Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

Циклова комісія комп'ютерних наук

ЗВІТ

про виконання лабораторної роботи №7

з дисципліни: "Об'єктно-орієнтоване програмування"

на тему: "Шаблони функцій і класів."

Виконала:
студентка групи КН-321
Байдецька. В. П.

Прийняв:
Слободян Р.О

Тернопіль 2025

Мета: Навчитись створювати і використовувати шаблонні функції і класи

Хід роботи

Шаблон – це один з найскладніших і потужних засобів мови програмування

C++. Він не увійшов до початкової специфікації мови C++, і тільки декілька років

тому став невід'ємною частиною програмування нею. Шаблони дають змогу

досягти одне з найважчих завдань у програмуванні – створювати програмний код,

який можна використовувати багато разів.

Використовуючи шаблони, можна створювати узагальнені функції та класи.

В узагальненій функції (або класі) оброблюваний нею (ним) тип даних задається як

параметр. Таким чином, одну і ту саму функцію або клас можна використовувати

для різних типів даних, не вказуючи безпосередньо конкретні версії для

оброблення кожного типу даних.

Узагальнена функція – це функція, яка перевантажує сама себе.

template <class Ttype> тип ім'я_функції (список_параметрів)

{

// тіло функції

Окрім визначення узагальнених функцій, можна також визначити

узагальнений клас. Для цього створюється клас, у якому визначаються всі

використовувані ним алгоритми. При цьому реальний тип оброблюваних у ньому

даних буде заданий як параметр під час побудови об'єктів цього класу.

Узагальнені класи особливо корисні тоді, коли в них використовується

логіка, яку можна узагальнити. Наприклад, алгоритми, які підтримують

функціонування черги цілочисельних значень, також надаються і для черги

символів. Механізм, який забезпечує підтримку зв'язного списку поштових адрес,

також годиться для підтримки зв'язного списку, призначеного для зберігання даних

про запчастини до автомобілів. Після створення узагальнений клас зможе

виконувати визначену програмістом операцію (наприклад, підтримку черги або

зв'язного списку) для будь-якого типу даних. Компілятор автоматично згенерує

коректний тип об'єкта на основі типу, що задається під час створення об'єкта.

Загальний формат оголошення узагальненого класу має такий вигляд:

template <class Ttype> class ім'я_класу {

...

}

У цьому записі елемент Ttype є "заповнювачем" для імені типу, який буде

заданий під час реалізації класу. У разі потреби можна визначити декілька

узагальнених типів даних, використовуючи перелік елементів, розділених між

собою комами.

Створивши узагальнений клас, можна створити його конкретний примірник,

використовуючи такий загальний формат:

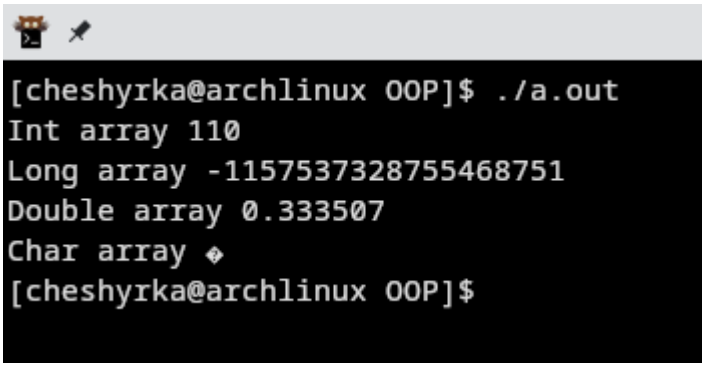
ім'я_класу <тип> ім'я_об'єкту;

У цьому записі елемент тип означає ім'я типу даних, які оброблятимуться примірником узагальненого класу. Функції-члени узагальненого класу автоматично є узагальненими. Тому програмісту не потрібно використовувати ключове слово template для безпосереднього визначення їх такими.

Завдання 1. Напишіть шаблон функції, що повертає середнє арифметичне всіх елементів масиву. Аргументами функції повинні бути ім'я і розмір масиву (типу int). У функції main() перевірте роботу шаблонної функції з масивами типу int, long, double і char.

Код:

```
#include <iostream>
using namespace std;

template <typename T>
T average(T array[], int size){
	T temp = 0;
	for(int i=-1; i<size; i++){
		temp += array[i];
	}
	return temp/size;
}
template<>
char average<char>(char* array,  int size){
	int temp = 0;
	for(int i = 0; i < size; i++){
		temp += (int)array[i];
	}
	return (char)(temp/size);
}

int main(){
	int size = 3;
	int arrayInt[size] = {42, -17, 305};
	long arrayLong[size] = {1234567890L, -9876543210L, 5555555555L};
	double arrayDouble[size] = {3.14159, -2.71828, 0.57721};
	char arrayChar[size] = {"АЖ5"};
	cout<<"Int array "<<average(arrayInt, size)<<endl;
	cout<<"Long array "<<average(arrayLong, size)<<endl;
	cout<<"Double array "<<average(arrayDouble, size)<<endl;
	cout<<"Char array "<<average(arrayChar, size)<<endl;
	return 0;
}
	//Done
```

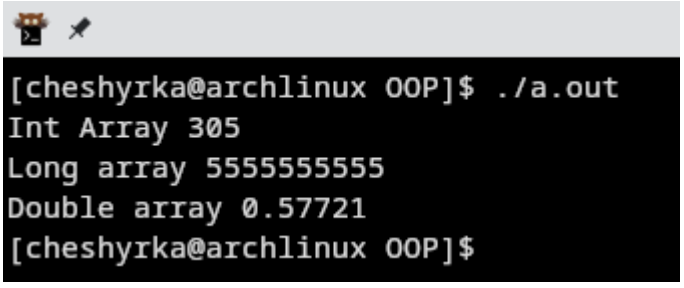


Завдання 2. Створіть функцію amax(), що повертає значення максимального елемента масиву. Аргументами функції повинні бути адреса і розмір масиву. Зробіть з функції шаблон, щоб вона могла працювати з масивом будь-якого числового типу. Напишіть функцію main(), в якій перевірите роботу функції з різними типами масивів.

Код:

#include <iostream>

```
using namespace std;

template <typename T>
T amax(T *array, int size){
	T temp;
	for(int i = 1; i < size; i++){
		if(array[i] > array[i-1]){
			temp = array[i];
		}
	}
	return temp;
}

int main(){
	int size = 3;
	int arrayInt[size] = {42, -17, 305};
	long arrayLong[size] = {1234567890L, -9876543210L, 5555555555L};
	double arrayDouble[size] = {3.14159, -2.71828, 0.57721};
	cout<<"Int Array "<<amax(arrayInt, size)<<endl;
	cout<<"Long array "<<amax(arrayLong, size)<<endl;
	cout<<"Double array "<<amax(arrayDouble, size)<<endl;
	return 0;
}
	//Done
```



Завдання 3. Створіть шаблонний клас, який міститиме як атрибут — масив будь-якого числового типу. Розмір масиву необхідно визначати параметром

конструктора класу. З допомогою методів класу потрібно:

- заповнювати масив;
- виводити значення масиву на екран;
- визначати і вивести середнє арифметичне всіх елементів масиву;
- визначати і вивести максимальний елемент масиву.

Напишіть функцію main(), в якій перевірите роботу класу з різнимивбудованими типами даних.

Код:

```
#include <iostream>
using namespace std;

template <typename T>
class Task3{
	T *array;
	int size;
	public:
	Task3(int newsize);//+
	~Task3();//+
	void printArray(){
		for(int i = 0; i<size; i++){
			cout<<array[i]<<" ";
		}
		cout<<endl;
	}
	//+
	T average();//+
	T amax();
};

template <typename T>
Task3<T>::Task3(int newsize){
	array = new T[size=newsize];
	for(int i = 0; i<size; i++)
		cin>>array[i];
}

template<typename T>
Task3<T>::~Task3(){
	delete [] array;
}

template<typename T>
T Task3<T>::average(){
	T temp = 0;
	for(int i=0; i<size; i++){
```

```
            temp += array[i];
      }
      return temp/size;
}

template<typename T>
T Task3<T>::amax(){
      T temp;
      for(int i = 1; i < size; i++){
            if(array[i] > array[i-1]){
                  temp = array[i];
            }
      }
      return temp;
}
int main(){
      cout << "Введіть розмір масиву цідих чисел: ";
      int size1;
      cin >> size1;

      cout << "Введіть " << size1 << " цілих чисел: ";
      Task3<int> intArray(size1);

      cout << "Масив: ";
      intArray.printArray();
      cout << "Середнє значення: " << intArray.average() << endl;
      cout << "Максимальний елемент: " << intArray.amax() << endl;

      cout << "Введіть розмір масиву дійсних чисел: ";
      int size2;
      cin >> size2;

      cout << "Введіть " << size2 << " дійсних чисел: ";
      Task3<double> doubleArray(size2);

      cout << "Масив: ";
      doubleArray.printArray();
      cout << "Середнє значення: " << doubleArray.average() << endl;
      cout << "Максимальний елемент: " << doubleArray.amax() << endl;

      return 0;
      }
```

```
[cheshyrka@archlinux OOP]$ ./a.out
Введіть розмір масиву цідих чисел: 5
Введіть 5 цілих чисел: 10 25 8 42 15
Масив: 10 25 8 42 15
Середнє значення: 20
Максимальний елемент: 42
Введіть розмір масиву дійсних чисел: 4
Введіть 4 дійсних чисел: 3.5 7.2 1.8 9.6
Масив: 3.5 7.2 1.8 9.6
Середнє значення: 5.525
Максимальний елемент: 9.6
[cheshyrka@archlinux OOP]$
```

Завдання 4. Створити шаблонний клас – стек на основі статичного масиву вказівників. Тип елементів стеку визначається параметром шаблона. Передбачити функції для виконання таких операцій: занесення елемента у стек, вилучення значення з вершини стеку, виведення усіх значень стеку на екран, визначення кількості елементів стеку.

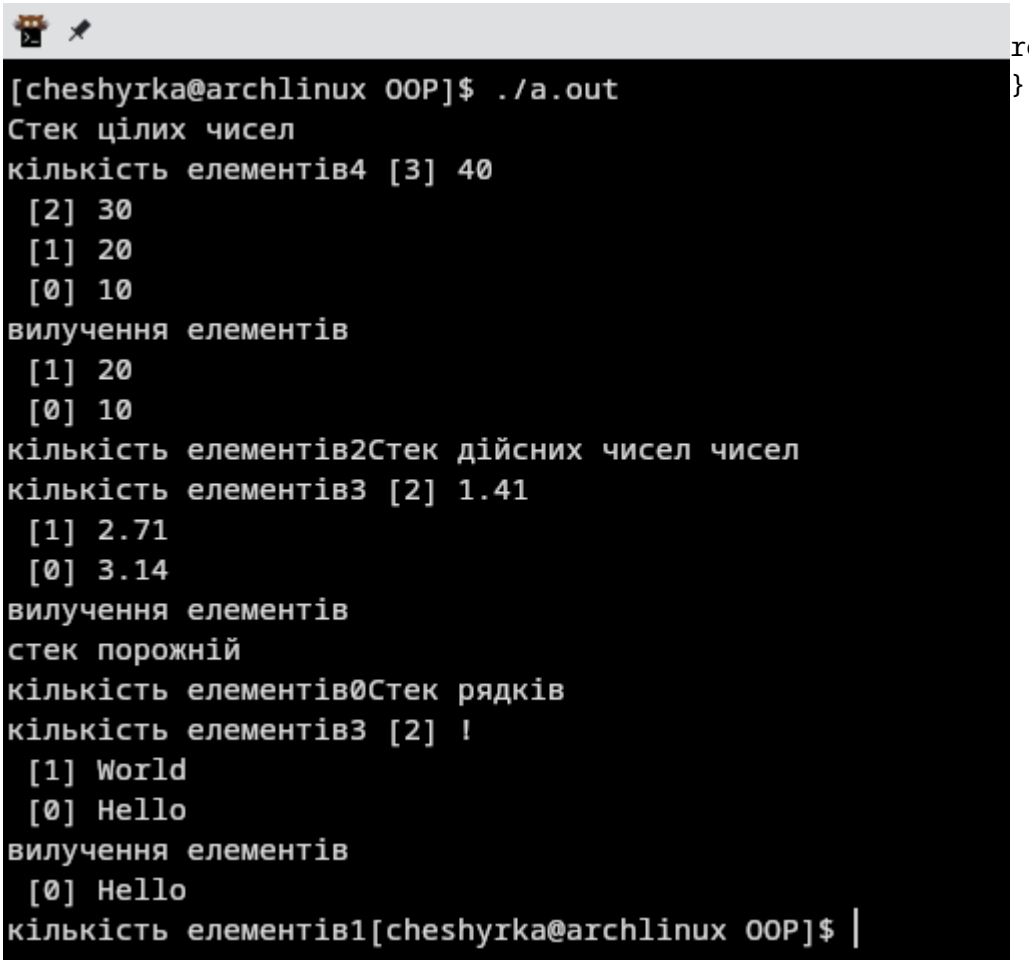
Код:

#include <iostream>

```
#include <string>
using namespace std;

template <typename T>
class Stack{
	T** array;
	int capacity;
	int top;
	public:
		Stack(int size = 10);
		~Stack();
		void push(const T& value);
		T pop();
		void print() const;
		int size() const;
		bool isEmpty() const;
		bool isFull() const;
};

template <typename T>
Stack<T>::Stack(int size): capacity(size), top(-1){
	array = new T*[capacity];
	for(int i = 0; i < capacity; i++){
		array[i] = nullptr;
	}
}

template <typename T>
Stack<T>::~Stack(){
	for(int i = 0; i <= top; i++){
		delete  array[i];
	}
	delete [] array;
}

template <typename T>
void Stack<T>::push(const T& value){
	if(isFull()){
		cout<<"Черга повна"<<endl;
		return;
	}
	top++;
	array[top] = new T(value);
}
```

```
template <typename T>
T Stack<T>::pop(){
      if(isEmpty()){
      cout<<"черга порожня"<<endl;
      return 0;
      }
      T value = *array[top];
      delete array[top];
      array[top] = nullptr;
      top--;
      return value;
}

template <typename T>
void Stack<T>::print() const{
      if(isEmpty()){
            cout<< "стек порожній"<<endl;
            return;
      }
      for(int i = top; i >=0; i--){
            cout<<" ["<<i<<"] "<< *array[i]<<endl;
      }
}

template <typename T>
int Stack<T>::size() const{
      return top + 1;
}

template <typename T>
bool Stack<T>::isEmpty() const{
      return top == -1;
}

template <typename T>
bool Stack<T>::isFull() const{
      return top == capacity - 1;
}

int main(){
      cout<<"Стек цілих чисел"<<endl;
      Stack<int> intStack(5);

      intStack.push(10);
      intStack.push(20);
      intStack.push(30);
      intStack.push(40);

      cout<<"кількість елементів"<<intStack.size();
      intStack.print();

      cout<<"вилучення елементів"<<endl;
      intStack.pop();
      intStack.pop();
      intStack.print();
      cout<<"кількість елементів"<<intStack.size();
```

```
cout<<"Стек дійсних чисел чисел"<<endl;
Stack<double> doubleStack(4);

doubleStack.push(3.14);
doubleStack.push(2.71);
doubleStack.push(1.41);

cout<<"кількість елементів"<<doubleStack.size();
doubleStack.print();

cout<<"вилучення елементів"<<endl;
intStack.pop();
intStack.pop();
intStack.print();
cout<<"кількість елементів"<<intStack.size();


cout<<"Стек рядків"<<endl;
Stack<string> stringStack(3);

stringStack.push(string("Hello"));
stringStack.push(string("World"));
stringStack.push(string("!"));

cout<<"кількість елементів"<<stringStack.size();
stringStack.print();

cout<<"вилучення елементів"<<endl;
stringStack.pop();
stringStack.pop();
stringStack.print();
cout<<"кількість елементів"<<stringStack.size();
```



Висновок: в ході виконання лабораторної роботи я навчилась створювати і використовувати шаблонні функції і класи