Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра “Електронних обчислювальних машин”



**Звіт з лабораторної роботи №8**

на тему:

**“ Шаблони”**

**Виконала:**

ст. гр. КІ-15

Аркавенко Л.С.

**Перевірив:**

асист. каф. ЕОМ

Козак Н. Б.

Львів – 2020

**Мета**: познайомитися із створенням шаблонів.

**Теоритичні відомості:**

Шаблони являють собою схематичний опис побудови класів та функцій. Використовуючи шаблони, з'являється можливість створювати узагальнені специфікації для класів та функцій, що найчастіше носять назву параметризованих класів (generic classes) та параметризованих функцій (generic functions).

Шаблони не прив‘язані до конкретних типів даних і описують алгоритми, незалежно від типів даних. Дані алгоритми мають функціонувати однаково для різних типів даних. Такий опис дозволяє описати один раз функції, методи чи класи і на їх базі генерувати функції, методи і класи для кожного конкретного набору параметрів, що економить зусилля і час розробки програмного забезпечення. Після визначення загального шаблона, якщо для одного, кількох або всіх параметрів поведінка класу чи функції відрізнятиметься від описаної в загальному шаблоні, то створюється спеціалізація для конкретного набору параметрів. Спеціалізація може бути звичайною (неявною), явною або частковою.

Призначенням шаблонів є створення екземплярів (instantiating) шаблону, які вже є реальними функціями чи класами. При цьому відбувається прив'язування параметрів шаблону до даних визначеного типу. Цей процес називається конкретизацією. Cпроба компілятором створити екземпляр шаблону є генерацією програми. Тому зустрічаючи спробу створити екземпляр шаблону компілятор перемикається в режим його вивчення та запам'ятовування, а це - часові витирати. Типи загального призначення, якими оперують шаблони, називаються шаблонними типами (template type), а їх сукупність параметрами шаблона (template parameters). Параметри шаблону як множина шаблонних типів може містити також преозначені і вбудовані типи С++. Шаблонний тип Т є невизначеним узагальненим типом. По мірі використання шаблонів компілятор автоматично замінить тип Т іменем реального типу. Як правило, для імені шаблонного типу використовують ідентифікатори T чи Type. Проте це не обов'язково: ім'я можна декларувати будь-яким допустимим в С++ ідентифікатором. Шаблонний тип можна повноцінно використовувати в тілі шаблону, але це не є строгою вимогою.

Шаблон допускає використання параметрів, які ініціалізуються аргументами за замовчуванням, згідно з методологією оголошення і використання таких аргументів. Типи аргументів по замовчуванні можуть бути лише преозначеними або вбудованими. Використання шаблонних типів як аргументів по замовчуванню не допускається.

Під шаблон пам'ять не виділяється. Якщо екземпляр шаблону не створюється, то компілятор навіть не буде транслювати код шаблону. Це спричинює труднощі з використанням файлів заголовків, які містять лише оголошення шаблонів, а їх реалізація знаходиться у сpp-файлі. Для подолання цих недоліків треба підключати сpp-файл, а не файл заголовку, або код шаблону вносити у файл заголовку. Ранні версії С++ компіляторів не перевіряли синтаксис тіла незалежно від створення екземпляр шаблону. Сучасні компілятори відразу аналізують синтаксис коду тіла при першому знаходженні оголошення шаблону, а тому позбавлені цих недоліків.

Використання шаблонів може значно скоротити час створення програми. Це досягається тим, що з'являється можливість перенести незалежний від типу даних, один раз написаний і перевірений код спільний для множини різнорідних функцій в одну програмну конструкцію - шаблон.

Основним застереженням при роботі з шаблонами є правильне використання операцій до змінних шаблонних типів. Тобто усі операції, які використовувались до змінної шаблонної типу, повинні мати місце для того типу, яким буде заміщений даний шаблонний тип. Таким чином, за допомогою реалізації узагальнених функцій можна зменшити розмір та складність програми. Особливо корисними шаблони є саме в бібліотеках класів – тут вони вказують програмісту необхідні специфікації, приховуючи при цьому деталі справжньої реалізації.

**Індивідуальне завдання:**

Контейнерний клас описує та забезпечує набір дій над даними параметризованого масиву, розмірність якого визначається під час роботи програми. Усі обчислення та перетворення повинні бути реалізовані у вигляді методів класу.

**Варіант 1**

В масиві обчислити різницю елементів масиву, що розташовані між першим від'ємним та другим додатним елементами.

Код виконання програми:

**Файл Mas.h**

#pragma once

#include <iostream>

using namespace std;

#ifndef Myc\_ls

#define Myc\_ls

template <typename T>

class MyClass {

private: int size;

T \*arr = new T[size];

int pos = 0, neg = 0, sch = 0;

T sum = 0;

public: ~MyClass() {

}

MyClass()

{

this->Enter\_long\_mas();

}

MyClass(int size) {

this->size = size + 1;

}

void Enter\_long\_mas() {

cout << "Enter the size of the array: ";

cin >> size;

size++;

}

void Enter\_mas() {

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

{

cout << "Enter elemen number " << i+1 << ": ";

cin >> arr[i];

}

}

void search() {

int sch = 0;

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

if (neg == 0) {

if (arr[i] < 0)

neg = i;

}

if (arr[i] > 0)

{

if (sch < 2) {

pos = i;

sch = sch + 1; }

}

}

}

T calculation() {

if (neg < pos) {

sum = arr[neg + 1];

for (int i = neg + 2; i < pos; i++)

{

sum = sum - arr[i];

}

}

else {

sum = arr[pos + 1];

{

for (int i = pos + 2; i < neg; i++)

{

sum = sum - arr[i];

}

}

}

return sum;

}

};

#endif

**Файл main.cpp**

#include <iostream>

#include "Mas.h"

using namespace std;

int main() {

double sum;

MyClass <double> a;

a.Enter\_mas();

a.search();

sum = a.calculation();

cout << sum;

return sum;

}

**Скріншот виконання програми:**

