**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра CАПР**

отчет

**по лабораторной работе №2**

**по дисциплине «Алгоритмы и структуры данных»**

**Вариант 1**

**Тема: Алгоритмы сортировки и поиска**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9302 |  | Баязитов О. О. |
| Преподаватель |  | Тутуева.А.В |

Санкт-Петербург

2020

**Постановка задачи**

Реализовать алгоритмы поиска и сортировки целочисленных массивов.

**Описание реализуемых алгоритмов**

|  |  |
| --- | --- |
| Название метода | Описание |
| int BinarySearch(int\* array, int left, int right, int key) | Бинарный поиск |
| void QuickSort(int\* array, int low, int high) | Быстрая сортировка |
| void InsertionSort(int\* array, int n) | Сортировка вставками |
| void BogoSort(int\* array, int n) | Болотная сортировка |
| void CountingSort(int\* array, int n) | Сортировка подсчетом |

**Оценка временной сложности каждого алгоритма**

|  |  |
| --- | --- |
| Название метода | Оценка временной сложности |
| BinarySearch | O(log2(n)) |
| QuickSort | O(n\*log(n)) |
| InsertionSort | O(n^2) |
| BogoSort | O(n\*n!) |
| CountingSort | O(n^2) |

**Сравнение временной сложности алгоритмов QuickSort и BubbleSort**

QuickSort(10) QuickSort(100) QuickSort(1000) QuickSort(10000)

|  |  |
| --- | --- |
| Замер | Время в мс |
| 1 | <1 |
| 2 | <1 |
| 3 | <1 |
| 4 | <1 |
| 5 | <1 |
| 6 | <1 |
| 7 | <1 |
| 8 | <1 |
| 9 | <1 |
| 10 | <1 |
| Среднее | <1 |

|  |  |
| --- | --- |
| 1 | <1 |
| 2 | <1 |
| 3 | <1 |
| 4 | <1 |
| 5 | <1 |
| 6 | <1 |
| 7 | <1 |
| 8 | <1 |
| 9 | <1 |
| 10 | <1 |
| Среднее | <1 |

|  |  |
| --- | --- |
| 1 | <1 |
| 2 | <1 |
| 3 | <1 |
| 4 | <1 |
| 5 | <1 |
| 6 | <1 |
| 7 | <1 |
| 8 | <1 |
| 9 | <1 |
| 10 | <1 |
| Среднее | <1 |

|  |  |
| --- | --- |
| 1 | <1 |
| 2 | 1 |
| 3 | <1 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | <1 |
| 9 | 1 |
| 10 | <1 |
| Среднее | 1 |

QuickSort(100000)

|  |  |
| --- | --- |
| 1 | 10 |
| 2 | 9 |
| 3 | 10 |
| 4 | 9 |
| 5 | 9 |
| 6 | 9 |
| 7 | 9 |
| 8 | 9 |
| 9 | 9 |
| 10 | 9 |
| Среднее | 9.2 |

InsertionSort(10) InsertionSort(100) InsertionSort(1000) InsertionSort(10000)

|  |  |
| --- | --- |
| 1 | <1 |
| 2 | <1 |
| 3 | <1 |
| 4 | <1 |
| 5 | <1 |
| 6 | <1 |
| 7 | <1 |
| 8 | <1 |
| 9 | <1 |
| 10 | <1 |
| Среднее | <1 |

|  |  |
| --- | --- |
| 1 | <1 |
| 2 | <1 |
| 3 | <1 |
| 4 | <1 |
| 5 | <1 |
| 6 | <1 |
| 7 | <1 |
| 8 | <1 |
| 9 | <1 |
| 10 | <1 |
| Среднее | <1 |

|  |  |
| --- | --- |
| 1 | <1 |
| 2 | 1 |
| 3 | <1 |
| 4 | 1 |
| 5 | <1 |
| 6 | 1 |
| 7 | 1 |
| 8 | <1 |
| 9 | 1 |
| 10 | 1 |
| Среднее | 1 |

|  |  |
| --- | --- |
| 1 | 43 |
| 2 | 44 |
| 3 | 43 |
| 4 | 44 |
| 5 | 44 |
| 6 | 44 |
| 7 | 43 |
| 8 | 44 |
| 9 | 44 |
| 10 | 44 |
| Среднее | 43.7 |

InsertionSort(100000)

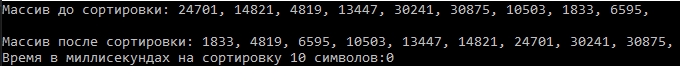
|  |  |
| --- | --- |
| 1 | 4302 |
| 2 | 4294 |
| 3 | 4307 |
| 4 | 4305 |
| 5 | 4293 |
| 6 | 4303 |
| 7 | 4290 |
| 8 | 4295 |
| 9 | 4310 |
| 10 | 4299 |
| Среднее | 4229.8 |

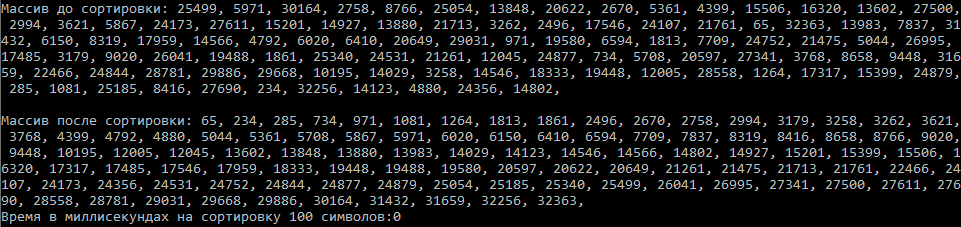
**Описание реализованных unit-тестов**

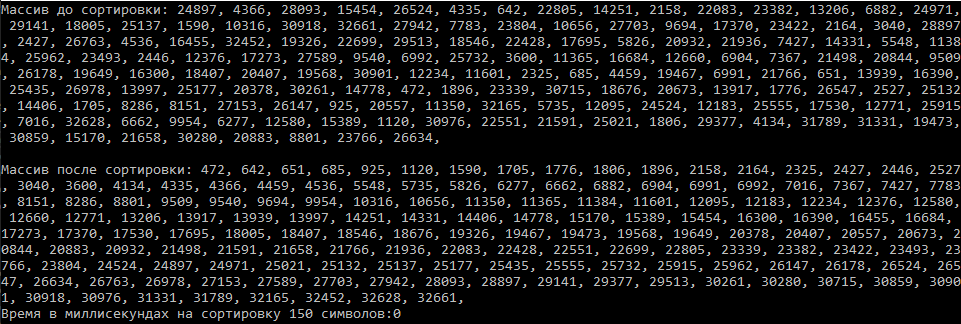
В таблице ниже представлены названия unit-тестов. Названия совпадают с названиями проверяемых функций.

|  |
| --- |
| Название теста |
| binarySearch |
| qickSort |
| insertionSort |
| bogoSort |
| countingSort |

**Пример работы**

****





**Листинг**

sort.h

#pragma once

int BinarySearch(int\* array, int left, int right, int key);

void QuickSort(int\* array, int left, int right);

void InsertionSort(int\* array, int sizeOfArray);

void BogoSort(int\* array, int sizeOfArray);

void CountingSort(int\* array, int sizeOfArray);

sort.cpp

#include <iostream>

#include "sort.h"

using namespace std;

int BinarySearch(int\* array, int leftB, int rightB, int key)

{

int center = 0;

while (true) {

center = (leftB + rightB) / 2;

if (key < array[center])

rightB = center - 1;

else if (key > array[center])

leftB = center + 1;

else

return center;

if (leftB > rightB)

return -1;

}

}

void QuickSort(int\* array, int leftB, int rightB)

{

int i = leftB;

int j = rightB;

int core = array[(i + j) / 2];

int temp;

while (i <= j) {

while (array[i] < core)

i++;

while (array[j] > core)

j--;

if (i <= j)

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

}

if (j > leftB)

QuickSort(array, leftB, j);

if (i < rightB)

QuickSort(array, i, rightB);

}

void InsertionSort(int\* array, int sizeOfArray)

{

int key;

int j;

for (int i = 1; i < sizeOfArray; i++) {

key = array[i];

j = i - 1;

while (j >= 0 && array[j] > key) {

array[j + 1] = array[j];

j--;

}

array[j + 1] = key;

}

}

void BogoSort(int\* array, int sizeOfArray)

{

bool sortcheck = false;

int index;

int temp;

while (sortcheck == false) {

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

index = rand() % sizeOfArray;

temp = array[i];

array[i] = array[index];

array[index] = temp;

}

for (int i = 0; i < sizeOfArray - 1; i++) {

sortcheck = true;

if (array[i] > array[i + 1]) {

sortcheck = false;

break;

}

}

}

}

void CountingSort(int\* array, int sizeOfArray)

{

int countArray[256] = { 0 };

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

countArray[array[i]] = countArray[array[i]] + 1;

int counter = 0;

for (int i = 0; i < 256; i++) {

for (int j = 0; j < countArray[i]; j++) {

array[counter] = i;

counter++;

}

}

}

main.cpp

#include <iostream>

#include <ctime>

#include <iomanip>

#include "sort.h"

#include <time.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

int array[150];

double time = 0;

int N = sizeof(array) / sizeof(array[0]);

cout << "Массив до сортировки: ";

for (int i = 0; i < N-1; i++) {

array[i] = rand();

cout << array[i]<< ", ";

}

cout << endl;

cout << endl;

double start = clock();

InsertionSort(array,N-1);

double end = clock();

time = end - start;

cout << "Массив после сортировки: ";

for (int i = 0; i < N - 1; i++) {

cout << array[i]<< ", ";

}

cout << endl;

cout << "Время в миллисекундах на сортировку " << N << " символов:"<< time;

return 0;

}