

АЛГОРИТМИ СОРТУВАННЯ ЛІНІЙНИХ СТРУКТУР

Алгоритми

Лабораторна робота №5

Варіант 7

Виконав: Конча Вадим

Завдання:

Вимоги до виконання роботи

1. Складіть програму для сортування елементів масиву за зростанням або спаданням (в залежності від умов вашого індивідуального завдання) за алгоритмом Shellsort або Quicksort. При цьому для демонстрації роботи алгоритму передбачте можливість:

- введення масиву з клавіатури, файлу або безпосередньо в тексті програми – як вважаєте за доцільне;
- друк елементів масиву на екран або у файл **після кожного проходу** по масиву та **відсортованого масиву**.

Пам'ятайте, що сортування підмасивів у Shellsort відбувається за допомогою алгоритму прямих включень.

2. Доповніть вашу програму лічильником кількості порівнянь та обмінів елементів при сортуванні та проаналізуйте їх залежність від кількості елементів масиву.

3. Додайте до програми процедури, необхідні для виконання вашого індивідуального завдання. Вхідні дані та результати роботи програми виведіть на екран.

Для варіантів з парними номерами реалізуйте алгоритм Shellsort, з непарними – Quicksort.

7. Заданий двовірний масив розмірністю $[1..N, 1..N]$. Виконати сортування стовпців по спаданню елементів останнього рядка. Вивести на екран вхідний і отриманий масив у вигляді матриці.

**заповнення матриці реалізовано через рандом.*

- введення масиву з клавіатури, файлу або безпосередньо в тексті програми – як вважаєте за доцільне;

“безпосередньо в тексті програми”

Код:

```
#include <iostream>
#include <chrono> //для "хроно" штуки, которая в последствии будет в сиде
#include <random> //для функций рандома

using namespace std;
```

```

int rows_cols = 5; //N - размер матрицы
int rowssave_forprint = rows_cols;
int ITER = 0;
int timetravel = 0, shutka = 0; //штук чтобы в функции можно было границы уменьшать со временем

```

```

void printMATRIX(int rows_cols, int** arr)
{

```

```

    //выводим массив
    cout << endl << ITER << ") " << endl;
    cout << endl;
    for (int j = 0; j < rows_cols; j++)
    {
        for (int i = 0; i < rows_cols; i++)
        {
            cout << arr[j][i] << " ";
        }
        cout << endl;
    }
}

```

```

bool check(int *arr)
{

```

```

    //проверка перед рекурсией
    bool helper = true;
    for (int l = 0; l < rows_cols - 1; l++)
    {
        if (arr[l] < arr[l + 1]) helper = false;
    }
    if (arr[rows_cols - 1] > arr[rows_cols - 2]) helper = false;
    return helper;
}

```

```

void QuickSort(int* arr, int rows, int mainindex, int** arirMAXIMA)

```

```

{//элемент из столбца от конца на один + индекс главного элемента

```

```

    int MARSLET = *(arirMAXIMA[rows-1] + mainindex);
    int saver1main = 0, saver2bigger = 0, saver3prost = 0;

```

```

    for (int i = mainindex + 1; i < rows; i++) //для всех последующих элементов, которые БОЛЬШЕ ////////////В ЭТОЙ СТРОЧКЕ ОСНОВНОЙ СМЫСЛ
    {

```

```

        if (ITER > pow(rows_cols,2) ) break;

```

```

        if (arirMAXIMA[rows_cols - 1][i] >= arirMAXIMA[rows_cols - 1][mainindex]) //в случае нахождение элемента больше или такого же
        {

```

```

            if (arirMAXIMA[rows_cols - 1][mainindex + 1] <= arirMAXIMA[rows_cols - 1][mainindex]) //и это не БУКВАЛЬНО следующий элемент
            {

```

```

                for (int k = 0; k < rows_cols; k++)
                {

```

```

//cout << "\nAZAZA 1\n";
//сейвы
saver1main = arirMAXIMA[k][mainindex];
saver2bigger = arirMAXIMA[k][i];
saver3prost = arirMAXIMA[k][mainindex + 1];

//cout << "\n" << MARSLET << " is = " << mainindex
<< "\n";

//переназначаем элементы
arirMAXIMA[k][mainindex + 1] = saver1main;
//главный элемент шагает 1 раз вперёд
arirMAXIMA[k][i] = saver3prost; //элемент на
который наступили, наступает на тот "большой"
arirMAXIMA[k][mainindex] = saver2bigger;
//большой переселяется за главный слева
}
mainindex++; //индекс на 1 больше, ведь
"главный" элемент всё таки шагнул, и его индекс изменился
ITER++; //одно переставление
printMATRIX(rowssave_forprint, arirMAXIMA);

}
else
{

//cout << "\nAZAZA 2\n";
for (int k = 0; k < rows_cols; k++)
{
//сейвы
saver1main = arirMAXIMA[k][mainindex];
saver2bigger = arirMAXIMA[k][i];

//cout << "\n" << MARSLET << " is = " << mainindex << "\n";

//переназначаем элементы
arirMAXIMA[k][i] = saver1main; //главный элемент шагает 1
раз вперёд
arirMAXIMA[k][mainindex] = saver2bigger; //большой
переселяется за главный слева
}
mainindex++; //индекс на 1 больше, ведь
"главный" элемент всё таки шагнул, и его индекс изменился
ITER++; //одно переставление
printMATRIX(rowssave_forprint, arirMAXIMA);

}

}

}
// cout << "\n" << MARSLET << " is = " << mainindex << "\n";

```

```

int prikol = rows - mainindex - 1 - timetravel;
// тут ещё -1 потому что ровсколс не имеет нулевого значения в отличии от майниндекса
if (prikol > 1 && check(arr) != true)
{
    //остатки справа
    cout << "\nRECURS RIGHT IS START\n";
    QuickSort(arr, rows_cols, mainindex + 1, arirMAXIMA);
    timetravel += mainindex;
}
//////////((int i = mainindex + 1; i < rows; i++) /
    if (mainindex - shutka > 1 && check(arr) != true)
    {
        //остатки слева
        cout << "\nRECURS LEFT IS START\n";
        QuickSort(arr, mainindex, 0, arirMAXIMA);
        shutka += mainindex;
    }

    if (mainindex == 0 && check(arr) != true)
    {
        QuickSort(arr, rows_cols, mainindex + 1, arirMAXIMA);
    }

}

//rows/2 работает только если штука находится посередине ----- && mainindex < rows_cols/2
//////////ПРОБЛЕМА КАЖЕТСЯ В ТОМ, ЧТО Я КРИВО В ПОСЛЕДУЮЩИЕ
РАЗЫ КИДАЮ 4Й И 1Й АРГУМЕНТЫ//////////
int main()
{
    //создаём динамический массив
    int** arr = new int* [rows_cols];
    for (int i = 0; i < rows_cols; i++)
    {
        arr[i] = new int[rows_cols];
    }

    //заполняем массива рандомом
    unsigned seed = chrono::system_clock::now().time_since_epoch().count(); //формировка
седа, с помощью времени хроно
    default_random_engine generator(seed); //обозначение генератора
    uniform_int_distribution<int> distribution(1, 9); //определение границ и функции вызова
    for (int j = 0; j < rows_cols; j++)
    {
        for (int i = 0; i < rows_cols; i++)
        {
            arr[j][i] = distribution(generator);
        }
    }

    printMATRIX(rowssave_forprint, arr);

    QuickSort(arr[rows_cols - 1], rows_cols, 0, arr);
    cout << "\n\nF\t\tN\tA\tL\n\n";
}

```

```

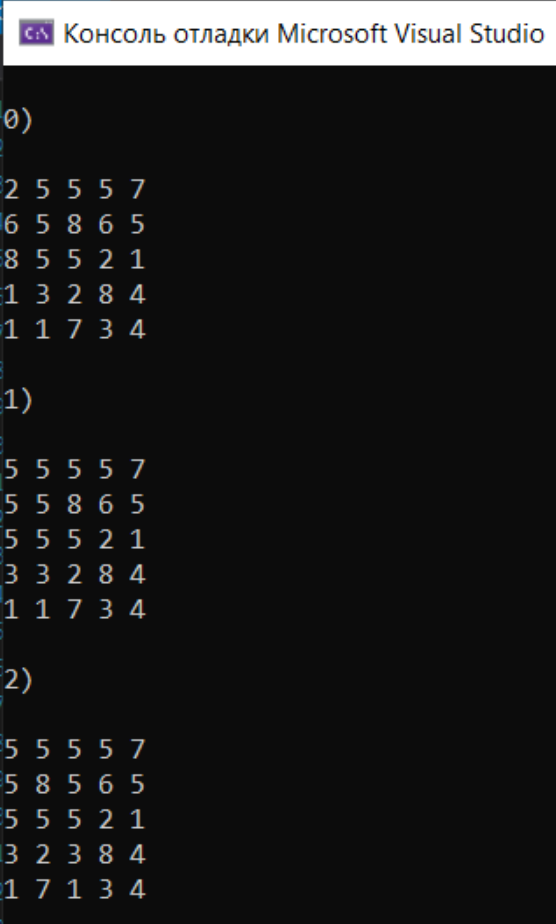
printMATRIX(rowssave_forprint, arr);

//delete чистим память
for (int i = 0; i < rows_cols; i++)
{
    delete [] arr[i];
}
delete[] arr;
}

```

РЕЗУЛЬТАТИ:

N=5



```

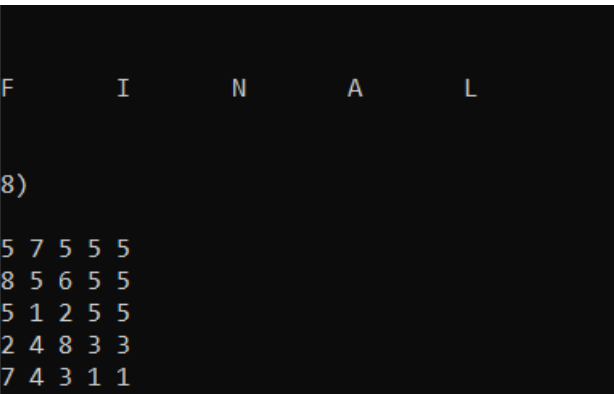
Консоль отладки Microsoft Visual Studio

0)
2 5 5 5 7
6 5 8 6 5
8 5 5 2 1
1 3 2 8 4
1 1 7 3 4

1)
5 5 5 5 7
5 5 8 6 5
5 5 5 2 1
3 3 2 8 4
1 1 7 3 4

2)
5 5 5 5 7
5 8 5 6 5
5 5 5 2 1
3 2 3 8 4
1 7 1 3 4

```



```

8)
5 7 5 5 5
8 5 6 5 5
5 1 2 5 5
2 4 8 3 3
7 4 3 1 1

9)
F      I      N      A      L

```

N=22

```
Консоль отладки Microsoft Visual Studio
0)
6 4 7 1 4 5 4 3 6 6 7 3 7 9 6 7 8 8 2 1 2 8
1 2 3 6 5 4 7 8 1 4 7 8 2 9 4 9 8 1 6 4 1 1
1 7 6 7 5 7 7 7 9 2 6 8 2 3 9 5 5 5 9 5 2 7
2 9 2 1 4 2 2 1 1 4 1 3 4 8 3 9 1 5 5 7 9 5
1 3 4 6 5 3 8 8 5 7 2 9 5 9 1 7 3 2 6 6 9 7
2 5 9 5 5 6 6 6 4 1 6 9 2 5 3 4 2 2 7 4 3 9
5 6 5 4 7 5 2 1 1 2 6 2 6 6 8 4 7 4 2 8 5 3
8 8 6 8 2 2 2 2 5 9 3 7 2 2 8 4 2 7 7 1 7 4
4 1 1 1 4 6 3 1 5 3 6 4 9 1 2 1 3 5 8 6 2 4
9 4 8 7 2 4 9 1 4 2 3 3 2 2 3 1 5 8 3 9 8 6
1 2 3 7 7 1 3 1 4 8 8 3 8 8 6 9 4 9 3 3 8 1
3 5 8 1 5 2 3 6 3 2 5 5 2 5 1 6 8 3 6 6 6 2
2 2 5 2 6 9 9 4 4 7 7 7 9 8 5 3 7 5 5 9 8 7
9 3 2 5 3 8 3 7 9 6 8 5 4 5 9 1 8 6 6 6 3 9
6 1 9 4 3 3 2 3 5 3 4 9 2 8 9 2 4 9 7 8 8 9
7 8 7 4 7 6 9 7 4 4 5 2 9 9 7 5 1 2 8 2 3 8
5 1 3 9 8 3 3 8 8 6 4 7 7 1 1 9 8 1 8 8 7 6
8 9 4 9 2 6 3 7 4 4 6 9 2 8 5 9 7 7 5 8 3 8
8 1 3 5 5 8 6 7 4 4 3 1 7 1 5 1 6 3 6 9 1 8
4 9 7 7 4 3 4 5 3 7 8 1 9 6 4 2 9 9 5 3 6 4
6 8 5 6 5 1 5 7 8 5 9 9 8 3 5 5 8 6 4 9 5 7
4 4 5 3 1 6 8 2 2 3 4 9 3 7 9 4 9 5 6 5 6 9

Консоль отладки Microsoft Visual Studio
F I N A L
90)
6 8 8 8 4 9 5 5 5 7 7 7 4 4 4 4 1 1 1 6 6 4
4 8 1 1 7 9 4 4 4 3 3 3 2 2 2 2 6 6 6 1 1 5
9 5 7 7 7 3 7 7 7 6 6 6 7 7 7 7 7 7 9 9 5
3 1 5 5 2 8 2 2 2 2 2 2 9 9 9 9 1 1 1 1 1 4
1 3 7 7 8 9 3 3 3 4 4 4 3 3 3 3 6 6 6 5 5 5
3 2 9 9 6 5 6 6 6 9 9 9 5 5 5 5 5 5 4 4 5
8 7 3 3 2 6 5 5 5 5 5 5 6 6 6 6 4 4 4 1 1 7
8 2 4 4 2 2 2 2 2 6 6 6 8 8 8 8 8 8 5 5 2
2 3 4 4 3 1 6 6 6 1 1 1 1 1 1 1 1 1 5 5 4
3 5 6 6 9 2 4 4 4 8 8 8 4 4 4 4 7 7 7 4 4 2
6 4 1 1 3 8 1 1 1 3 3 3 2 2 2 2 7 7 7 4 4 7
1 8 2 2 3 5 2 2 2 8 8 8 5 5 5 5 1 1 1 3 3 5
5 7 7 7 9 8 9 9 9 5 5 5 2 2 2 2 2 2 4 4 6
9 8 9 9 3 5 8 8 8 2 2 2 3 3 3 3 5 5 5 9 9 3
9 4 9 9 2 8 3 3 3 9 9 9 1 1 1 1 4 4 4 5 5 3
7 1 8 8 9 9 6 6 6 7 7 7 8 8 8 8 4 4 4 4 4 7
1 8 6 6 3 1 3 3 3 3 3 3 1 1 1 1 1 9 9 9 8 8 8
5 7 8 8 3 8 6 6 6 4 4 4 9 9 9 9 9 9 4 4 2
5 6 8 8 6 1 8 8 8 3 3 3 1 1 1 1 5 5 5 4 4 5
4 9 4 4 4 6 3 3 3 7 7 7 9 9 9 9 7 7 7 3 3 4
5 8 7 7 5 3 1 1 1 5 5 5 8 8 8 8 6 6 6 8 8 5
9 9 9 9 8 7 6 6 6 5 5 5 4 4 4 4 3 3 3 2 2 1
```

Джерела:

[Быстрая сортировка — Википедия \(wikipedia.org\)](https://uk.wikipedia.org/wiki/Быстрая_сортировка)