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.

Виконала: ст.гр. ПЗ-23

Кохман О.В.

Прийняла: доцент каф. ПЗ

Коротєєва Т.О.

«_____»_____2022р.

∑ ____ _____.

Львів – 2022

Тема: шаблони класів.

Мета: навчитись створювати шаблони класів та екземпляри шаблонів.

Теоретичні відомості

У мові C++ **шаблони функцій — це функції, які служать взірцем для створення інших подібних функцій**. Головна ідея — створення функцій без вказівки точного типу(ів) деяких або всіх змінних. Для цього ми визначаємо функцію, вказуючи **тип параметра шаблону**, який використовується замість будь-якого типу даних. Після того, як ми створили функцію з типом параметра шаблону, ми фактично створили «трафарет функції».

Детальне оголошення параметрів шаблону:

Спочатку пишемо **ключове слово template**, яке повідомляє компілятору, що далі ми будемо оголошувати параметри шаблону.

Параметри шаблону функції вказуються в кутових дужках (<>).

Для створення типів параметрів шаблону використовуються **ключові слова typename і class**. В базових випадках використання шаблонів функцій різниці між typename і class немає, тому ви можете вибрати будь-яке з двох. Якщо ви використовуєте ключове слово class, то фактичний тип параметрів не обов'язково повинен бути класом (це може бути змінна фундаментального типу даних, вказівник або щось інше).

Потім даємо назву типу параметра шаблону (зазвичай T).

Шаблони класів працюють точно так же, як і шаблони функцій: компілятор копіює шаблон класу, замінюючи типи параметрів шаблону класу на фактичні (передані) типи даних, а потім компілює цю копію. Якщо у вас є шаблон класу, але ви його не використовуєте, то компілятор не буде його навіть компілювати.

Шаблони класів ідеально підходять для реалізації контейнерних класів, тому що дуже часто таким класам доводиться працювати з різними типами даних, а шаблони дозволяють це організувати в мінімальній кількості коду. Хоча синтаксис трохи потворний, і повідомлення про помилки іноді можуть бути «об'ємними», шаблони класів дійсно є однією з кращих і найбільш корисних конструкцій мови C++.

Індивідуальне завдання

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.

Код програми

Назва файлу: SomeArray.h

```
#ifndef SOMEARRAY_H
#define SOMEARRAY_H
#pragma once
#include <iostream>
#include <algorithm>
#include <random>
#include "SomeString.h"
template <typename T>
class SomeArray {
private:
	T* array;
	int size;
public:
	SomeArray();
	SomeArray(int size);
	SomeArray(T* array, int size);
	~SomeArray();
	void printArray();
	void sortAscending();
	void sortDescending();
	T min();
};
#endif
```

Назва файлу: SomeArray.cpp

```
#include "SomeArray.h"
template <typename T>
SomeArray<T>::SomeArray() {
	this->size = 0;
	array = new T[size];
}
template <typename T>
SomeArray<T>::SomeArray(int size) {
	this->size = size;
	array = new T[size];
}
template <typename T>
SomeArray<T>::SomeArray(T* _array, int size) {
	this->size = size;
	array = new T[size];
	for (int i = 0; i < size; i++) {
		this->array[i] = _array[i];
	}
}
template <typename T>
```

```
SomeArray<T>::~SomeArray() {
	delete[] array;
}
template <typename T>
void SomeArray<T>::printArray() {
	for (int i = 0; i < size; i++) {
		std::cout << array[i] << " ";
	}
	std::cout << "\n";
}
template <typename T>
void SomeArray<T>::sortAscending() {
	int length = size;
	while (length >= 0) {
		for (int i = 1; i < size; i++) {
			if (array[i - 1] > array[i]) {
				T temp = array[i];
				array[i] = array[i - 1];
				array[i - 1] = temp;
			}
		}
		length--;
	}
}
template <typename T>
void SomeArray<T>::sortDescending() {
	int length = size;
	while (length >= 0) {
		for (int i = 1; i < size; i++) {
			if (array[i - 1] < array[i]) {
				T temp = array[i];
				array[i] = array[i - 1];
				array[i - 1] = temp;
			}
		}
		length--;
	}
}
template <typename T>
T SomeArray<T>::min() {
	T min = array[0];
	for (int i = 0; i < size; i++) {
		if (array[i] < min) {
			min = array[i];
		}
	}
	return min;
}
```

Назва файлу: SomeString.h

```
#ifndef SOMESTRING_H
#define SOMESTRING_H
#pragma once
#include <cstring>
#include <iostream>
class SomeString {
private:
	char* array;
	int length;
public:
	SomeString();
	SomeString(int length);
```

```
	SomeString(char* array, int length);
	friend bool operator<(const SomeString& first, const SomeString& second);
	friend bool operator>(const SomeString& first, const SomeString& second);
	friend std::ostream& operator<<(std::ostream& out, const SomeString&
object);
};
#endif
```

Назва файлу: SomeString.cpp

```
#include "SomeString.h"
SomeString::SomeString() {
	int length = 0;
	char* array = new char[length];
}
SomeString::SomeString(int length) {
	this->length = length;
	array = new char[length];
}
SomeString::SomeString(char* _array, int length) {
	this->length = length;
	this->array = new char[length];
	for (int i = 0; i < length; i++) {
		array[i] = _array[i];
	}
}
bool operator< (const SomeString& first, const SomeString& second) {
	if (strcmp(first.array, second.array) < 0) {
		return true;
	}
	else {
		return false;
	}
}
bool operator>(const SomeString& first, const SomeString& second) {
	if (strcmp(first.array, second.array) > 0) {
		return true;
	}
	else {
		return false;
	}
}
std::ostream& operator<<(std::ostream& out, const SomeString& object) {
	for (int i = 0; i < object.length; i++) {
		out << object.array[i];
	}
	return out;
}
```

Назва файлу: main.cpp

```
int main(int args, char* argv) {
	int size = 5;
	char one[] = "first";
	char two[] = "second";
	char three[]  = "third";
	char four[] = "fourth";
	char five[] = "fifth";
	SomeString first = SomeString(one, sizeof(one) / sizeof(char));
	SomeString second = SomeString(two, sizeof(two) / sizeof(char));
	SomeString third = SomeString(three, sizeof(three) / sizeof(char));
	SomeString fourth = SomeString(four, sizeof(four) / sizeof(char));
	SomeString fifth = SomeString(five, sizeof(five) / sizeof(char));
```

```
        SomeString* array = new SomeString[size]{ first, second, third , fourth,
fifth};
        SomeArray<SomeString> object(array, size);
        object.printArray();
        std::cout << object.min() << std::endl;
        object.sortAscending();
        object.printArray();
        object.sortDescending();
        object.printArray();
        return 0;
}
```

Протокол роботи

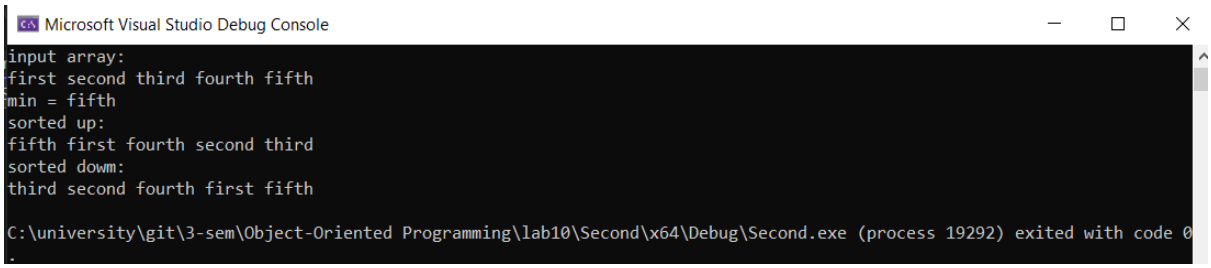

Рис. 1 Результат роботи програми

Висновок

На цій лабораторній роботі я навчилась створювати шаблони класів, використовувати їх та створювати їх екземпляри та продемонструвала результати роботи у консолі в Visual Studio 2022.