Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 6**

***«****Перевантаження функцій та операцій, дружні функції****»***

**з дисципліни *«****Об’єктно–орієнтоване програмування****»***

**Лектор:**

ст. викладач кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-11

Артех О.Ю.

**Прийняла:**

асистент кафедри ПЗ

Терендій О.В.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Тема роботи:** Перевантаження функцій і операцій, дружні функції.

**Мета роботи:** Навчитися використовувати механізм перевантаження функцій та операцій. Навчитися створювати та використовувати дружні функції. Ознайомитися з статичними полями та методами, а також навчитися їх використовувати при створенні класів.

**Теоретичні відомості**

С++ підтримує спеціальні засоби, які дозволяють перевизначити вже існуючі операції. Наприклад, для операції + можна ввести своє власне визначення, яке реалізує операцію додавання для об’єктів певного класу. Фактично пере визначення для операцій існує і в мові С. Так, операція + може використовувати як об’єкти типу int, так і об’єкти типу float. С++ розширює цю ідею.

Для визначення операції використовується функція, що вводиться користувачем. Тип функції визначається іменем класу, далі записується  ключове слово operator, за яким слідує символ операції, в круглих дужках дається перелік параметрів, серед яких хоча б один типу клас.

Дружньою функцією класу називається функція, яка сама не є членом класу, але має повні права на доступ до закритих та захищених елементів класу. Оскільки така функція не є членом класу, то вона не може бути вибрана з допомогою операторів (.) та (->), Дружня функція класу викликається звичайним способом. Для опису дружньої функції використовується ключове слово **friend**.

До тепер рахувалось, що дані в кожному об’єкті є власністю саме цього об’єкту та не використовуються іншими об’єктами цього класу. Та іноді приходиться слідкувати за накопиченням даних. Наприклад, необхідно з’ясувати скільки об’єктів даного класу було створено на даний момент та скільки з них існує. Статичні змінні-члени досяжні для всіх екземплярів класу. Це є компроміс між глобальними даними, які досяжні всім елементам програми, та даними, що досяжні тільки об’єктам даного класу.

Статична змінні створюється в одному екземплярі для всіх об’єктів даного класу.

Статичні функції класу подібні до статичних змінних: вони не належать одному об’єкту, а знаходяться в області дії всього класу. Статичні функції-члени не мають вказівника this. Відповідно їх не можна оголосити як const. Статичні функції-члени не можуть звертатись до нестатичних змінних. До статичних функцій-членів можна звертатись з об’єкту їх класу, або вказавши повне ім’я, включаючи ім’я об’єкту.

**Індивідуальне завдання**

На основі класу з попередньої лабораторної:

1. Перевантажити як мінімум три функції-члени з попереднього завдання.
2. Перевантажити операції згідно з варіантом (№2). Для операцій, для яких не вказані символи, вибрати символи самостійно.

2 3. Клас Vector – вектор в просторі.

Перевантажити операції, як функції члени:

Додавання

Віднімання

Скалярний добуток

Векторний добуток

Перевантажити операції, як дружні-функції:

Добуток вектора на скаляр

Введення вектора з форми ("<<")

Виведення вектора на форму (">>")

Більше (">")

Менше ("<")

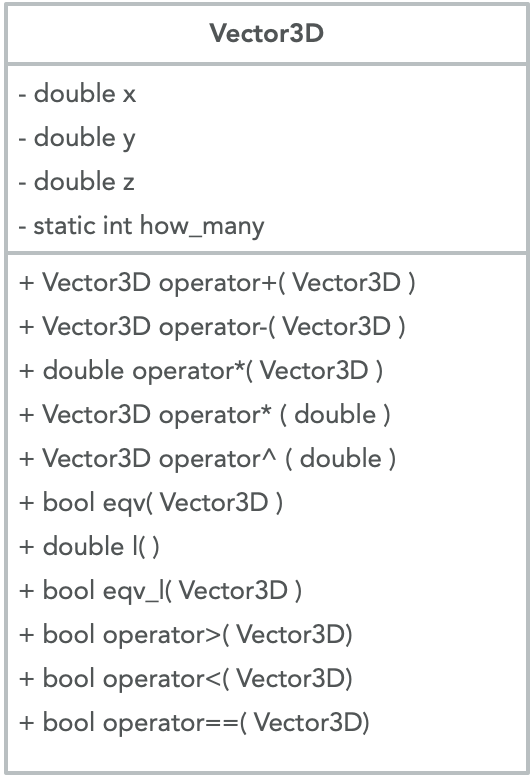
Рівне ("==") (при порівнянні порівнювати довжини векторів).

Створити статичне поле, в якому б містилась інформація про кількість створених об’єктів, а також статичні функції для роботи з цим полем.Створити дружні функції згідно з варіантом.

1. Створити статичні поля та статичні методи згідно з варіантом.
2. Продемонструвати розроблені можливості класу завдяки створеному віконному застосуванню.

**Хід роботи**

UML діаграма класу:



Код програми:

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_set\_v1\_clicked**();

void **on\_set\_v2\_clicked**();

void **on\_add\_clicked**();

void **on\_sub\_clicked**();

void **on\_set\_as\_v1\_1\_clicked**();

void **on\_set\_as\_v1\_2\_clicked**();

void **on\_set\_as\_v2\_1\_clicked**();

void **on\_set\_as\_v2\_2\_clicked**();

void **on\_scalyar\_clicked**();

void **on\_choose\_v1\_1\_clicked**();

void **on\_choose\_v2\_1\_clicked**();

void **on\_on\_const\_clicked**();

void **on\_choose\_v1\_2\_clicked**();

void **on\_choose\_v2\_2\_clicked**();

void **on\_lenght\_clicked**();

void **on\_eqv\_clicked**();

void **on\_eqv\_l\_clicked**();

void **on\_vect\_clicked**();

void **on\_set\_as\_v1\_3\_clicked**();

void **on\_set\_as\_v2\_3\_clicked**();

void **on\_show\_how\_many\_clicked**();

void **on\_bigger\_clicked**();

void **on\_smaller\_clicked**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

vectors.h

#ifndef CLASS567\_CLASS\_H

#define CLASS567\_CLASS\_H

#include <iostream>

#include <cmath>

*class* **Vector3D** {

*private* :

double x,y,z;

*static* int how\_many;

*public* :

**Vector3D**(){how\_many++;};

**Vector3D**(double x, double y, double z);

~**Vector3D**(){how\_many--;};

*static* int **get\_how\_many**(){*return* how\_many;};

double **get\_x**();

double **get\_y**();

double **get\_z**();

void **set\_x**(double x);

void **set\_y**(double y);

void **set\_z**(double z);

Vector3D *operator*+(Vector3D v2);

Vector3D *operator*-(Vector3D v2);

double *operator*\*( Vector3D v2);

Vector3D *operator*^(Vector3D v2);

bool **eqv**(Vector3D v2);

double **l**();

*friend* Vector3D *operator*\*(Vector3D v, double c);

*friend* std::ostream& *operator*<< (std::ostream &out, *const* Vector3D &vector);

*friend* std::istream& *operator*>> (std::istream &in, Vector3D &vector);

*friend* bool *operator*> (Vector3D v1, Vector3D v2);

*friend* bool *operator*< (Vector3D v1, Vector3D v2);

*friend* bool *operator*==(Vector3D v1, Vector3D v2);

};

#endif

main.cpp

#include "mainwindow.h"

#include <QApplication>

int **main**(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "vectors.h"

#include "stdio.h"

#include "string.h"

#include "stdlib.h"

Vector3D vector\_1, vector\_2, vector\_add, vector\_sub, vector\_vect;

int on\_const\_choose = 1, lenght\_choose = 1;

int Vector3D::how\_many = 0;

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**on\_set\_v1\_clicked**()

{

vector\_1.set\_x(ui->v1\_x->text().toDouble());

vector\_1.set\_y(ui->v1\_y->text().toDouble());

vector\_1.set\_z(ui->v1\_z->text().toDouble());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_1.get\_x(),vector\_1.get\_y(),vector\_1.get\_z());

ui->label\_v1->setText(coord);

}

void MainWindow::**on\_set\_v2\_clicked**()

{

vector\_2.set\_x(ui->v2\_x->text().toDouble());

vector\_2.set\_y(ui->v2\_y->text().toDouble());

vector\_2.set\_z(ui->v2\_z->text().toDouble());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_2.get\_x(),vector\_2.get\_y(),vector\_2.get\_z());

ui->label\_v2->setText(coord);

}

void MainWindow::**on\_add\_clicked**()

{

vector\_add = vector\_1 + vector\_2 ;

char res[100];

sprintf(res,"%5.2lf, %5.2lf, %5.2lf",vector\_add.get\_x(),vector\_add.get\_y(),vector\_add.get\_z());

ui->add\_res->setText(res);

}

void MainWindow::**on\_sub\_clicked**()

{

vector\_sub = vector\_1 - vector\_2;

char res[100];

sprintf(res,"%5.2lf, %5.2lf, %5.2lf",vector\_sub.get\_x(),vector\_sub.get\_y(),vector\_sub.get\_z());

ui->sub\_res->setText(res);

}

void MainWindow::**on\_set\_as\_v1\_1\_clicked**()

{

vector\_1.set\_x(vector\_add.get\_x());

vector\_1.set\_y(vector\_add.get\_y());

vector\_1.set\_z(vector\_add.get\_z());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_1.get\_x(),vector\_1.get\_y(),vector\_1.get\_z());

ui->label\_v1->setText(coord);

}

void MainWindow::**on\_set\_as\_v1\_2\_clicked**()

{

vector\_1.set\_x(vector\_sub.get\_x());

vector\_1.set\_y(vector\_sub.get\_y());

vector\_1.set\_z(vector\_sub.get\_z());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_1.get\_x(),vector\_1.get\_y(),vector\_1.get\_z());

ui->label\_v1->setText(coord);

}

void MainWindow::**on\_set\_as\_v2\_1\_clicked**()

{

vector\_2.set\_x(vector\_add.get\_x());

vector\_2.set\_y(vector\_add.get\_y());

vector\_2.set\_z(vector\_add.get\_z());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_2.get\_x(),vector\_2.get\_y(),vector\_2.get\_z());

ui->label\_v2->setText(coord);

}

void MainWindow::**on\_set\_as\_v2\_2\_clicked**()

{

vector\_2.set\_x(vector\_sub.get\_x());

vector\_2.set\_y(vector\_sub.get\_y());

vector\_2.set\_z(vector\_sub.get\_z());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_2.get\_x(),vector\_2.get\_y(),vector\_2.get\_z());

ui->label\_v2->setText(coord);

}

void MainWindow::**on\_scalyar\_clicked**()

{

double dres = vector\_1 \* vector\_2;

char res[100];

sprintf(res,"%lf",dres);

ui->scalyar\_res->setText(res);

}

void MainWindow::**on\_choose\_v1\_1\_clicked**()

{

ui->choose\_v1\_1->setChecked(*true*);

ui->choose\_v2\_1->setChecked(*false*);

on\_const\_choose = 1;

}

void MainWindow::**on\_choose\_v2\_1\_clicked**()

{

ui->choose\_v1\_1->setChecked(*false*);

ui->choose\_v2\_1->setChecked(*true*);

on\_const\_choose = 2;

}

void MainWindow::**on\_on\_const\_clicked**()

{

*if*(on\_const\_choose == 1)

{

vector\_1 = vector\_1\*(ui->const\_value->text().toDouble());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_1.get\_x(),vector\_1.get\_y(),vector\_1.get\_z());

ui->label\_v1->setText(coord);

}

*else*

{

vector\_2 = vector\_2\*(ui->const\_value->text().toDouble());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_2.get\_x(),vector\_2.get\_y(),vector\_2.get\_z());

ui->label\_v2->setText(coord);

}

}

void MainWindow::**on\_choose\_v1\_2\_clicked**()

{

ui->choose\_v1\_2->setChecked(*true*);

ui->choose\_v2\_2->setChecked(*false*);

lenght\_choose = 1;

}

void MainWindow::**on\_choose\_v2\_2\_clicked**()

{

ui->choose\_v1\_2->setChecked(*false*);

ui->choose\_v2\_2->setChecked(*true*);

lenght\_choose = 2;

}

void MainWindow::**on\_lenght\_clicked**()

{

double dres;

*if*( lenght\_choose == 1)

dres = vector\_1.l();

*else*

dres = vector\_2.l();

char res[100];

sprintf(res,"%lf",dres);

ui->lenght\_res->setText(res);

}

void MainWindow::**on\_eqv\_clicked**()

{

int i = vector\_1.eqv(vector\_2);

*if*(i == 0)

ui->eqv\_res->setText("false");

*else*

ui->eqv\_res->setText("true");

}

void MainWindow::**on\_eqv\_l\_clicked**()

{

int i = (vector\_1 == vector\_2);

*if*(i == 0)

ui->eqv\_l\_res->setText("false");

*else*

ui->eqv\_l\_res->setText("true");

}

void MainWindow::**on\_vect\_clicked**()

{

vector\_vect = vector\_1^vector\_2 ;

char res[100];

sprintf(res,"%5.2lf, %5.2lf, %5.2lf",vector\_vect.get\_x(),vector\_vect.get\_y(),vector\_vect.get\_z());

ui->vect\_res->setText(res);

}

void MainWindow::**on\_set\_as\_v1\_3\_clicked**()

{

vector\_1.set\_x(vector\_vect.get\_x());

vector\_1.set\_y(vector\_vect.get\_y());

vector\_1.set\_z(vector\_vect.get\_z());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_1.get\_x(),vector\_1.get\_y(),vector\_1.get\_z());

ui->label\_v1->setText(coord);

}

void MainWindow::**on\_set\_as\_v2\_3\_clicked**()

{

vector\_2.set\_x(vector\_vect.get\_x());

vector\_2.set\_y(vector\_vect.get\_y());

vector\_2.set\_z(vector\_vect.get\_z());

char coord[100];

sprintf(coord,"%5.2lf, %5.2lf, %5.2lf",vector\_2.get\_x(),vector\_2.get\_y(),vector\_2.get\_z());

ui->label\_v2->setText(coord);

}

void MainWindow::**on\_show\_how\_many\_clicked**()

{

int q = Vector3D::get\_how\_many();

char str[10];

sprintf(str,"%d",q);

ui->res\_show\_how\_many->setText(str);

}

void MainWindow::**on\_bigger\_clicked**()

{

bool q = (vector\_1 > vector\_2);

*if* ( q )

ui->res\_is\_bs->setText("true");

*else*

ui->res\_is\_bs->setText("false");

}

void MainWindow::**on\_smaller\_clicked**()

{

bool q = (vector\_1 < vector\_2);

*if* ( q )

ui->res\_is\_bs->setText("true");

*else*

ui->res\_is\_bs->setText("false");

}

vectors.cpp

#include "vectors.h"

Vector3D::**Vector3D**(double x, double y, double z){

*this*->x =x;

*this*->y =y;

*this*->z =z;

how\_many++;

}

double Vector3D::**get\_x**(){

*return* x;

}

double Vector3D::**get\_y**(){

*return* y;

}

double Vector3D::**get\_z**(){

*return* z;

}

void Vector3D::**set\_x**(double x){

*this*->x = x;

}

void Vector3D::**set\_y**(double y){

*this*->y = y;

}

void Vector3D::**set\_z**(double z){

*this*->z = z;

}

Vector3D Vector3D::*operator*+( Vector3D v2){

Vector3D temp;

temp.set\_x(*this*->x+v2.x);

temp.set\_y(*this*->y+v2.y);

temp.set\_z(*this*->z+v2.z);

*return* temp;

}

Vector3D Vector3D::*operator*-( Vector3D v2){

Vector3D temp;

temp.set\_x(*this*->x - v2.x);

temp.set\_y(*this*->y - v2.y);

temp.set\_z(*this*->z - v2.z);

*return* temp;

}

double Vector3D::*operator*\*(Vector3D v2){

*return* *this*->x\*v2.x + *this*->y\*v2.y + *this*->z\*v2.z;

}

Vector3D *operator*\*( Vector3D v, double c){

Vector3D temp;

temp.set\_x(v.get\_x()\*c);

temp.set\_y(v.get\_y()\*c);

temp.set\_z(v.get\_z()\*c);

*return* temp;

}

Vector3D Vector3D::*operator*^(Vector3D v2){

Vector3D temp;

temp.set\_x(*this*->y\*v2.z-v2.y\**this*->z);

temp.set\_y(-(*this*->x\*v2.z-v2.x\**this*->z));

temp.set\_z(*this*->x\*v2.y-v2.x\**this*->y);

*return* temp;

}

bool Vector3D::**eqv**(Vector3D v2){

*if*(*this*->x != v2.x)

*return* *false*;

*if*(*this*->y != v2.y)

*return* *false*;

*return* *this*->z == v2.z;

}

double Vector3D::**l**(){

*return* pow(*this*->x\**this*->x + *this*->y\**this*->y + *this*->z\**this*->z,1.0/2) ;

}

bool *operator*==( Vector3D v1, Vector3D v2){

*return* (v1.l() == v2.l());

}

std::ostream& *operator*<< (std::ostream &out, *const* Vector3D &vector)

{

out << "(" << vector.x << ", " << vector.y << ", " << vector.z << ")";

*return* out;

}

std::istream& *operator*>> (std::istream &in, Vector3D &vector)

{

in >> vector.x;

in >> vector.y;

in >> vector.z;

*return* in;

}

bool *operator*> (Vector3D v1, Vector3D v2)

{

*return* ( v1.l() > v2.l()) ;

}

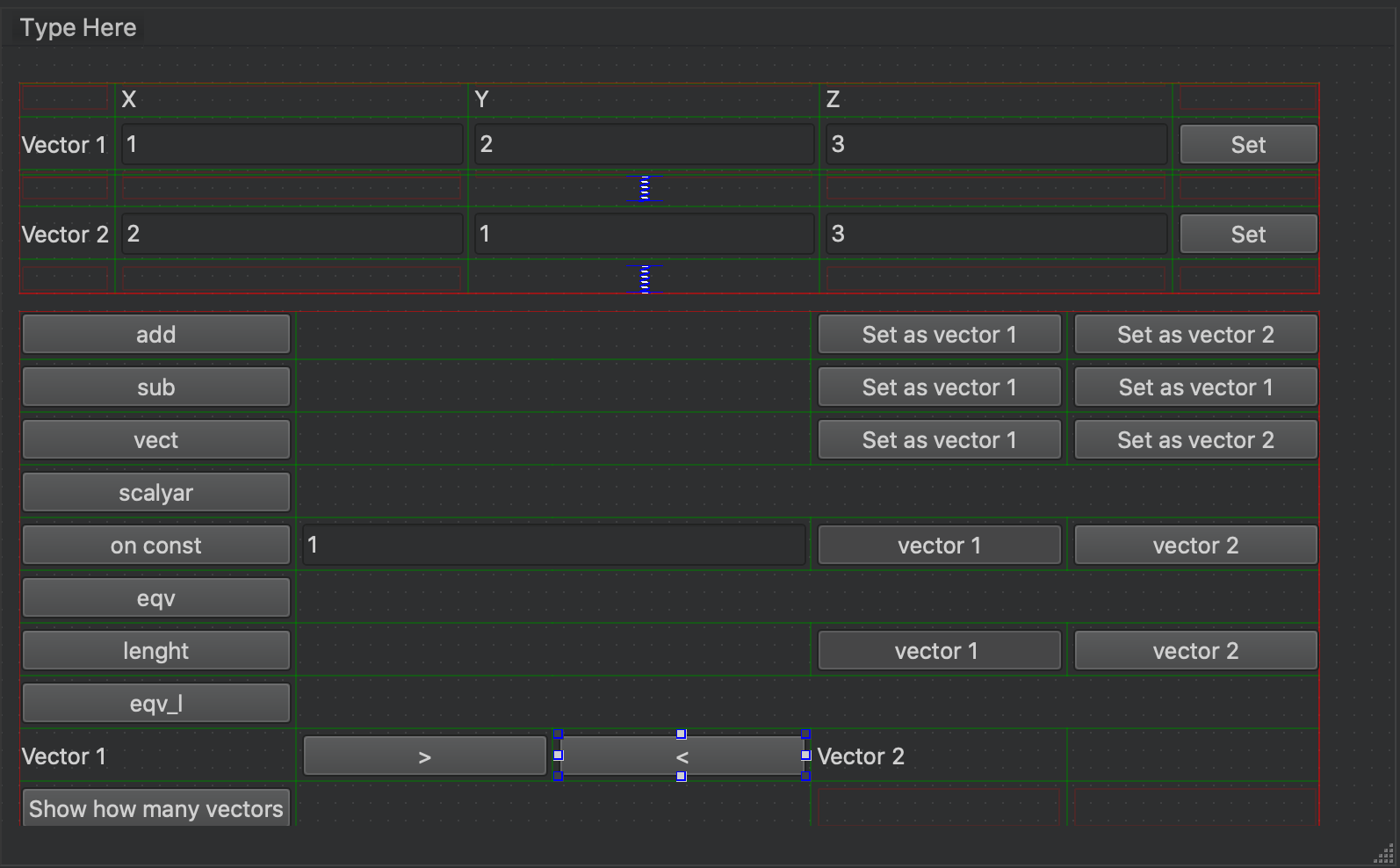
bool *operator*< (Vector3D v1, Vector3D v2)

{

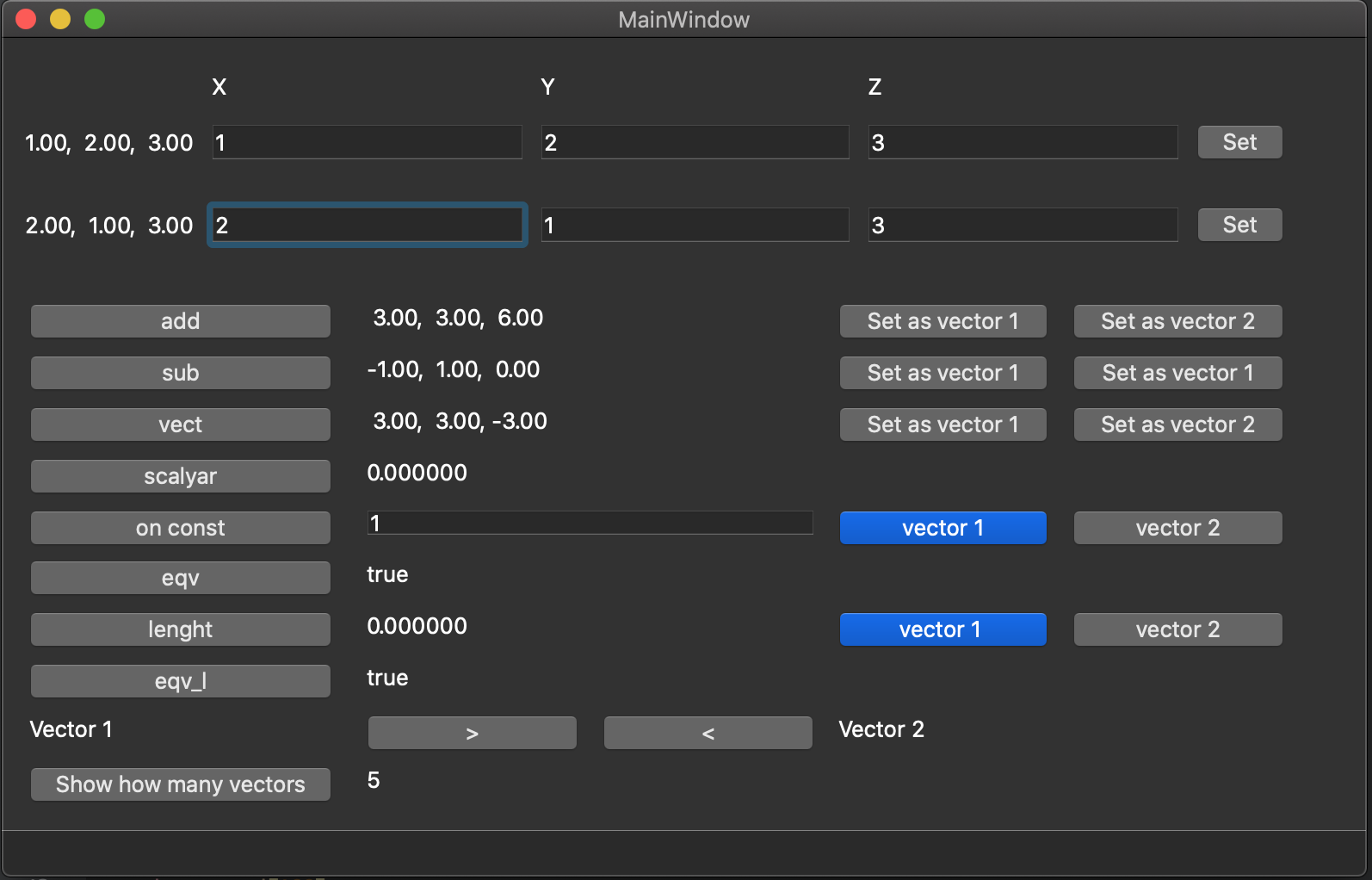
*return* ( v1.l() < v2.l()) ;

}

Форма MainWindow:



Результат виконання:



**Висновки**

Під час виконання даної лабораторній роботі я навчилася використовувати механізм перевантаження функцій та операцій, створювати та використовувати дружні функції. Ознайомився з статичними полями та методами. Отримані знання освоїв, створивши графічний застосунок для роботи зі створеним класом Vector3D із перевантаженими базовими операторами та набором перевантажених функцій.