Министерство цифрового развития, связи и массовых коммуникаций РФ  
Федеральное государственное бюджетное образовательное учреждение высшего образования

Уральский технический институт связи и информатики (филиал) ФГБОУ ВО "Сибирский государственный университет телекоммуникаций и информатики" в г. Екатеринбурге (УрТИСИ СибГУТИ)



КАФЕДРА

ВМиФ

**ОТЧЕТ**

По дисциплине «Структуры и алгоритмы обработки данных»

лабораторная работа № 3

«Сортировки»

|  |  |
| --- | --- |
| Выполнил: | Студент гр. ПЕ-12б  Цыганков Валерий Сергеевич |
| Проверил: | преподаватель  Белкина А.В. |

Екатеринбург, 2022

**1. Цель:**

1.1. Приобрести навыки в реализации алгоритмов основных сортировок.

1.2. Научиться оценивать временные и пространственные характеристики реализованного алгоритма.

**2. Текст решения:**

#include <iostream>

#include <ctime>

#include <locale.h>

#include <string>

using namespace std;

template <class T>

void arrOut(T arr[], int n);

void arrRand(int arr[], int n, int maxim);

void bubbleSort(int arr[], int n);

void shakerSort(int arr[], int n);

void insertSort(int arr[], int n);

void podshetSort(int arr[], int n);

void extractSort(int arr[], int n);

void treeSort(int arr[], int n);

void shellSort(int arr[], int n);

void kStat(int arr[], int n, int k);

void mergeSort(int arr[], int n);

void radixSortStrings(string arrStr[], int n);

void radixSortDates(int arr[], int size);

void dateToNumbers(string arr[], int a[], int n);

void numbersToDate(int arr[], string a[], int n);

int main()

{

    setlocale(LC\_ALL, "Rus");

    int i, loop = 1, command = 0;

    srand(time(NULL));

    const int n = 8, maxim = 100;

    // STRINGS

    /\*

        string arrStr[n] = {"week", "eye", "ilya", "window", "nevod", "il", "igla", "vili"};

        //{"неделя", "око", "илья", "окно", "невод", "ил", "игла", "вилы"};

        arrOut(arrStr, n);

        radixSortStrings(arrStr, n);

        arrOut(arrStr, n);

    \*/

    // DATES

    /\*

    puts("----------------------------------");

    const int k = 5;

    string arrDate[k] = {"23.01.19",

                         "12.02.19",

                         "15.12.18",

                         "16.02.18",

                         "04.01.17"};

    int ar[k];

    dateToNumbers(arrDate, ar, k);

    arrOut(ar, k);

    radixSortDates(ar, k);

    numbersToDate(ar, arrDate, k);

    arrOut(arrDate, k);

    \*/

    // MAIN

    ////

    int arr[n];

    arrRand(arr, n, maxim);

    arrOut(arr, n);

    while (loop == 1)

    {

        arrRand(arr, n, maxim);

        cout << "Choose sort method: " << endl;

        cout << "1: Bubble Sort" << endl;

        cout << "2: Shaker Sort" << endl;

        cout << "3: Insert Sort" << endl;

        cout << "4: Podshet Sort" << endl;

        cout << "5: Extract Sort" << endl;

        cout << "6: Tree Sort" << endl;

        cout << "7: Shell Sort" << endl;

        cout << "8: k-Statistic" << endl;

        cout << "9: Merge Sort" << endl;

        cin >> command;

        arrOut(arr, n);

        switch (command)

        {

        case 1:

            bubbleSort(arr, n);

            arrOut(arr, n);

            break;

        case 2:

            shakerSort(arr, n);

            arrOut(arr, n);

            break;

        case 3:

            insertSort(arr, n);

            arrOut(arr, n);

            break;

        case 4:

            podshetSort(arr, n);

            arrOut(arr, n);

            break;

        case 5:

            extractSort(arr, n);

            arrOut(arr, n);

            break;

        case 6:

            treeSort(arr, n);

            arrOut(arr, n);

            break;

        case 7:

            shellSort(arr, n);

            arrOut(arr, n);

            break;

        // FIXME

        case 8:

            int a;

            cout << "Vvedite posiciyu chisla" << endl;

            cin >> a;

            kStat(arr, n, a);

            break;

        case 9:

            mergeSort(arr, n);

            arrOut(arr, n);

            break;

        default:

            break;

        }

        cout << "1 - continue" << endl;

        cin >> loop;

        cout << "---------------------" << endl;

    }

    return 0;

}

void arrRand(int arr[], int n, int maxim)

{

    for (int i = 0; i < n; i++)

    {

        arr[i] = rand() % maxim;

    }

}

template <class T>

void arrOut(T arr[], int n)

{

    cout << "Array: " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << ' ';

    }

    cout << endl;

}

void bubbleSort(int arr[], int n)

{

    int i, j;

    bool flag = false;

    for (i = 0; i < n - 1; i++)

    {

        if (flag)

            break;

        flag = true;

        for (j = 0; j < n - i - 1; j++)

            if (arr[j] > arr[j + 1])

            {

                swap(arr[j], arr[j + 1]);

                flag = false;

            }

    }

}

void shakerSort(int arr[], int n)

{

    int left = 0, right = n - 1, i;

    while (left <= right)

    {

        for (i = right; i >= left; i--)

        {

            if (arr[i - 1] > arr[i])

            {

                swap(arr[i - 1], arr[i]);

            }

        }

        left++;

        for (i = left; i <= right; i++)

        {

            if (arr[i - 1] > arr[i])

            {

                swap(arr[i - 1], arr[i]);

            }

        }

        right--;

        arrOut(arr, n);

    }

}

void insertSort(int arr[], int n)

{

    int i, j = 0, temp;

    for (i = 1; i < n - 1; i++)

    {

        temp = arr[i];

        j = i;

        while ((j > 0) && (arr[j - 1] > temp))

        {

            arr[j] = arr[j - 1];

            j--;

        }

        arr[j] = temp;

    }

}

void podshetSort(int arr[], int n)

{

    int \*tempArr = new int[n];

    int k;

    for (int i = 0; i < n - 1; i++)

    {

        tempArr[i] = 0;

    }

    for (int i = 0; i < n - 1; i++)

    {

        k = 0;

        for (int j = 0; j < n - 1; j++)

        {

            if ((arr[j] < arr[i]) || ((arr[j] == arr[i]) && (j < i)))

            {

                k = k + 1;

            }

        }

        tempArr[k] = arr[i];

    }

    for (int i = 0; i < n - 1; i++)

    {

        arr[i] = tempArr[i];

    }

    delete[] tempArr;

}

void extractSort(int arr[], int n)

{

    int min, temp;

    for (int i = 0; i < n - 1; i++)

    {

        min = i;

        for (int j = i + 1; j < n; j++)

        {

            if (arr[j] < arr[min])

                min = j;

        }

        swap(arr[i], arr[min]);

    }

}

void kucha(int arr[], int n, int i)

{

    int maxim = i;

    int l = 2 \* i + 1, r = 2 \* i + 2;

    if (l < n && arr[l] > arr[maxim])

        maxim = l;

    if (r < n && arr[r] > arr[maxim])

        maxim = r;

    if (maxim != i)

    {

        swap(arr[i], arr[maxim]);

        kucha(arr, n, maxim);

    }

}

void treeSort(int arr[], int n)

{

    for (int i = n / 2 - 1; i >= 0; i--)

    {

        kucha(arr, n, i);

    }

    for (int i = n - 1; i >= 0; i--)

    {

        swap(arr[0], arr[i]);

        kucha(arr, i, 0);

        arrOut(arr, n);

    }

}

void shellSort(int arr[], int n)

{

    arrOut(arr, n);

    int d = n / 2;

    while (d > 0)

    {

        for (int i = 0; i < n - d - 1; i++)

        {

            bool flag = false;

            int j = i;

            arrOut(arr, n);

            cout << arr[j] << " - " << arr[j + d] << " ::" << d << endl;

            while (j >= 0 && arr[j] > arr[j + d])

            {

                swap(arr[j], arr[j + d]);

                j = j - d;

            }

        }

        d /= 2;

    }

}

void kStat(int arr[], int n, int k)

{

    cout << "START" << endl;

    arrOut(arr, n);

    int l = 0, r = n - 1, i = 0, j = 0, x = 0;

    while (l < r - 1)

    {

        x = arr[k];

        i = l;

        j = r;

        do

        {

            cout << "Vhod: " << endl;

            arrOut(arr, n);

            while (arr[i] < x)

                i++;

            while (x < arr[j])

                j = j - 1;

            if (i <= j)

            {

                cout << arr[i] << "-i   j-" << arr[j] << endl;

                swap(arr[i], arr[j]);

                i++;

                j--;

            }

            arrOut(arr, n);

        } while (i < j);

        if (j < k)

            l = i;

        if (i > k)

            r = j;

    }

    arrOut(arr, n);

    cout << "Nuwnoe chislo: " << arr[k] << endl;

}

void mergeSort(int arr[], int n)

{

    int rght, left, wid, rend;

    int i, j, m, t;

    int b[n];

    for (int k = 1; k < n; k \*= 2)

    {

        arrOut(arr, n);

        cout << "k = " << k \* 2 << endl;

        for (left = 0; left + k < n; left += k \* 2)

        {

            rght = left + k;

            rend = rght + k;

            if (rend > n)

                rend = n;

            m = left;

            i = left;

            j = rght;

            while (i < rght && j < rend)

            {

                if (arr[i] <= arr[j])

                {

                    b[m] = arr[i];

                    i++;

                }

                else

                {

                    b[m] = arr[j];

                    j++;

                }

                m++;

            }

            while (i < rght)

            {

                b[m] = arr[i];

                i++;

                m++;

            }

            while (j < rend)

            {

                b[m] = arr[j];

                j++;

                m++;

            }

            for (m = left; m < rend; m++)

            {

                arr[m] = b[m];

            }

        }

        arrOut(arr, n);

        cout << "--------------------" << endl;

    }

}

// Strings

int getMax(string arr[], int n)

{

    int max = arr[0].size();

    for (int i = 1; i < n; i++)

    {

        if (arr[i].size() > max)

            max = arr[i].size();

    }

    return max;

}

void countingSortString(string arr[], int n, int k)

{

    cout << k + 1 << endl;

    const int lenC = 257;

    string \*b = new string[n];

    int \*c = new int[lenC];

    for (int i = 0; i < lenC; i++)

    {

        c[i] = 0;

    }

    for (int j = 0; j < n; j++)

    {

        if (k < arr[j].size())

        {

            c[arr[j][k] + 1]++;

        }

        else

        {

            c[0]++;

        }

    }

    for (int f = 1; f < lenC; f++)

    {

        c[f] += c[f - 1];

    }

    for (int r = n - 1; r >= 0; r--)

    {

        if (k < arr[r].size())

        {

            b[c[arr[r][k] + 1] - 1] = arr[r];

        }

        else

        {

            b[c[0] - 1] = arr[r];

        }

        if (k < arr[r].size())

        {

            c[arr[r][k] + 1]--;

        }

        else

        {

            c[0]--;

        }

    }

    for (int l = 0; l < n; l++)

    {

        arr[l] = b[l];

        cout << arr[l] << " ";

    }

    cout << endl;

    delete[] b;

    delete[] c;

}

void radixSortStrings(string arrStr[], int n)

{

    int max = getMax(arrStr, n);

    for (int digit = max; digit > 0; digit--)

    {

        countingSortString(arrStr, n, digit - 1);

    }

}

// Dates

void dateToNumbers(string arr[], int a[], int n)

{

    string t;

    for (int i = 0; i < n; i++)

    {

        t = arr[i].substr(6, 7) + arr[i].substr(3, 4).substr(0, 2) + arr[i].substr(0, 2);

        a[i] = stoi(t);

    }

}

void numbersToDate(int arr[], string a[], int n)

{

    string t;

    for (int i = 0; i < n; i++)

    {

        t = to\_string(arr[i]).substr(4, 6) + "." + to\_string(arr[i]).substr(2, 3).substr(0, 2) + "." + to\_string(arr[i]).substr(0, 2);

        a[i] = t;

    }

}

void countingSortInt(int arr[], int n, int div)

{

    cout << "K= " << div << endl;

    int output[n];

    int count[10] = {0};

    for (int i = 0; i < n; i++)

        count[(arr[i] / div) % 10]++;

    for (int i = 1; i < 10; i++)

        count[i] += count[i - 1];

    for (int i = n - 1; i >= 0; i--)

    {

        output[count[(arr[i] / div) % 10] - 1] = arr[i];

        count[(arr[i] / div) % 10]--;

    }

    for (int i = 0; i < n; i++)

    {

        arr[i] = output[i];

        cout << arr[i] << endl;

    }

}

void radixSortDates(int arr[], int size)

{

    int div = 1;

    int max = 6;

    for (int k = max; k > 0; k--)

    {

        countingSortInt(arr, size, div);

        div \*= 10;

    }

}

**3. Контрольные вопросы:**

**3.1 Как классифицируются методы сортировки?**

Сортировки классифицируют по месту проведения сортировки:

* Внутренняя – работает с данными в оперативной памяти с произвольным доступом к любой ячейке.
* Внешняя – работает с запоминающими устройствами большого объёма, также доступ к информации осуществляется последовательно, а не произвольно.

**3.2 Сортировка простым включением относится к устойчивым алгоритмам сортировки?**

Да, так как если мы найдём элемент равный уже найденному, мы поставим его после.

**3.3 В чем состоит практическое значение изучения простейших методов сортировки?**

Понимание алгоритмов сортировок позволяет решать типичные проблемы, которые будут появляться других задачах. С помощью знаний, полученных на сортировках, большинство задач можно решить гораздо быстрее, так как их решение не будет сводиться к решению “в лобовую”.

**3.4 Какова времеменная и пространственная сложность сортировки подсчетом?**

O(n+k), где n-кол-во элементов, а k – диапазон элементов. Также потребуется создать массив длины k, поэтому V(k).

**3.5 Что такое барьерный (опорный) элемент?**

Опорный элемент – элемент, с которым происходят сравнения левых и правых (от опорного элементов).