Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет

Кафедра **«Информационные технологии и автоматизированные системы»**

направление подготовки: 09.03.04 - «Программная инженерия»

Лабораторная работа

По теме **«Простые сортировки»**

Выполнял:

студент группы РИС-24-1б

Морозова Н.С.

Проверял:

Доцент кафедры ИТАС

Полякова О.А.

Постановка задачи:

1. Создать три разных массива из случайных чисел для каждой сортировки, вывести их.
2. Провести в них сортировку тремя способами: подсчётом, блочной и слиянием.
3. Вывести все отсортированные массивы.

Код

#include <iostream>

#include <cmath>

#include <vector>

#include <algorithm>

#include <Windows.h>

using namespace std;

int\* count(int arr[], int size)

{

int min = arr[0];

int max = arr[0];

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

{

if (arr[i] < min)

min = arr[i];

if (arr[i] > max)

max = arr[i];

}

int range = max - min + 1; //вспомогательный массив

int\* count = new int[range](); //инициализация нулями

for (int i = 0; i < size; i++) //подсчёт каждого элемента

count[arr[i] - min]++;

for (int i = 1; i < range; i++) //накопление суммы

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

int\* out = new int[size];

for (int i = size - 1; i >= 0; i--)

{

out[count[arr[i] - min] - 1] = arr[i];

count[arr[i] - min]--;

}

for (int i = 0; i < size; i++) //восстанавление массива

arr[i] = out[i];

delete[] count;

delete[] out;

return arr;

}

int\* block(int arr[], int n)

{

int min = arr[0];

int max = arr[0];

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

{

if (arr[i] < min)

min = arr[i];

if (arr[i] > max)

max = arr[i];

}

int num = sqrt(n); //количество блоков

int size = (max - min) / num + 1; //размер каждого блока

vector<vector<int>> bl(num); //динамические блоки

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

{

int ind = (arr[i] - min) / size;

if (ind >= num)

ind = num - 1;

bl[ind].push\_back(arr[i]);

}

for (int i = 0; i < num; i++) //сортировка каждого блока

{

sort(bl[i].begin(), bl[i].end());

int\* blockArray = bl[i].data(); // Получаем указатель на данные вектора

print(blockArray, bl[i].size());

}

int idx = 0;

for (int i = 0; i < num; i++) //исходный массив

for (int j = 0; j < bl[i].size(); j++)

arr[idx++] = bl[i][j];

return arr;

}

void merge(int arr[], int left, int mid, int right)

{

int lR = mid - left + 1;

int rR = right - mid;

int\* lArr = new int[lR];

int\* rArr = new int[rR];

for (int i = 0; i < lR; i++) //заполнение левого и правого диапазонов

lArr[i] = arr[left + i];

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

rArr[i] = arr[mid + 1 + i];

int indl = 0;

int indr = 0;

int indmer = left;

while (indl < lR && indr < rR)

{

if (lArr[indl] <= rArr[indr])

{

arr[indmer] = lArr[indl];

indl++;

}

else

{

arr[indmer] = rArr[indr];

indr++;

}

indmer++;

print(arr, 20);

}

while (indl < lR) //восстановление массива

{

arr[indmer] = lArr[indl];

indl++;

indmer++;

}

while (indr < rR)

{

arr[indmer] = rArr[indr];

indr++;

indmer++;

}

delete[] lArr;

delete[] rArr;

}

void mergeS(int arr[], int st, int end)

{

if (st >= end) return;

int mid = st + (end - st) / 2;

mergeS(arr, st, mid);

mergeS(arr, mid + 1, end);

merge(arr, st, mid, end);

}

void print(int\* arr)

{

for (int i = 0; i < 20; i++)

cout << arr[i] << ' ';

cout << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int masC[] = {5, -6, -6, 19, 8, 20, 6, 19, 0, 4, 15, 2, -2, 13, 19, 3, 2, 11, -3, -5, 7};

cout << "Массив для сортировки подсчётом: ";

print(masC);

count(masC, 20);

cout << "Отсортированный массив: ";

print(masC);

cout << endl;

int masB[] = {9, 345, 1, 5095, 800, 43, 670, 15, 10030, 65, 709, 55, 3, 3045, 21, 8210, 6, 90, 3906, 25};

cout << "Массив для блочной сортировки: ";

print(masB);

block(masB, 19);

cout << "Отсортированный массив: ";

print(masB);

cout << endl;

int masM[] = {56, 1, 890, -45, 87, 9, -3, 23, 0, 456, 7, 43, 15, 152, 74, 13, 900, -11, 44, -5, 98};

cout << "Массив для сортировки слиянием: ";

print(masM);

mergeS(masM, 0, 20);

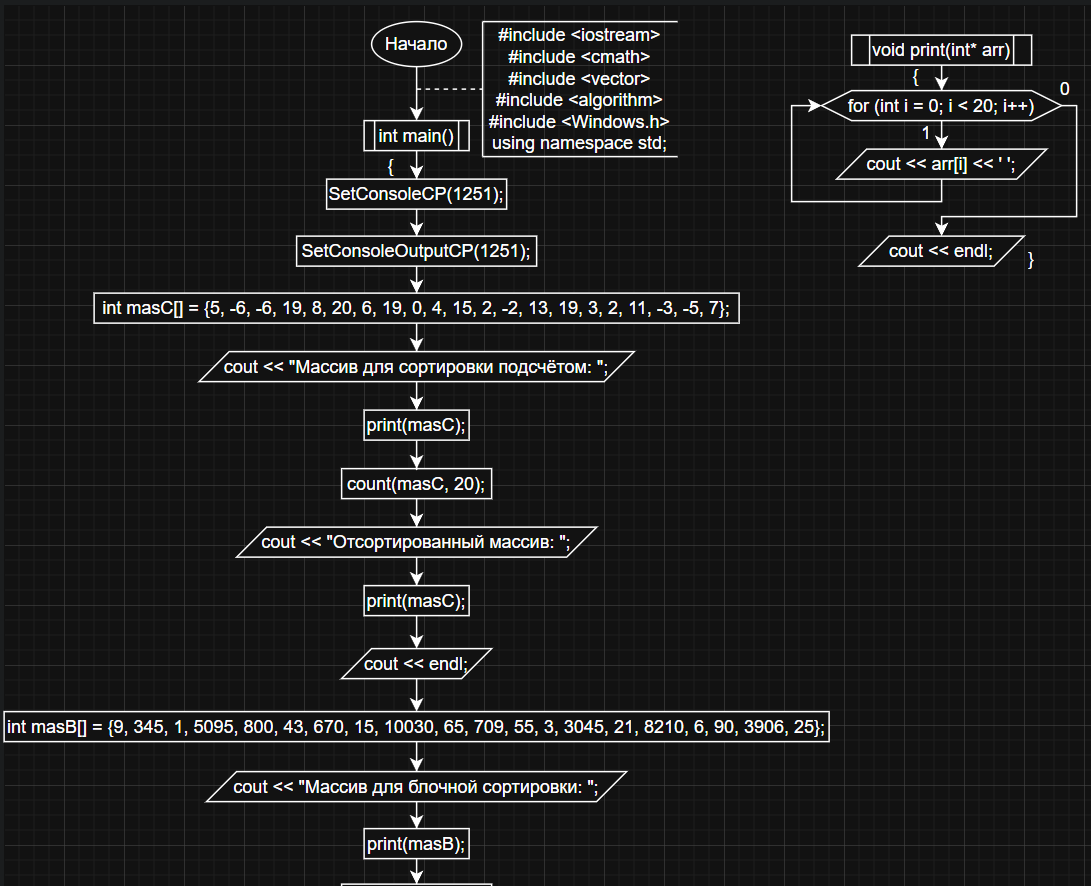
cout << "Отсортированный массив: ";

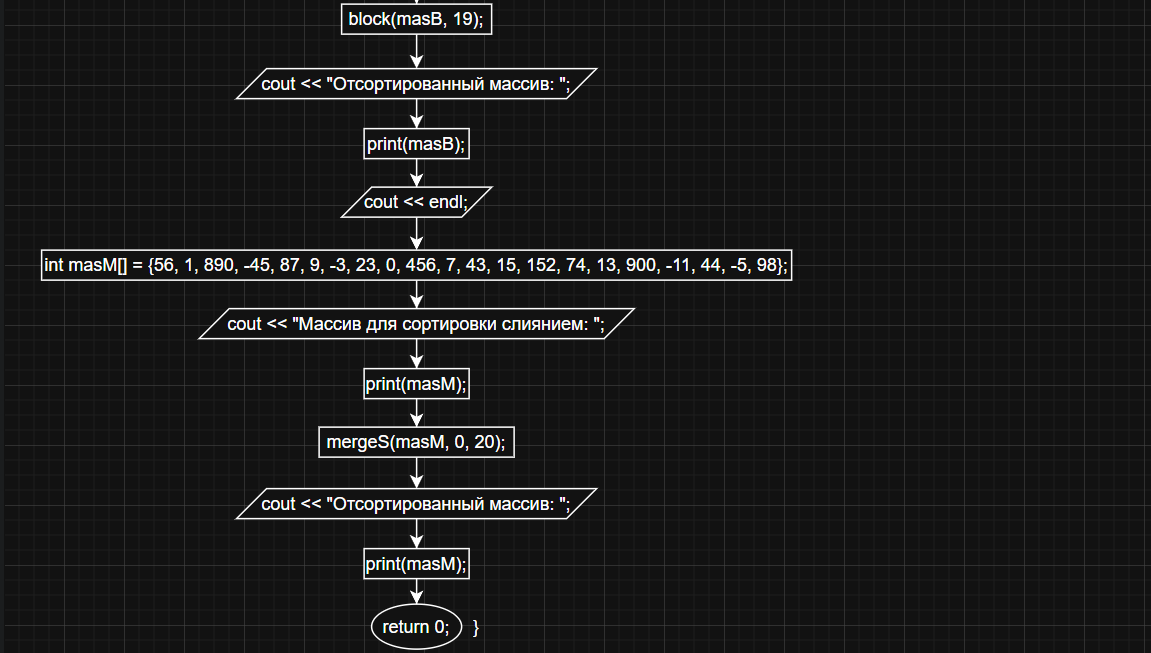
print(masM);

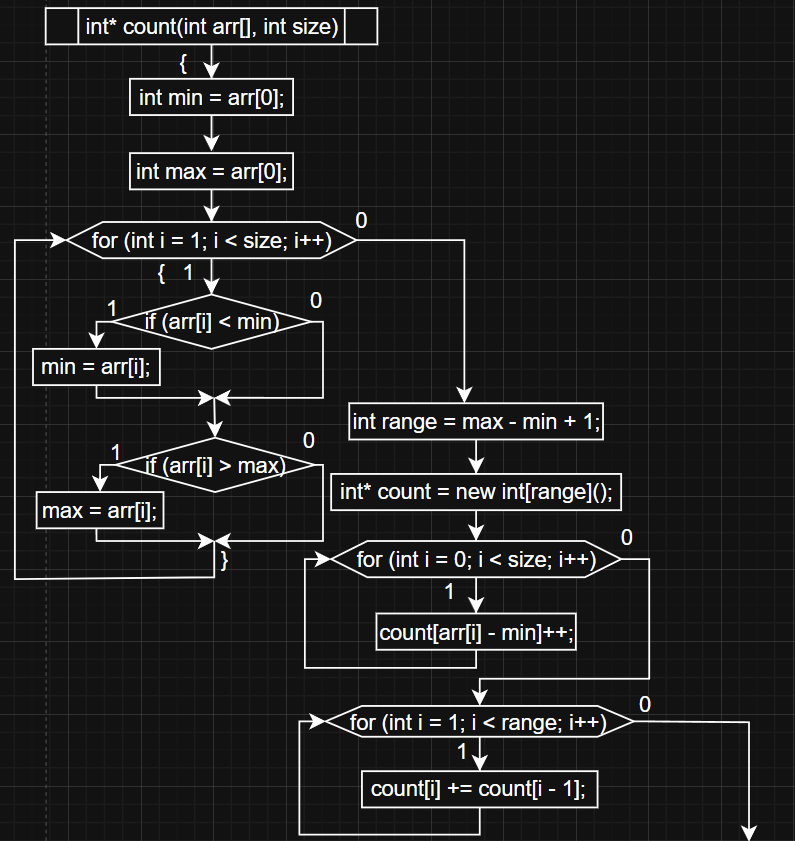
return 0;

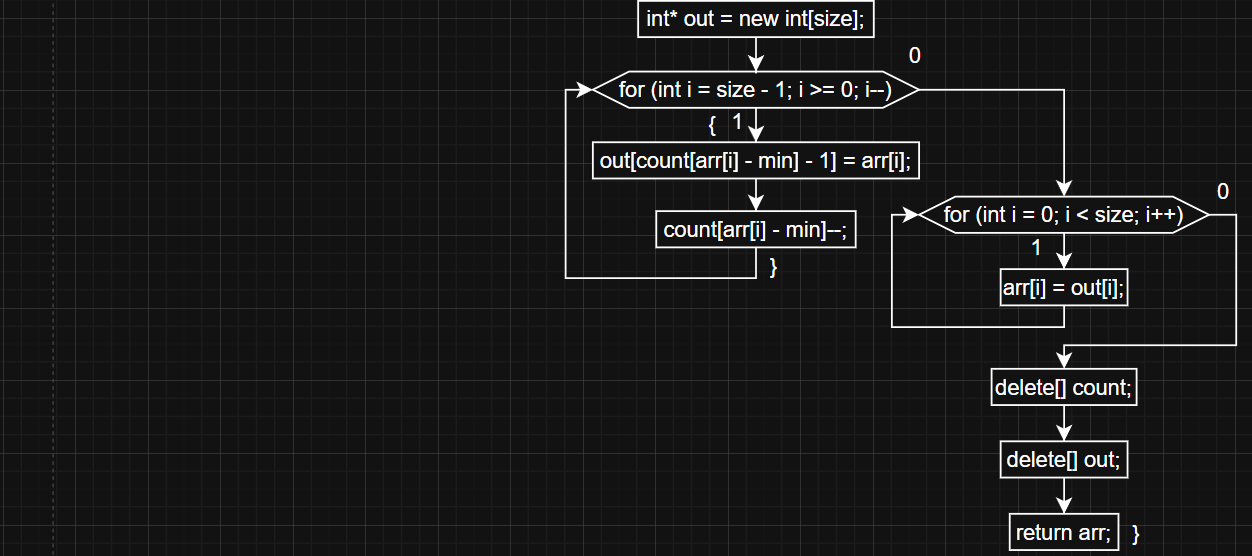
}

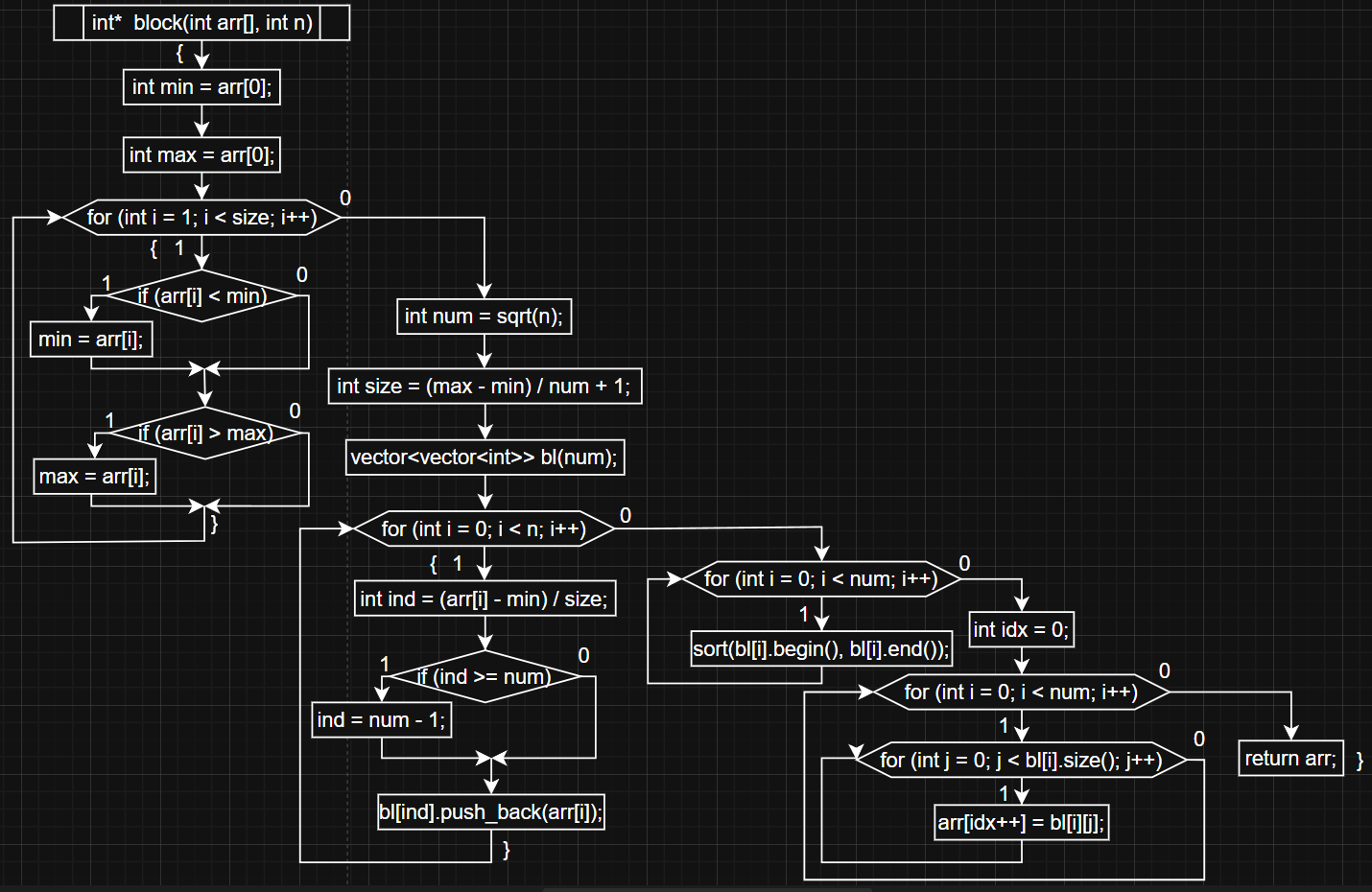
Блок-схема

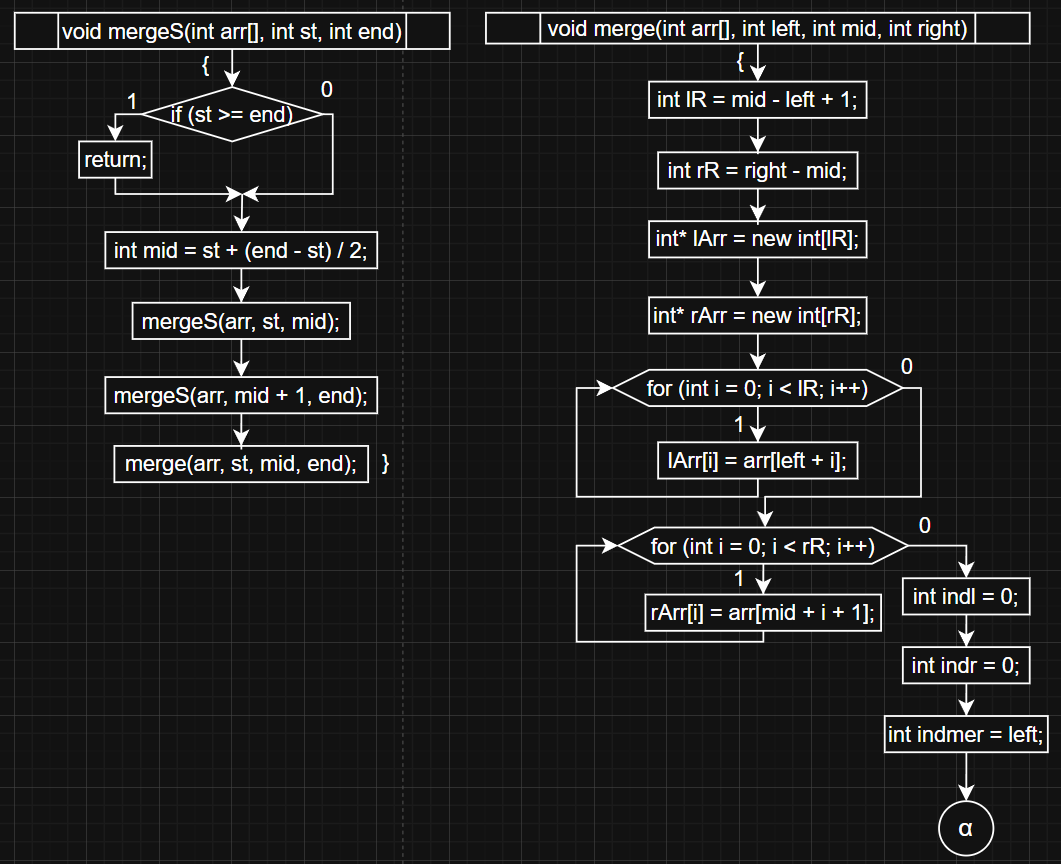


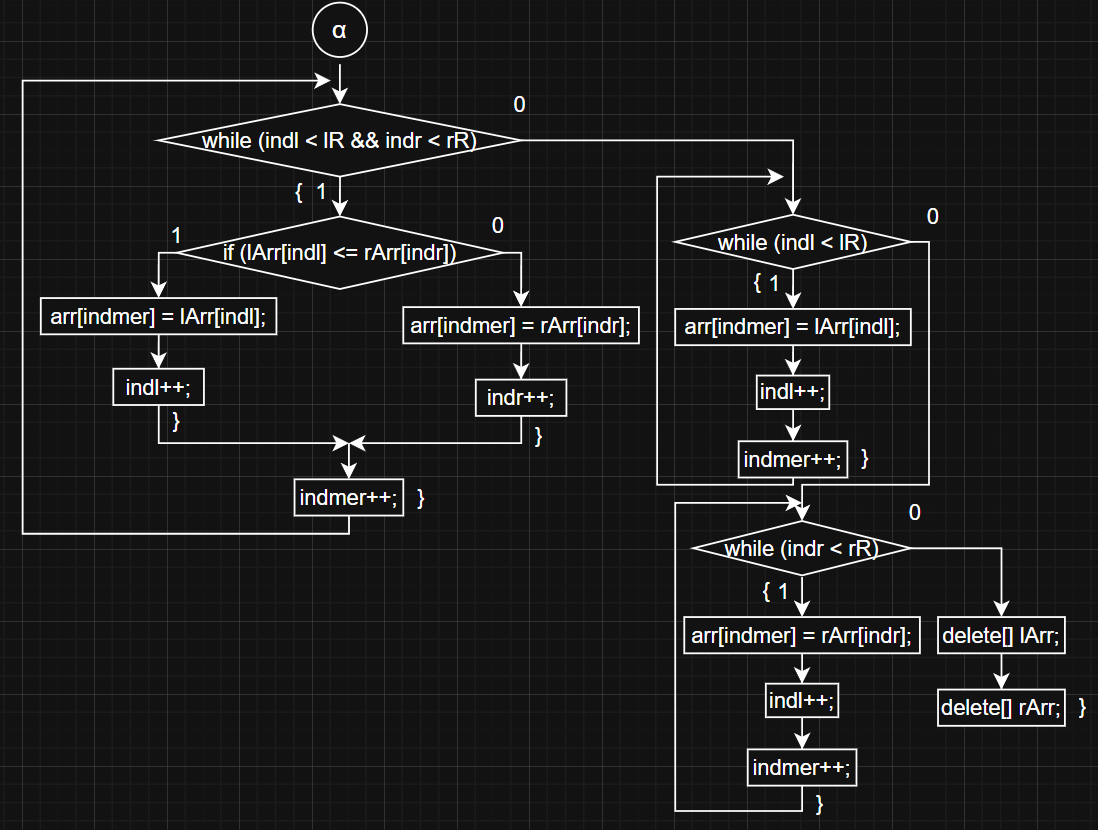












Результат работы программы:

