|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра МОСИТ |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ЛАБОРАТОРНОЙ РАБОТЕ №8** | |
| **по дисциплине** |  |
| **«**Структуры и АОД**»** |  |
| Выполнил студент группы ИКБО-03-19 | Макаров Д.В |
| Принял профессор кафедры математического обеспечения и стандартизации информационных технологий, канд. техн.  наук | Кораблин Ю.П |

|  |  |  |
| --- | --- | --- |
| Лабораторная (Практическая) | « » 202 г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
| работа выполнена |  |  |
| «Зачтено» | « » 202 г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |

Москва 2020

**Содержание**

[Постановка задачи: 3](#_Toc37239379)

[Исходный код проекта: 3](#_Toc37239380)

[Результаты работы: 4](#_Toc37239381)

[Вывод: 5](#_Toc37239382)

# **Постановка задачи:**

1. Составить программу последовательного (с барьером) поиска заданного элемента по ключу в одномерном целочисленном массиве A[n].  
 2. Составить программы поиска заданного элемента по ключу в одномерном упорядоченном целочисленном массиве A[n] для двух алгоритмов поиска:  
 1)бинарного(двоичного)поиска:  
 2) Фибоначчиева поиска.

# **Исходный код проекта:**

#include <iostream>

#include <time.h>

#include <string>

int compbinary = 0;

int compfibonachy = 0;

//k- максимальное число, которые есть в массиве

//right - самое большое число

//left - cамое маленьекое

int binaryfind(int\* A, int key, int left, int right) {

//найдем средний элемент

int mid = (left + right) / 2;

// если mid =key, вернём true

if (key == mid)

{

compbinary++;

return mid;

}

//если не совпадает, ищем дальше

if (key < mid)

{

compbinary++;

binaryfind(A, key, left, mid);

}

else

{

binaryfind(A, key, mid, right);

}

}

//left- минимальное число

//n - максимум

int fibonachyfind(int arr[], int left, int right)

{

//p- первое число фибоначи

//q- второе

//sumqp- их сумма

//borbder - граница выхода за массив

int p = 0, q = 1, sumqp = p + q, border = -1;;

//устанавливаем значение q и p

while (sumqp < right)

{

p = q;

q = sumqp;

sumqp = p + q;

}

while (sumqp > 1)

{

int buff;

//проверка,является ли p допустимым

//считать здесь сравнение не будем, т.к проверки в бинарном поиске на границу мы не сделали

if (border + p <= right - 1) {

buff = border + p;

}

else

buff = right - 1;

//конец проверки

//ищем значение

if (arr[buff] < left)

{

compfibonachy++;

sumqp = q;

q = p;

p = sumqp - q;

border = buff;

}

else if (arr[buff] > left)

{

compfibonachy++;

sumqp = p;

q = q - p;

p = sumqp - q;

}

else return buff;

}

//если значение не найдено

if (q && arr[border + 1] == left)return border + 1;

return -1;

}

int main() {

int LargeOfMassive = 100000000;

int key;

int\* arr;

arr = new int[LargeOfMassive];

std::cout << "Key::";

std::cin >> key;

std::cout << " " << std::endl;

//генерируем пронумерованный массив

for (int i = 0; i < LargeOfMassive; i++)

{

arr[i] = i;

}

//считаем время

clock\_t start = clock();

std::cout << binaryfind(arr, key, arr[0], LargeOfMassive - 1) << std::endl;

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("binary find : %f seconds\n", seconds);

clock\_t start2 = clock();

std::cout << fibonachyfind(arr, key, LargeOfMassive - 1) << std::endl;

clock\_t end2 = clock();

double seconds2 = (double)(end2 - start2) / CLOCKS\_PER\_SEC;

printf("fubonachi find : %f seconds\n", seconds2);

std::cout << "comp of binary" << compbinary << std::endl;

std::cout << "comp of fibonachy" << compfibonachy;

return 0;

}

# **Результаты работы:**

Таблица 1.

|  |  |  |
| --- | --- | --- |
| n | Бинарный поиск | Поиск Фибоначчи |
| 100 | 0.000000 seconds | 0.000000 seconds |
| 1000 | 0.000000 seconds | 0.000000 seconds |
| 10000 | 0.000000 seconds | 0.000000 seconds |
| 100000 | 0.000000 seconds | 0.000000 seconds |
| 1000000  миллион | 0.001000 seconds | 0.001000 seconds |

# **Вывод:**

На основание замером, мы можем сделать вывод, что скорость выполнения у обоих методов поиска очень быстрая. Но если сравнивать количество операций сравнение, то бинарный поиск сравнивает меньшее количество элементов.