МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 3

по дисциплине: Алгоритмы и структуры данных
тема: «Сравнительный анализ методов сортировки (Pascal/C)»

Выполнил: ст. группы ПВ-202
Аладиб язан

Проверил:
Кабалянц Петр Степанович
Маньшин Илья Михайлович

Лабораторная работа № 3
«Сравнительный анализ методов сортировки (Pascal/C)»

Цель работы: изучение методов сортировки массивов и приобретение навыков в проведении сравнительного анализа различных методов сортировки.

Задания к работе:

1. Изучить временные характеристики алгоритмов.
2. Изучить методы сортировки:
    1) включением;
    2) выбором;
    3) обменом:
        3.1) улучшенная обменом 1;
        3.2) улучшенная обменом 2;
    4) Шелла;
    5) Хоара;
    6) пирамидальная.
3. Программно реализовать методы сортировки массивов.
4. Разработать и программно реализовать средство для проведения экспериментов по определению временных характеристик алгоритмов сортировки.
5. Провести эксперименты по определению временных характеристик алгоритмов сортировки. Результаты экспериментов представить в виде таблицы 9, клетки которой содержат количество операций сравнения при выполнении алгоритма сортировки массива с заданным количеством элементов. Провести эксперимент для упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов (для каждого типа массива заполнить отдельную таблицу).
6. Построить график зависимости количества операций сравнения от количества элементов в сортируемом массиве.
7. Определить аналитическое выражение функции зависимости количества операций сравнения от количества элементов в массиве.
8. Определить порядок функций временной сложности алгоритмов сортировки при сортировке упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов.

Программная реализация методов сортировки массивов:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int numCompares = 0;

/* функция сортировки включением */
void Sis(int A[], int nn) {
    numCompares = 0;
    int i, j, k;
    for (j = 1; j < nn; j++) {
        k = A[j];
        i = j - 1;
        numCompares++;
        while (k < A[i] && i >= 0) {
            numCompares++;
            A[i + 1] = A[i];
            i -= 1;
        }
        A[i + 1] = k;
    }
}

/*функция сортировки выбором */
void StrSel(int A[], int nn) {
    numCompares = 0;
    int i, j, x, k;
    for (i = 0; i < nn - 1; i++) {
        x = A[i];
        k = i;
        for (j = i + 1; j < nn; j++) {
            numCompares++;
            if (A[j] < x) {
                k = j;
                x = A[k];
            }
        }
        A[k] = A[i];
        A[i] = x;
    }
}

/* функция сортировки обменом */
void BblSort(int A[], int nn) {
    numCompares = 0;
    int i, j, k, p;
    for (i = 0; i < nn - 1; i++) {
        p = 0;
        for (j = nn - 1; j > i; j--) {
            numCompares++;
            if (A[j] < A[j - 1]) {
                k = A[j];
                A[j] = A[j - 1];
                A[j - 1] = k;
                p = 1;
            }
        }
/* Если перестановок не было, то сортировка выполнена */
        if (p == 0)
            break;
    }
```

```
}

/* функция сортировки методом Шелла */
void ShellSort(int a[], int n) {
    numCompares = 0;
    int i, j, k, hh, t, s;
    int h[1000];
    t = round(log(n) / log(3)) - 1;

    if (t < 1) {
        t = 0;
    };

    h[t] = 1;
    for (k = t - 1; k >= 1; k--) {
        h[k - 1] = 3 * h[k] + 1;
    }

    for (s = t; s >= 0; s--) {
        hh = h[s];
        for (j = hh; j <= n; j++) {
            i = j - hh;
            k = a[j];
            numCompares++;
            while ((k <= a[i]) && (i >= 0)) {
                numCompares++;
                a[i + hh] = a[i];
                i = i - hh;
            };
            a[i + hh] = k;
        }
    }
}

void QSort(int a[], int L, int R) {
    int x = a[L], i = L, j = R, t; // в качестве разделителя выбираем первый элемент
    while (i <= j) {
        numCompares++;
        while (a[i] < x) {
            numCompares++;
            i++;
        }

        numCompares++;
        while (a[j] > x) {
            numCompares++;
            j--;
        }

        if (i <= j) {
            t = a[i];
            a[i] = a[j];
            a[j] = t;
            i++;
            j--;
        }
    }
    if (L < j)
        QSort(a, L, j);
    if (i < R)
        QSort(a, i, R);
}
```

```
/*функция сортировки методом Хоара*/
void HoarSort(int a[], int n) {
    numCompares = 0;
    QSort(a, 1, n);
}

/* ========================================== */
void Sift(int A[], int L, int R) {
    int i, j, x, k;
    i = L;
    j = 2 * L + 1;
    x = A[L];

    numCompares++;
    if ((j < R) && (A[j] < A[j + 1]))
        j++;
    numCompares++;
    while ((j <= R) && (x < A[j])) {
        numCompares++;
        k = A[i];
        A[i] = A[j];
        A[j] = k;
        i = j;
        j = 2 * j + 1;
        numCompares++;
        if ((j < R) && (A[j] < A[j + 1]))
            j++;
    }
}

/* пирамидальная функция сортировки */
void HeapSort(int A[], int nn) {
    numCompares = 0;
    int L, R, x, i;
    L = nn / 2;
    R = nn - 1;
/* Построение пирамиды из исходного массива */
    while (L > 0) {
        L = L - 1;
        Sift(A, L, R);
    }
/* Сортировка: пирамида => отсортированный массив */
    while (R > 0) {
        x = A[0];
        A[0] = A[R];
        A[R] = x;
        R--;
        Sift(A, L, R);
    }
}

void getSortedArray(int *a, int n) {
    for (int i = 0; i < n; i++) {
        a[i] = i;
    }
}

void getReversedArray(int *a, int n) {
    for (int i = 0; i < n; i++) {
        a[i] = n - 1 - i;
    }
}
```

```
void getRandomArray(int *a, int n) {
    for (int i = 0; i < n; i++) {
        a[i] = rand() % n;
    }
}

int main() {
    srand(42);
    int array[1000];

    for (int i = 5; i <= 50; i += 5) {
        getSortedArray(array, i);
        HoarSort(array, i);

        printf("%d,", numCompares);
    }

    printf("\n");

    for (int i = 5; i <= 50; i += 5) {
        getReversedArray(array, i);
        HoarSort(array, i);

        printf("%d,", numCompares);
    }

    printf("\n");

    for (int i = 5; i <= 50; i += 5) {
        getRandomArray(array, i);
        HoarSort(array, i);

        printf("%d,", numCompares);
    }

    printf("\n");
}
```

Результаты эксперимента представлены на рисунке 1-3.

Сортировка	Количество элементов в массиве								
	500	1000	1500	2000	2500	3000	3500	4000	4500
Включением	0	0	0	0	0	0	0	0	0
Выбором	0,001	0,003	0,005	0,009	0,014	0,02	0,029	0,037	0,047
Обменом	0,001	0,004	0,010	0,017	0,026	0,037	0,052	0,067	0,085
Обменом 1	0,001	0,004	0,010	0,016	0,025	0,038	0,051	0,068	0,083
Обменом 2	0,001	0,005	0,013	0,021	0,027	0,041	0,055	0,07	0,089
Шелла	0,001	0,001	0,001	0,001	0,002	0,002	0,002	0,003	0,003
Хоара	0,001	0,001	0,001	0,001	0,001	0,002	0,002	0,002	0,003
Пирамидальная	0,002	0,003	0,005	0,007	0,009	0,011	0,014	0,016	0,017

Рисунок 1. Наилучшие случаи

Сортировка	Количество элементов в массиве								
	500	1000	1500	2000	2500	3000	3500	4000	4500
Включением	0	0,002	0,006	0,010	0,015	0,02	0,026	0,034	0,043
Выбором	0	0,002	0,006	0,009	0,014	0,02	0,028	0,036	0,046
Обменом	0,002	0,007	0,016	0,028	0,045	0,065	0,09	0,118	0,152
Обменом 1	0,001	0,007	0,016	0,029	0,047	0,068	0,094	0,123	0,158
Обменом 2	0,001	0,005	0,011	0,020	0,032	0,046	0,065	0,086	0,11
Шелла	0	0,00	0,005	0,009	0,014	0,020	0,028	0,035	0,046
Хоара	0	0	0	0	0,001	0,001	0,001	0,001	0,001
Пирамидальная	0,003	0,006	0,01	0,013	0,017	0,020	0,025	0,028	0,032

Рисунок 2. Средние случаи

Сортировка	Количество элементов в массиве								
	500	1000	1500	2000	2500	3000	3500	4000	4500
Включением	0,001	0,004	0,01	0,017	0,028	0,039	0,053	0,068	0,089
Выбором	0	0,003	0,005	0,009	0,014	0,02	0,029	0,037	0,047
Обменом	0,002	0,009	0,018	0,032	0,050	0,075	0,101	0,131	0,166
Обменом 1	0,002	0,009	0,02	0,037	0,057	0,081	0,109	0,141	0,180
Обменом 2	0,002	0,006	0,015	0,025	0,039	0,057	0,077	0,101	0,128
Шелла	0,001	0,004	0,01	0,018	0,028	0,041	0,055	0,072	0,091
Хоара	0,001	0,001	0,004	0,005	0,009	0,012	0,017	0,023	0,035
Пирамидальная	0,002	0,006	0,008	0,012	0,0014	0,018	0,021	0,024	0,027

Рисунок 3. Наихудшие случаи

Из-за того, что функции количества операций сравнения и временной сложности имеют один порядок, графики их также совпадают. Для иллюстрации графиков зависимости количества операций сравнения от размера массива, воспользуемся всеми тремя таблицами и для каждой сортировки построим на одной плоскости график наилучшего, среднего и наихудшего времени выполнения. Эти графики, а также ФВС данных сортировок изображены на рис. 4 – 11.

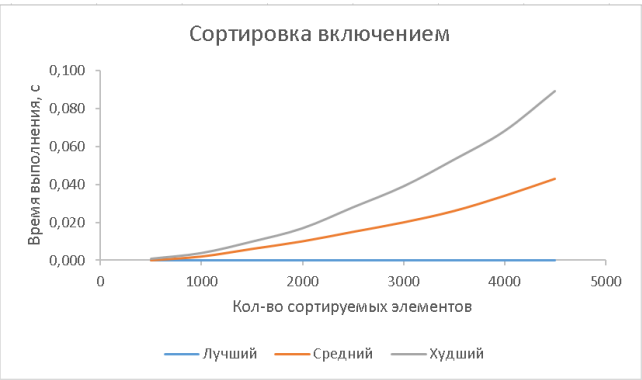

Рисунок 4

ФВС сортировки включением в случае упорядоченного массива O(N), в случае упорядоченности в обратном порядке $O(N^2)$, а в среднем случае она состовляет $O(N^2)$

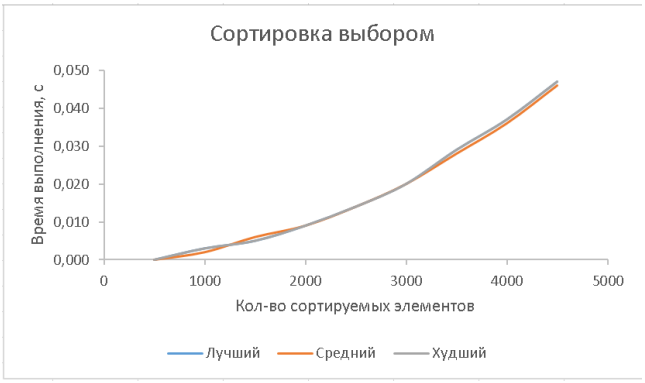

Рисунок 5

ФВС сортировки выбором во всех случаях составляет $O(N^2)$

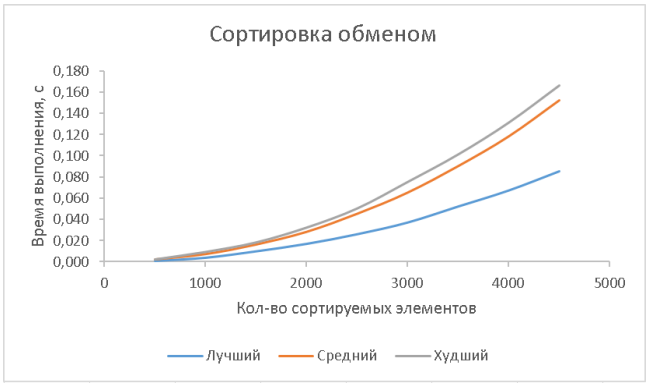

Рисунок 6

ФВС сортировки обменом в случаях упорядоченности в обратном порядке и в среднем составляет $O(N^2)$, в лучшем случае $O(N)$

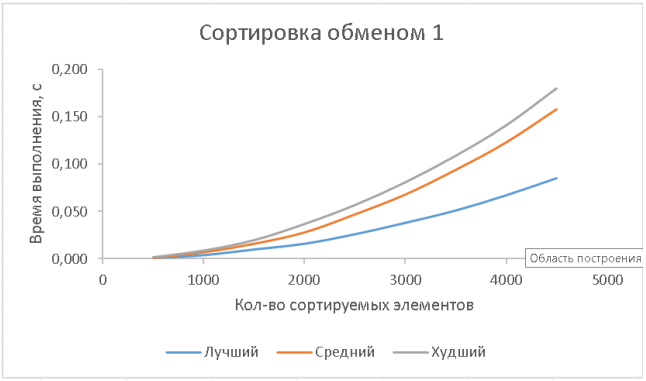

Рисунок 7

ФВС сортировки обменом в случаях упорядоченности в обратном порядке и в среднем составляет $O(N^2)$, в лучшем случае $O(N)$

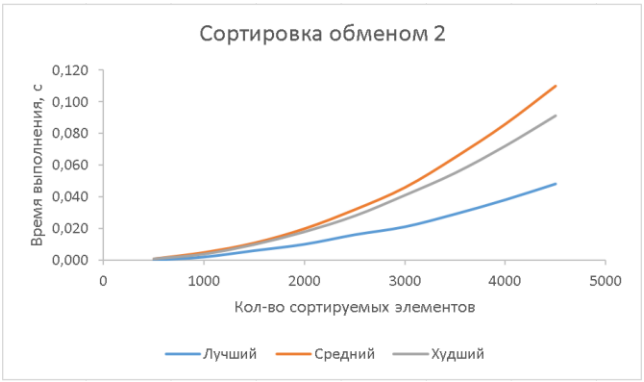

Рисунок 8

ФВС сортировки обменом в случаях упорядоченности в обратном порядке и в среднем составляет $O(N^2)$, в лучшем случае $O(N)$

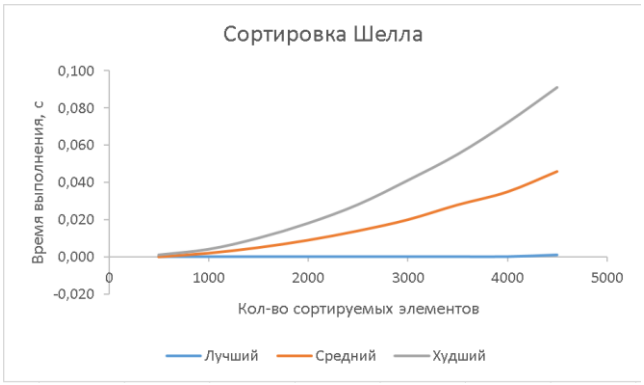

Рисунок 9

Первые подмассивы, создаваемые с первым шагом, сортируются со сложностью $O(N^2)$, но так как они довольно малы, то она выполняется быстро. Ближе к концу массив в целом становится более упорядоченным, поэтому порядок ФВС сортировки Шелла лежит в пределах между $O(N)$ и $O(N^2)$

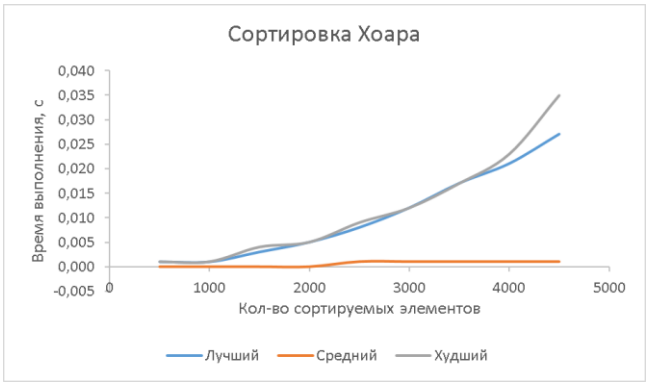

Рисунок 10

Если массив разделяется каждый раз на две равные части, то порядок ФВС равен $O(N \log_2 N)$. Если каждый раз одна из частей массива содержит не более одного элемента, то порядок ФВС равен $O(N^2)$

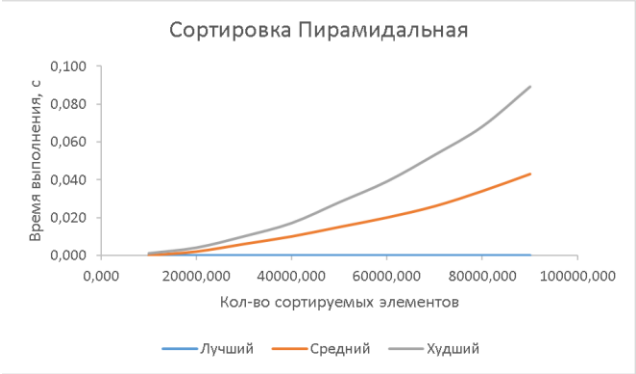

Рисунок 10

Вывод :

Я изучил методы сортировки массивов и приобрел навыки в проведении сравнительного анализа различных методов сортировки.