**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе № 5 на тему:   
**«Методы поиска элемента в массиве»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-312, Серафимович Г. П.

«10» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ Серафимович Г. П./

Принял: к.ф.-м.н., доцент, И.А. Моисеев

«10» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Основная часть**

**Цель работы:**

Целью данной работы является реализация двух алгоритмов поиска (линейного и бинарного) на основе массива целых чисел, считанных из файла. Программа должна продемонстрировать эффективность различных методов поиска, а также подсчитать количество операций сравнения, необходимых для выполнения этих операций.

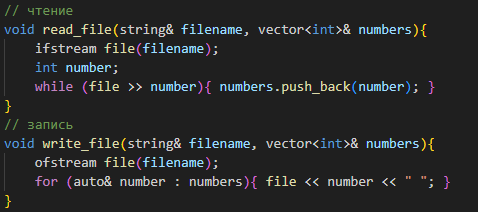
**Результаты выполнения работы:**

Были реализованы основные два метода:

1. Метод бинарного поиска

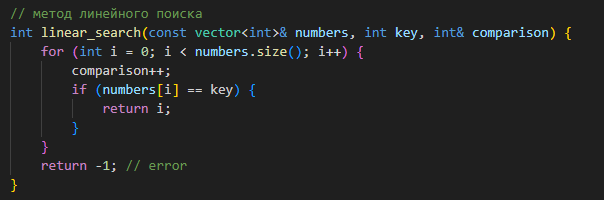
2. Метод линейного поиска

Также была реализована функция для чтения файла (Рисунок 1)



**Рисунок 1. Чтение файла**

Функция линейного поиска (Рисунок 2)

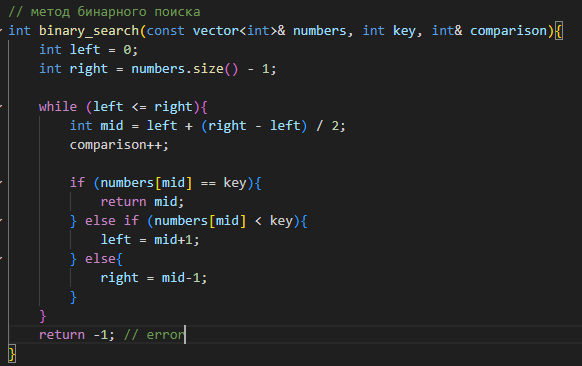


**Рисунок 2 Функция линейного поиска**

В этом коде реализован метод линейного поиска, который ищет заданное значение key в векторе целых чисел numbers. Он перебирает элементы вектора по одному, увеличивая счетчик comparison при каждой итерации. Если найден элемент, равный key, метод возвращает его индекс.

Если элемент не найден, функция возвращает -1, указывая на ошибку. Этот алгоритм имеет линейную временную сложность O(n).

Функция бинарного поиска(Рисунок 3)



**Рисунок 3 Функция бинарного поиска**

В этом коде реализован метод бинарного поиска, который ищет заданное значение key в отсортированном векторе целых чисел numbers. Алгоритм использует два указателя: left и right, чтобы определить текущий диапазон поиска. На каждой итерации вычисляется средний индекс mid, и если элемент по этому индексу равен key, возвращается его индекс. Если элемент меньше key, поиск продолжается в правой половине, иначе — в левой. Если элемент не найден, функция возвращает -1, указывая на ошибку.

Общий вывод 1 пример (Рисунок 6)



**Рисунок 6 Общий вывод**

Общий вывод 2 пример (Рисунок 7)



**Рисунок 7 Общий вывод**

**Выводы:**

В ходе выполнения задачи я многое понял. Бинарный поиск, примененный к отсортированному массиву, также показал свою эффективность по сравнению с линейным поиском, особенно на больших объемах данных.

**Приложение**

Листинг программы:

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

using namespace std;

// чтение

void read\_file(string& filename, vector<int>& numbers) {

ifstream file(filename);

if (!file.is\_open()) { cerr << "Ошибка при открытии файла!" << endl; exit(1); }

int number;

while (file >> number) {

numbers.push\_back(number);

}

}

// метод бинарного поиска

int binary\_search(const vector<int>& numbers, int key, int& comparison) {

int left = 0;

int right = numbers.size() - 1;

while (left <= right) {

comparison++;

int mid = left + (right - left) / 2;

if (numbers[mid] == key) {

return mid;

} else if (numbers[mid] > key) {

left = mid + 1; // перемещение вправо

} else {

right = mid - 1; // перемещение влево

}

}

return -1; // error

}

// метод линейного поиска

int linear\_search(const vector<int>& numbers, int key, int& comparison) {

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

comparison++;

if (numbers[i] == key) {

return i;

}

}

return -1; // error

}

int main() {

string inputFile = "sorted\_input.txt";

vector<int> numbers;

read\_file(inputFile, numbers);

// поиск

int key;

cout << "Введи число для поиска: ";

cin >> key;

int linearSearchCount = 0;

int linearResult = linear\_search(numbers, key, linearSearchCount);

int binarySearchCount = 0;

int binaryResult = binary\_search(numbers, key, binarySearchCount);

// вывод результатов поиска

if (linearResult != -1) {

cout << "Индекс от linear search: " << linearResult << " (" << linearSearchCount << " операции сравнения)" << endl;

} else {

cout << "Элемент не найден линейным поиском (" << linearSearchCount << " операции сравнения)" << endl;

}

if (binaryResult != -1) {

cout << "Индекс от binary search: " << binaryResult << " (" << binarySearchCount << " операции сравнения)" << endl;

} else {

cout << "Элемент не найден бинарным поиском (" << binarySearchCount << " операции сравнения)" << endl;

}

return 0;

}