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-211

Чувилко Илья Романович

Проверил:

Синюк Василий Григорьевич

Белгород 2022 г.

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

Задания

1. Изучить временные характеристики алгоритмов.
2. Изучить методы сортировки:
   - 2.1 включением;
   - 2.2 выбором
   - 2.3 обменом:
     - 2.3.1 улучшенная обменом 1;
     - 2.3.2 улучшенная обменом 2;
   - 2.4 Шелла;
   - 2.5 Хоара;
   - 2.6 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.

```c
#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");
}
```

Задание 5

Результаты экспериментов (Неупорядоченный массив)

Сортировка	Количество элементов в массиве								
	5	10	15	20	25	30	35	40	45
Включением	29	109	239	419	649	929	1259	1639	2069
Выбором	29	109	239	419	649	929	1259	1639	2069
Обменом	29	109	239	419	649	929	1259	1639	2069
Обменом 1	32	117	252	437	672	957	1292	1677	2112
Обменом 2	34	119	254	439	674	959	1294	1679	2114
Шелла	31	91	153	249	303	404	577	643	716
Хоара	53	128	233	358	513	688	893	1118	1373
Пирамидальная	38	103	181	280	365	478	582	687	810

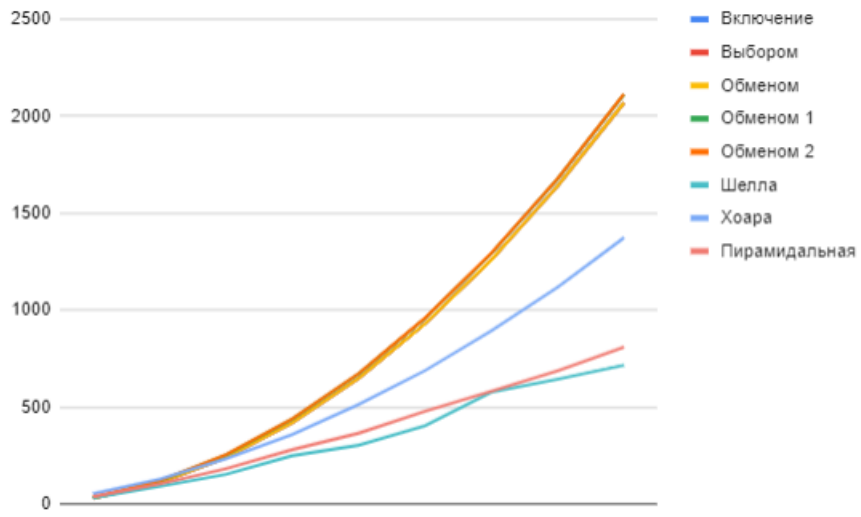
Результаты экспериментов (Упорядоченный массив)

Сортировка	Количество элементов в массиве								
	5	10	15	20	25	30	35	40	45
Включением	13	28	43	58	73	88	103	118	133
Выбором	29	109	239	419	649	929	1259	1639	2069
Обменом	29	109	239	419	649	929	1259	1639	2069
Обменом 1	10	20	30	40	50	60	70	80	90
Обменом 2	6	11	16	21	26	31	36	41	46
Шелла	26	73	109	195	243	291	440	497	563
Хоара	50	125	225	350	500	675	875	1100	1350
Пирамидальная	44	127	218	332	445	559	689	826	954

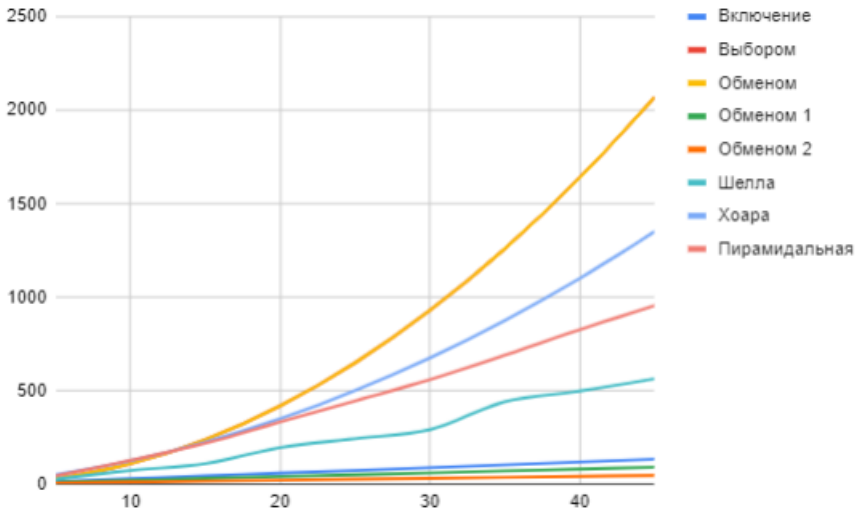
Результаты экспериментов (Упорядоченный в обратном порядке)

Сортировка	Количество элементов в массиве								
	5	10	15	20	25	30	35	40	45
Включением	21	71	157	251	359	572	772	781	1168
Выбором	29	109	239	419	649	929	1259	1639	2069
Обменом	29	109	239	419	649	929	1259	1639	2069
Обменом 1	19	95	252	436	650	957	1291	1677	2066
Обменом 2	30	60	163	317	505	697	910	1157	1496
Шелла	32	88	160	261	350	417	570	691	790
Хоара	46	119	179	258	358	415	506	615	728
Пирамидальная	40	123	199	306	416	523	662	739	897

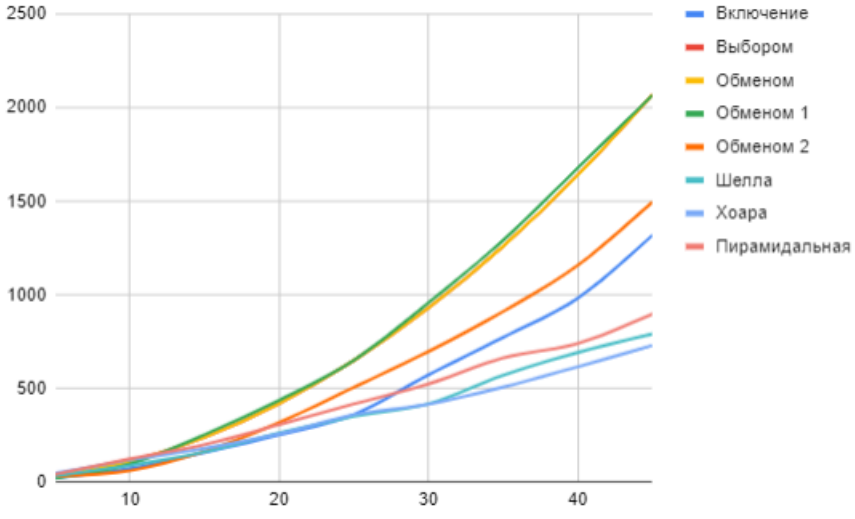
Задание 6
Неупорядоченный массив



Упорядоченный массив



Упорядоченный в обратном порядке



Задание 7 Аналитические выражение функции зависимости количества операций сравнения от количества элементов в массиве.

включением	лучший случай: $N-1$; средний и худший: $(N-1)*N/2$
выбором	$(N-1)*N/2$
обменом	$(N-1)*N/2$
обменом 1	лучший случай: $N-1$; средний и худший: $(N-1)*N/2$
обменом 2	лучший случай: $N-1$; средний и худший: $(N-1)*N/2$
Шелла	зависит от выбранных шагов
Хоара	удачный разделитель:$N+2*(N/2)+\ldots+m*(N*m), m=logN$
пирамидальная	$[log_2N]+[log_2(N-1)]+\ldots+[log_22]$

Задание 8 Определить порядок функций временной сложности алгоритмов сортировки при сортировке упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов.

Название сортировки	лучший случай	средний случай	худший случай
включением	$O(N)$	$O(N^2)$	$O(N^2)$
выбором	$O(N^2)$	$O(N^2)$	$O(N^2)$
обменом	$O(N^2)$	$O(N^2)$	$O(N^2)$
обменом 1	$O(N)$	$O(N^2)$	$O(N^2)$
обменом 2	$O(N)$	$O(N^2)$	$O(N^2)$
Шелла	$O(N)$	$O(N^2)$	$O(N*log^2N)$
Хоара	$O(N*logN)$	$O(N^2)$	$O(N*logN)$
пирамидальная	$O(N*logN)$	$O(N*logN)$	$O(N*logN)$

К о н т р о л ь н ы е в о п р о с ы

1. Что такое временная сложность алгоритма?

2. Почему функцию временной сложности нельзя использовать для оценки алгоритма?

3. Что такое порядок функции? Как определяется порядок функции, заданной

многочленом?
4. Как можно определить порядок функции временной сложности алгоритма?
5. Что называется сортировкой?
6. В каком случае метод сортировки называется устойчивым?
7. Как выполняется сортировка включением?
8. Зависит ли время сортировки включением от упорядоченности массива?
9. Зависит ли порядок функции временной сложности сортировки включением от упорядоченности массива?
10. Выполните анализ сортировки включением.
11. Реализуйте алгоритм сортировки включением на языке программирования.
12. Как выполняется сортировка выбором?
13. Зависит ли время сортировки выбором от упорядоченности массива?
14. Зависит ли порядок функции временной сложности сортировки выбором от упорядоченности массива?
15. Выполните анализ сортировки выбором.
16. Реализуйте алгоритм сортировки выбором на языке программирования.
17. Как выполняется сортировка обменом?
18. Зависит ли время сортировки обменом от упорядоченности массива?
19. Зависит ли порядок функции временной сложности сортировки обменом от упорядоченности массива?
20. Выполните анализ сортировки обменом.
21. Реализуйте алгоритм сортировки обменом на языке программирования.
22. Как можно улучшить сортировку обменом?
23. Почему сортировка Шелла быстрее сортировки вставками?
24. Выполните итеративную реализацию сортировки Хоара.
25. Чем пирамидальная сортировка отличается от сортировки выбором?

Вывод: В ходе выполнения данной лабораторной работы были изучены методы сортировки, приобретены навыки в проведении сравнительного анализа различных методов сортировки