Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

Лабораторная работа  
«Методы Сортировки. Сложные соритровки»

Выполнил:   
студент группы РИС-23-1б   
Мокрушин Никита Дмитриевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

Пермь 2024 г.

**Сортировка Естественным слиянием**

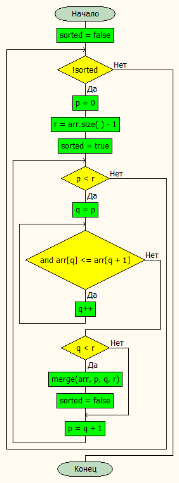
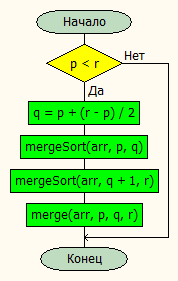
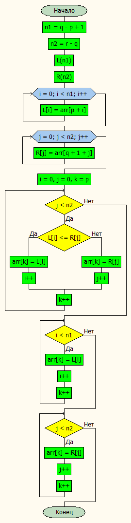
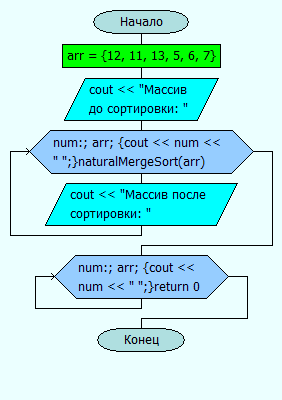
**Постановка задачи:**

Отсортировать массив, состоящий из 25-ти элементов, используя метод «Естественного слияния». Для каждого метода подобрать подходящий под метод массив.

**Анализ задачи:**

* Алгоритм начинает считать элементы массива до тех пор, пока они упорядочены.
* Как только обнаруживается элемент, нарушающий порядок, выполняется слияние всех упорядоченных элементов.
* После слияния массив просматривается снова, чтобы найти следующий отсортированный фрагмент.
* Этот процесс повторяется до тех пор, пока весь массив не будет отсортирован.

**Блок-схема:**



**Код на языке с++:**

#include <iostream>

#include <vector>

using namespace std;

void merge(vector<int>& arr, int p, int q, int r) {

int n1 = q - p + 1;

int n2 = r - q;

vector<int> L(n1);

vector<int> R(n2);

for (int i = 0; i < n1; i++) {

L[i] = arr[p + i];

}

for (int j = 0; j < n2; j++) {

R[j] = arr[q + 1 + j];

}

int i = 0, j = 0, k = p;

while (i < n1 and j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(vector<int>& arr, int p, int r) {

if (p < r) {

int q = p + (r - p) / 2;

mergeSort(arr, p, q);

mergeSort(arr, q + 1, r);

merge(arr, p, q, r);

}

}

void naturalMergeSort(vector<int>& arr) {

bool sorted = false;

while (!sorted) {

int p = 0;

int r = arr.size() - 1;

sorted = true;

while (p < r) {

int q = p;

while (q < r and arr[q] <= arr[q + 1]) {

q++;

}

if (q < r) {

merge(arr, p, q, r);

sorted = false;

}

p = q + 1;

}

}

}

int main() {

vector<int> arr = { 12, 11, 13, 5, 6, 7 };

cout << "Массив до сортировки: ";

for (int num : arr) {

cout << num << " ";

}

naturalMergeSort(arr);

cout << "Массив после сортировки: ";

for (int num : arr) {

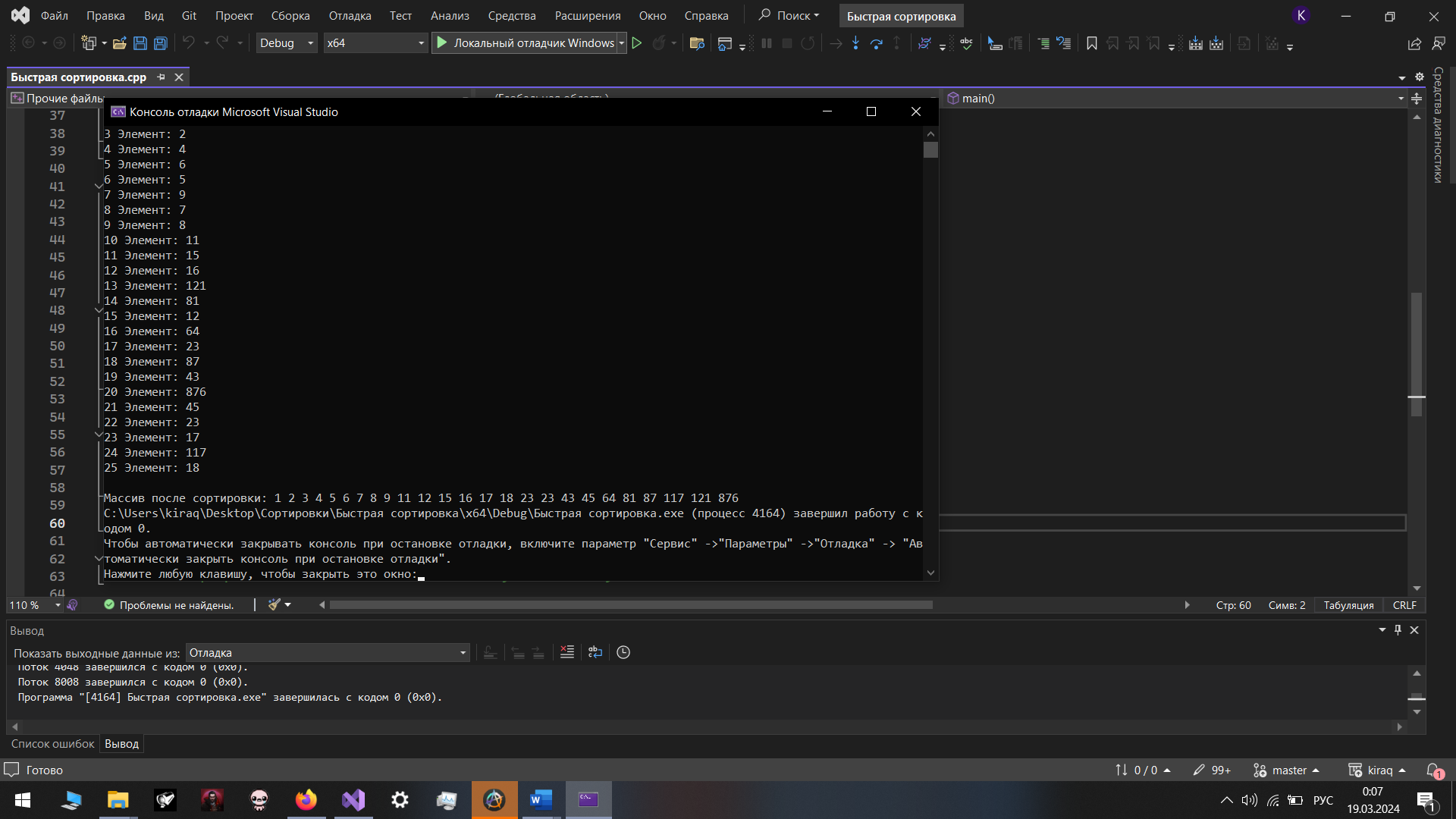
cout << num << " ";

}

return 0;

}

**Пример работы кода:**



**Вывод:**

Код отработал, как и задумывалось по алгоритму

**Многофазная сортировка**

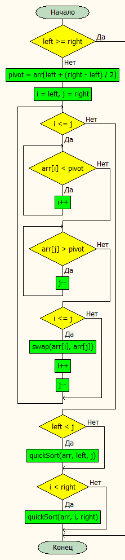
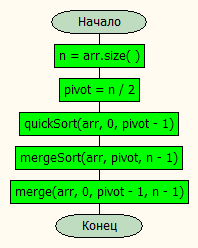
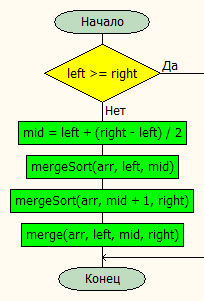
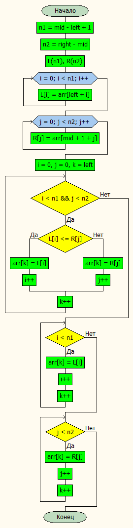
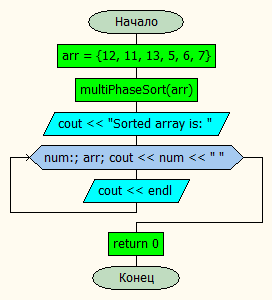
**Постановка задачи:**

Отсортировать массив, состоящий из 25-ти элементов, используя метод «Многофазной сортировки». Для каждого метода подобрать подходящий под метод массив.

**Анализ задачи:**

* Алгоритм начинает считать элементы массива до тех пор, пока они упорядочены.
* Как только обнаруживается элемент, нарушающий порядок, выполняется слияние всех упорядоченных элементов.
* После слияния массив просматривается снова, чтобы найти следующий отсортированный фрагмент.
* Этот процесс повторяется до тех пор, пока весь массив не будет отсортирован.

**Блок-схема:**

****

**Код на языке с++:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void merge(vector<int>& arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

vector<int> L(n1), R(n2);

for (int i = 0; i < n1; i++)

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

for (int j = 0; j < n2; j++)

R[j] = arr[mid + 1 + j];

int i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(vector<int>& arr, int left, int right) {

if (left >= right) return;

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

void quickSort(vector<int>& arr, int left, int right) {

if (left >= right) return;

int pivot = arr[left + (right - left) / 2];

int i = left, j = right;

while (i <= j) {

while (arr[i] < pivot) i++;

while (arr[j] > pivot) j--;

if (i <= j) {

swap(arr[i], arr[j]);

i++;

j--;

}

}

if (left < j) quickSort(arr, left, j);

if (i < right) quickSort(arr, i, right);

}

void multiPhaseSort(vector<int>& arr) {

int n = arr.size();

int pivot = n / 2;

quickSort(arr, 0, pivot - 1);

mergeSort(arr, pivot, n - 1);

merge(arr, 0, pivot - 1, n - 1);

}

int main() {

vector<int> arr = { 12, 11, 13, 5, 6, 7 };

multiPhaseSort(arr);

cout << "Sorted array is: ";

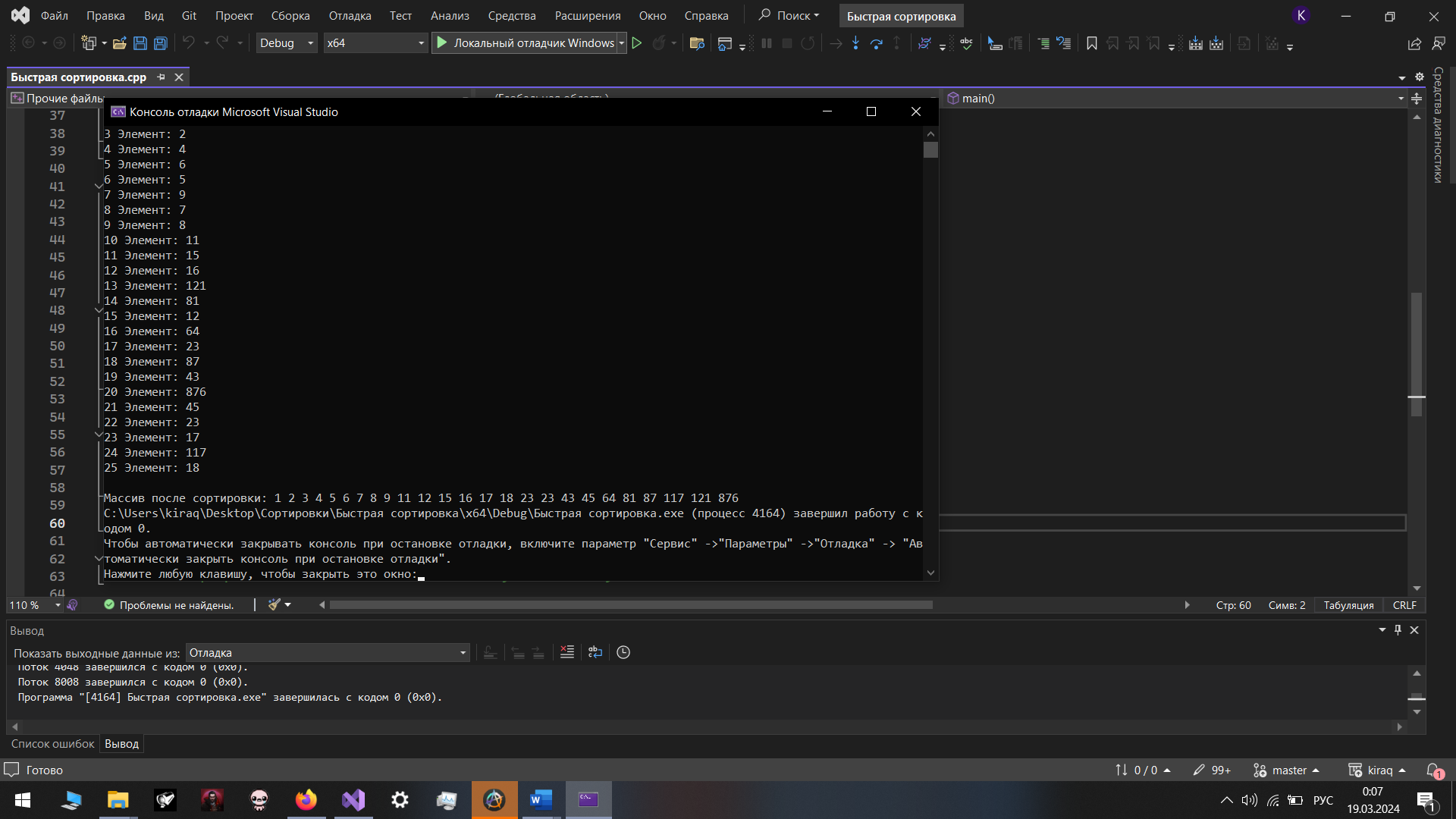
for (int num : arr)

cout << num << " ";

cout << endl;

return 0;

}**Пример работы кода:**



**Вывод:**

Код отработал, как и задумывалось по алгоритму