**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: Одномерные статистические массивы

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4372 |  | Кубрина А.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение одномерных статистических массивов.

**Основные теоретические положения.**

Одномерные статические массивы являются одним из основных инструментов для хранения и обработки данных в языке программирования C++. Они позволяют организовать данные одного типа в упорядоченную последовательность, что делает работу с ними более удобной и эффективной. В этом отчете мы рассмотрим основные характеристики одномерных статических массивов, их использование, преимущества и недостатки, а также примеры кода.

1. Определение и структура одномерных статических массивов

1.1. Определение

Одномерный статический массив — это коллекция элементов одного типа, хранящихся в непрерывной области памяти. Размер массива задается во время компиляции и остается фиксированным в течение всего времени выполнения программы.

1.2. Объявление массива

Для объявления одномерного статического массива в C++ используется следующий синтаксис:

тип\_данных имя\_массива[размер];

Пример:

int numbers[5]; // Объявление массива из 5 целых чисел

В данном примере numbers — это массив, который может хранить 5 целых чисел.

2. Инициализация массивов

2.1. Инициализация при объявлении

Массивы можно инициализировать сразу при их объявлении:

int numbers[5] = {1, 2, 3, 4, 5}; // Инициализация массива значениями

Если количество инициализируемых элементов меньше размера массива, оставшиеся элементы будут инициализированы нулями:

int numbers[5] = {1, 2}; // numbers[2], numbers[3], numbers[4] будут равны 0

2.2. Частичная инициализация

Если массив не полностью инициализирован, то неуказанные элементы получают значение по умолчанию:

int numbers[5] = {}; // Все элементы будут равны 0

3. Доступ к элементам массива

Доступ к элементам массива осуществляется с помощью индексов. Индексация начинается с нуля:

numbers[0] = 10; // Присвоение значения первому элементу

int firstElement = numbers[0]; // Получение значения первого элемента

Для обхода массива можно использовать цикл:

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

std::cout << numbers[i] << " "; // Вывод всех элементов массива

}

4. Преимущества и недостатки статических массивов

4.1. Преимущества

• Простота использования: Статические массивы легко объявлять и использовать в коде.

• Быстрый доступ: Доступ к элементам массива осуществляется за константное время O(1).

• Эффективное использование памяти: Память выделяется единожды, что делает работу с памятью более предсказуемой.

4.2. Недостатки

• Фиксированный размер: Размер массива необходимо задавать заранее, что может привести к неэффективному использованию памяти.

• Отсутствие динамичности: Необходимо использовать другие структуры данных (например, динамические массивы или контейнеры STL), если требуется изменяемый размер.

• Отсутствие встроенных методов: В отличие от контейнеров STL, статические массивы не имеют встроенных методов для работы с данными (например, сортировка или поиск).

**Постановка задачи.**

Необходимо написать программу, которая:

1)    Создает целочисленный массив размерности *N* = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.

2)    Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.

3)    Найти максимальный и минимальный элемент массива. Подсчитайте время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.

4)    Выводит среднее значение (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Выводит индексы всех элементов, которые равны этому значению, и их количество. Подсчитайте время поиска.

5)    Выводит количество элементов в отсортированном массиве, которые меньше числа *a*, которое инициализируется пользователем.

6)    Выводит количество элементов в отсортированном массиве, которые больше числа *b*, которое инициализируется пользователем.

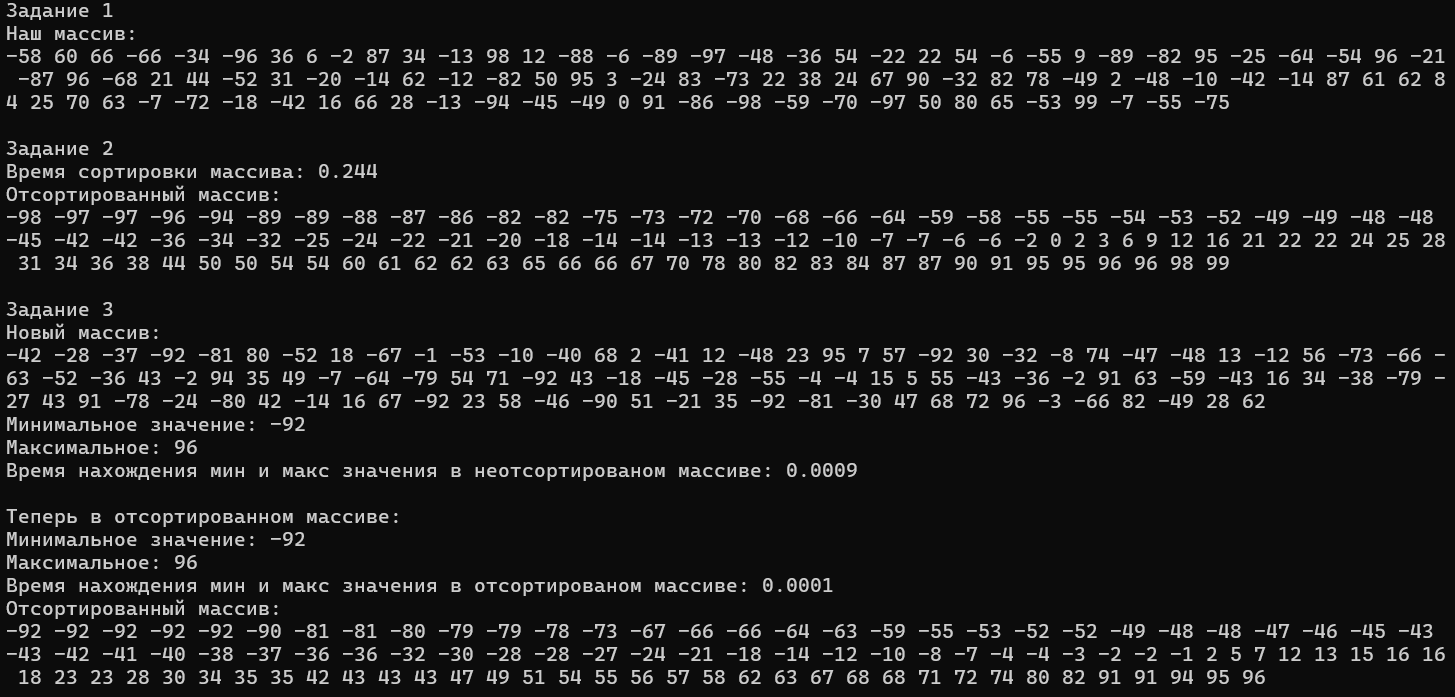
7)    Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором. (\*)

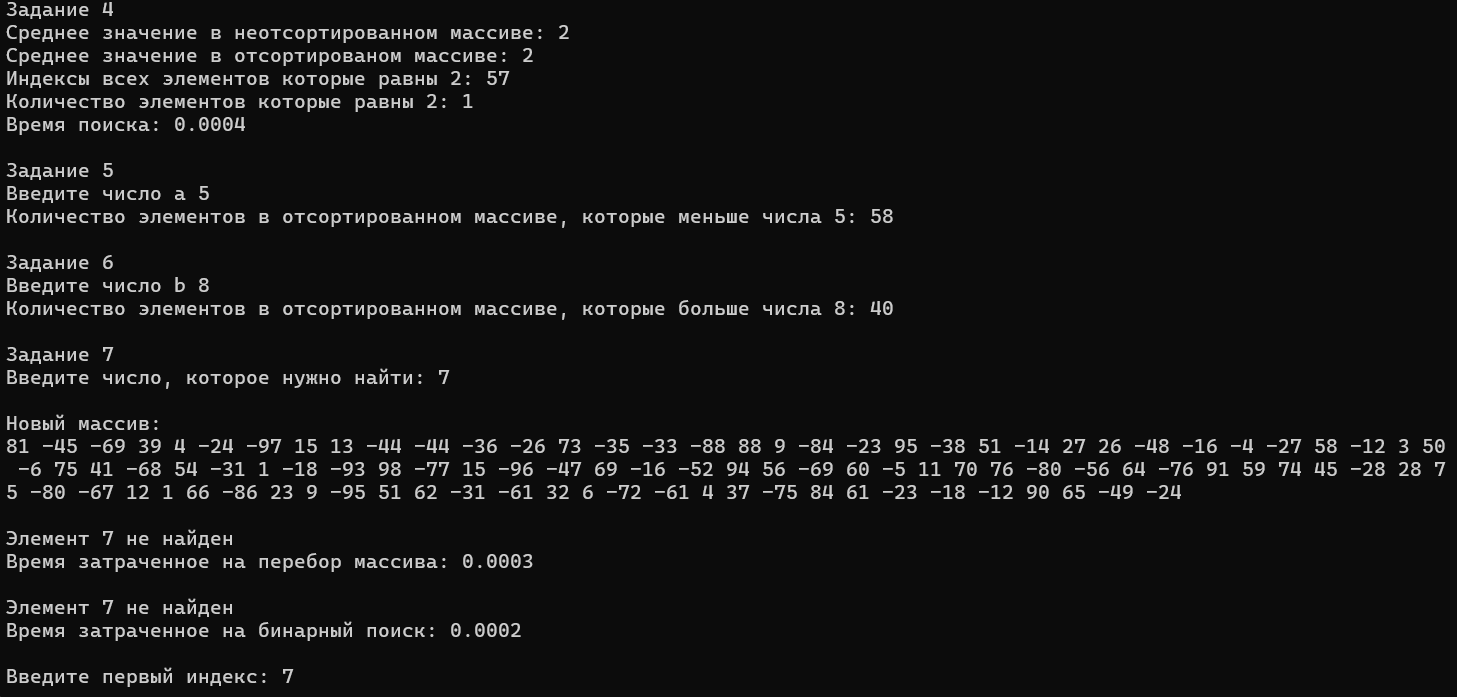
8)     Меняет местами элементы массива, индексы которых вводит пользователь. Выведите скорость обмена, используя библиотеку chrono.

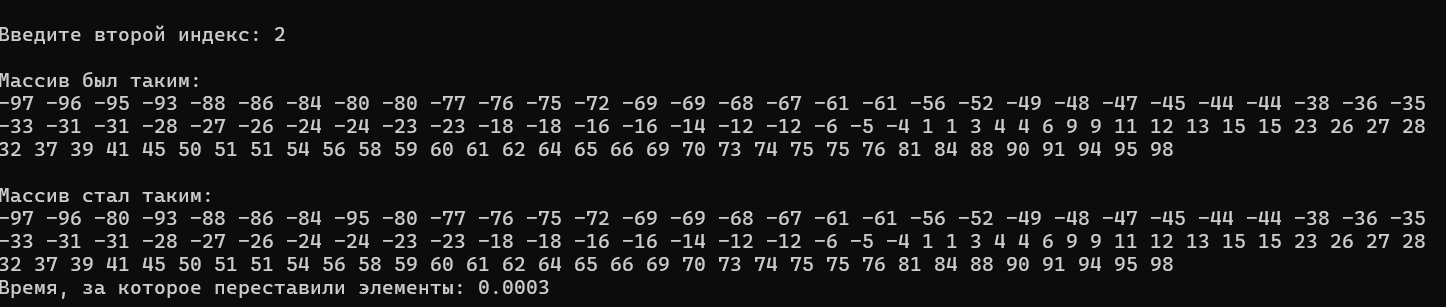
**Выполнение работы.**

Код программы представлен в приложении А.

Блок скриншотов работы программы







**Выводы.**

Одномерные статические массивы являются важным инструментом для организации данных в C++. Они просты в использовании и обеспечивают быстрый доступ к элементам. Однако их фиксированный размер и отсутствие динамичности могут ограничивать применение в некоторых ситуациях. Для более сложных задач рекомендуется использовать динамические структуры данных или контейнеры стандартной библиотеки.

Приложение А

рабочий код

