

Triangle triangle(arr);

Triangle triangle2(arr);

cout << "Прямокутний: " << triangle.type() << endl;

cout << "Площа трикутника: " << triangle.square() << endl;

cout << "Периметр: " << triangle.perimetr() << endl;

cout<<triangle(30)<<endl;

bool result1 = triangle==triangle2;

bool result2 = triangle!= triangle2;

triangle++;

cout<<triangle;

for (int i = 0; i < 2; i++)

delete[] arr[i];

delete[] arr;

}

Скріншот виконання програми:

