Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**Тема:** Лабораторная работа №12\_сортировки

Семестр: 2

Выполнил студент ИВТ-23-2б:

Синицын Владислав Сергеевич

(дата, подпись)

Проверила:

Полякова Ольга Андреевна

(дата, подпись)

Пермь 2024

Задание:

1.блочная сортировка

2.слияние

3.быстрая

4.подсчетом

5.Шелл

6.Хоар

7.естественное слияние

8.двухфазная

Программное решение:

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

static void BucketSort(int count) {

int data[] = { -200, 1340, 14223, 1, 4,

-4, 7, 1313,-143124, 31,

0, 12, -3, 767, 3123,

12234, 2345, -23412, 42, 23,

26, 722, -312, -1, -131331};

int minValue = data[0];

int maxValue = data[0];

cout << "Массив до сортировки: \n";

for (int i = 0; i != 25; i++) { //вывод массива

cout << data[i] << endl;

}

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

{

if (data[i] > maxValue)

maxValue = data[i];

if (data[i] < minValue)

minValue = data[i];

}

int bucketLength = maxValue - minValue + 1;

vector<int>\* bucket = new vector<int>[bucketLength];

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

{

bucket[i] = vector<int>();

}

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

{

bucket[data[i] - minValue].push\_back(data[i]);

}

int k = 0;

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

{

int bucketSize = bucket[i].size();

if (bucketSize > 0)

{

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

{

data[k] = bucket[i][j];

k++;

}

}

}

cout << "\nМассив после сортировки: \n";

for (int i=0; i != 25; i++) { //вывод массива

cout << data[i] << endl;

}

}

void merge(int array[], int const left,

int const mid, int const right)

{

auto const subArrayOne = mid - left + 1;

auto const subArrayTwo = right - mid;

auto\* leftArray = new int[subArrayOne],

\* rightArray = new int[subArrayTwo];

for (auto i = 0; i < subArrayOne; i++)

leftArray[i] = array[left + i];

for (auto j = 0; j < subArrayTwo; j++)

rightArray[j] = array[mid + 1 + j];

auto indexOfSubArrayOne = 0,

indexOfSubArrayTwo = 0;

int indexOfMergedArray = left;

while (indexOfSubArrayOne < subArrayOne &&

indexOfSubArrayTwo < subArrayTwo)

{

if (leftArray[indexOfSubArrayOne] <=

rightArray[indexOfSubArrayTwo])

{

array[indexOfMergedArray] =

leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

}

else

{

array[indexOfMergedArray] =

rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

}

indexOfMergedArray++;

}

while (indexOfSubArrayOne < subArrayOne)

{

array[indexOfMergedArray] =

leftArray[indexOfSubArrayOne];

indexOfSubArrayOne++;

indexOfMergedArray++;

}

while (indexOfSubArrayTwo < subArrayTwo)

{

array[indexOfMergedArray] =

rightArray[indexOfSubArrayTwo];

indexOfSubArrayTwo++;

indexOfMergedArray++;

}

}

void mergeSort(int array[],

int const begin,

int const end)

{

if (begin >= end)

return;

auto mid = begin + (end - begin) / 2;

mergeSort(array, begin, mid);

mergeSort(array, mid + 1, end);

merge(array, begin, mid, end);

}

int partition(int arrs[], int start, int end)

{

int pivot = arrs[start];

int count = 0;

for (int i = start + 1; i <= end; i++) {

if (arrs[i] <= pivot)

count++;

}

int pivotIndex = start + count;

swap(arrs[pivotIndex], arrs[start]);

int i = start, j = end;

while (i < pivotIndex && j > pivotIndex) {

while (arrs[i] <= pivot) {

i++;

}

while (arrs[j] > pivot) {

j--;

}

if (i < pivotIndex && j > pivotIndex) {

swap(arrs[i++], arrs[j--]);

}

}

return pivotIndex;

}

void quickSort(int arr[], int start, int end)

{

if (start >= end)

return;

int p = partition(arr, start, end);

quickSort(arr, start, p - 1);

quickSort(arr, p + 1, end);

}

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

int arr1[25];

int count\_arr[25];

int x = arr[0];

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

if (arr[i] > x)

x = arr[i];

}

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

count\_arr[i] = 0;

}

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

count\_arr[arr[i]]++;

}

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

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

}

for (int i = n - 1; i >= 0; i--) {

arr1[count\_arr[arr[i]] - 1] = arr[i];

count\_arr[arr[i]]--;

}

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

arr[i] = arr1[i];

}

for (int i = 0; i != 25; i++) { //вывод массива

cout << arr[i] << " ";

}

}

int shellSort(int arr[], int n)

{

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

{

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

{

int temp = arr[i];

int j;

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

arr[j] = arr[j - gap];

arr[j] = temp;

}

}

return 0;

}

void printArray(int arr[], int n)

{

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

cout << arr[i] << " ";

}

void hoarasort(int\* a, int first, int last)

{

int i = first, j = last;

double tmp, x = a[(first + last) / 2];

do {

while (a[i] < x)

i++;

while (a[j] > x)

j--;

if (i <= j)

{

if (i < j)

{

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

i++;

j--;

}

} while (i <= j);

if (i < last)

hoarasort(a, i, last);

if (first < j)

hoarasort(a, first, j);

}

void merge(vector<int>& arr, int l, int m, int r) {

int n1 = m - l + 1;

int n2 = r - m;

vector<int> L(n1);

vector<int> R(n2);

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

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

}

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

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

}

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

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 naturalMergeSort(vector<int>& arr, int l, int r) {

if (l < r) {

int m = l + (r - l) / 2;

naturalMergeSort(arr, l, m);

naturalMergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

void polyphaseSort(vector<int>& arr, int k) {

sort(arr.begin(), arr.end()); // Начальная сортировка массива

int n = arr.size();

vector<int> temp(n);

int batch\_size = n / k;

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

sort(arr.begin() + i, arr.begin() + min(i + k, n)); // Сортировка каждой подгруппы размером k

// Выполняем слияние

if (i + k < n) {

merge(arr.begin() + i, arr.begin() + i + k, arr.begin() + i + k, arr.begin() + min(i + 2 \* k, n), temp.begin());

copy(temp.begin(), temp.begin() + min(2 \* k, n - i), arr.begin() + i);

}

}

}

int main() {

system("chcp 1251 > NULL");

cout << "Введите номер метода: " << endl << "1.блочная сортировка" << endl << "2.слияние" << endl << "3.быстрая" << endl

<< "4.подсчетом" << endl << "5.Шелл" << endl << "6.Хоар" << endl << "7.слияние" << endl << "8.двухфазная" << endl << endl;

int i;

cin >> i;

cout << endl;

int arr[] = { 12, 11, 13, 5, 6, //массив для 2 м 5 метода

7, 9, 10, 14, 15

,17 , 19, 1, 0, 34,

2, 6, 8 ,24, 62,

21, 3, 36, 7, 3 };

int arrs[] = { 12, 11, 13, 5, 6, //массив для 3 метода

7, 9, 10, 14, 15

,17 , 19, 1, 0, 34,

2, 6, 8 ,24, 62,

21, 3, 36, 7, 3 };

int arrf[] = { 2, 2, 2, 3, 3, 4, 8, 1, 2, 3, //массив для 4 метода

4, 5, 2, 2, 4,

2, 2, 2, 2, 2,

2, 2, 2, 2, 2 };

int arrd[] = { 2, 2, 2, 3, 3, 4, 8, 1, 2, 3, //массив для 6 метода

4, 5, 2, 2, 4,

2, 2, 2, 2, 2,

2, 2, 2, 2, 2 };

std::vector<int> arrg = { 12, 11, 13, 5, 6, //массив для 7 метода

7, 8, 9, 10, 11,

12, 12, 13, 1, 2,

6, 7, 13, 15, 1,

2, 5, 6, 23, 32};

int k = 3; // Размер подгруппы

int n = 25;

auto arr\_size = sizeof(arr) / sizeof(arr[0]);

switch (i) {

case 1:

BucketSort(25);

break;

case 2:

cout << "Исходный массив " << endl;

printArray(arr, arr\_size);

mergeSort(arr, 0, arr\_size - 1);

cout << "\nОтсортированный массив " << endl;

printArray(arr, arr\_size);

break;

case 3:

cout << "Массив до сортировки: \n";

printArray(arrs, n);

cout << "\nМассив после сортировки: \n";

quickSort(arrs, 0, n - 1);

printArray(arrs, 25);

break;

case 4:

cout << "Массив до сортировки: \n";

printArray(arrf, n);

cout << "\nМассив после сортировки: \n";

n = sizeof(arrf) / sizeof(arrf[0]);

countSort(arrf, n);

break;

case 5:

n = sizeof(arr) / sizeof(arr[0]);

cout << "Массив до сортировки: \n";

printArray(arr, n);

shellSort(arr, n);

cout << "\nМассив после сортировки: \n";

printArray(arr, n);

break;

case 6:

cout << "Массив до сортировки: ";

printArray(arrd, 25);

cout << "\nМассив после сортировки: ";

hoarasort(arrd, 0, n - 1);

printArray(arrd, 25);

break;

case 7:

cout << "Массив до сортировки: ";

for (int num : arrg) {

cout << num << " ";

}

naturalMergeSort(arrg, 0, arrg.size() - 1);

cout << "\nМассив после сортировки: ";

for (int num : arrg) {

cout << num << " ";

}

break;

case 8:

cout << "Массив до сортировки: ";

for (int num : arrg) {

cout << num << " ";

}

polyphaseSort(arrg, k);

cout << "\nМассив после сортировки: ";

for (int num : arrg) {

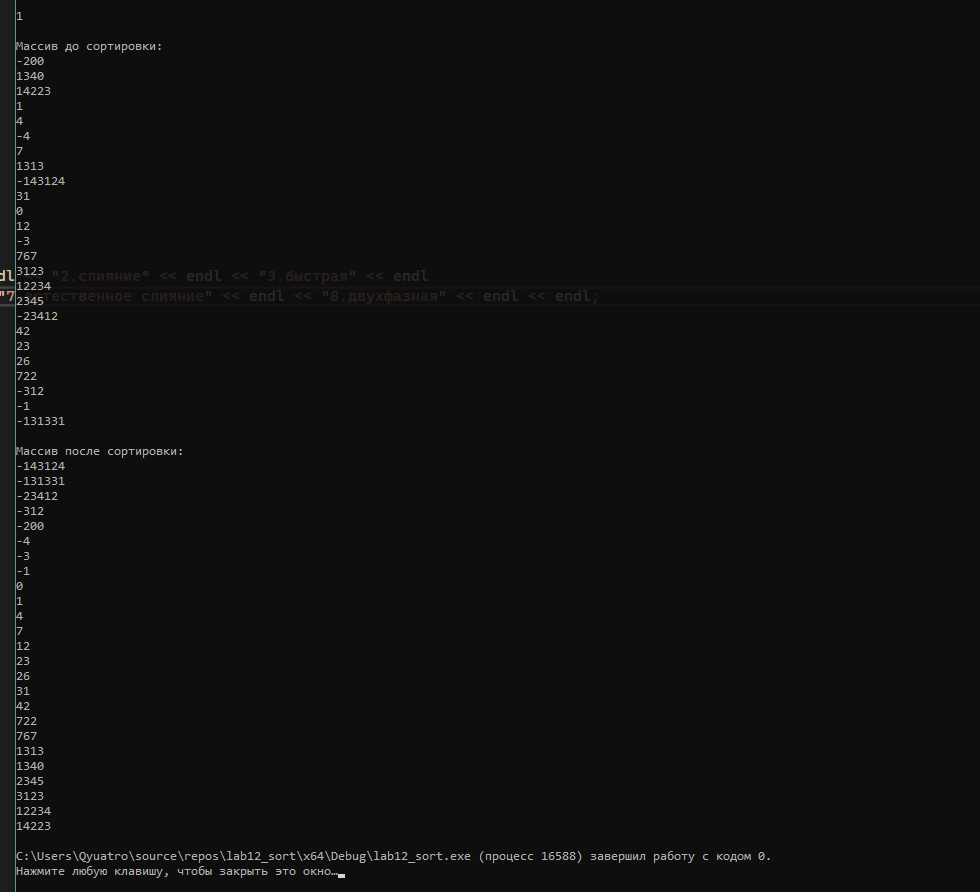
cout << num << " ";

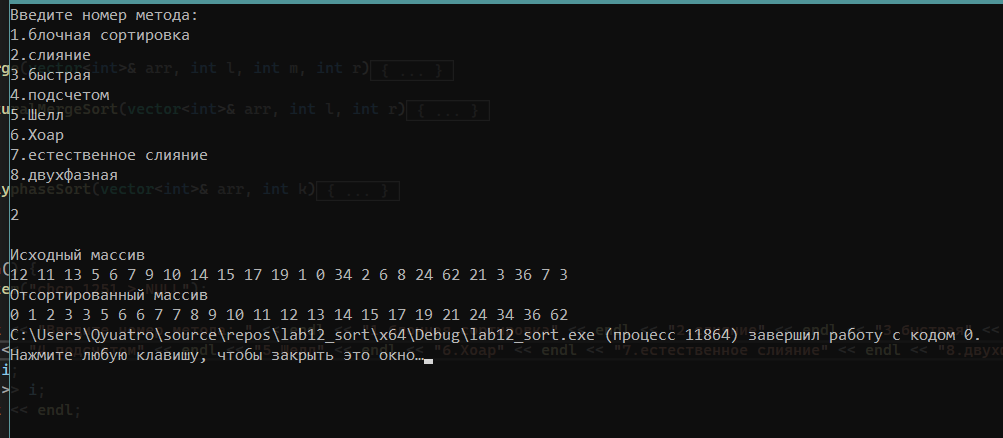
}

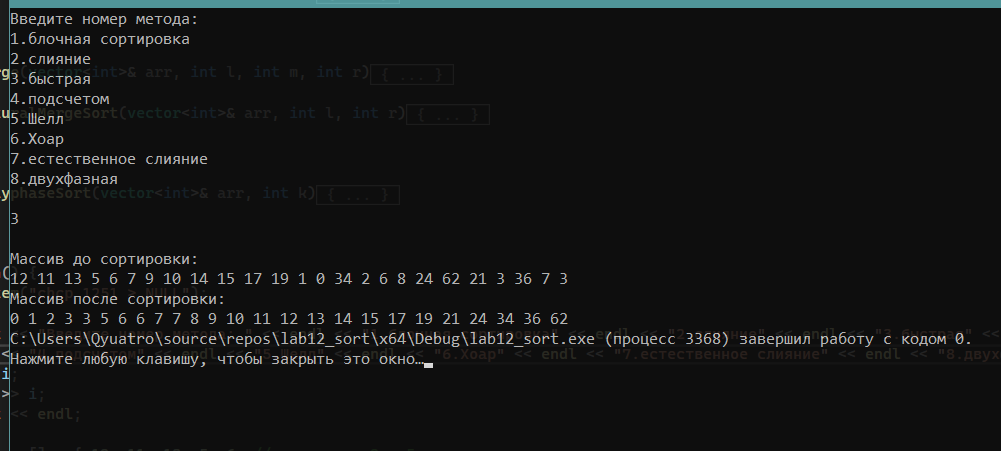
break;

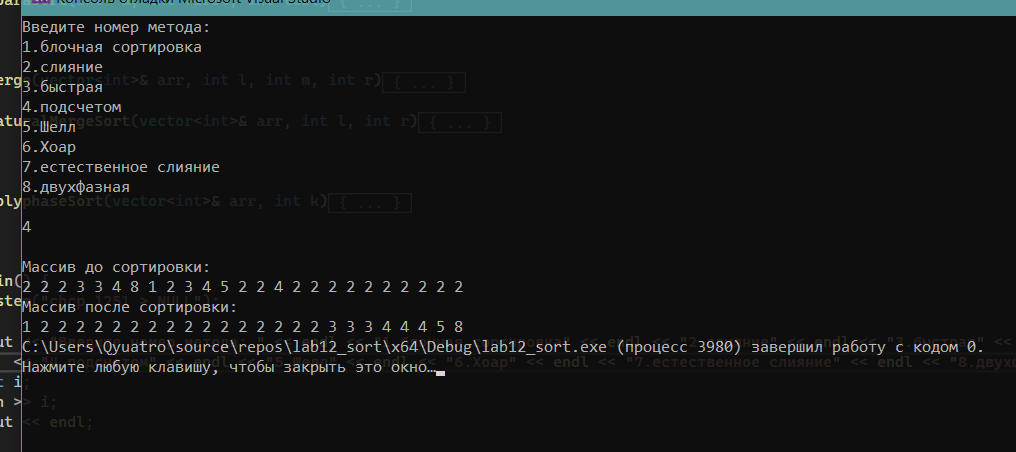
}

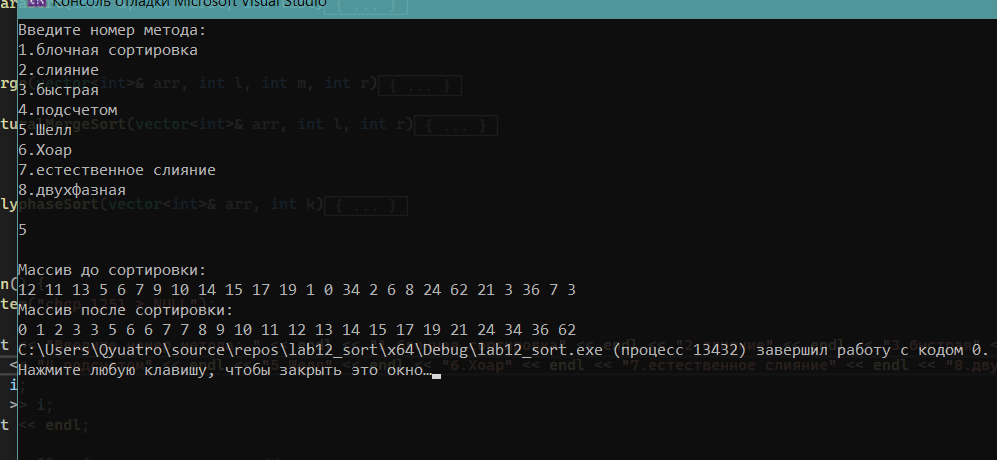
}

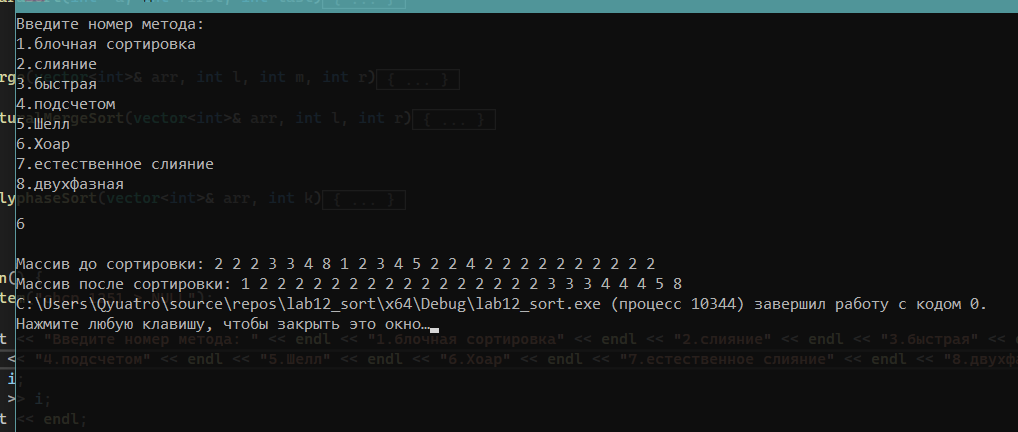


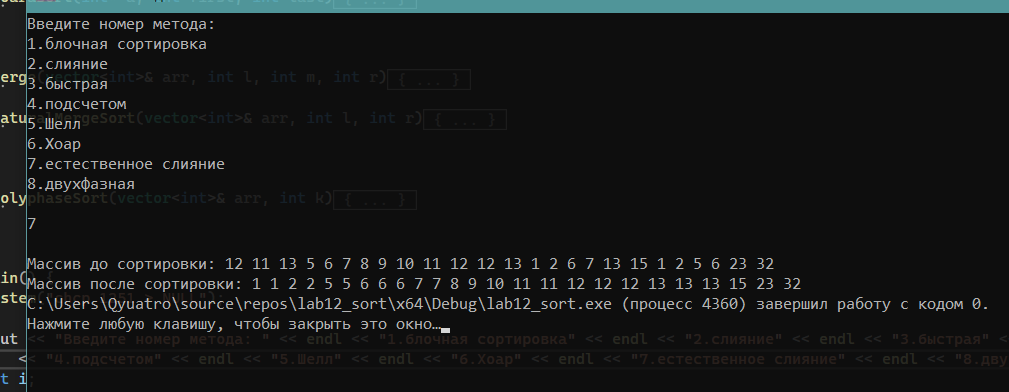














Main:

