МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Информатика

Отчет по лабораторной работе №2

Выполнила бригада группы М3О-211Б-21

Багиров Э. Р.

Нуриев Н. Н.

Оглавление

[Задание 3](#_Toc117518159)

[Код программы 4](#_Toc117518160)

[Сравнительные графики зависимости времени работы (в мкс) алгоритмов поиска от количества элементов массива 7](#_Toc117518161)

[Вывод 8](#_Toc117518162)

# Задание

Для алгоритмов **BLS** и **SLS** в качестве входного массива использовать одну и ту же последовательность значений (функция **rand**( )).

Для алгоритмов **Т** и **В** – значения массива должны быть отсортированы по неубыванию, одна и та же последовательность чисел (можно использовать соответствующую функцию из первой лабораторной работы).

Оценить длительность поиска для различных значений размеров последовательностей (начиная с 10000 до 200000 элементов массива, провести измерения не менее, чем для 10 разных размерностей).

Для каждой размерности рассматриваются случаи нахождения ключа поиска в начале, в середине и в конце массива.

Для алгоритмов **BLS** и **SLS** кроме подсчета **времени,** необходимого для поиска, требуется определить сколько раз выполняются операции **сравнения** (сравнение ключа с элементом массива, а также в одном из этих двух алгоритмов добавляется подсчет сравнений при анализе индекса элемента массива в цикле… ).

Все результаты оформить в виде таблиц и графиков. На графиках **- только временные характеристики** поиска.

По результатам сделать выводы об эффективности того или иного алгоритма поиска.

# Код программы

#include <iostream>

#include <ctime>

#include <chrono>

using namespace std;

void InsSort();

int BLS();

int SLS();

int T();

int B();

void InsSort(int\* arr, int length) {

int count = 0;

int temp;

for (int i = 1; i < length; i++)

for (int j = i; j > 0 && arr[j - 1] > arr[j]; j--) {

count++;

temp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = temp;

}

}

int BLS(int\* arr, int length, int key, int\* comparisons) {

\*comparisons = 0;

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

(\*comparisons)++;

if (arr[i] == key)

return i;

}

return -1; // NOT FOUND

}

int SLS(int\* arr, int length, int key, int\* comparisons) {

\*comparisons = 0;

int last = arr[length-1];

arr[length - 1] = key;

int i = 0;

while (arr[i] != key) {

(\*comparisons)++;

i++;

}

(\*comparisons)++; // учесть сравнение после выхода из while

arr[length - 1] = last;

(\*comparisons)++; // учесть сравнение соответсвия конечному индексу

if (i < length - 1) {

return i;

}

(\*comparisons)++; // учесть сравнение последнего элемента с ключем

if (arr[length - 1] == key) {

return i;

}

return -1; // NOT FOUND

}

int T(int\* arr, int length, int key) {

arr[length] = 2147483647;

int i = 0;

while (key > arr[i]) i++;

if (arr[i] == key) return i;

return -1; // NOT FOUND

}

int B(int\* arr, int length, int key) {

int p = 0;

int r = length-1;

int q = 0;

while (true) {

int q = (p + r) / 2;

if (key < arr[q]) r = q - 1;

else if (key > arr[q]) p = q + 1;

else return q;

if (p > r) return -1; // NOT FOUND

}

}

int main() {

srand(0);

setlocale(LC\_ALL, "rus");

int length;

cout << "Введите длину массива: ";

cin >> length;

cout << endl;

int\* arr = new int[length + 1];

for (int i = 0; i < length; i++) arr[i] = rand();

int key;

cout << "Введите ключ поиска: ";

cin >> key;

int pick;

int indexForKey;

int tmp;

cout << endl << "Хотите вставить ключ в массив?\n1.Да\n2.Нет\nВаш выбор: ";

cin >> pick;

if (pick == 1) {

cout << endl << "Введите индекс, в который вставить ключ поиска: ";

cin >> indexForKey;

cout << endl;

tmp = arr[indexForKey];

arr[indexForKey] = key;

}

int indexToFound;

int comparisons;

auto startTime = chrono::steady\_clock::now();

indexToFound = BLS(arr, length, key, &comparisons);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nЭлемент не найден в поиске BLS.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

cout << "Было сделано " << comparisons << " сравнений\n\n";

}

else {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Результат работы алгоритма поиска BLS: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

cout << "Было сделано " << comparisons << " сравнений\n\n";

}

startTime = chrono::steady\_clock::now();

indexToFound = SLS(arr, length, key, &comparisons);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Элемент не найден в поиске SLS.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

cout << "Было сделано " << comparisons << " сравнений\n\n";

}

else {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Результат работы алгоритма поиска SLS: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

cout << "Было сделано " << comparisons << " сравнений\n\n";

}

startTime = chrono::steady\_clock::now();

InsSort(arr, length);

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nВремя сортировки: " << time.count() << " мкс.\n\n";

startTime = chrono::steady\_clock::now();

indexToFound = T(arr, length, key);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nЭлемент не найден в поиске T.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

else {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nРезультат работы алгоритма поиска T: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

startTime = chrono::steady\_clock::now();

indexToFound = B(arr, length, key);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nЭлемент не найден в поиске B.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

else {

auto endTime = chrono::steady\_clock::now();

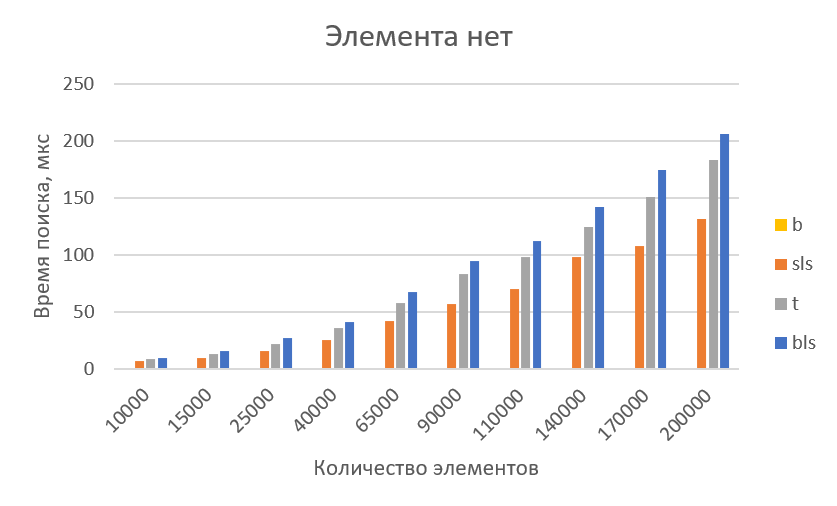
time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nРезультат работы алгоритма поиска B: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

}

# Сравнительные графики зависимости времени работы алгоритмов поиска от количества элементов массива







# Вывод

Были изучены основные принципы работы алгоритмов поиска BLS – последовательный поиск, SLS – быстрый последовательный поиск, T – последовательный поиск в упорядоченном массиве и B – бинарный поиск, а также исследовал их свойства. Программа написана для осуществления поиска среди массива целых чисел. Было измерено время работы алгоритмов поиска с помощью библиотеки chrono. Алгоритм BLS, состоящий в последовательном сравнении каждого элемента с ключом поиска, работает медленнее, чем SLS, заключающийся Был сделан вывод, что для неупорядоченных массивов лучше использовать алгоритм SLS, для упорядоченных массивов – B.