Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ

по теме:

СОРТИРОВКИ

Выполнила: студентка группы РИС-22-1б

Черкасова А.А\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Полякова О.А\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_/\_\_\_\_\_\_\_/\_\_\_\_\_г.

Пермь 2023

**Содержание**

[**Введение** 3](#_Toc127379672)

[**Анализ задачи** 3](#_Toc127379673)

[**Блок – схема** 4](#_Toc127379674)

[**Приложение А** 7](#_Toc127379675)

[**Приложение Б** 9](#_Toc127379676)

# **Введение**

**Постановка задачи:** Произвести сортировку массива:

* Блочной сортировкой (+)
* Сортировкой Шелла (+)
* Быстрой сортировкой (сортировкой Хоара) (+)
* Сортировкой подсчётом(+)
* Сортировкой слиянием(+)

**Цель:** Отсортировать массив.

Для достижения поставленной цели, необходимо решить следующие **задачи**:

* Провести анализ задачи
* Реализовать задачу на языке С++
* Составить блок-схему

# **Анализ задачи**

*А) Сортировка слиянием:*

При сортировке слиянием, массив разделяется пополам до тех пор, пока каждый участок не станет длиной в один элемент. Затем эти участки возвращаются на место (сливаются) в правильном порядке. Алгоритм сортировки слиянием представлен на рисунке.

*Б) Быстрая сортировка:*

* 1. Выбрать ключевой индекс и разделить по нему массив на две части. Это можно реализовать разными способами;
  2. Переместить все элементы больше ключевого в правую часть массива, а все элементы меньше ключевого - в левую. Таким образом, ключевой элемент находится в правильной позиции - ключевой элемент больше любого элемента слева и меньше любого элемента справа;
  3. Первые два шага повторяются, пока массив не будет полностью отсортирован.

*В) Сортировка подсчётом:*

Идея алгоритма состоит в предварительном подсчете количества элементов с различными ключами в исходном массиве и разделении результирующего массива на части соответствующей длины (блоки). Затем при повторном проходе исходного массива каждый его элемент копируется в специально отведенный его ключу блок, в первую свободную ячейку.

Таким образом после завершения алгоритма в результирующем массиве содержится исходная последовательность в отсортированном виде, так как блоки расположены по возрастанию соответствующих ключей.

*Г) Сортировка Шелла*

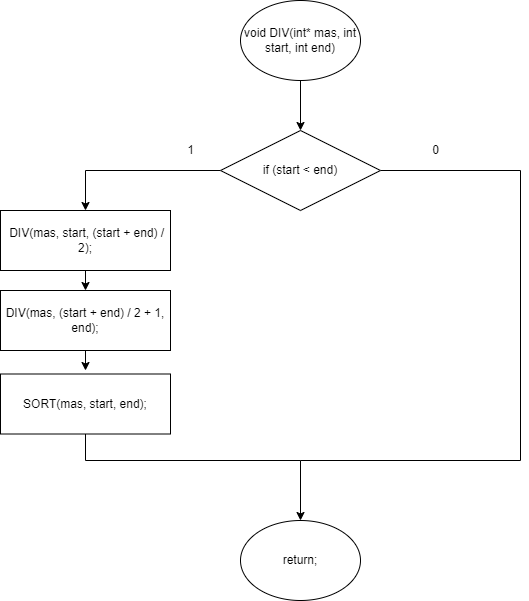
Сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или N/2, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии N/2; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на d/2, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на d=1 проход по массиву происходит в последний раз.

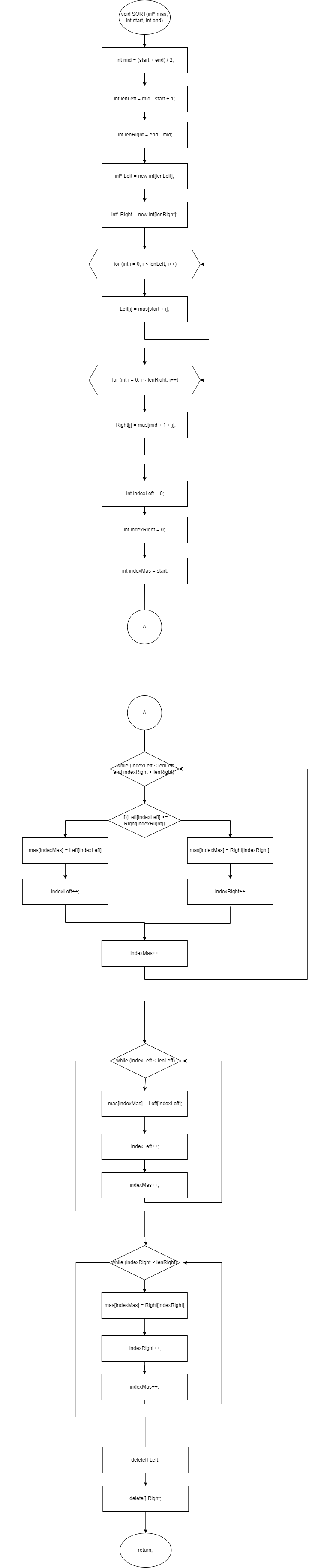
*Д) Блочная сортировка*

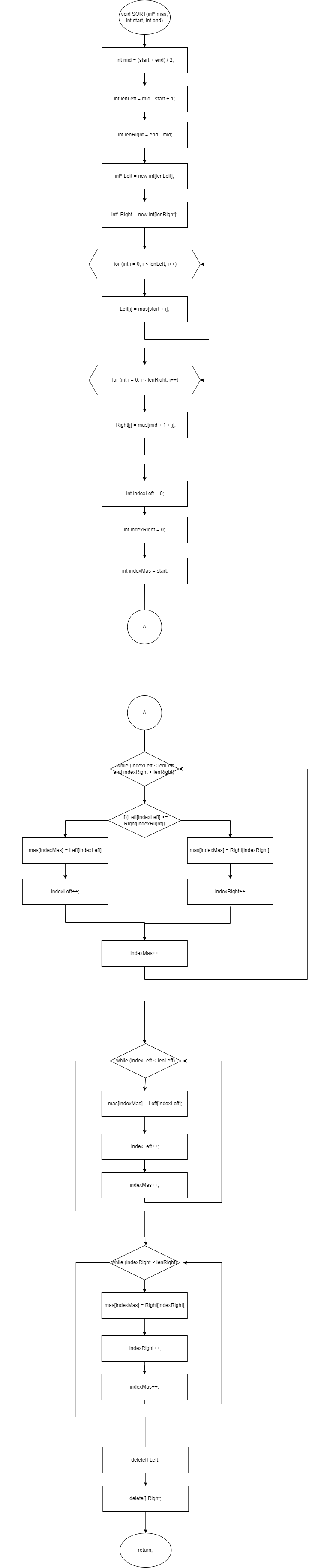
Алгоритм блочной (корзинной) сортировки разделяет элементы массива входных данных на некоторое количество блоков - k, количество блоков зависит от количества исходного множества данных. Далее каждый из таких блоков сортируется либо другой сортировкой, либо рекурсивно тем же методом разбиения. После сортировок внутри каждых блоков данные записываются в исходный массив в порядке разбиения на блоки.

# **Блок – схема**

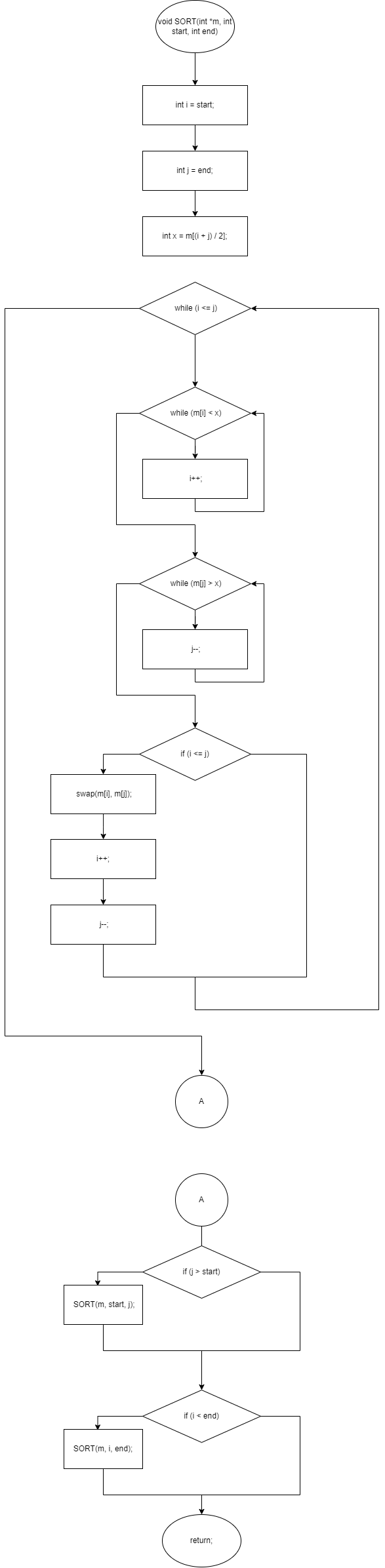
А) Сортировка слиянием

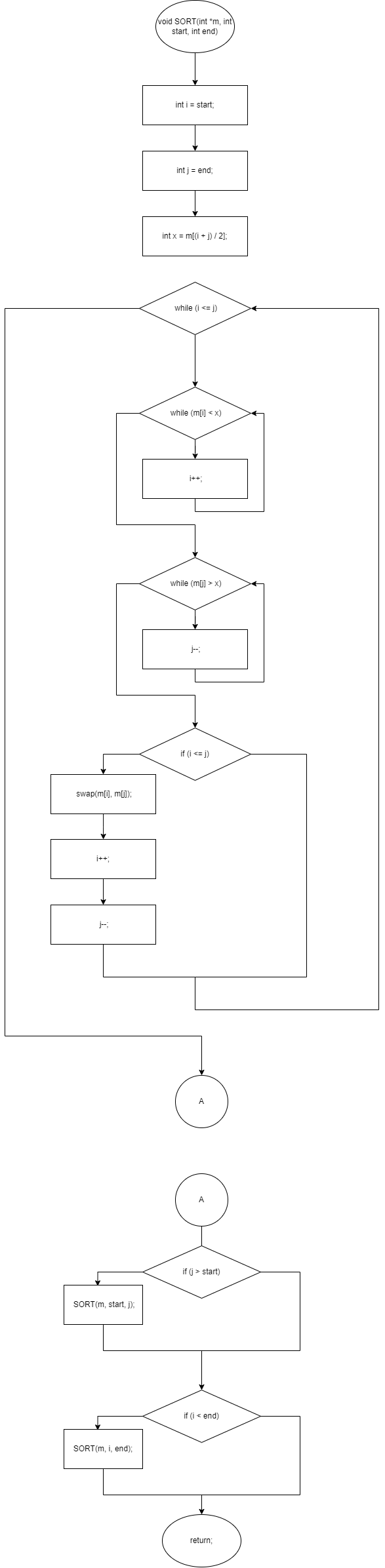
**

**

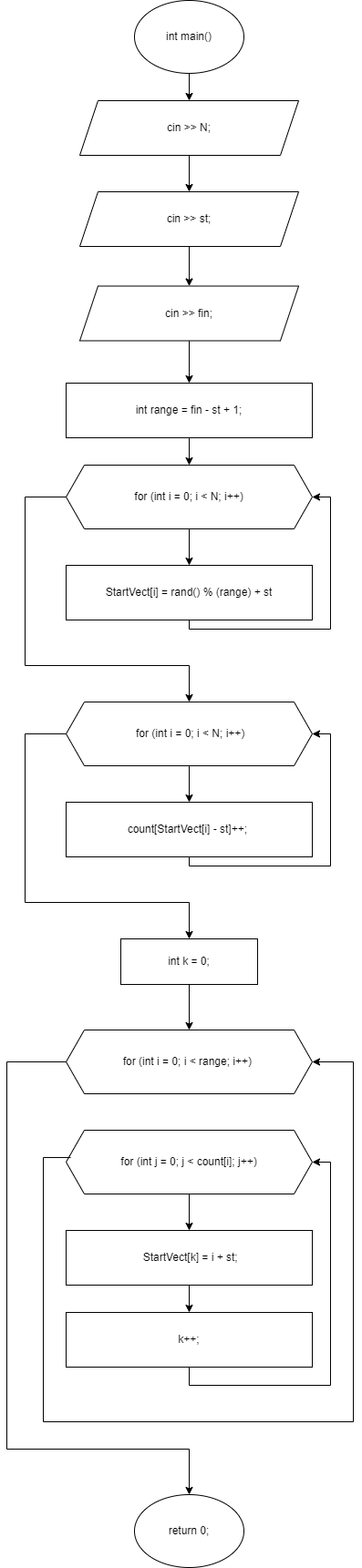
**

Б) Быстрая сортировка*:*

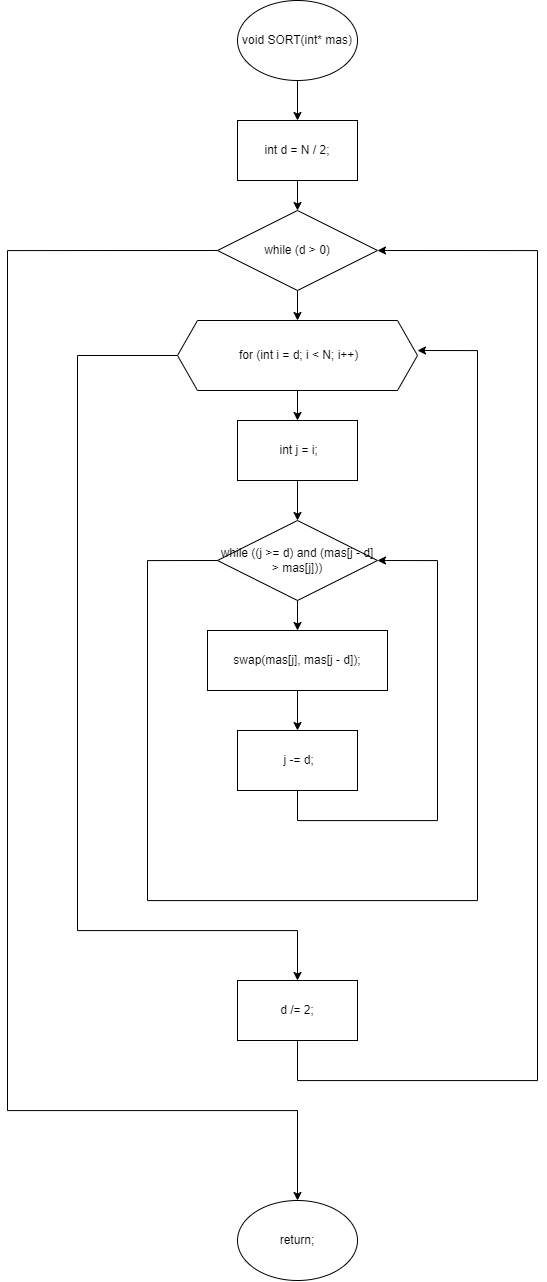




В) Сортировка подсчётом:



Г) Сортировка Шелла



**Приложение А**

# **Листинг программы**

А) Сортировка слиянием

//сортировка слиянием

#include <iostream>

#include <ctime>

using namespace std;

int N;

void SORT(int\*, int, int);

void DIV(int\* mas, int start, int end) //деление массива пополам

{

if (start < end)

{

DIV(mas, start, (start + end) / 2); // левая часть

DIV(mas, (start + end) / 2 + 1, end); // правая

SORT(mas, start, end);

}

}

void SORT(int\* mas, int start, int end)

{

int mid = (start + end) / 2;

int lenLeft = mid - start + 1;

int lenRight = end - mid;

int\* Left = new int[lenLeft];

int\* Right = new int[lenRight];

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

{

Left[i] = mas[start + i];

}

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

{

Right[j] = mas[mid + 1 + j];

}

int indexLeft = 0;

int indexRight = 0;

int indexMas = start;

while (indexLeft < lenLeft and indexRight < lenRight)

{

if (Left[indexLeft] <= Right[indexRight])

{

mas[indexMas] = Left[indexLeft];

indexLeft++;

}

else

{

mas[indexMas] = Right[indexRight];

indexRight++;

}

indexMas++;

}

while (indexLeft < lenLeft)

{

mas[indexMas] = Left[indexLeft];

indexLeft++;

indexMas++;

}

while (indexRight < lenRight)

{

mas[indexMas] = Right[indexRight];

indexRight++;

indexMas++;

}

delete[] Left;

delete[] Right;

return;

}

void PRINT(int\* mas) // вывод массива

{

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

cout << mas[i] << " ";

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "RUS");

cout << "Введите N: ";

cin >> N;

int\* mas = new int[N];

srand(time(NULL));

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

{

mas[i] = rand() % 30 - 15;

}

cout << "начальный массив: ";

PRINT(mas);

DIV(mas, 0, N - 1);

cout << "конечный массив: " ;

PRINT(mas);

delete[] mas;

mas = nullptr;

}

Б) Быстрая сортировка (сортировка Хоара)

//быстрая сортировка

#include <iostream>

#include <ctime>

#include <vector>

using namespace std;

int N;

void SORT(int \*m, int start, int end)

{

int i = start;

int j = end;

int x = m[(i + j) / 2];

while (i <= j)

{

while (m[i] < x)

i++;

while (m[j] > x)

j--;

if (i <= j)

{

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

i++;

j--;

}

}

if (j > start)

SORT(m, start, j);

if (i < end)

SORT(m, i, end);

}

void PRINT(int\* temp) // вывод массива

{

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

cout << temp[i] << " ";

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "RUS");

cout << "Введите N: ";

cin >> N;

int\* mas = new int[N];

srand(time(NULL));

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

{

mas[i] = rand() % 30 - 15;

}

cout << "начальный массив: ";

PRINT(mas);

SORT(mas, 0, N - 1);

cout << "конечный массив: ";

PRINT(mas);

delete[] mas;

mas = nullptr;

}

В) Сортировка подсчётом

//сортировка подсчётом

#include <iostream>

#include <vector>

#include <ctime>

using namespace std;

int N;

void PRINT(vector <int> temp) // вывод массива

{

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

cout << temp[i] << " ";

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "RUS");

cout << "Введите N: ";

cin >> N;

int st, fin;

cout << "Введите диапазон значений" << endl;

cout << "от: ";

cin >> st;

cout << "до: ";

cin >> fin;

vector <int> StartVect(N);

srand(time(NULL));

int range = fin - st + 1; // диапазон + длина массива, хранящего количество повтор.

for (int i = 0; i < N; i++) //заполнение рандомными числами из диапазона

{

StartVect[i] = rand() % (range) + st;

}

cout << "начальный массив: ";

PRINT(StartVect);

vector <int> count(range); // массив, хран. кол-во повторов

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

count[StartVect[i] - st]++; // смещение на левую границу дапазона

cout << "массив счётчика: ";

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

cout << count[i] << " ";

cout << endl;

int k = 0;

for (int i = 0; i < range; i++) // проходимся по каждому эл-ту массива счетчика

{

for (int j = 0; j < count[i]; j++) // теперь цикл выполняется столько раз, сколько повторений было каждого числа

{

StartVect[k] = i + st; // занесение эл-та массива с учетом смещения

k++;

}

}

cout << "конечный массив: ";

PRINT(StartVect);

}

Г) Сортировка Шелла

//сортировка Шелла

#include <iostream>

#include <ctime>

using namespace std;

int N;

void SORT(int\* mas)

{

int d = N / 2; //шаг

while (d > 0)

{

for (int i = d; i < N; i++)

{

int j = i;

while ((j >= d) and (mas[j - d] > mas[j]))

{

swap(mas[j], mas[j - d]);

j -= d;

}

}

d /= 2;

}

}

void PRINT(int\* mas) // вывод массива

{

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

cout << mas[i] << " ";

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "RUS");

cout << "Введите N: ";

cin >> N;

int\* mas = new int[N];

srand(time(NULL));

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

{

mas[i] = rand() % 30 - 15;

}

cout << "начальный массив: ";

PRINT(mas);

SORT(mas);

cout << "конечный массив: ";

PRINT(mas);

delete[] mas;

mas = nullptr;

}

Д) Блочная сортировка

//блочная сортировка

#include <iostream>

#include <ctime>

using namespace std;

int N;

const int offset = 105;

const int maxSize = 100;

void SORT(int\* arr)

{

int j;

int buckets[offset];

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

buckets[i] = 0;

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

buckets[arr[i]]++;

for (int i = 0, j = 0; i < offset; i++)

{

while (buckets[i] > 0)

{

arr[j] = i;

buckets[i]--;

j++;

}

}

}

void PRINT(int\* temp) // вывод массива

{

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

cout << temp[i] << " ";

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "RUS");

cout << "Введите N: ";

cin >> N;

int\* mas = new int[N];

srand(time(NULL));

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

{

mas[i] = rand() % 100;

}

cout << "начальный массив: ";

PRINT(mas);

SORT(mas);

cout << "конечный массив: ";

PRINT(mas);

delete[] mas;

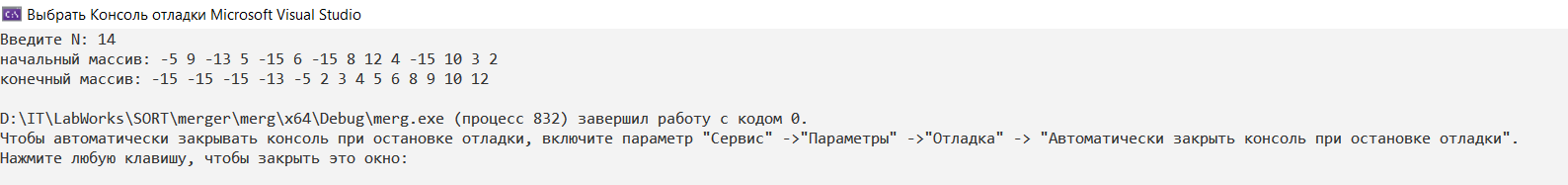
mas = nullptr;

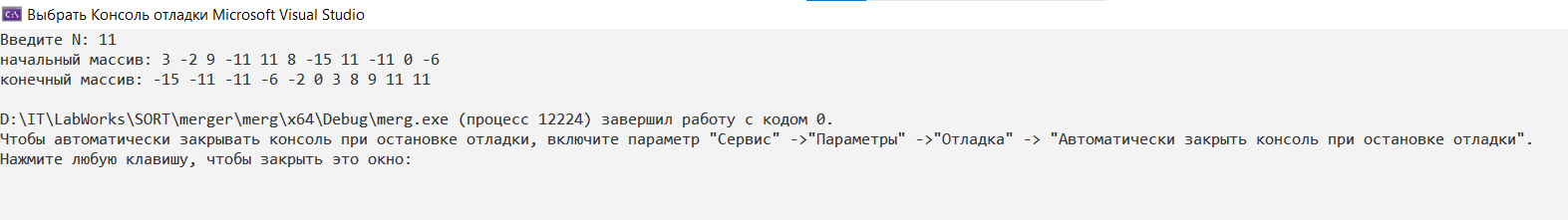
}

# **Приложение Б**

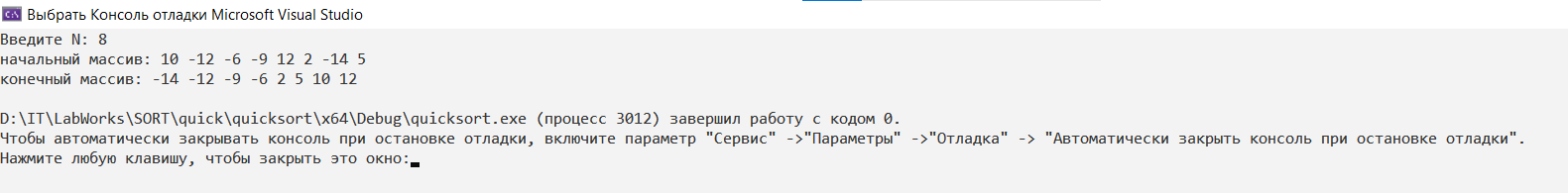
# **Результаты выполнения программы**

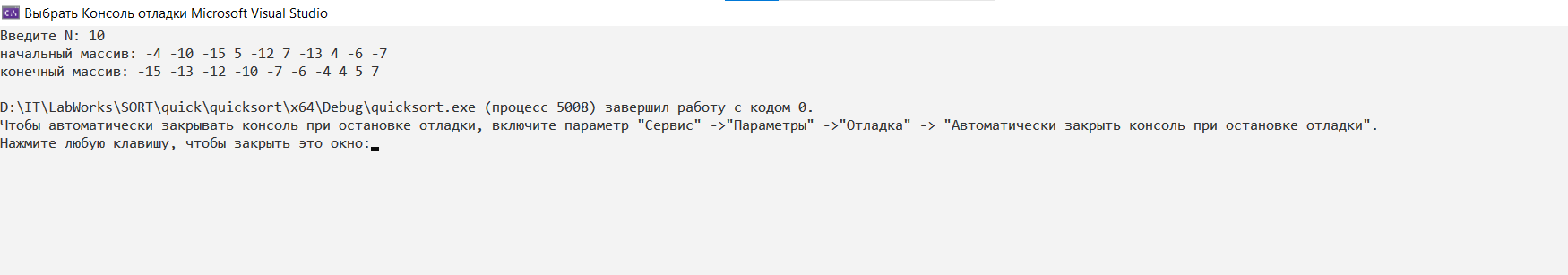
А) Сортировка слиянием



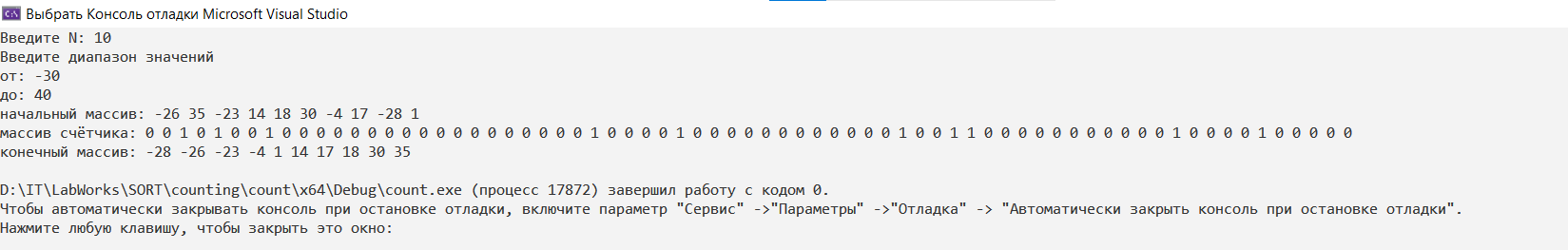


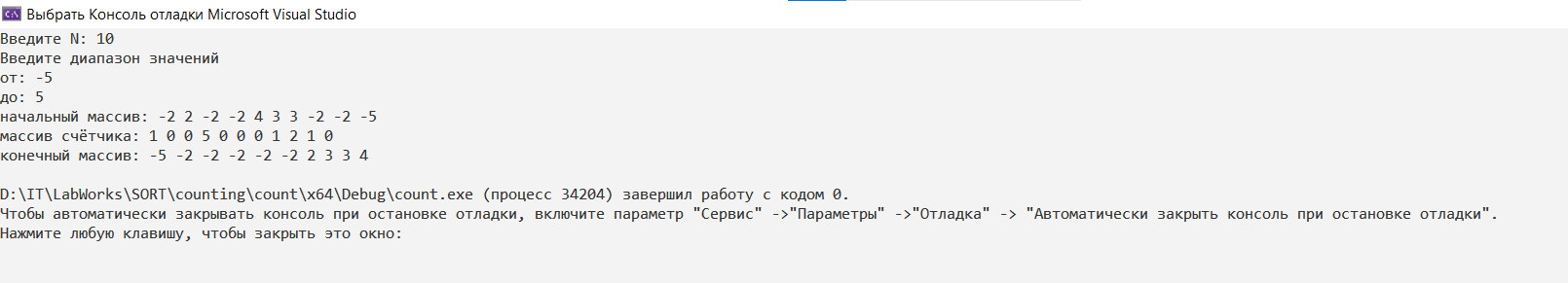
Б) Быстрая сортировка (сортировка Хоара)



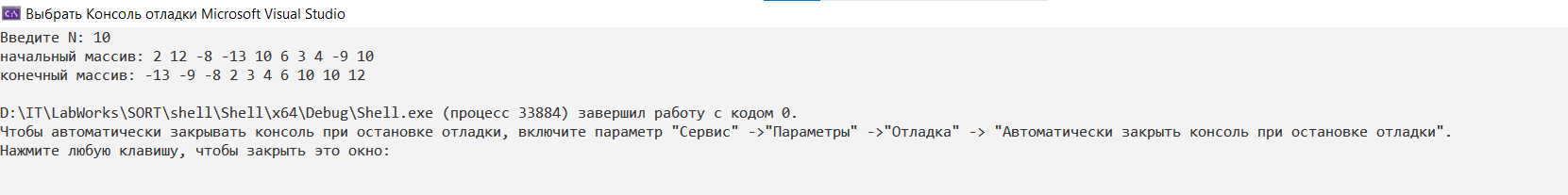


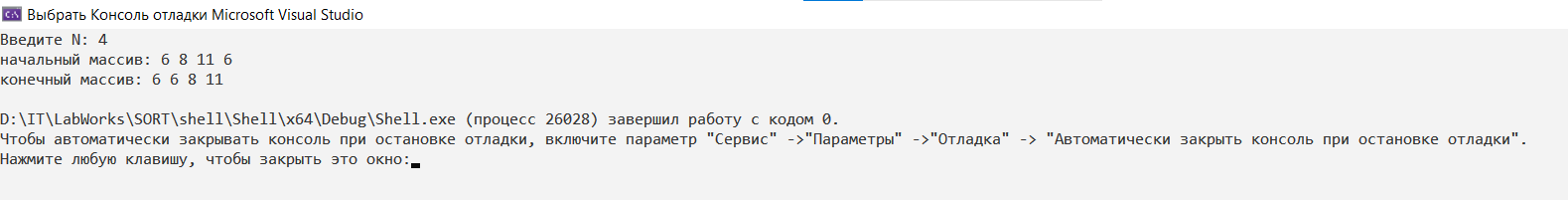
В) Сортировка подсчётом





Г) Сортировка Шелла





Д) Блочная сортировка

