ВИДЫ СОРТИРОВКИ:

1 – Сортировка пузырьком

#include <iostream>

using namespace std;

int main()

{

/\* Установим размер массива \*/

int n; // Кол-во элементов

cout << "Количество элементов: ";

cin >> n;

/\* Заполним массив значениями \*/

int mass[n];

for(int i = 0; i < n; ++i)

{

cout << i+1 << "-ый элемент: ";

cin >> mass[i];

}

/\* Выведем исходный массив \*/

cout << "Исходный массив: ";

for(int i = 0; i < n; ++i)

{

cout << mass[i] << " ";

}

cout << endl;

/\* Отсортируем массив по убыванию \*/

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

{

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

{

if(mass[r] < mass[r+1])

{

// Обмен местами

int temp = mass[r];

mass[r] = mass[r+1];

mass[r+1] = temp;

}

}

}

/\* Выведем отсортированный массив \*/

cout << "Отсортированный массив: ";

for(int i = 0; i < n; ++i)

{

cout << mass[i] << " ";

}

cout << endl;

return 0;

}

2- Сортировка выбором.

#define \_CRT\_SECURE\_NO\_WARNINGS // для корректной работы scanf()

#include <stdio.h>

#define SIZE 10 // размер массива сортируемых элементов

// Функция сортировки прямыми включениями

void selectionSort(int\* num, int size)

{

int min, temp; // для поиска минимального элемента и для обмена

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

{

min = i; // запоминаем индекс текущего элемента

// ищем минимальный элемент чтобы поместить на место i-ого

for (int j = i + 1; j < size; j++) // для остальных элементов после i-ого

{

if (num[j] < num[min]) // если элемент меньше минимального,

min = j; // запоминаем его индекс в min

}

if (min == i) continue; // чтобы не менять элемент сам с собой

temp = num[i]; // меняем местами i-ый и минимальный элементы

num[i] = num[min];

num[min] = temp;

}

}

int main()

{

int a[SIZE]; // Объявляем массив из 10 элементов

// Вводим значения элементов массива

for (int i = 0; i < SIZE; i++)

{

printf("a[%d] = ", i);

scanf("%d", &a[i]);

}

selectionSort(a, SIZE); // вызываем функцию сортировки

// Выводим отсортированные элементы массива

for (int i = 0; i < SIZE; i++)

printf("%d ", a[i]);

getchar();

getchar();

return 0;

}

3 - Сортировка вставкой.

#include <iostream>

using namespace std;

int main(){

// несортированный список

int unsorted[] = {9,8,7,6,5,4,3,3,2,1};

// размер списка

int size\_unsorted = sizeof(unsorted) / sizeof(unsorted[0]);

//выводим несортированный список

cout << "\nUnsorted: ";

for(int i = 0 ; i < size\_unsorted ; i++){

cout << unsorted[i] << " ";

}

int current\_element,temp;

for(int i = 1; i < size\_unsorted; i++){

current\_element = unsorted[i];

for(int j = i-1; j >= 0 && unsorted[j] > current\_element; j--){

// замена, если текущий элемент меньше

temp = unsorted[j+1];

unsorted[j+1] = unsorted[j];

unsorted[j] = temp;

}

}

// печать отсортированного списка

cout << "\nSorted: ";

for(int i = 0 ; i < size\_unsorted ; i++){

cout << unsorted[i] << " ";

}

return 0;

}

4 – Сортировка подсчетом.

// сортировка подсчётом

let countingSort = (arr, min, max) => {

// начинаем с минимального значения диапазона

let i = min,

// вспомогательная переменная для цикла

j = 0,

// получаем длину массива

len = arr.length,

// вспомогательный массив, где будем хранить результаты подсчёта

count = [];

// сначала заполняем нулями массив с результатом подсчёта

for (i; i <= max; i++) {

count[i] = 0;

}

// потом проходим по всему исходному массиву

for (i = 0; i < len; i++) {

// и увеличиваем на единичку ячейки под тем же номером в массиве с результатом подсчёта

count[arr[i]] += 1;

}

// а затем проходим по массиву с результатом

for (i = min; i <= max; i++) {

// пока в очередной ячейке значение больше нуля

while (count[i] > 0) {

// добавляем номер ячейки в исходный массив

arr[j] = i;

// переходим к следующему элементу в исходном массиве

j++;

// уменьшаем на единицу содержимое ячейки в массиве с подсчётом

count[i]--;

}

}

// как всё сделали — возвращаем отсортированный массив

return arr;

};

// исходный массив

let arr = [0,3,5,2,4,5,2,3];

// сортируем его

countingSort(arr,0,5);

// и выводим результат

console.log(arr)

5 - Цифровая сортировка.

// Цифровая сортировка

// Дано n строк, выведите их порядок после k фаз цифровой сортировки.

// Формат входного файла

// В первой строке входного файла содержатся числа n — число строк, m — их длина

// и k – число фаз цифровой сортировки (1<=n<=10^6, 1<=k<=m<=10^6, n\*m<=5\*10^7).

// Далее находится описание строк, но в нетривиальном формате. Так, i-ая строка

// (1<=i<=n) записана в i-ых символах второй, …, (m+1)-ой строк входного файла.

// Иными словами, строки написаны по вертикали. Это сделано специально, чтобы

// сортировка занимала меньше времени.

// Строки состоят из строчных латинских букв: от символа "a" до символа "z"

// включительно. В таблице символов ASCII все эти буквы располагаются подряд

// и в алфавитном порядке, код буквы "a" равен 97, код буквы "z" равен 122.

// Формат выходного файла

// Выведите номера строк в том порядке, в котором они будут после k фаз цифровой

// сортировки.

#include <fstream>

void radixsort(int \*\*a, int n, int f)

{

int i;

int \*\*b = new int \* [n + 1];

int c[27] = {0};

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

c[a[i][f]]++;

for (i = 1; i < 27; ++i)

c[i] += c[i - 1];

for (i = n; i > 0; --i) {

b[c[a[i][f]]] = a[i];

c[a[i][f]]--;

}

a = b;

delete b;

}

int main()

{

std::ifstream fin("input.txt");

std::ofstream fout("output.txt");

int n, m, k, i, j;

fin >> n >> m >> k;

int \*\*array = new int \* [n + 1];

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

array[i] = new int [m + 1];

array[i][0] = i;

}

char temp;

for (i = 1; i < m + 1; ++i) {

for (j = 1; j < n + 1; ++j) {

fin >> temp;

array[j][i] = temp - 96;

}

}

for (i = 1; i <= k; ++i)

radixsort(array, n, m - i + 1);

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

fout << array[i][0] << ' ';

fout << '\n';

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

delete [] array[i];

delete [] array;

return 0;

}

void Tools::showFiveDaysForecast(nlohmann::json data) {

int i = 0;

int j = i + 1;

std::string dateDay;

std::string dateDayJ;

int counter = 0;

double sum = 0;

while (i < data["list"].size()) {

dateDay = data["list"][i]["dt\_txt"].get<std::string>().substr(8, 2);

counter = 1;

sum = data["list"][i]["main"]["temp"].get<double>() - 273.15;

j = i + 1;

if (j >= data["list"].size()) {

std::cout << data["list"][i]["dt\_txt"].get<std::string>().substr(0, 10);

std::cout << " Average Temp: " << (int)(sum / counter) << std::endl;

i++;

continue;

}

dateDayJ = data["list"][j]["dt\_txt"].get<std::string>().substr(8, 2);

while (dateDay == dateDayJ) {

counter++;

sum += data["list"][j]["main"]["temp"].get<double>() - 273.15;

j++;

if (j >= data["list"].size()) {

break;

}

dateDayJ = data["list"][j]["dt\_txt"].get<std::string>().substr(8, 2);

}

std::cout << data["list"][i]["dt\_txt"].get<std::string>().substr(0, 10);

std::cout << " Average Temp: " << (int)(sum / counter) << std::endl;

i = j;

}

}