Чернівецький національний університет імені Юрія Федьковича

Навчально-науковий інститут фізико-технічних та комп’ютерних наук

Кафедра математичних проблем управління і кібернетики

Звіт

про виконання лабораторної роботи №7

«Параметризовані типи. Шаблони функції. Шаблони класів. Ітератори.»

з дисципліни

«Об’єктно-орієнтоване програмування»

Виконала: студентка 141-А+Б групи Жданюк А.

Перевірив: канд.фіз.-мат. наук,

доцент Лазорик В.В.

Оцінка:

Дата захисту:

Чернівці 2024

**Тема:Параметризовані типи. Шаблони функції. Шаблони класів. Ітератори.**

Виконання лабораторної роботи

1. Зайти в свій обліковий запис на github.com. Зайти в github classroom.

2. Клонувати репозиторій <https://classroom.github.com/a/7BE-gIB9>

на робочий комп’ютер. В репозиторії знаходиться в функції main()

викликається демонстраційний приклад виконання подібної лабораторної роботи його замінити, який можна використовувати (модифікувати та використовувати) для розв’язання задачі власних задач.

3. Розробити параметризовані функції або класи згідно варіантів.

4. Реалізувати створені шаблони для декількох типів. Передбачити введення початкових даних: з клавіатури, файлу та використовуючи датчик випадкових чисел.

5. В процесі написання функцій розв’язання задач лабораторної роботи періодично здійснювати синхронізацію з репозиторієм на github.com.

6. Оформити звіт про виконання лабораторної роботи. При захисті роботи студент усно повинен знати відповіді на контрольні питання.

7. Звіт відправити в для оцінювання в [https://moodle.chnu.edu.ua](https://moodle.chnu.edu.ua/).

**Мої варіанти задач**:

1.Описати функції-шаблони для знаходження мінімального та

максимального елемента пари чисел. Написати специфікацію функціїшаблон для типу char\*.

2.Написати функцію-шаблон функцію впорядкування методом

«Вилучення». Написати специфікацію функції-шаблон для типу char\*

3.Створити параметризованний клас однозв'язного списку.

4.Написати програму, яка тестує створений ітератор. Побудувати

клас, що описує масив. Побудувати клас ітератор, що дозволяє проходити масив. Написати програму, яка тестує створений ітератор

**Завдання 1**



#include <iostream>

#include <utility>

#include <cstring>

// Шаблон функції для знаходження мінімального елемента

template <typename T>

T min(T a, T b) {

return (a < b) ? a : b;

}

// Шаблон функції для знаходження максимального елемента

template <typename T>

T max(T a, T b) {

return (a > b) ? a : b;

}

// Спеціалізована функція-шаблон для типу char\*

template <>

const char\* min<const char\*>(const char\* a, const char\* b) {

return (strcmp(a, b) < 0) ? a : b;

}

// Спеціалізована функція-шаблон для типу char\*

template <>

const char\* max<const char\*>(const char\* a, const char\* b) {

return (strcmp(a, b) > 0) ? a : b;

}

int main() {

// Приклад використання для цілих чисел

int int1 = 10, int2 = 20;

std::cout << "Min of " << int1 << " and " << int2 << " is: " << min(int1, int2) << std::endl;

std::cout << "Max of " << int1 << " and " << int2 << " is: " << max(int1, int2) << std::endl;

// Приклад використання для рядків

const char\* str1 = "apple";

const char\* str2 = "banana";

std::cout << "Min of \"" << str1 << "\" and \"" << str2 << "\" is: " << min(str1, str2) << std::endl;

std::cout << "Max of \"" << str1 << "\" and \"" << str2 << "\" is: " << max(str1, str2) << std::endl;

return 0;

}

**Завдання 2**



#include <iostream>

#include <cstring>

template <typename T>

void selectionSort(T\* array, size\_t size) {

for (size\_t i = 0; i < size - 1; ++i) {

size\_t minIndex = i;

for (size\_t j = i + 1; j < size; ++j) {

if (array[j] < array[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

std::swap(array[i], array[minIndex]);

}

}

}

// Спеціалізація для типу char\*

template <>

void selectionSort(const char\*\* array, size\_t size) {

for (size\_t i = 0; i < size - 1; ++i) {

size\_t minIndex = i;

for (size\_t j = i + 1; j < size; ++j) {

if (strcmp(array[j], array[minIndex]) < 0) {

minIndex = j;

}

}

if (minIndex != i) {

std::swap(array[i], array[minIndex]);

}

}

}

int main() {

int intArray[] = {3, 1, 4, 1, 5, 9, 2, 6, 5};

const char\* strArray[] = {"apple", "banana", "orange", "grape"};

selectionSort(intArray, sizeof(intArray) / sizeof(int));

selectionSort(strArray, sizeof(strArray) / sizeof(const char\*));

std::cout << "Sorted Integer Array: ";

for (int i : intArray) {

std::cout << i << " ";

}

std::cout << std::endl;

std::cout << "Sorted String Array: ";

for (const char\* str : strArray) {

std::cout << str << " ";

}

std::cout << std::endl;

return 0;

}

**Завдання 3**



#include <iostream>

template<typename T>

class LinkedListNode {

public:

T data;

LinkedListNode\* next;

LinkedListNode(const T& value) : data(value), next(nullptr) {}

};

template<typename T>

class LinkedList {

private:

LinkedListNode<T>\* head;

public:

LinkedList() : head(nullptr) {}

~LinkedList() {

LinkedListNode<T>\* current = head;

while (current != nullptr) {

LinkedListNode<T>\* next = current->next;

delete current;

current = next;

}

}

void insert(const T& value) {

LinkedListNode<T>\* newNode = new LinkedListNode<T>(value);

if (head == nullptr) {

head = newNode;

} else {

LinkedListNode<T>\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void display() {

LinkedListNode<T>\* current = head;

while (current != nullptr) {

std::cout << current->data << " ";

current = current->next;

}

std::cout << std::endl;

}

};

int main() {

// Приклад використання параметризованого класу LinkedList з типом int

LinkedList<int> intList;

intList.insert(1);

intList.insert(2);

intList.insert(3);

intList.display();

// Приклад використання параметризованого класу LinkedList з типом std::string

LinkedList<std::string> stringList;

stringList.insert("Hello");

stringList.insert("world");

stringList.display();

return 0;

}

**Завдання 4**



#include <iostream>

template <typename T>

class Array {

private:

T\* data;

size\_t size;

public:

// Конструктор за замовчуванням

Array(size\_t s) : size(s) {

data = new T[size];

}

// Деструктор

~Array() {

delete[] data;

}

// Оператор індексації

T& operator[](size\_t index) {

return data[index];

}

// Клас ітератора

class Iterator {

private:

T\* ptr;

public:

// Конструктор

Iterator(T\* p) : ptr(p) {}

// Перевантажений оператор розіменування

T& operator\*() {

return \*ptr;

}

// Перевантажений оператор інкремента

Iterator& operator++() {

++ptr;

return \*this;

}

// Перевантажений оператор порівняння

bool operator!=(const Iterator& other) const {

return ptr != other.ptr;

}

};

// Початковий ітератор

Iterator begin() {

return Iterator(data);

}

// Кінцевий ітератор

Iterator end() {

return Iterator(data + size);

}

};

int main() {

Array<int> arr(5);

// Заповнення масиву

for (int i = 0; i < 5; ++i) {

arr[i] = i + 1;

}

// Виведення масиву за допомогою ітератора

std::cout << "Array elements: ";

for (Array<int>::Iterator it = arr.begin(); it != arr.end(); ++it) {

std::cout << \*it << " ";

}

std::cout << std::endl;

return 0;

}