Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 10**

*«Шаблони класів»*

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

**Лектор:**

ст. викладач кафедри ПЗ

Коротєєва Т.О.

**Виконав:**

студ. групи ПЗ-12

Артех О.Ю.

**Прийняла:**

асистент кафедри ПЗ

Терендій О.В.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Тема роботи:** Шаблони класу.

**Мета роботи:** Навчитись створювати шаблон класу та екземпляри шаблонів.

**Теоретичні відомості**

Шаблони дозволяють визначити конструкції (функції, класи), які використовують певні типи, але на момент написання коду точно не відомо, що це будуть за типи. Іншими словами, шаблони дозволяють визначити універсальні конструкції, які НЕ залежать від певного типу.

Для застосування шаблонів перед класом вказується ключове слово template, після якого йдуть кутові дужки. У кутових дужках після слова typename йде параметр шаблону. Можна визначити кілька параметрів шаблону, в прикладі вище застосовується тільки один параметр.

template <typename T>

Параметр шаблону представляє довільний ідентифікатор, в якості якого, як правило, застосовують великі літери, наприклад, T. Але це необов'язково. Тобто в даному випадку параметр T представлятиме певний тип, який стає відомим під час компіляції. Це може бути і тип int, і double, і string, і будь-який інший тип.

**Індивідуальне завдання**

1.Створити шаблон класу Number. Шаблон класу повинен давати

можливість вивести число на екран, додавати, віднімати, множити, а

також порівнювати  числа.

2. Використати  функціонал шаблону на розробленому користувацькому

типі з лаб.роботи 5 ( Vector3D ).

3. Продемонструвати його можливості завдяки створеному віконному

застосуванню.

4. Оформити звіт до лабораторної роботи..

**Хід роботи**

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\_button\_plus\_clicked**();

void **on\_button\_minus\_clicked**();

void **on\_button\_mult\_clicked**();

void **on\_button\_big\_clicked**();

void **on\_botton\_sm\_clicked**();

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

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

***number.h***

#ifndef NUMBER\_H

#define NUMBER\_H

#include <iostream>

#include <string>

#include <sstream>

*using* *namespace* std;

*template*<*class* T>

*class* **Number** {

T data;

*public* :

**Number**()= *default*;;

**Number**(T *const* &temp){

*this*->data = temp;

};

Number *operator*+(Number other) {

*return* *this*->data + other.data;

}

Number *operator*-(Number other) {

*return* *this*->data - other.data;

}

Number *operator*\*(Number other) {

*return* (*this*->data) \* other.data;

}

bool *operator*>(Number other) {

*return* (*this*->data) > other.data;

}

bool *operator*<(Number other) {

*return* (*this*->data) < other.data;

}

bool *operator*==(Number other) {

*return* (*this*->data) == other.data;

}

*friend* std::ostream &*operator*<<(std::ostream &out, *const* Number &obj) {

out << obj.data;

*return* out;

};

*friend* string **num\_to\_str**(*const* Number &obj){

ostringstream temp;

temp << obj.data;

string str = temp.str();

*return* str;

}

};

#endif *//* *NUMBER\_H*

***vectors.h***

#ifndef VECTORS\_H

#define VECTORS\_H

#include <iostream>

#include <cmath>

#include <string>

#include <sstream>

*using* *namespace* std;

*class* **Vector3D** {

*private* :

double x{}, y{}, z{};

*public* :

**Vector3D**() = *default*;;

**Vector3D**(Vector3D *const* &temp){

*this*->x = temp.x;

*this*->y = temp.y;

*this*->z = temp.z;

}

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

*this*->x = x;

*this*->y = y;

*this*->z = z;

};

~**Vector3D**() = *default*;;

double **get\_x**() {

*return* x;

};

double **get\_y**() {

*return* y;

};

double **get\_z**() {

*return* z;

};

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

*this*->x = x;

};

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

*this*->y = y;

};

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

*this*->z = z;

};

Vector3D& *operator*=(Vector3D v2){

*this*->x = v2.x;

*this*->y = v2.y;

*this*->z = v2.z;

*return* \**this*;

}

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 *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 *operator*\*(Vector3D v2) {

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

};

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 **eqv**(Vector3D v2) {

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

*return* *false*;

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

*return* *false*;

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

};

double **l**() {

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

};

*friend* 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;

};

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

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

*return* out;

};

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

in >> vector.x;

in >> vector.y;

in >> vector.z;

*return* in;

};

*friend* bool *operator*>(Vector3D v1, Vector3D v2) { *return* (v1.l() > v2.l()); };

*friend* bool *operator*<(Vector3D v1, Vector3D v2) { *return* (v1.l() < v2.l()); };

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

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

};

*friend* string **vector\_to\_str**(*const* Vector3D &obj){

ostringstream temp;

temp << obj;

string str = temp.str();

*return* str;

}

};

#endif *//* *VECTORS\_H*

***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 "number.h"

Vector3D v1,v2;

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

: QMainWindow(parent)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

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

{

*delete* ui;

}

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

{

v1.set\_x(ui->v1\_x->text().toInt());

v1.set\_y(ui->v1\_y->text().toInt());

v1.set\_z(ui->v1\_z->text().toInt());

Number<Vector3D> temp1(v1);

QString temp= QString::fromStdString(num\_to\_str(temp1));

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

}

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

{

v2.set\_x(ui->v2\_x->text().toInt());

v2.set\_y(ui->v2\_y->text().toInt());

v2.set\_z(ui->v2\_z->text().toInt());

Number<Vector3D> temp2(v2);

QString temp= QString::fromStdString(num\_to\_str(temp2));

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

}

void MainWindow::**on\_button\_plus\_clicked**()

{

Number<Vector3D> temp1(v1);

Number<Vector3D> temp2(v2);

Number<Vector3D> res;

res = v1 + v2;

QString temp= QString::fromStdString(num\_to\_str(res));

ui->label\_res->setText(temp);

}

void MainWindow::**on\_button\_minus\_clicked**()

{

Number<Vector3D> temp1(v1);

Number<Vector3D> temp2(v2);

Number<Vector3D> res;

res = v1 - v2;

QString temp= QString::fromStdString(num\_to\_str(res));

ui->label\_res->setText(temp);

}

void MainWindow::**on\_button\_mult\_clicked**()

{

Number<Vector3D> temp1(v1);

Number<Vector3D> temp2(v2);

Number<Vector3D> res;

res = v1 ^ v2;

QString temp= QString::fromStdString(num\_to\_str(res));

ui->label\_res->setText(temp);

}

void MainWindow::**on\_button\_big\_clicked**()

{

Number<Vector3D> temp1(v1);

Number<Vector3D> temp2(v2);

*if*( temp1 > temp2)

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

*else*

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

}

void MainWindow::**on\_botton\_sm\_clicked**()

{

Number<Vector3D> temp1(v1);

Number<Vector3D> temp2(v2);

*if*( temp1 < temp2)

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

*else*

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

}

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

{

Number<Vector3D> temp1(v1);

Number<Vector3D> temp2(v2);

*if*( temp1 == temp2)

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

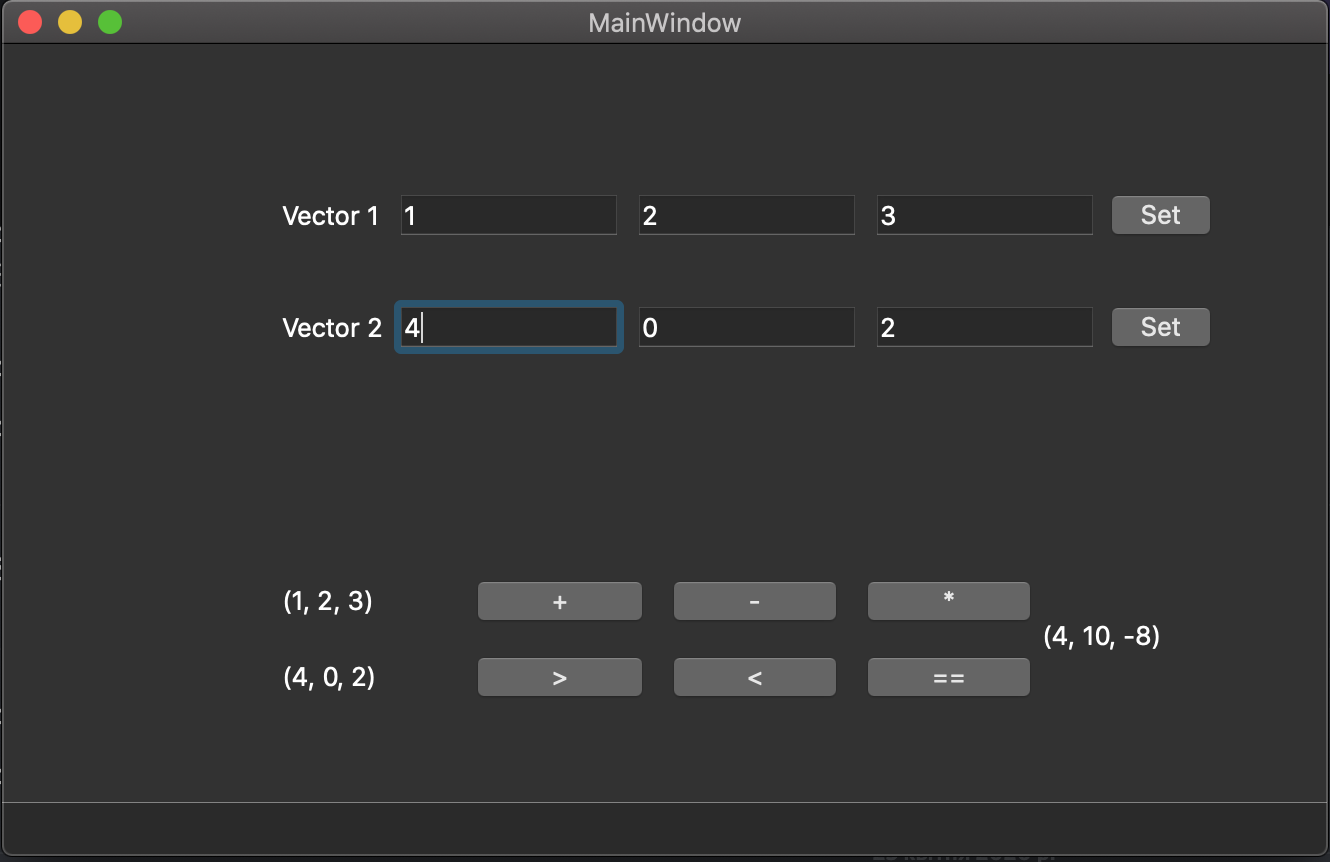
*else*

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

}

Форма та застосунок :





**Висновки**

Під час виконання лабораторної роботи я створив шаблон класу Set та продемонстрував його роботу на створеному користувацькому числа String.