**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И. РАЗЗАКОВА**

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**ОТЧЕТ**

**по лабораторной работе №8**

**Выполнила:** студент группы ПИ-3-21

Джанышбекова Акмарал

**Проверила:** Мусабаев Э. Б.

**Бишкек 2024**

**Тема: Шаблоны**

**Задание №1**

Создать шаблон функции, которая находит значение максимального элемента массива. Аргументами функции должны быть имя и размер массива (типа int). Создать шаблон функции, осуществляющей сортировку данных массива. Аргументами функции должны быть имя и размер массива (типа int). Продемонстрировать работу шаблонов на данных различных типов.

include <iostream>

#include <iomanip>

#include <algorithm>

#include <limits>

using namespace std;

// Шаблон функции для нахождения максимального элемента массива

template <typename T>

T findMaxElement(const T\* array, int size) {

T maxElement = array[0];

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

if (array[i] > maxElement + std::numeric\_limits<T>::epsilon()) {

maxElement = array[i];

}

}

return maxElement;

}

// Шаблон функции для сортировки массива

template <typename T>

void sortArray(T\* array, int size) {

sort(array, array + size, [](const T& a, const T& b) {

return a + std::numeric\_limits<T>::epsilon() < b;

});

}

int main() {

setlocale(LC\_ALL, "Ru");

// Ввод размера массива для целых чисел

int intSize;

cout << "Введите размер массива целых чисел: ";

cin >> intSize;

// Ввод значений для целых чисел

int\* intArray = new int[intSize];

cout << "Введите значения массива целых чисел: ";

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

cin >> intArray[i];

}

cout << "Максимальный элемент в intArray: " << findMaxElement(intArray, intSize) << std::endl;

sortArray(intArray, intSize);

cout << "Отсортированный intArray: ";

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

cout << intArray[i] << " ";

}

cout << std::endl;

delete[] intArray; // Освобождение памяти для целых чисел

// Ввод размера массива для чисел с плавающей точкой (double)

int doubleSize;

cout << "Введите размер массива с плавающей точкой: ";

cin >> doubleSize;

// Ввод значений для чисел с плавающей точкой

double\* doubleArray = new double[doubleSize];

cout << "Введите значения массива с плавающей точкой: ";

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

cin >> doubleArray[i];

}

cout << "Максимальный элемент в doubleArray: " << findMaxElement(doubleArray, doubleSize) << endl;

sortArray(doubleArray, doubleSize);

cout << "Отсортированный doubleArray: ";

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

cout << doubleArray[i] << " ";

}

cout << endl;

delete[] doubleArray; // Освобождение памяти для чисел с плавающей точкой

return 0;

}

**Задание №2**

Создать шаблон класса, у которого есть 2 поля в виде одномерных массивов одного размера. Заполнить массивы с помощью генератора случайных чисел. Найти сумму элементов этих массивов. Продемонстрировать работу шаблонов на данных различных типов.

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

template <typename T, int size>

class ArraySum {

public:

// Конструктор, который заполняет массивы случайными числами

ArraySum() {

srand(time(nullptr));

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

// array1[i] = static\_cast<T>(rand() % 100);

// array2[i] = static\_cast<T>(rand() % 100);

array1[i] = static\_cast<T>(rand()) / static\_cast<T>(RAND\_MAX / 100.0);

array2[i] = static\_cast<T>(rand()) / static\_cast<T>(RAND\_MAX / 100.0);

}

}

// Метод для нахождения суммы элементов массивов

T findSum() const {

T sum = 0;

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

sum += array1[i] + array2[i];

}

return sum;

}

// Метод для вывода содержимого массивов

void displayArrays() const {

std::cout << "Первый массив: ";

displayArray(array1);

std::cout << "Второй массив: ";

displayArray(array2);

}

private:

T array1[size];

T array2[size];

// Вспомогательный метод для вывода содержимого массива

void displayArray(const T\* arr) const {

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

std::cout << std::fixed << std::setprecision(1) << arr[i] << " ";

}

std::cout << std::endl;

}

};

int main() {

setlocale(LC\_ALL, "Ru");

// Примеры использования шаблона на различных типах данных

// Шаблон для массивов целых чисел

std::cout << "\nДля массивов целых чисел: " << std::endl;

ArraySum<int, 5> intArraySum;

intArraySum.displayArrays();

std::cout << "Сумма массивов: " << intArraySum.findSum() << std::endl;

std::cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " << std::endl;

// Шаблон для массивов чисел с плавающей точкой

std::cout << "\nДля массивов чисел с плавающей точкой: " << std::endl;

ArraySum<double, 5> doubleArraySum;

doubleArraySum.displayArrays();

std::cout << "Сумма массивов: " << doubleArraySum.findSum() << std::endl;

return 0;

}

**Задание №3**

Создать шаблон класса «стек». Использовать его при решении задачи 3 из лабораторной № 1.

#include <iostream>

#include <string>

// Шаблон класса "стек"

template <typename T, size\_t Capacity>

class Stack {

private:

T elements[Capacity];

size\_t top;

public:

Stack() : top(0) {}

void push(const T& value) {

if (top < Capacity) {

elements[top++] = value;

}

else {

std::cerr << "Stack overflow!" << std::endl;

}

}

T pop() {

if (top > 0) {

return elements[--top];

}

else {

std::cerr << "Stack underflow!" << std::endl;

return T(); // Возвращаем значение по умолчанию в случае ошибки

}

}

bool isEmpty() const {

return top == 0;

}

size\_t size() const {

return top;

}

};

// Структура для представления информации о работнике

struct Worker {

std::string surname;

int age;

std::string specialization;

double averageSalary;

};

int main() {

setlocale(LC\_ALL, "Ru");

const size\_t maxWorkers = 100; // Максимальное количество работников на одном заводе

Stack<Worker, maxWorkers> workerStack;

// Ввод информации о работниках завода

size\_t numberOfFactories;

std::cout << "Введите количество заводов: ";

std::cin >> numberOfFactories;

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

size\_t numberOfWorkers;

std::cout << "Введите количество работников на заводе " << i + 1 << ": ";

std::cin >> numberOfWorkers;

for (size\_t j = 0; j < numberOfWorkers; ++j) {

Worker worker;

std::cout << "Введите информацию о работнике " << j + 1 << " на заводе " << i + 1 << ":\n";

std::cout << "Фамилия: ";

std::cin >> worker.surname;

std::cout << "Возраст: ";

std::cin >> worker.age;

std::cout << "Специальность: ";

std::cin >> worker.specialization;

std::cout << "Средний оклад: ";

std::cin >> worker.averageSalary;

// Положим информацию о работнике в стек

workerStack.push(worker);

}

}

// Подсчет количества слесарей и токарей

size\_t numberOfLocksmiths = 0;

size\_t numberOfTurners = 0;

while (!workerStack.isEmpty()) {

Worker worker = workerStack.pop();

if (worker.specialization == "slesar") {

++numberOfLocksmiths;

}

else if (worker.specialization == "tokar") {

++numberOfTurners;

}

}

// Вывод результатов

std::cout << "Количество слесарей: " << numberOfLocksmiths << std::endl;

std::cout << "Количество токарей: " << numberOfTurners << std::endl;

return 0;

}

**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ №8**

1. Что такое обобщенная функция?

**Обобщенная функция** - это такая функция, которая умеет работать с разными типами данных или структурами, не требуя явного указания конкретного типа. Она гибкая и может использоваться для различных данных.

2. Шаблоны функции позволяют удобным способом создать семейство

а) переменных; **б) функций;** в) классов; г) программ.

Шаблоны функций позволяют создавать сразу несколько функций для разных типов данных, обеспечивая гибкость и универсальность.

3. Истинно ли утверждение, что шаблонная функция может иметь несколько аргументов?

**Да, истинно**. Шаблонная функция может иметь несколько аргументов, и эти аргументы могут быть различных типов.

4. Реальный код шаблонной функции генерируется при:

**а) объявлении функции в исходном коде;**

б) создании объекта;

в) запуске функции из её исходного кода.

1. Что такое обобщенный (шаблонный) класс?

**Обобщенный (шаблонный) класс** - это класс, который может работать с разными типами данных или объектами, аналогично обобщенной функции.

1. В каком случае его имеет смысл применять?

**Шаблонные классы имеют смысл использовать**, когда требуется создать универсальный класс, способный работать с различными типами данных, обеспечивая повторное использование кода без необходимости создания отдельного класса для каждого типа. Это обеспечивает гибкость и эффективное управление разнообразием данных.

1. Шаблонный класс:

**а) работает с разными типами данных;**

б) генерирует идентичные объекты;

в) генерирует классы с различным числом методов

Шаблонный класс создается для работы с разными типами данных, но не создает идентичные объекты и не меняет количество методов.

Простой пример:

#include <iostream>

using namespace std;

template <typename T1, typename T2>

T1 Sum(T1 a, T2 b) {

return a + b;

}

int main() {

setlocale(LC\_ALL, "Ru");

std::cout << Sum(5, 1.0) << std::endl;

std::cout << Sum(2.0, 1.0) << std::endl;

return 0;

}

