МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа:**

"Внешние сортировки"

Выполнила: студентка гр.РИС-23-3Б

Мокроусова Ангелина Андреевна

Проверила: доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2024

**Постановка задачи:**

Написать программы в которых реализованы следующие методы сортировки: многофазная, естественное слияние.

**Анализ задачи:**

1. Многофазная сортировка:

* Метод, который использует разбиение массива на несколько подмассивов и их последующее сортирование отдельно.
* Каждая фаза сортировки может выполняться разными методами, например, сортировка слиянием или пузырьковая сортировка.
* Результаты сортировки в каждой фазе объединяются в единый отсортированный массив.
* Имеет высокую эффективность и применяется в случаях, когда требуется отсортировать большие объемы данных.

2. Сортировка естественным слиянием:

* Метод, основанный на разбиении массива на подмассивы (заранее отсортированные) и последующем их слиянии.
* Сортировка происходит путем слияния двух подмассивов, при этом сохраняется порядок сортировки.
* Подходит для сортировки больших массивов данных, так как не требует дополнительной памяти для создания дополнительных структур данных.
* Один из методов внешней сортировки, который эффективен при работе с большими объемами данных, но требует более сложной реализации.

**Код на С++(Многофазная):**

#include <iostream>

using namespace std;

void merge(int arr[], int l, int m, int r) {

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

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

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

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

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

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 multiPhaseSort(int arr[], int n) {

int curr\_size;

int left\_start;

for (curr\_size = 1; curr\_size <= n - 1; curr\_size = 2 \* curr\_size) {

for (left\_start = 0; left\_start < n - 1; left\_start += 2 \* curr\_size) {

int mid = min(left\_start + curr\_size - 1, n - 1);

int right\_end = min(left\_start + 2 \* curr\_size - 1, n - 1);

merge(arr, left\_start, mid, right\_end);

}

}

}

int main() {

setlocale(LC\_ALL, "ru");

int arr[] = { 12, 11, 13, 5, 6, 7 };

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

cout << "До:" << endl;

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

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

}

cout << endl;

multiPhaseSort(arr, n);

cout << "После:" << endl;

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

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

}

std::cout << endl;

return 0;

}

**Код на С++(Естественное слияние):**

#include <iostream>

#include <vector>

using namespace std;

void naturalMergeSort(vector<int>& arr) {

int size = arr.size();

vector<int> result(size);

int i = 1;

while (i < size) {

int left = 0;

while (left < size) {

int mid = left + i;

int right = min(left + 2 \* i, size);

int leftIndex = left;

int rightIndex = mid;

int resultIndex = left;

while (leftIndex < mid && rightIndex < right) {

if (arr[leftIndex] < arr[rightIndex]) {

result[resultIndex++] = arr[leftIndex++];

}

else {

result[resultIndex++] = arr[rightIndex++];

}

}

while (leftIndex < mid) {

result[resultIndex++] = arr[leftIndex++];

}

while (rightIndex < right) {

result[resultIndex++] = arr[rightIndex++];

}

left += 2 \* i;

}

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

arr[j] = result[j];

}

i \*= 2;

}

}

int main() {

setlocale(LC\_ALL, "ru");

vector<int> arr = { 12, 11, -13, 0, 6, 6 };

cout << "До:" << endl;

for (int num : arr) {

cout << num << " ";

}

cout << endl;

naturalMergeSort(arr);

cout << "После:" << endl;

for (int num : arr) {

cout << num << " ";

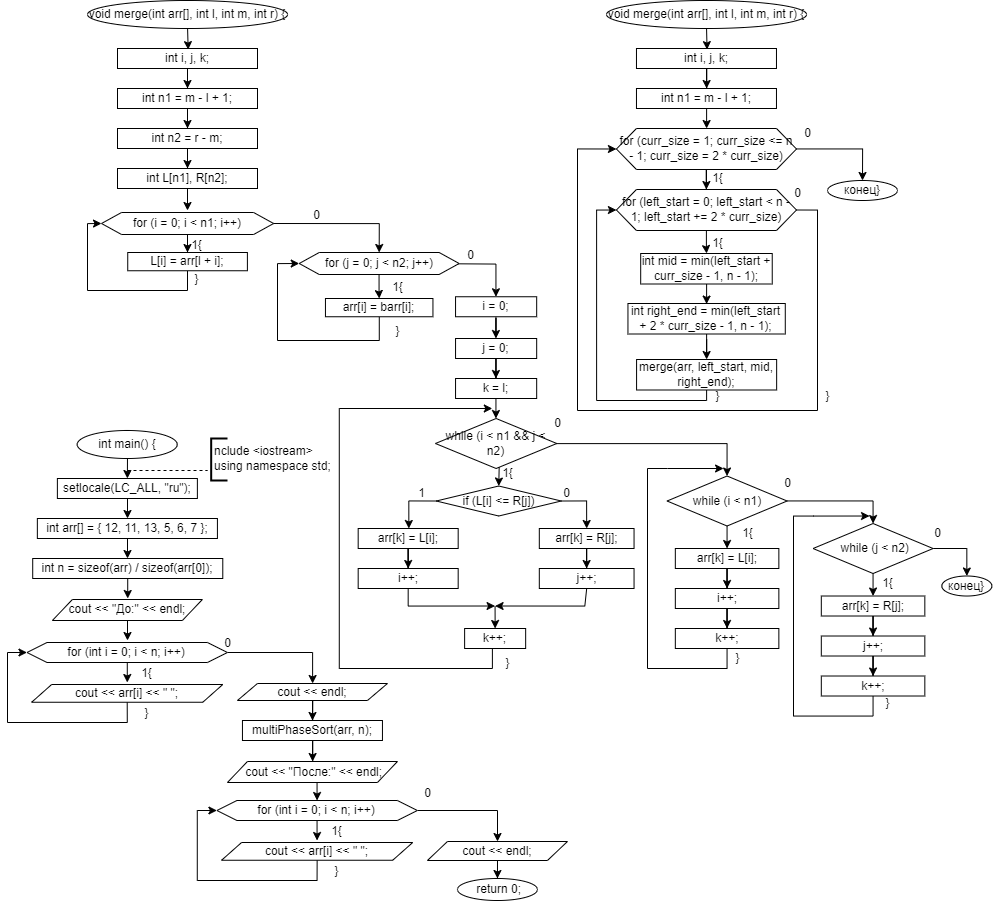
}

cout << endl;

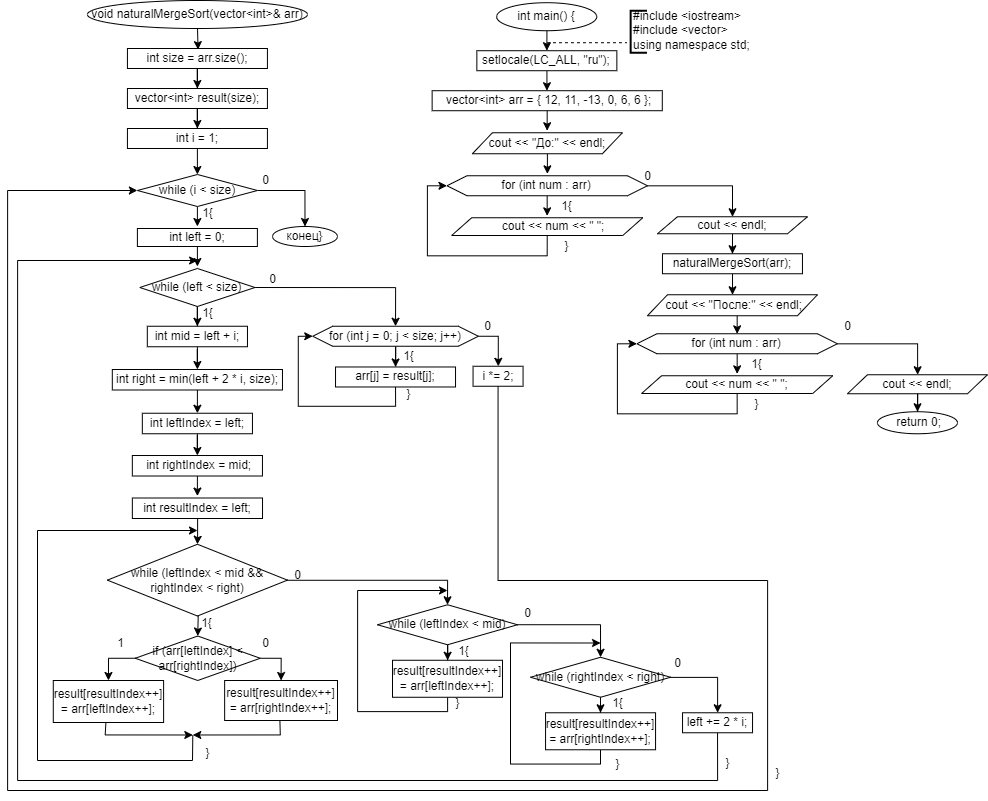
return 0;

}

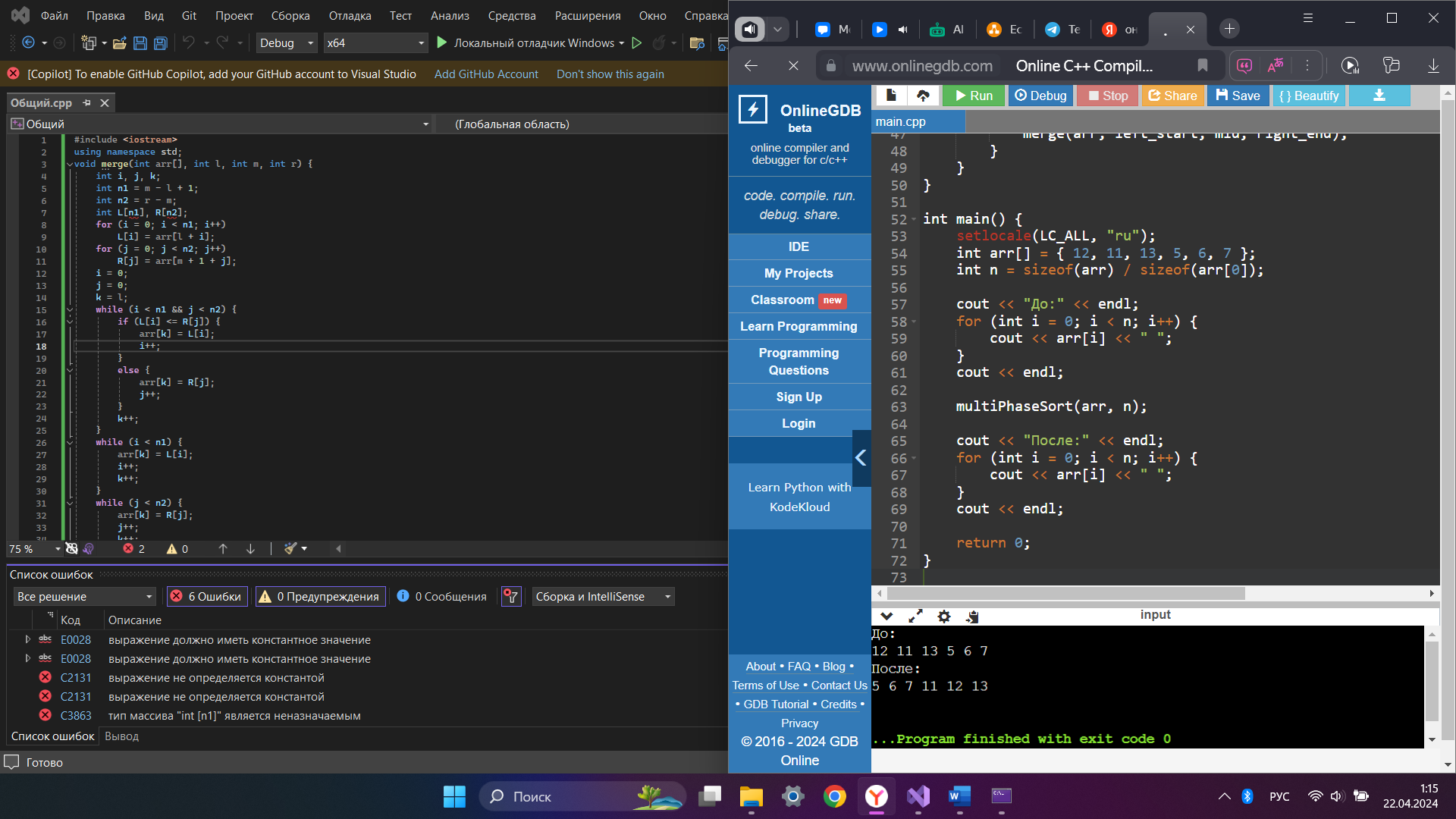
**Блок схема(Многофазная):**

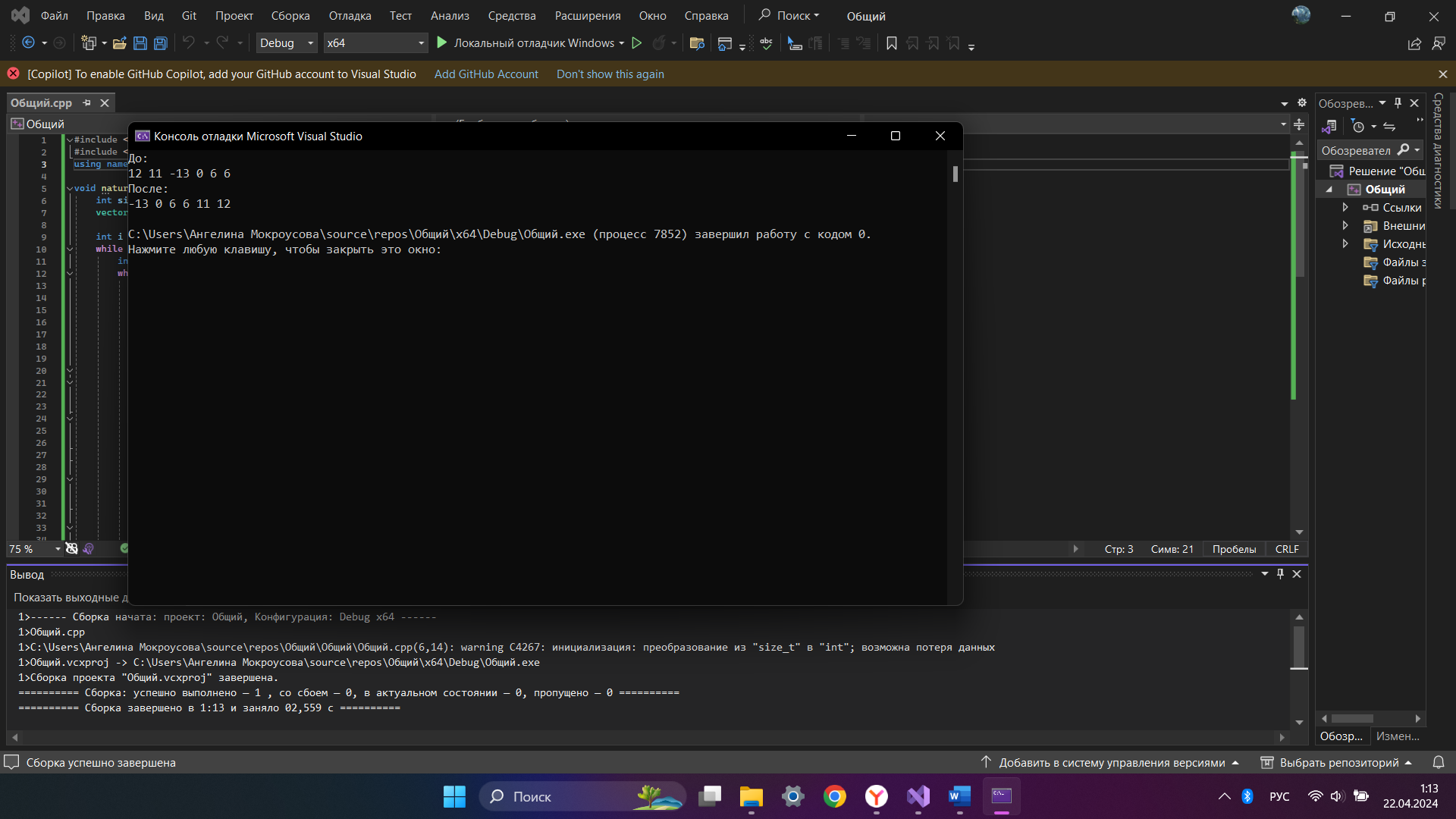


**Блок схема(Естественное слияние):**



**Скриншот работы программы:**





**Вывод:**

Программа успешно выполняет поставленную задачу.

**Скриншот с GitHab:**