**Лабораторная работа №1**

**"Библиотечные реализации алгоритмов и их скорость"**

*Оценка вычислительной сложности и потребления памяти: удаление элемента из конца одномерного Массива*

**Номер варианта: 11**

**Преподаватель:** Лебедев Евгений Денисович

Работу выполнил: Зазыкин Иван Дмитриевич

**Определение структур данных и операций**:

Работаем с одномерным массивом (или динамическим массивом — **std::vector**). Операция — удаление элемента с конца. Это можно реализовать с использованием функции **pop\_back()** для **std::vector**, или вручную для обычного массива.

**Ожидаемый результат:**

* Ожидаем увидеть, как меняется время работы программы при увеличении размера массива (по числам 10, 100, 1000 и так далее до 1.000.000.000).

**Справочная информация о процессоре:**

* **Процессор**: AMD Ryzen 7 8845HS
* **Тактовая частота:** 3.8 – 5.1 ГГц
* **Количество ядер**: 8 ядер, 16 потоков
* **Размер транзистора:** 4 нанометра
* **Кэш**: 16 MB L3 кэш
* **Архитектура набора команд: x86-64**

**Листинги кода реализации:**

**void removeLastElement(vector<int>& arr) {**

**if (!arr.empty()) {**

**arr.pop\_back();**

**}**

**}**

1. Этой функцией проверяем, не пуст ли вектор, и удаляем его последний элемент, если в нём есть данные.

**int main() {**

**setlocale(LC\_ALL, "Russian");**

1. Устанавливаю русскую локализацию для корректного вывода в консоль.

**vector<int> sizes = { 10, 100, 150, 1000, 10000, 10500, 100000, 1000000, 10000000, 100000000 };**

1. Создаём вектор с различными размерами массивов, которые будем тестировать

**auto start = chrono::high\_resolution\_clock::now();**

**removeLastElement(arr);**

**auto end = chrono::high\_resolution\_clock::now();**

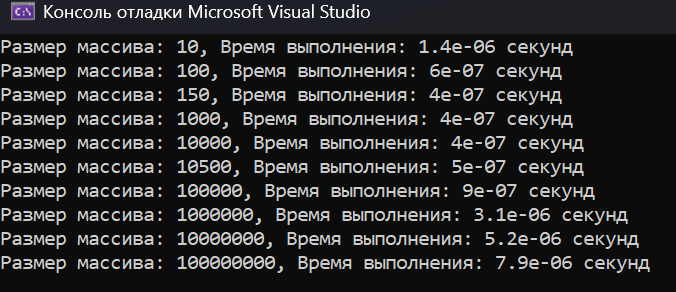
1. Засекаем начальное время, вызываем функцию **removeLastElement()**, а затем фиксируем время окончания.

**chrono::duration<double> duration = end - start;**

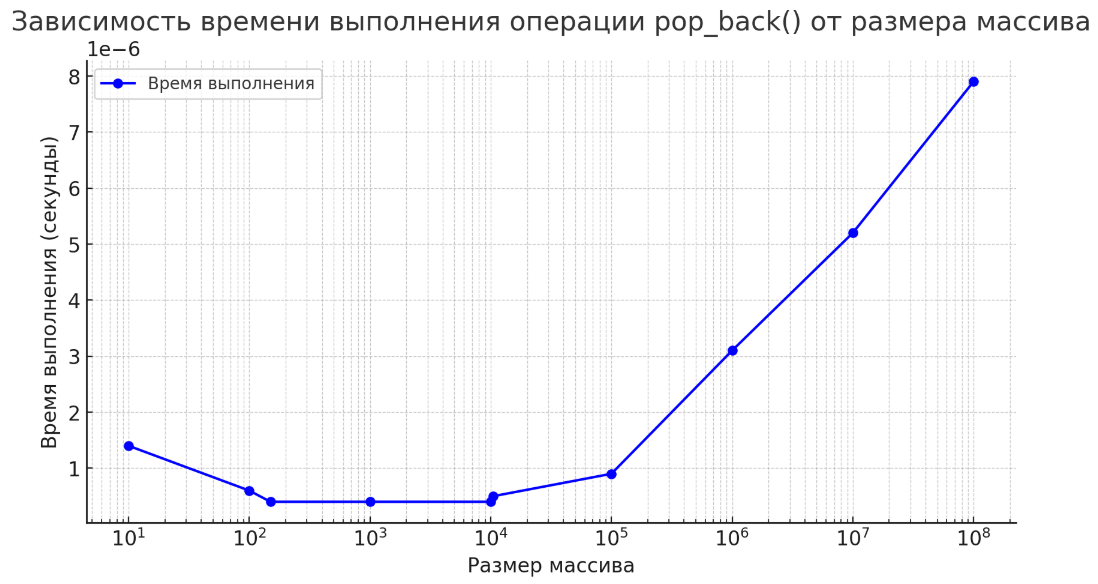
**cout << "Размер массива: " << size << ", Время выполнения: " << duration.count() << " секунд" << endl;**

1. Вычисляем и выводим время выполнения операции.

**График с измерениями:**

**Вывод из консоли: **

**График выполнения операции:**

****

**Вывод:**

Ожидаемый результат частично подтвердился: в теории операция **pop\_back()** выполняется за **O(1)** — постоянное время, так как удаление последнего элемента вектора не требует сдвига остальных элементов. Однако глядя на график, видим, что время выполнения **незначительно растёт** при увеличении размера массива.

Это может быть связано с влиянием системных факторов: особенностями работы кэша процессора, многозадачностью операционной системы или задержками при доступе к оперативной памяти. Несмотря на эти колебания, сложность операции **pop\_back()** остаётся **O(1)**, поскольку рост времени выполнения **не зависит линейно от размера массива**.