**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**



### ЗВІТ

До лабораторної роботи № 10

**На тему:** *“* *Шаблони класів”*

**З дисципліни:** *“Об’єктно-орієнтоване програмування”*

**Лектор:**

доцент каф.ПЗ

Коротєєва Т. О.

**Виконав:**

ст. гр. ПЗ-16

Шеремета А.І.

**Прийняв:**

асист. каф. ПЗ

Дивак І.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

**Тема роботи:** Шаблони класів.

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

**Теоретичні відомості**

Шаблони є основою для універсального програмування на C++. Як строго типізована мова C++ вимагає, щоб усі змінні мали конкретний тип, явно оголошений програмістом або компілятором. Однак багато структур та алгоритмів даних виглядають однаково незалежно від типу, на який вони працюють. Шаблони дозволяють визначити операції класу чи функції та дозволити користувачеві вказати, які типи цих операцій мають працювати.

Шаблон – це конструкція, яка створює звичайний тип або функцію під час компіляції на основі аргументів, які користувач надає для параметрів шаблону. Наприклад, можна визначити шаблон функції так:

template <typename T>

T minimum(const T& lhs, const T& rhs)

{

return lhs < rhs ? lhs : rhs;

}

Ми пишемо загальну функцію, яку можна використовувати для різних типів даних. Прикладами шаблонів функцій є sort(), max(), min(), printArray().

Як і шаблони функцій, шаблони класів корисні, коли клас визначає щось, що не залежить від типу даних. Може бути корисним для таких класів, як LinkedList, BinaryTree, Stack, Queue, Array тощо.

Як і звичайні параметри, ми можемо вказати аргументи за замовчуванням для шаблонів.

template <class T, class U = char> class A {

public:

T x;

U y;

A() { cout << "Constructor Called" << endl; }

};

Шаблони функцій дозволяють програмісту писати загальну функцію, яка не залежить від типу даних.

Використовуючи шаблони функцій, ми можемо зменшити розмір коду та полегшити обслуговування коду.

template <class T>

<return-type> <function-name> ( <parameters of type T> )

{

//function body

}

**Лабораторне завдання**

Створити шаблон класу та продемонструвати його роботу за індивідуальним варіантом. Оформити звіт до лабораторної роботи. Звіт має містити варіант завдання, код розробленої програми, результати роботи програми (скріншоти), висновок.

**Варіант**

2. Створити шаблон класу Array, який містить однотипні елементи. Шаблон класу повинен давати можливість вивести всі елементи на екран, відсортувати всі елементи в порядку зростання та спадання, а також мінімальний з елементів. Продемонструвати функціонал шаблону на створеному користувацькому типі String – символьна стрічка. Для порівняння стрічок використовувати алфавітний порядок.

**Хід роботи**

1. Реалізовую програму:

**arrayt.h**

#ifndef ARRAYT\_H

#define ARRAYT\_H

#include <QTableWidget>

#include <mystring.h>

template <typename T>

class ArrayT

{

public:

ArrayT();

ArrayT(int size, T arr[]){

m\_arr = new T[size];

m\_arr = arr;

this->size = size;

}

ArrayT(const ArrayT& a){

size = a.size;

if(a.m\_arr){

m\_arr = new T[size];

for(int i = 0; i < size; i++){

m\_arr[i] = a.m\_arr[i];

}

}else m\_arr = 0;

}

~ArrayT(){

delete [] m\_arr;

qDebug() << "Destructor called";}

void show(QTableWidget& table);

void sortUp();

void sortDown();

T getmin();

T& getByIndex(int index);

QString toQString(T){return QString(m\_arr);}

ArrayT operator=(const ArrayT& other)

{

if(this == &other){

return \*this;

}

delete [] m\_arr;

if(other.m\_arr){

size = other.size;

m\_arr = new T[size];

for(int i = 0; i < size; i++){

m\_arr[i] = other.m\_arr[i];

}

}

}

private:

int size;

T\* m\_arr;

};

template<typename T>

void ArrayT<T>::show(QTableWidget& table){

table.clear();

table.setRowCount(0);

for(int i = 0;i < size; i++){

table.insertRow(i);

table.setItem(table.rowCount()-1,

0,

(new QTableWidgetItem(m\_arr[i].getStr())));

}

}

template <typename T>

T ArrayT<T>::getmin(){

T min = m\_arr[0];

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

if(m\_arr[i].getStr() < min.getStr()) min = m\_arr[i];

return min;

}

template<typename T>

T &ArrayT<T>::getByIndex(int index)

{

return m\_arr[index];

}

template<typename T>

void ArrayT<T>::sortUp(){

T temp;

for(int i = 0; i < size; i++){

{

for(int j=i+1;j<size;j++)

{

if(m\_arr[i].getStr() > m\_arr[j].getStr())

{

temp =m\_arr[i];

m\_arr[i]=m\_arr[j];

m\_arr[j]=temp;

}

}

}

}

}

template<typename T>

void ArrayT<T>::sortDown(){

T temp;

for(int i = 0; i < size; i++){

for(int j=i+1;j<size;j++)

{

if(m\_arr[i].getStr() < m\_arr[j].getStr())

{

temp =m\_arr[i];

m\_arr[i]=m\_arr[j];

m\_arr[j]=temp;

}

}

}

}

//template <typename T>

//const ArrayT<T>& operator[](const ArrayT<T> index){

// return m\_arr[index].getStr();

//}

#endif // ARRAYT\_H

**mystring.h**

#ifndef MYSTRING\_H

#define MYSTRING\_H

#include <QString>

class MyString

{

public:

MyString(){str = 0;}

MyString(QString s){str = s;}

QString getStr(){return str;}

void setString(QString a){str = a;}

private:

QString str;

};

#endif // MYSTRING\_H

**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\_Insert\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <mystring.h>

#include <arrayt.h>

#include <QString>

#include <QMessageBox>

#include <QPixmap>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->textEdit->setFocus();

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_Insert\_clicked()

{

if(ui->tableWidget->rowCount()>4){

ui->tableWidget->setGeometry(130, 140, 256, 300);

}

int count = 0;

ui->tableWidget->insertRow(count);

ui->tableWidget->setItem(ui->tableWidget->currentRow(), count+1, new QTableWidgetItem(ui->textEdit->toPlainText()));

ui->textEdit->clear();

ui->textEdit->setFocus();

count++;

}

void MainWindow::on\_pushButton\_clicked()

{

MyString strings[ui->tableWidget->rowCount()];

for(int i = 0; i < ui->tableWidget->rowCount(); i++){

strings[i] = QString(ui->tableWidget->item(i, 0)->text());

}

ArrayT<MyString> A(ui->tableWidget->rowCount(), strings);

A.sortUp();

A.show(\*ui->tableWidget);

}

void MainWindow::on\_pushButton\_2\_clicked()

{

MyString strings[ui->tableWidget->rowCount()];

for(int i = 0; i < ui->tableWidget->rowCount(); i++){

strings[i] = QString(ui->tableWidget->item(i, 0)->text());

}

ArrayT<MyString> A(ui->tableWidget->rowCount(), strings);

if(A.getmin().getStr() == "4"){

QMessageBox hi;

hi.setIconPixmap(QPixmap(":/pics/150.jpg"));

hi.exec();

}else

ui->label->setText(A.getmin().getStr());

}

void MainWindow::on\_pushButton\_3\_clicked()

{

MyString strings[ui->tableWidget->rowCount()];

for(int i = 0; i < ui->tableWidget->rowCount(); i++){

strings[i] = QString(ui->tableWidget->item(i, 0)->text());

}

ArrayT<MyString> A(ui->tableWidget->rowCount(), strings);

A.sortDown();

A.show(\*ui->tableWidget);

}

void MainWindow::on\_pushButton\_4\_clicked()

{

MyString strings[ui->tableWidget->rowCount()];

for(int i = 0; i < ui->tableWidget->rowCount(); i++){

strings[i] = QString(ui->tableWidget->item(i, 0)->text());

}

ArrayT<MyString> A(ui->tableWidget->rowCount(), strings);

int index = ui->textEdit\_2->toPlainText().toInt();

if(index < ui->tableWidget->rowCount()+1 && index > 0){

ui->label\_2->setText(A.getByIndex(index-1).getStr());

}

else {

QMessageBox hi;

hi.setIconPixmap(QPixmap(":/pics/indexerror.png"));

hi.exec();

}

}

**Виконання лабораторної роботи**

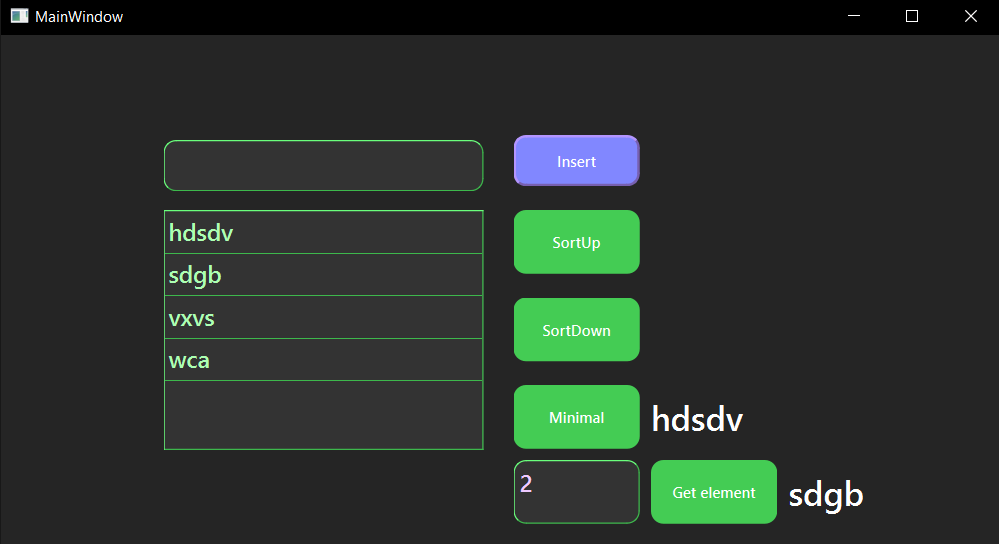


Рис. 1 Виконання програми

**Висновки**

На цій лабораторній роботі я навчився створювати шаблони класу, екземпляри шаблонів та використовувати шаблони з користувацьким типом.