МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка информационных систем (РИС)

**О Т Ч Ё Т**   
**о лабораторной работе №1**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Сортировки Шелла, Хоара»

Выполнил:

Студент группы РИС-24-3б

Суяргулов А.О.

Проверил:

Доцент кафедры ИТАС

Полякова О.А.

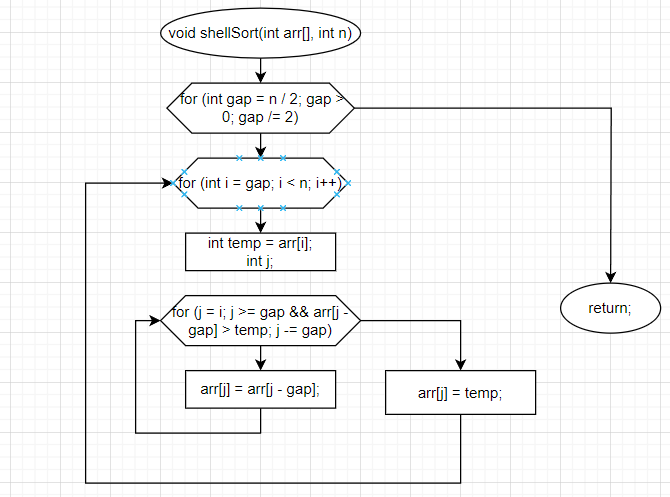
Пермь, 2025

ВВЕДЕНИЕ

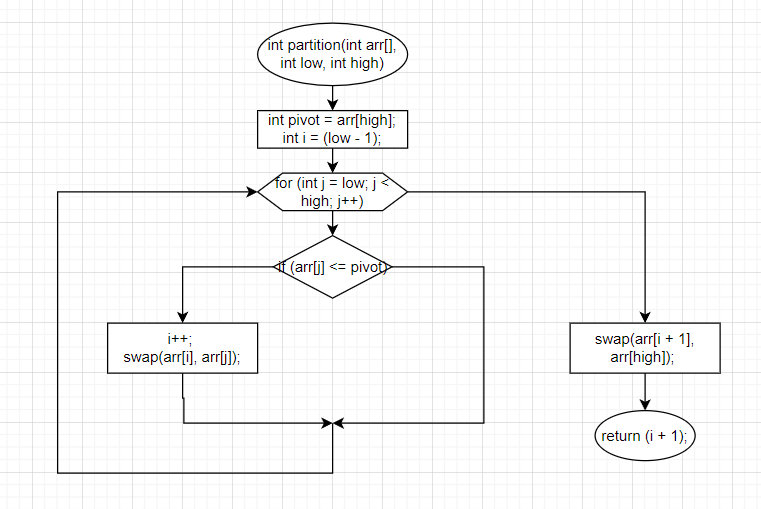
Реализовать метод Шелла и Хоара для сортировки массива.

ОСНОВНАЯ ЧАСТЬ

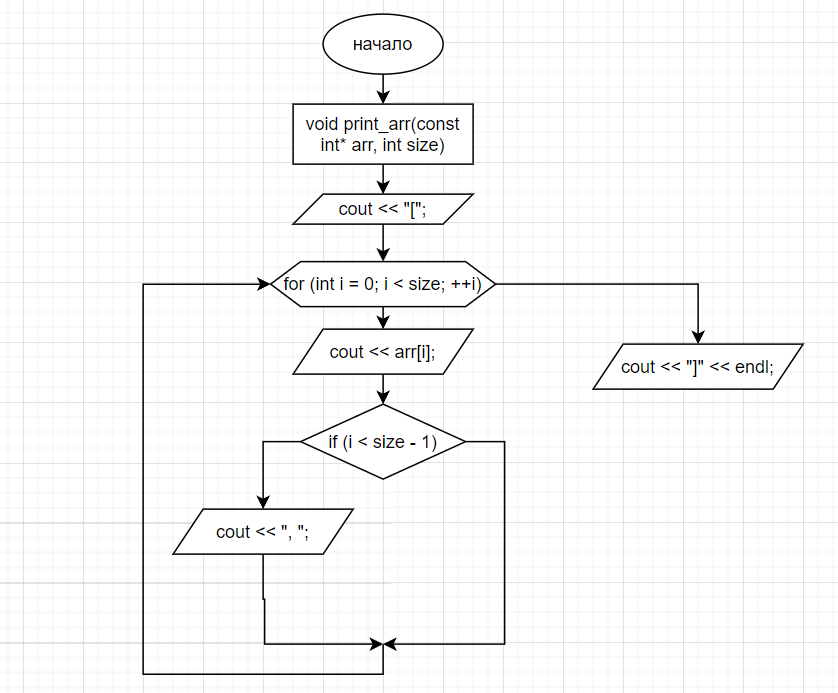
**Блок-схема метода Шелла:**

****

**Блок схема метода Хоара:**



**Функция для печати массива:**

****

**Код:**

#include <iostream>

using namespace std; // Используем пространство имен std

// Функция для сортировки методом Шелла

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

// Начинаем с большого шага и уменьшаем его

for (int gap = n / 2; gap > 0; gap /= 2) {

// Проходим по всем элементам массива, начиная с индекса gap

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

int temp = arr[i]; // Сохраняем текущий элемент

int j;

// Сравниваем и перемещаем элементы, которые находятся на расстоянии gap

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap) {

arr[j] = arr[j - gap]; // Сдвигаем элемент

}

arr[j] = temp; // Вставляем сохраненный элемент на правильное место

}

}

}

// Функция для разделения массива в быстрой сортировке

int partition(int arr[], int low, int high) {

int pivot = arr[high]; // Опорный элемент (последний элемент)

int i = (low - 1); // Индекс меньшего элемента

// Проходим по всем элементам от low до high

for (int j = low; j < high; j++) {

// Если текущий элемент меньше или равен опорному

if (arr[j] <= pivot) {

i++; // Увеличиваем индекс меньшего элемента

swap(arr[i], arr[j]); // Меняем местами

}

}

swap(arr[i + 1], arr[high]); // Меняем опорный элемент с элементом после меньшего

return (i + 1); // Возвращаем индекс опорного элемента

}

// Функция быстрой сортировки

void quickSort(int arr[], int low, int high) {

if (low < high) {

// Индекс, по которому массив будет разделен

int pi = partition(arr, low, high);

// Рекурсивно сортируем элементы до и после разделения

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

// Функция для вывода массива

void printArray(int arr[], int n) {

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

cout << arr[i] << " "; // Выводим каждый элемент

}

cout << endl; // Переход на новую строку

}

int main() {

setlocale(LC\_ALL, "Ru");

// Инициализация массива для сортировки методом Шелла

int arr1[] = { 12, 34, 54, 2, 3 };

int n1 = sizeof(arr1) / sizeof(arr1[0]); // Определяем размер массива

cout << "Исходный массив для сортировки Шелла: ";

printArray(arr1, n1); // Выводим исходный массив

shellSort(arr1, n1); // Сортируем массив методом Шелла

cout << "Отсортированный массив (метод Шелла): ";

printArray(arr1, n1); // Выводим отсортированный массив

// Инициализация массива для быстрой сортировки

int arr2[] = { 12, 34, 54, 2, 3 };

int n2 = sizeof(arr2) / sizeof(arr2[0]); // Определяем размер массива

cout << "Исходный массив для быстрой сортировки: ";

printArray(arr2, n2); // Выводим исходный массив

quickSort(arr2, 0, n2 - 1); // Сортируем массив методом Хоара

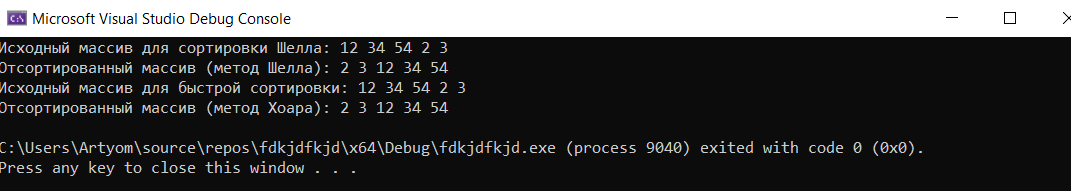
cout << "Отсортированный массив (метод Хоара): ";

printArray(arr2, n2); // Выводим отсортированный массив

return 0; // Завершаем программу

}

**Результат работы кода:**

****

**Ссылка на GitHub:**