

Министерство науки и высшего образования РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа киберфизических систем и управления
УДК 004.421

УТВЕРЖДАЮ

« ____ » _____ г.

Отчет

по дисциплине «Теория и технология программирования»
Лабораторная работа №6 «Методы сортировки», вариант 17

Выполнил:
Студент
гр.3530902/90001
Непушкин С.А.

подпись, дата

Доцент ВШКФСИУ,
Кандидат технических наук
С. В. Хлопин

подпись, дата

Санкт-Петербург

2020

1. Цель работы.

Цель задания - изучить методы сортировки данных.

2. Задание:

Составить программу для сортировки массива данных методами: пузырьковой, отбора, вставки, Шелла и быстрой сортировки. Вывести на экран неупорядоченную (один раз) и упорядоченные (для каждого из методов) массивы данных. Составить сравнительную таблицу эффективности методов, в которой необходимо указать число сравнений и перестановок переменных в каждом методе сортировки.

Неупорядоченная матрица задается один раз случайным образом, далее она используется для каждого из методов сортировки.

Индивидуальный вариант:

Упорядочить каждый столбец матрицы по убыванию.

3. Текст программы

```
#include <iostream>
#include <stdio.h>
#include <locale.h>
#include <time.h>
using namespace std;

int isDigit(char c)
{
    return ((c >= '0') && (c <= '9'));
}

int charToDigit(char c)
{
    if (isDigit(c))
    {
        return c - '0';
    }
    return -1;
}

//0 - Ошибка; 1 - Ввелось корректно
int inputInt(int &var)
{
    char c = 0;
    int value = -1;
    while ((c = getchar()) != '\n')
    {
        if (!isDigit(c))
        {
            return 0;
        }
        if (value == -1)
        {
            value = 0;
        }
        value = value * 10 + charToDigit(c);
    }
    if (value < 1)
    {

```

```

        return 0;
    }
    var = value;
    return 1;
}

void initMatrix(int** &matrix, int rows, int cols)
{
    matrix = new int*[rows];
    for (int i = 0; i < rows; i++)
    {
        matrix[i] = new int[cols];
        for (int j = 0; j < cols; j++)
        {
            matrix[i][j] = rand() % 100;
        }
    }
}

int** copyMatrix(int** matrix, int rows, int cols)
{
    int** res = nullptr;
    initMatrix(res, rows, cols);
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            res[i][j] = matrix[i][j];
        }
    }
    return res;
}

void deleteMatrix(int** &matrix, int rows, int cols)
{
    for (int i = 0; i < rows; i++)
    {
        delete[] matrix[i];
    }
    delete matrix;
}

void printMatrix(int** &matrix, int rows, int cols)
{
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            printf("%2d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void swap(int &a, int &b)
{
    int c = a;
    a = b;
    b = c;
}

void mixColMatrix(int** matrix, int rows, int col)
{
    for (int i = 0; i < rows; i++)
    {

```

```

        swap(matrix[i][col], matrix[rand() % rows][col]);
    }
}

void mixMatrix(int** matrix, int rows, int cols)
{
    for (int j = 0; j < cols; j++)
    {
        for (int i = 0; i < rows; i++)
        {
            swap(matrix[i][j], matrix[rand() % rows][j]);
        }
    }
}

pair<int, int> bubbleColsSort(int** inmatrix, int rows, int cols)
{
    int** matrix = copyMatrix(inmatrix, rows, cols);
    int per = 0;
    int srav = 0;

    for (int k = 0, shift = 1; k < cols; k++)
    {
        for (int i = 0; (i < rows) && (shift > 0); i++)
        {
            shift = 0;
            for (int j = 0; j < rows - i - 1; j++)
            {
                if (matrix[j][k] < matrix[j + 1][k])
                {
                    swap(matrix[j][k], matrix[j + 1][k]);
                    per++;
                    shift++;
                }
            }
            srav++;
        }
        shift = 1;
    }

    printf("Перестановки: %d \t Сравнения: %d \n", per, srav);
    printMatrix(matrix, rows, cols);
    deleteMatrix(matrix, rows, cols);

    return pair<int, int>(per, srav);
}

pair<int, int> selectionColsSort(int** inmatrix, int rows, int cols)
{
    int** matrix = copyMatrix(inmatrix, rows, cols);
    int per = 0;
    int srav = 0;
    if (rows != 1)
    {
        for (int k = 0; k < cols; k++)
        {
            for (int i = 0; i < rows - 1; i++)
            {
                int index = i;
                for (int j = i + 1; j < rows; j++)
                {
                    if (matrix[j][k] > matrix[index][k])
                    {
                        index = j;
                    }
                }
            }
        }
    }
}

```

```

        srav++;
    }
    if (index != i)
    {
        swap(matrix[index][k], matrix[i][k]);
        per++;
    }
}
}

printf("Перестановки: %d \t Сравнения: %d \n", per, srav);
printMatrix(matrix, rows, cols);
deleteMatrix(matrix, rows, cols);
return pair<int, int>(per, srav);
}

pair<int, int> insertColsSort(int** inmatrix, int rows, int cols)
{
    int** matrix = copyMatrix(inmatrix, rows, cols);
    int per = 0;
    int srav = 0;

    for (int k = 0; k < cols; k++)
    {
        for (int i = 1; i < rows; i++)
        {
            int temp = matrix[i][k];
            int j = 0;
            for (j = i; j > 0 && temp > matrix[j - 1][k]; j--)
            {
                matrix[j][k] = matrix[j - 1][k];
                per++;
                srav++;
            }
            matrix[j][k] = temp;
        }
    }

    printf("Перестановки: %d \t Сравнения: %d \n", per, srav);
    printMatrix(matrix, rows, cols);
    deleteMatrix(matrix, rows, cols);
    return pair<int, int>(per, srav);
}

pair<int, int> shellColsSort(int** inmatrix, int rows, int cols)
{
    int** matrix = copyMatrix(inmatrix, rows, cols);
    int per = 0;
    int srav = 0;

    for (int k = 0; k < cols; k++)
    {
        for (int diff = rows / 2; diff > 0; diff /= 2)
        {
            for (int i = diff; i < rows; i++)
            {
                int temp = matrix[i][k];
                int j = 0;
                for (j = i; j >= diff && temp > matrix[j - diff][k]; j -= diff)
                {
                    matrix[j][k] = matrix[j - diff][k];
                    srav++;
                    per++;
                }
            }
        }
    }
}

```

```

        }
        matrix[j][k] = temp;
    }
}

printf("Перестановки: %d \t Сравнения: %d \n", per, srav);
printMatrix(matrix, rows, cols);
deleteMatrix(matrix, rows, cols);
return pair<int, int>(per, srav);
}

pair<int, pair<int, int>> partition(int** inmatrix, int low, int high, int col)
{
    int per = 0;
    int srav = 0;
    int pivot = inmatrix[high][col];
    int i = low;
    for (int j = low; j <= high - 1; j++)
    {
        if (inmatrix[j][col] > pivot)
        {
            swap(inmatrix[i][col], inmatrix[j][col]);
            i++;
            per++;
        }
        srav++;
    }
    if (i != high)
    {
        swap(inmatrix[i][col], inmatrix[high][col]);
        per++;
    }
    return pair<int, pair<int, int>>(i, pair<int, int>(per, srav));
}

pair<int, int> quickSort(int** inmatrix, int low, int high, int col)
{
    pair<int, int> res(0, 0);
    if (low < high)
    {
        pair<int, pair<int, int>> pi = partition(inmatrix, low, high, col);
        res.first += pi.second.first;
        res.second += pi.second.second;
        quickSort(inmatrix, low, pi.first - 1, col);
        quickSort(inmatrix, pi.first + 1, high, col);
    }
    return res;
}

pair<int, int> quickColsSort(int** inmatrix, int rows, int cols)
{
    int** matrix = copyMatrix(inmatrix, rows, cols);
    int per = 0;
    int srav = 0;

    for (int k = 0; k < cols; k++)
    {
        pair<int, int> temp = quickSort(matrix, 0, rows - 1, k);
        per += temp.first;
        srav += temp.second;
    }

    printf("Перестановки: %d \t Сравнения: %d \n", per, srav);
    printMatrix(matrix, rows, cols);
}

```

```

        deleteMatrix(matrix, rows, cols);
        return pair<int, int>(per, srav);
    }

int colSorted(int** matrix, int rows, int col)
{
    for (int i = 0; i < rows - 1; i++)
    {
        if (matrix[i][col] < matrix[i + 1][col])
        {
            return 0;
        }
    }
    return 1;
}

pair<int, int> monkeyColSort(int** inmatrix, int rows, int cols)
{
    int** matrix = copyMatrix(inmatrix, rows, cols);
    int per = 0;
    int srav = 0;
    for (int k = 0; k < cols; k++)
    {
        while (!colSorted(matrix, rows, k)) {
            mixColMatrix(matrix, rows, k);
        }
        cout << endl;
        printMatrix(matrix, rows, cols);
    }
    printf("Перестановки: %d \t Сравнения: %d \n", per, srav);
    printMatrix(matrix, rows, cols);
    deleteMatrix(matrix, rows, cols);
    return pair<int, int>(per, srav);
}

int main()
{
    srand(time(0));
    setlocale(LC_ALL, "Russian");

    printf("Введите количество строк матрицы : ");
    int n = 0;
    if (!inputInt(n))
    {
        printf("Некорректный ввод числа\n");
        return 0;
    }

    printf("Введите количество столбцов матрицы : ");
    int m = 0;
    if (!inputInt(m))
    {
        printf("Некорректный ввод числа\n");
        return 0;
    }

    int** matrix = nullptr;
    initMatrix(matrix, n, m);
    printMatrix(matrix, n, m);
    //monkeyColSort(matrix, n, m);

    pair<int, int> res[5];

    printf("\n\nСортировка пузырьком:\n");
    res[0] = bubbleColsSort(matrix, n, m);

```

```

printf("\n\nСортировка отбором:\n");
res[1] = selectionColsSort(matrix, n, m);

printf("\n\nСортировка вставками:\n");
res[2] = insertColsSort(matrix, n, m);

printf("\n\nСортировка Шелла:\n");
res[3] = shellColsSort(matrix, n, m);

printf("\n\nБыстрая сортировка:\n\n");
res[4] = quickColsSort(matrix, n, m);

deleteMatrix(matrix, n, m);

printf("_____ \n");
printf("_____ |Сортировка пузырьком| Сортировка отбором |Сортировка вставками|
Сортировка Шелла | Быстрая сортировка | \n");

printf("Перестановки|");
for (int i = 0; i < 5; i++)
{
    printf("%20d|", res[i].first);
}
printf("\nСравнения |");
for (int i = 0; i < 5; i++)
{
    printf("%20d|", res[i].second);
}
printf("\n_____ | _____ | _____ | _____ |
_____ | _____ | \n");

return 0;
}

```

4. Пример работы программы

```

Консоль отладки Microsoft Visual Studio
Введите количество строк матрицы : 6
Введите количество столбцов матрицы : 1
93
11
54
76
8
4

Сортировка пузырьком:
Перестановки: 3      Сравнения: 12
93
76
54
11
8
4

Сортировка отбором:
Перестановки: 1      Сравнения: 15
93
76
54
11
8
4

Сортировка вставками:
Перестановки: 3      Сравнения: 3
93
76
54
11
8
4

```



```
Консоль отладки Microsoft Visual Studio

Сортировка Шелла:
Перестановки: 3      Сравнения: 3
93
76
54
11
8
4

Быстрая сортировка:
Перестановки: 5      Сравнения: 5
93
76
54
11
8
4



|              | Сортировка пузырьком | Сортировка отбором | Сортировка вставками | Сортировка Шелла | Быстрая сортировка |
|--------------|----------------------|--------------------|----------------------|------------------|--------------------|
| Перестановки | 3                    | 1                  | 3                    | 3                | 5                  |
| Сравнения    | 12                   | 15                 | 3                    | 3                | 5                  |



G:\Project6.3\Debug\Project6.3.exe (процесс 13320) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу...
```

```
Консоль отладки Microsoft Visual Studio

Введите количество строк матрицы : 5
Введите количество столбцов матрицы : 10
64 6 24 52 10 22 74 40 36 84
92 81 59 39 0 36 99 86 56 82
97 57 10 99 73 28 39 86 30 53
55 65 36 22 57 55 77 75 82 17
7 65 50 74 17 87 3 38 16 72

Сортировка пузырьком:
Перестановки: 46      Сравнения: 92
97 81 59 99 73 87 99 86 82 84
92 65 50 74 57 55 77 86 56 82
64 65 36 52 18 36 74 75 36 72
55 57 24 39 17 28 39 40 30 53
7 6 10 22 0 22 3 38 16 17

Сортировка отбором:
Перестановки: 28      Сравнения: 100
97 81 59 99 73 87 99 86 82 84
92 65 50 74 57 55 77 86 56 82
64 65 36 52 18 36 74 75 36 72
55 57 24 39 17 28 39 40 30 53
7 6 10 22 0 22 3 38 16 17

Сортировка вставками:
Перестановки: 46      Сравнения: 46
97 81 59 99 73 87 99 86 82 84
92 65 50 74 57 55 77 86 56 82
64 65 36 52 18 36 74 75 36 72
55 57 24 39 17 28 39 40 30 53
7 6 10 22 0 22 3 38 16 17
```

```
Консоль отладки Microsoft Visual Studio

Сортировка Шелла:
Перестановки: 34      Сравнения: 34
97 81 59 99 73 87 99 86 82 84
92 65 50 74 57 55 77 86 56 82
64 65 36 52 18 36 74 75 36 72
55 57 24 39 17 28 39 40 30 53
7 6 10 22 0 22 3 38 16 17

Быстрая сортировка:
Перестановки: 30      Сравнения: 40
97 81 59 99 73 87 99 86 82 84
92 65 50 74 57 55 77 86 56 82
64 65 36 52 18 36 74 75 36 72
55 57 24 39 17 28 39 40 30 53
7 6 10 22 0 22 3 38 16 17



|              | Сортировка пузырьком | Сортировка отбором | Сортировка вставками | Сортировка Шелла | Быстрая сортировка |
|--------------|----------------------|--------------------|----------------------|------------------|--------------------|
| Перестановки | 46                   | 28                 | 46                   | 34               | 30                 |
| Сравнения    | 92                   | 100                | 46                   | 34               | 40                 |



G:\Project6.3\Debug\Project6.3.exe (процесс 14480) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу...
```

Вывод

В ходе лабораторной работы я:

- столкнулся с проблемой подсчета сравнений в быстрой сортировке, но успешно решил ее
- узнал о классе «pair»
- узнал, как можно оптимизировать метод сортировки «пузырьком»
- подробно разобрал метод быстрой сортировки и его алгоритм
- столкнулся с проблемой сохранения неотсортированного массива, но успешно решил ее