**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №8**

Выполнил: студент группы ПИ-3-21

Закиров Мустафа

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

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.

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

using namespace std;

// Функция для поиска максимального элемента в массиве

template<typename T>

T findMax(const T array[], int size) {

T max = array[0]; // Инициализация max первым элементом массива

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

if (array[i] > max) {

max = array[i]; // Обновление max, если найден больший элемент

}

}

return max;

}

// Функция для сортировки массива по возрастанию

template<typename T>

void sortArray(T array[], int size) {

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

for (int j = 0; j < size - i - 1; ++j) {

if (array[j] > array[j + 1]) {

// Обмен элементов, если они стоят в неправильном порядке

T temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

// Целочисленный массив

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

int intSize = sizeof(intArray) / sizeof(int);

cout << "Максимальное целое число: " << findMax(intArray, intSize) << endl;

sortArray(intArray, intSize);

cout << "Отсортированный целочисленный массив: ";

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

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

}

cout << endl;

// Массив чисел с плавающей запятой

double doubleArray[] = { 3.14, 1.41, 2.71, 2.55, 6.02 };

int doubleSize = sizeof(doubleArray) / sizeof(double);

cout << "Максимальное число с плавающей запятой: " << findMax(doubleArray, doubleSize) << endl;

sortArray(doubleArray, doubleSize);

cout << "Отсортированный массив чисел с плавающей запятой: ";

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

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

}

cout << endl;

// Массив символов

char charArray[] = { 'a', 'C', 'F', 'x', '!', '#', '[', '&', '?', '1', 'n' };

int charSize = sizeof(charArray) / sizeof(char);

cout << "Максимальный символ: " << findMax(charArray, charSize) << endl;

sortArray(charArray, charSize);

cout << "Отсортированный массив символов: ";

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

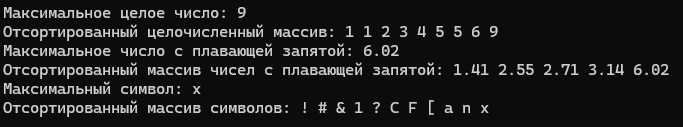
cout << charArray[i] << " ";

}

cout << endl;

return 0;

}

**Тесты:**

# **Задание №2**

**Постановка задачи:**

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.

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <random>

template<typename T, size\_t N>

class ArrayPair {

private:

T array1[N];

T array2[N];

public:

ArrayPair() {

std::random\_device rd;

std::mt19937 mersenne(rd());

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

array1[i] = mersenne();

}

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

array2[i] = mersenne();

}

}

void printArrays() {

std::cout << "Массив №1: " << std::endl;

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

std::cout << array1[i] << " ";

}

std::cout << std::endl;

std::cout << "Массив №2: " << std::endl;

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

std::cout << array2[i] << " ";

}

std::cout << std::endl;

}

T sum() const {

T total = 0;

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

total += array1[i];

total += array2[i];

}

return total;

}

};

int main() {

ArrayPair<int, 5> intArrays;

intArrays.printArrays();

std::cout << "Сумма целых чисел: " << intArrays.sum() << std::endl;

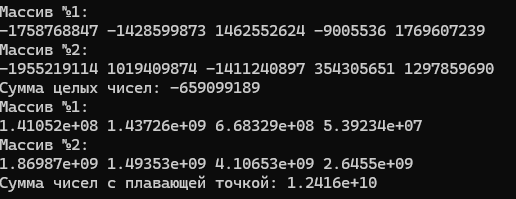
ArrayPair<double, 4> doubleArrays;

doubleArrays.printArrays();

std::cout << "Сумма чисел с плавающей точкой: " << doubleArrays.sum() << std::endl;

return 0;

}

**Тесты:**

# **Задание №3**

**Постановка задачи:**

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

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

template<typename T>

class Stack {

private:

vector<T> elements;

public:

void push(const T& element) {

elements.push\_back(element);

}

void pop() {

if (!elements.empty()) {

elements.pop\_back();

}

}

const T& top() const {

if (!elements.empty()) {

return elements.back();

}

throw std::out\_of\_range("Stack<>::top(): стек пуст");

}

bool empty() const {

return elements.empty();

}

size\_t size() const {

return elements.size();

}

};

class TrainCar {

public:

int carNumber;

TrainCar(int number) : carNumber(number) {}

};

class TrainSorter {

private:

Stack<TrainCar> mainTrack;

Stack<TrainCar> sideTrackA;

Stack<TrainCar> sideTrackB;

public:

void addToMainTrack(const TrainCar& car) {

mainTrack.push(car);

}

void moveToSideTrackA() {

if (!mainTrack.empty()) {

sideTrackA.push(mainTrack.top());

mainTrack.pop();

}

}

void moveToSideTrackB() {

if (!mainTrack.empty()) {

sideTrackB.push(mainTrack.top());

mainTrack.pop();

}

}

void popFromSideTrackA() {

if (!sideTrackA.empty()) sideTrackA.pop();

}

void popFromSideTrackB() {

if (!sideTrackB.empty()) sideTrackB.pop();

}

void displayTrainComposition() const {

cout << "Основной путь: ";

displayStack(mainTrack);

cout << "Боковой путь A: ";

displayStack(sideTrackA);

cout << "Боковой путь B: ";

displayStack(sideTrackB);

}

private:

void displayStack(const Stack<TrainCar>& stack) const {

Stack<TrainCar> tempStack = stack;

while (!tempStack.empty()) {

std::cout << tempStack.top().carNumber << " ";

tempStack.pop();

}

std::cout << std::endl;

}

};

void readTrainFromFile(TrainSorter& sorter, const string& filename) {

ifstream inputFile(filename);

if (!inputFile.is\_open()) {

cout << "Не удалось открыть файл: " << filename << std::endl;

return;

}

int carNumber;

while (inputFile >> carNumber) {

sorter.addToMainTrack(TrainCar(carNumber));

}

inputFile.close();

}

int main() {

TrainSorter trainSorter;

readTrainFromFile(trainSorter, "trains.txt");

int num;

do {

cout << "Что делаем?" << endl

<< "1. Добавить вагон на основной путь" << endl

<< "2. Переместить вагон на боковой путь A" << endl

<< "3. Переместить вагон на боковой путь B" << endl

<< "4. Убрать вагон с бокового пути A" << endl

<< "5. Убрать вагон с бокового пути B" << endl

<< "6. Вывести информацию о составе поезда" << endl

<< "0. Выйти" << endl;

cin >> num;

switch (num) {

case 1: {

int carNumber;

std::cout << "Введите номер вагона: ";

std::cin >> carNumber;

trainSorter.addToMainTrack(TrainCar(carNumber));

break;

}

case 2:

trainSorter.moveToSideTrackA();

break;

case 3:

trainSorter.moveToSideTrackB();

break;

case 4:

trainSorter.popFromSideTrackA();

break;

case 5:

trainSorter.popFromSideTrackB();

break;

case 6:

trainSorter.displayTrainComposition();

break;

case 0:

cout << "Выход из программы" << endl;

break;

default:

cout << "Некорректный ввод" << endl;

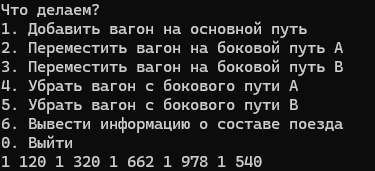
}

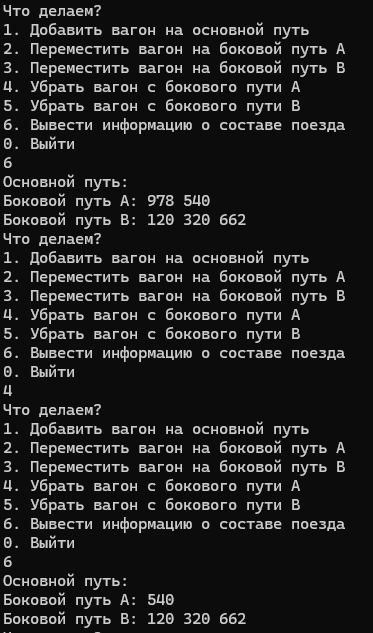
} while (num);

return 0;

}

**Тесты:**





# **Ответы на вопросы:**

1. Что такое обобщенная функция?

Шаблон функции (обобщенные функции)

Функции, в которых тип данных передается, как формальный параметр, называются шаблонами функций.

2. Шаблоны функции позволяют удобным способом создать семейство.  
а) переменных;  
б) функций;  
в) классов;  
г) программ.

Б) Функций.

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

Да, истинно.

template <class Т> void chang(Т &a, Т &b){

Т t; //ввели доп. переменную t обобщенного типа, как и у аргументов

t=a; //сохранили в ней значение первого аргумента

a=b; // 1-му аргументу присвоили значение второго аргумента

b=t; // 2-му аргументу присвоили значение введенной переменной t

}

int main() {

double x=2.3,y=4.4;

int m=5,n=6;

**//вызов шаблона функции**

chang(x,y); //вызов функции с аргументами типа double

cout<<"x: "<<x<<endl;

cout<<"y: "<<y<<endl;

chang(m,n); //вызов функции с аргументами типа int

cout<<"m: "<<m<<endl;

cout<<"n: "<<n<<endl;

return 0;}

4. Реальный код шаблонной функции генерируется при:   
а) объявлении функции в исходном коде;  
б) создании объекта;   
в) запуске функции из её исходного кода.

А) Объявлении функции в исходном коде.

5. Что такое обобщенный (шаблонный) класс?

Шаблон класса (обобщенный класс).

Классы, в которых тип данных передается как формальный параметр, называются обобщенными.

6. В каком случае его имеет смысл применять?

Обобщенные классы имеют смысл применять, когда требуется написать универсальный код, который работает с разными типами данных, обеспечивая при этом повторное использование кода и увеличивая его гибкость.

7. Шаблонный класс:  
а) работает с разными типами данных;   
б) генерирует идентичные объекты;  
в) генерирует классы с различным числом методов

А) Работает с различными типами данных.