Министерство общего и профессионального

образования Российской Федерации

Пермский государственный технический университет

Лабораторная работа

"Методы сортировки"

Выполнила студентка группы РИС 23-3б:

Федорова О.И.

Проверила

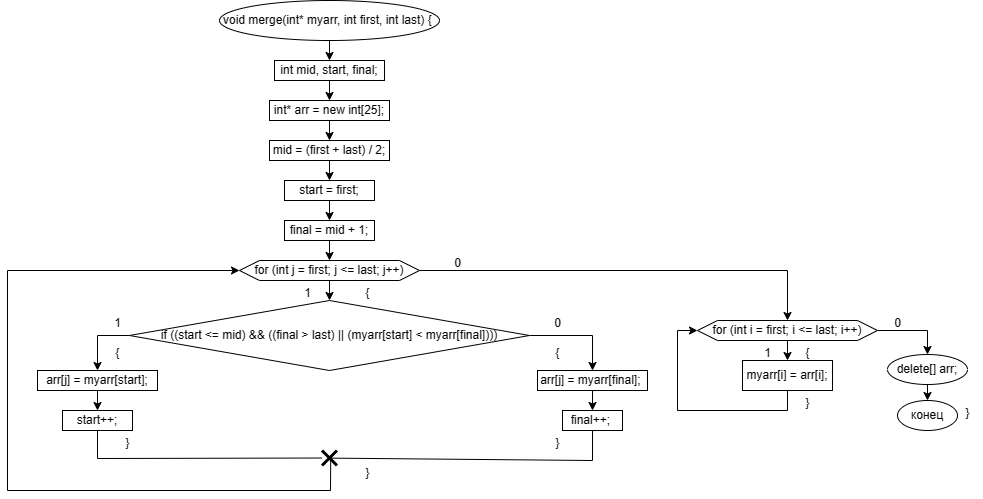
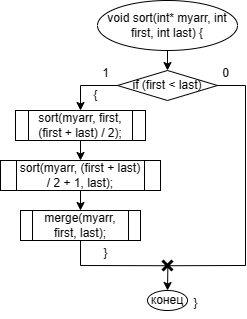
Доцент кафедры ИТАС

Полякова О.А.

Пермь 2024

1. **Сортировка слиянием**
2. Анализ задачи:
3. Понадобится функция, которая будет разбивать массива на группы. (sort)
4. Вторая функция(merge) будет сортировать элементы входящей группы с помощью второго массива, который будет запоминать элементы, которые нужно поменять местами.
5. sort будет разбивать массив сначала на самые маленькие группы, а после все крупнее и крупнее, чтобы в итоге был отсортирован весь массив.
6. Блок-схема:





1. Код программы:

#include<iostream>

using namespace std;

void merge(int\* myarr, int first, int last) { // функция для сортировки групп элементов

int mid, start, final;

int\* arr = new int[25]; // второй массив для запоминания элементов

mid = (first + last) / 2; // нахождение середины группы

start = first;

final = mid + 1;

for (int j = first; j <= last; j++) { // пройдется по группе элементов

if ((start <= mid) && ((final > last) || (myarr[start] < myarr[final]))) { // условия для сортировки

arr[j] = myarr[start]; // запоминание элементов для перемещения

start++;

}

else {

arr[j] = myarr[final];

final++;

}

}

for (int i = first; i <= last; i++) {

myarr[i] = arr[i]; // переставление элементов в правильном порядке

}

delete[] arr;

}

void sort(int\* myarr, int first, int last) { // функция для разделения элементов рна группы

if (first < last) {

sort(myarr, first, (first + last) / 2); // разбивает на максимально возможную маленькую группу

sort(myarr, (first + last) / 2 + 1, last); // разбиение на группы последующих элементов

merge(myarr, first, last); // сортировка образованных групп элементов

}

}

int main() {

setlocale(LC\_ALL, "ru");

int n = 25;

int\* myarr = new int[n]; // создание массива

srand(time(0));

for (int i = 0; i < n; i++) { // заполнение массива рандомными числами

myarr[i] = rand() % 100 + 1;

}

cout << "Исходный массив: " << endl;

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

cout << endl << "Отсортированный массив: " << endl;

sort(myarr, 0, n-1); // сортировка массива

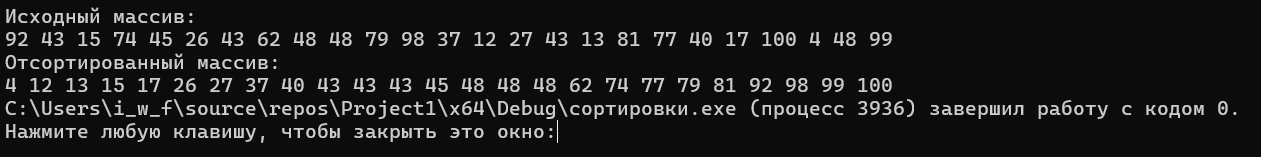
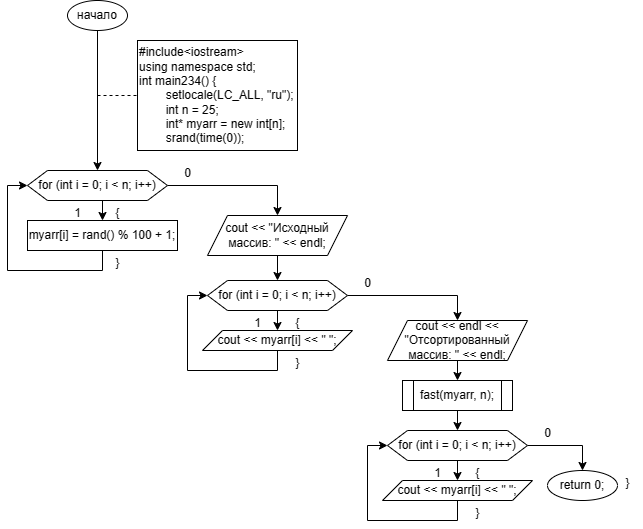
for (int i = 0; i < n; i++) { // вывод отсортированного массива

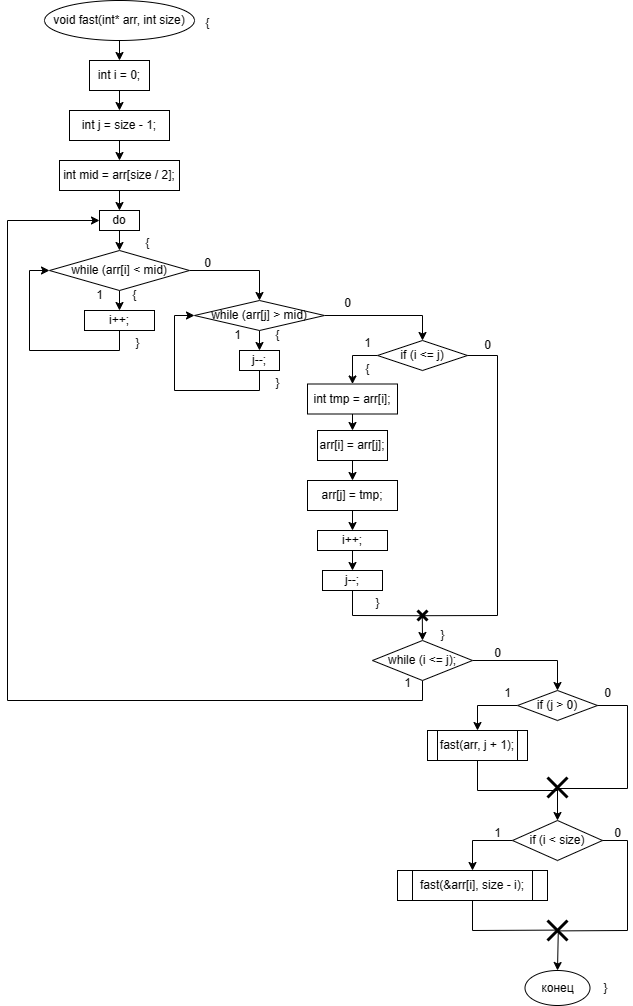
cout << myarr[i] << " ";

}

return 0;

}

1. Результат работы программы:
2. **Быстрая сортировка.**
3. Анализ задачи.
4. Понадобится функция, которая будет выполнять сортировку на разных отрезках массива и с разными ключами(fast).
5. В функции fast будет выполняться цикл, в котором элементы слева и справа будут сравниваться с ключом и меняться в случае неправильного их расположения.
6. Функция будет вызываться до тех пор, пока каждый элемент не побудет ключом, чтобы отсортировать весь массив.
7. Блок-схема:



1. Код программы:

#include<iostream>

using namespace std;

void fast(int\* arr, int size) { // функция для сортировки

int i = 0; // левая граница(меньших чисел)

int j = size - 1; // правая граница(больших чисел)

int mid = arr[size / 2]; // ключ, от которого будем отталкиваться

do {

while (arr[i] < mid) { // ищем элемент, больший ключа

i++;

}

while (arr[j] > mid) { // ищем элемент, меньший ключа

j--;

}

if (i <= j) { // расстановка в правильном порядке

int tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

i++;

j--;

}

} while (i <= j); // цикл пока границы не сойдутся в середине

if (j > 0) fast(arr, j + 1); // вызов функции с другими границами и ключом в левой стороне

if (i < size) fast(&arr[i], size - i); // вызов функции с другими границами и ключом в правой стороне

}

int main() {

setlocale(LC\_ALL, "ru");

int n = 25;

int\* myarr = new int[n]; // создание массива

srand(time(0));

for (int i = 0; i < n; i++) { // заполнение массива рандомными числами

myarr[i] = rand() % 100 + 1;

}

cout << "Исходный массив: " << endl;

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

cout << endl << "Отсортированный массив: " << endl;

fast(myarr, n); // сортировка массива

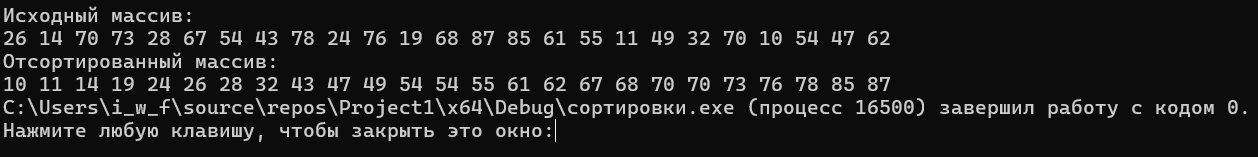
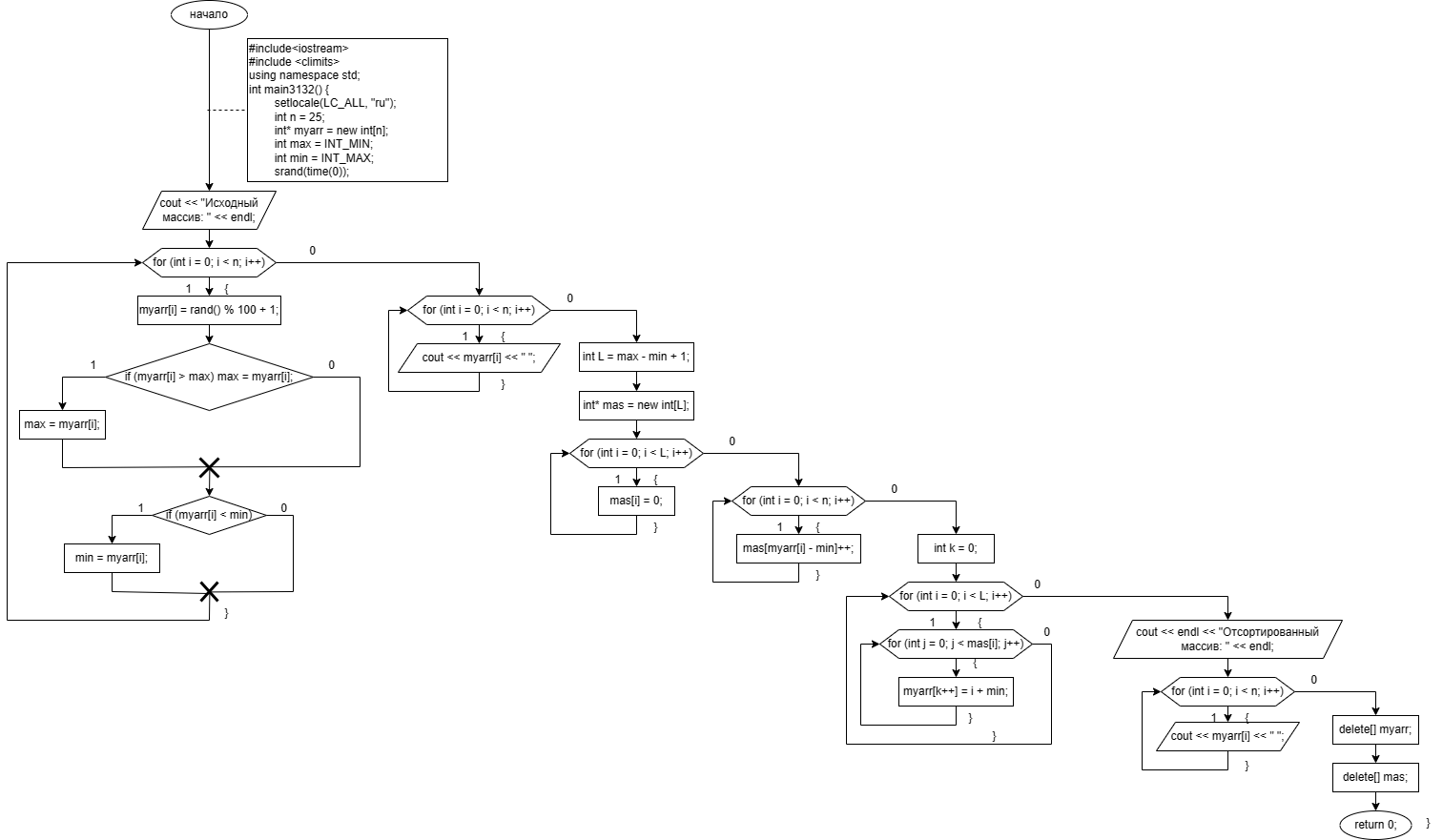
for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

return 0;

}

1. Результат работы программы:
2. **Сортировка подсчетом.**
3. Анализ задачи:
4. При создании массива параллельно производится поиск максимального и минимального чисел, для дальнейшего выявления количества элементов нового массива.
5. Создание нового массива, изначально содержащего одни 0.
6. Массив заполняется счетчиками количества каждого числа в изначальном массиве.
7. Далее благодаря массиву со счетчиками в массив записываются элементы в нужном порядке.
8. Блок-схема:
9. Код программы:

#include<iostream>

#include <climits>

using namespace std;

int main() {

setlocale(LC\_ALL, "ru");

int n = 25;

int\* myarr = new int[n]; // создание массива

int max = INT\_MIN;

int min = INT\_MAX;

srand(time(0));

cout << "Исходный массив: " << endl;

for (int i = 0; i < n; i++) { // заполнение массива рандомными числами и нахождение среди них максимального и минимального

myarr[i] = rand() % 100 + 1;

if (myarr[i] > max) max = myarr[i];

if (myarr[i] < min) min = myarr[i];

}

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

int L = max - min + 1; // количество элементов нового массива

int\* mas = new int[L]; // новый массив для подсчета количества каждого числа

for (int i = 0; i < L; i++) { // заполнение нового массива 0

mas[i] = 0;

}

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

mas[myarr[i] - min]++; // счетчик количества чисел в новом массиве увеличивается

}

int k = 0;

for (int i = 0; i < L; i++) {

for (int j = 0; j < mas[i]; j++) {

myarr[k++] = i + min; // запись в массив элементов в нужном порядке

}

}

cout << endl << "Отсортированный массив: " << endl;

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

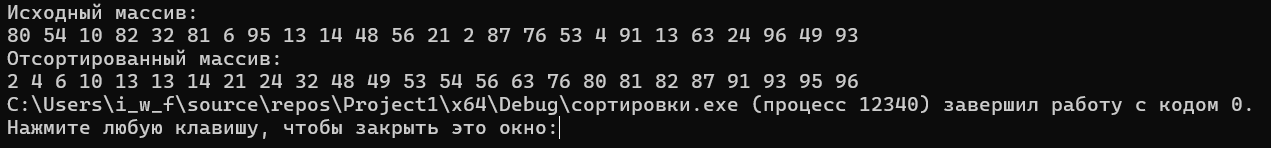
}

delete[] myarr;

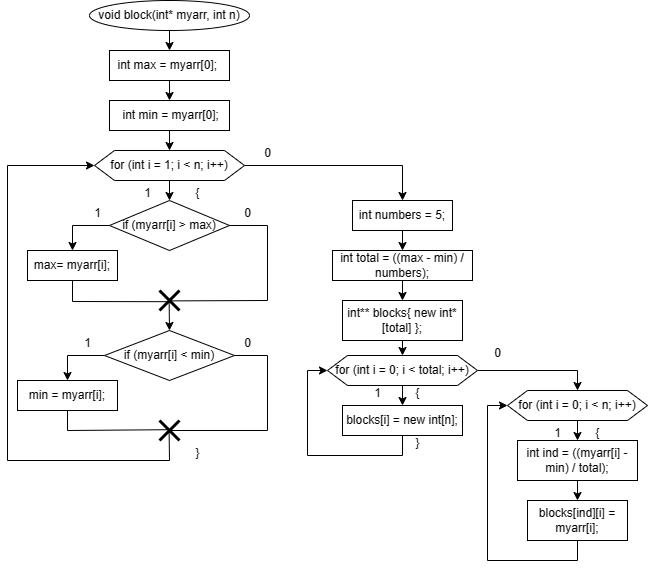
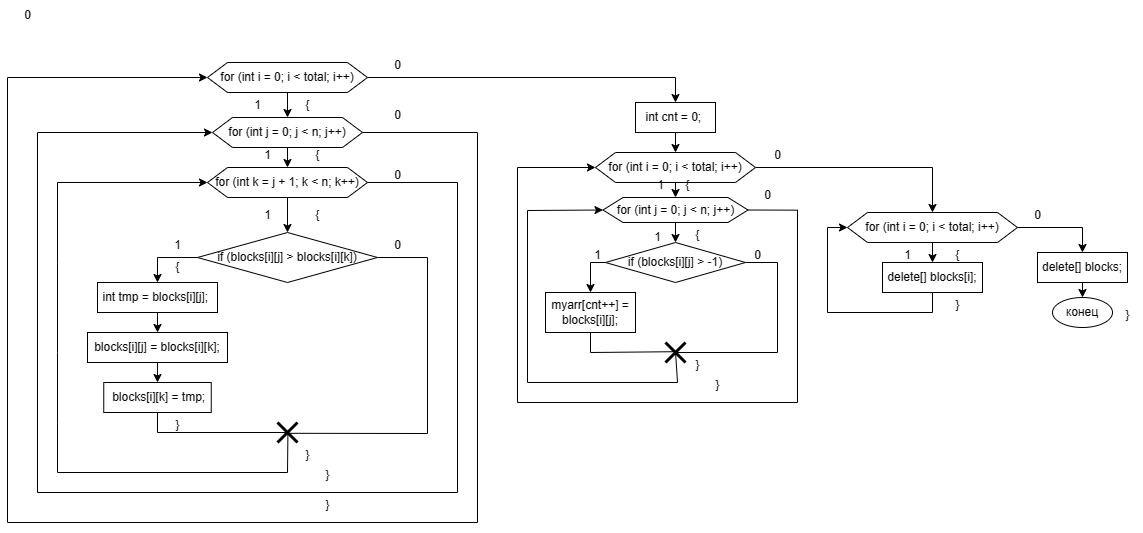
delete[] mas;

return 0;

}

4. Результат работы программы:

1. **Блочная сортировка**
2. Анализ задачи:
3. В цикле находится максимальный и минимальный элементы.
4. После этого в переменную total рассчитывается количество блоков и создается двумерный массив.
5. Цикл определяет в какой блок распределить число.
6. Далее элементы помещаются новый двумерный массив, отсортированные методом вставки.
7. В массив переписываются элементы в верном порядке.
8. Блок-схема:



1. Код программы:

#include<iostream>

using namespace std;

void block(int\* myarr, int n) { // функция для сортировки

int max = myarr[0];

int min = myarr[0];

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

if (myarr[i] > max) { // поиск максимального элемента

max= myarr[i];

}

if (myarr[i] < min) { // поиск минимального элемента

min = myarr[i];

}

}

int numbers = 5; // количество чисел в блоке

int total = ((max - min) / numbers); // вычисление количества блоков

int\*\* blocks{ new int\* [total] }; // создание двумерного массива

for (int i = 0; i < total; i++) {

blocks[i] = new int[n]; // создание столбцов массива

}

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

int ind = ((myarr[i] - min) / total); // определение, в какой блок поместить число

blocks[ind][i] = myarr[i];

}

for (int i = 0; i < total; i++) {

for (int j = 0; j < n; j++) {

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

if (blocks[i][j] > blocks[i][k]) { // сортировка методом вставки

int tmp = blocks[i][j];

blocks[i][j] = blocks[i][k]; // расставление элементов

blocks[i][k] = tmp;

}

}

}

}

int cnt = 0;

for (int i = 0; i < total; i++) {

for (int j = 0; j < n; j++) {

if (blocks[i][j] > -1) {

myarr[cnt++] = blocks[i][j]; // запись в массив элементов в нужном порядке

}

}

}

for (int i = 0; i < total; i++) {

delete[] blocks[i]; // очищение памяти

}

delete[] blocks; // очищение памяти

}

int main1() {

setlocale(LC\_ALL, "ru");

int n = 25;

int\* myarr = new int[n]; // создание массива

srand(time(0));

for (int i = 0; i < n; i++) { // заполнение массива рандомными числами

myarr[i] = rand() % 100 + 1;

}

cout << "Исходный массив: " << endl;

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

cout << endl << "Отсортированный массив: " << endl;

block(myarr, n); // вызов функции для сортировки массива

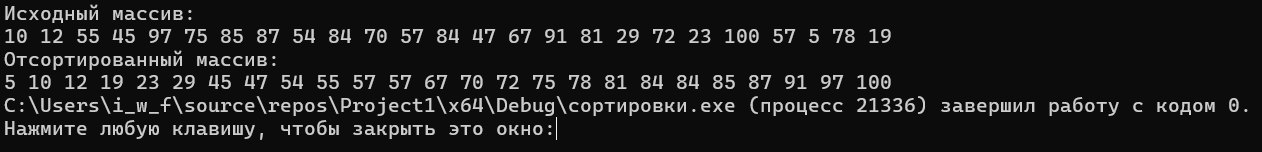
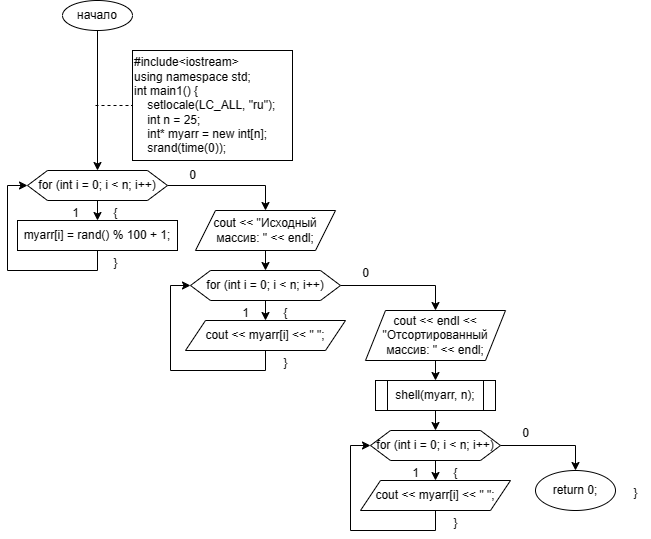
for (int i = 0; i < n; i++) { // вывод массива

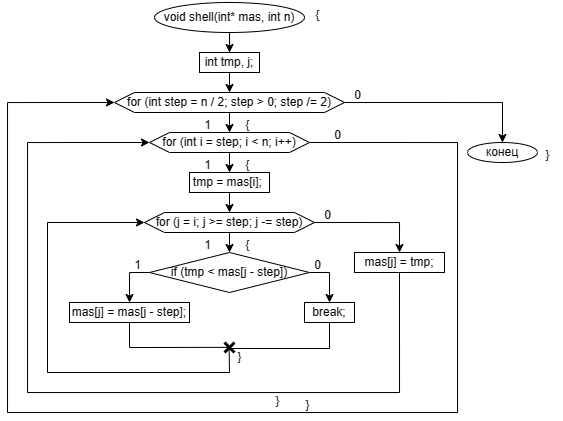
cout << myarr[i] << " ";

}

return 0;

}

1. Результат работы программы:
2. **Сортировка методом Шелла**
3. Анализ задачи:
4. Создание функции, которая будет производить сортировку с помощью цикла, который делит массива на списки шагом (step), который тоже будет меняться.
5. В функции начинается второй цикл, с помощью которого будут рассматриваться каждый список по отдельности.
6. Если элементы списка расположены в неверном порядке, то элементы поменяются местами.
7. После того, как программа обработает все списки с разными шагами, массив будет отсортирован.
8. Блок-схема:



1. Код программы:

#include<iostream>

using namespace std;

void shell(int\* mas, int n) {

int tmp, j;

for (int step = n / 2; step > 0; step /= 2) { // деление массива

for (int i = step; i < n; i++) { // прохождение по части массива

tmp = mas[i];

for (j = i; j >= step; j -= step) { // прохождение по массиву с шагом step

if (tmp < mas[j - step]) { // если элементы в неправильном порядке

mas[j] = mas[j - step];

}

else {

break; // прерывание цикла

}

}

mas[j] = tmp; // замена элемента

}

}

}

int main() {

setlocale(LC\_ALL, "ru");

int n = 25;

int\* myarr = new int[n]; // создание массива

srand(time(0));

for (int i = 0; i < n; i++) { // заполнение массива рандомными числами

myarr[i] = rand() % 100 + 1;

}

cout << "Исходный массив: " << endl;

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

cout << endl << "Отсортированный массив: " << endl;

shell(myarr, n); // вызов функции для сортировки массива

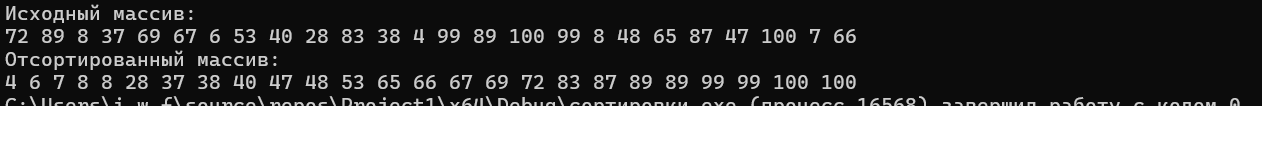
for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

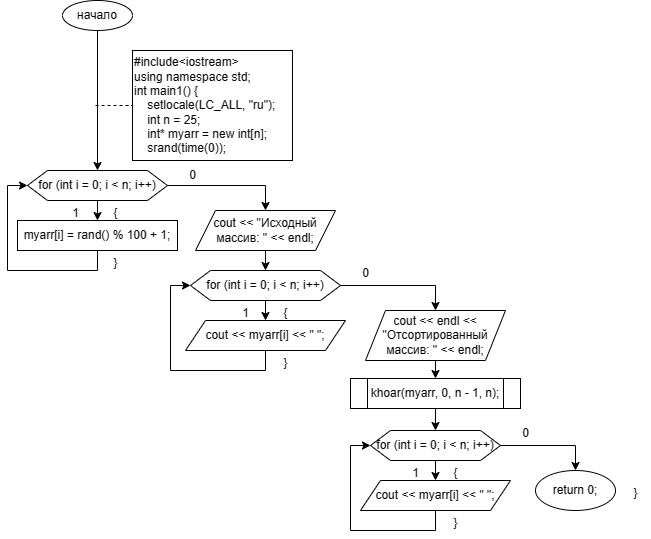
}

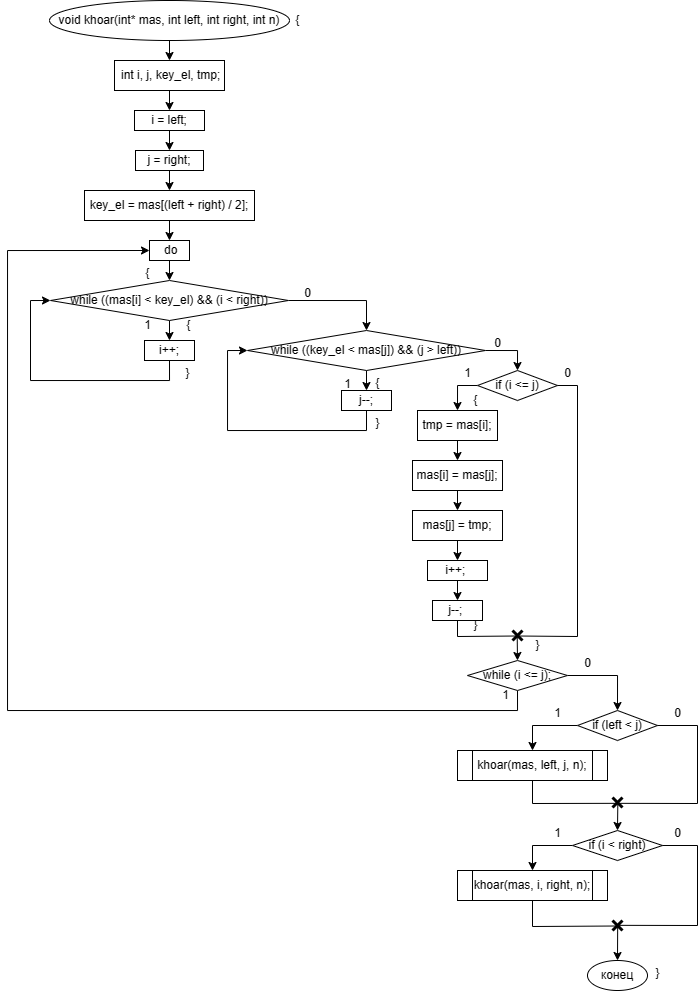
return 0;

}

1. Результат работы программы:

**6. Сортировка методом Хоара**

1. Анализ задачи:
2. Создание функции, которая принимает массив с границами, изначально это 1 и последний элемент.
3. При каждом вызове функции рассчитывается опорный элемент.
4. Затем начинаются два цикла: первый ищет в левой части элемент больше опорного; второй ищет в правой части элемент меньше опорного.
5. После нахождения элементы меняются местами (сортируются).
6. Так, функция будет вызываться с другими параметрами до тех пор, пока весь массив не будет отсортирован.
7. Блок-схема:



1. Код программы:

#include<iostream>

using namespace std;

void khoar(int\* mas, int left, int right, int n) {

int i, j, key\_el, tmp;

i = left; // левая граница

j = right; // правая граница

key\_el = mas[(left + right) / 2]; // создание опорного элемента, равен середине

do {

while ((mas[i] < key\_el) && (i < right)) i++; // поиск элемента в левой части

while ((key\_el < mas[j]) && (j > left)) j--; // поиск элемента в правой части

if (i <= j) {

tmp = mas[i];

mas[i] = mas[j]; // замена элементов

mas[j] = tmp; // замена элементов

i++;

j--;

}

} while (i <= j);

if (left < j) khoar(mas, left, j, n); // вызов функции с другой правой границей

if (i < right) khoar(mas, i, right, n); // вызов функции с другой левой границей

}

int main() {

setlocale(LC\_ALL, "ru");

int n = 25;

int\* myarr = new int[n]; // создание массива

srand(time(0));

for (int i = 0; i < n; i++) { // заполнение массива рандомными числами

myarr[i] = rand() % 100 + 1;

}

cout << "Исходный массив: " << endl;

for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

cout << endl << "Отсортированный массив: " << endl;

khoar(myarr, 0, n - 1, n); // вызов функции для сортировки массива

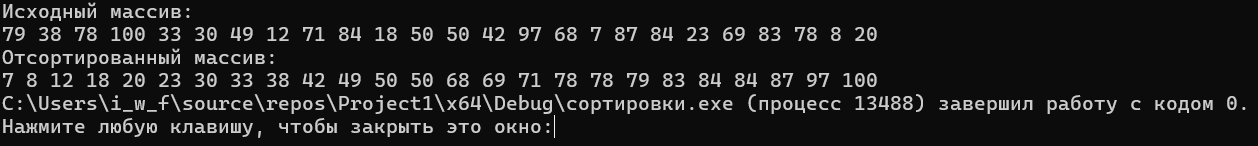
for (int i = 0; i < n; i++) { // вывод массива

cout << myarr[i] << " ";

}

return 0;

}

1. Результат работы программы: