Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Сортировки и поиски

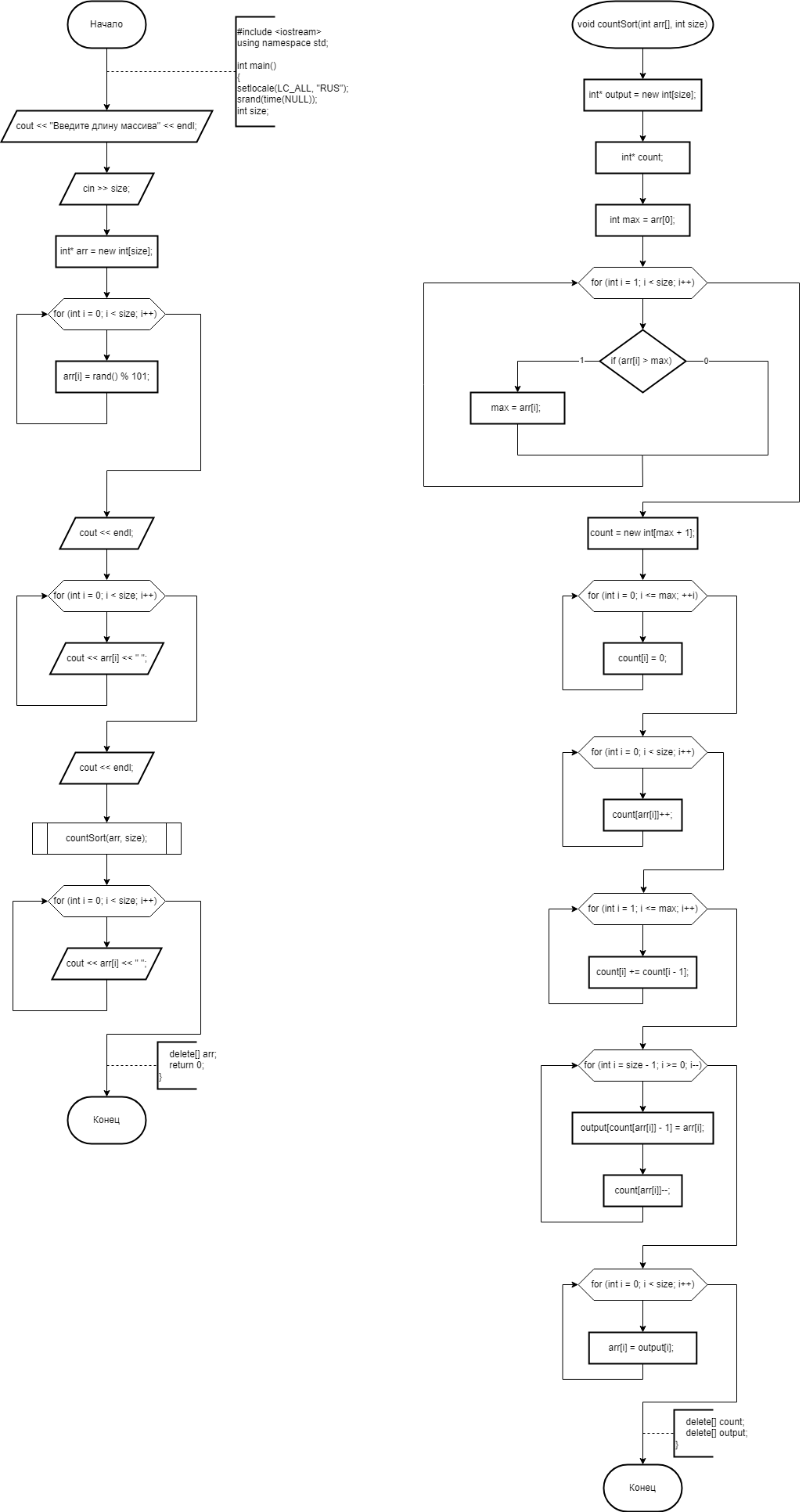
Выполнил работу  
Студент группы РИС-22-1Б  
Ишемцева Мария Андреевна  
Проверил  
Доцент кафедры ИТАС  
Полякова Ольга Андреевна

**Постановка**

Реализация простых сортировок и поисков  
  
Сортировки:

Подсчетом

Блок схема:



Код:

void chet() {

int n;

int a, maxi = 0;

cout << "Введите количество элементов в массиве: ";

cin >> n;

int\* arr = new int[n];

cout << "Введите элементы массива" << endl;

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

cin >> a;

if (a > maxi) maxi = a;

arr[i] = a;

}

int\* sort\_arr = new int[maxi];

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

sort\_arr[i] = 0;

}

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

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

}

int y = 0;

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

if (sort\_arr[i] > 0) {

for (int j = 0; j < sort\_arr[i]; j++) {

arr[y] = i;

y++;

}

}

}

cout << "Массив отсортированный подсчетом: ";

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

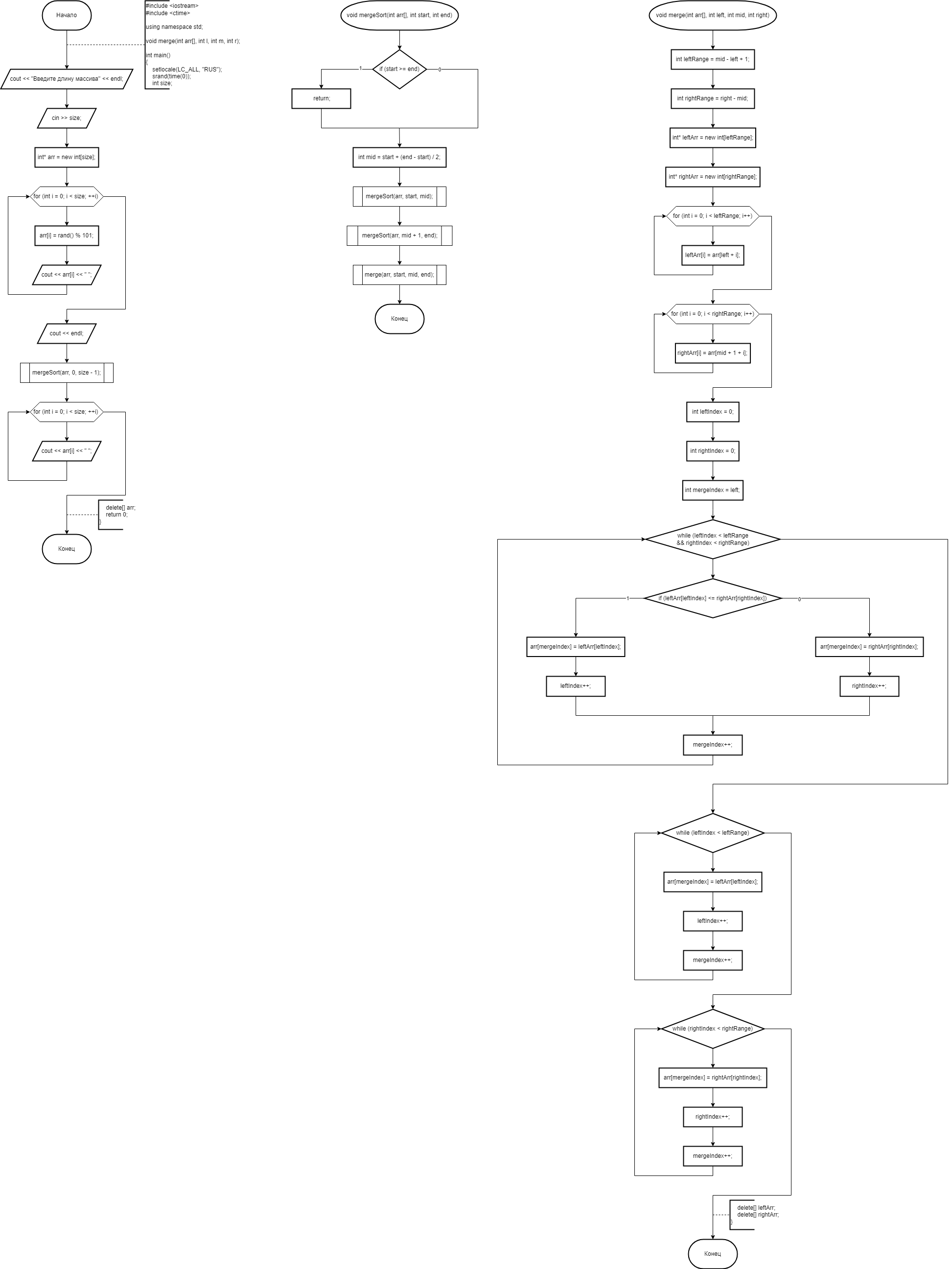
cout << arr[i] << " ";

}

cout << endl;

}

Слиянием:



Код:  
void merge(int arr[], int l, int m, int r);

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

{

if (start >= end)

{

return;

}

int mid = start + (end - start) / 2;

mergeSort(arr, start, mid);

mergeSort(arr, mid + 1, end);

merge(arr, start, mid, end);

}

void merge(int arr[], int left, int mid, int right)

{

int leftRange = mid - left + 1;

int rightRange = right - mid;

int\* leftArr = new int[leftRange];

int\* rightArr = new int[rightRange];

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

{

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

}

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

{

rightArr[i] = arr[mid + 1 + i];

}

int leftIndex = 0;

int rightIndex = 0;

int mergeIndex = left;

while (leftIndex < leftRange && rightIndex < rightRange)

{

if (leftArr[leftIndex] <= rightArr[rightIndex])

{

arr[mergeIndex] = leftArr[leftIndex];

leftIndex++;

}

else

{

arr[mergeIndex] = rightArr[rightIndex];

rightIndex++;

}

mergeIndex++;

}

while (leftIndex < leftRange)

{

arr[mergeIndex] = leftArr[leftIndex];

leftIndex++;

mergeIndex++;

}

while (rightIndex < rightRange)

{

arr[mergeIndex] = rightArr[rightIndex];

rightIndex++;

mergeIndex++;

}

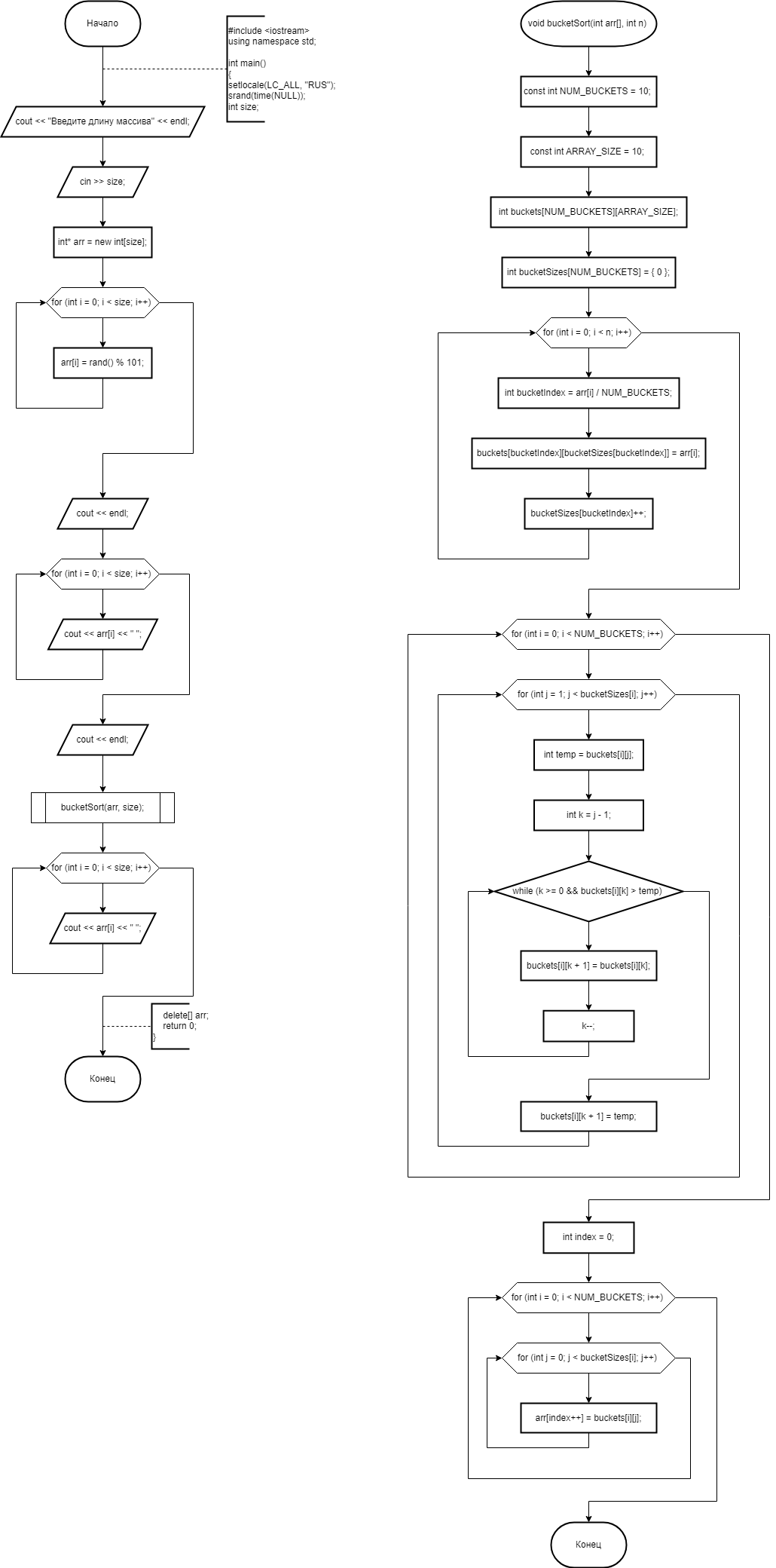
delete[] leftArr;

delete[] rightArr;

}

Блочная:

Блок схема:



Код:

void bucketSort() {

int n;

cout << "Введите размер массива: ";

cin >> n;

int\* arr = new int[n];

cout << "Введите элементы массива:\n";

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

{

cin >> arr[i];

}

const int num = 10;

const int arr\_len = 10;

int buckets[num][arr\_len];

int bucketSizes[num] = { 0 };

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

int index = arr[i] / num;

buckets[index][bucketSizes[index]] = arr[i];

bucketSizes[index]++;

}

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

for (int j = 1; j < bucketSizes[i]; j++) {

int temp = buckets[i][j];

int k = j - 1;

while (k >= 0 && buckets[i][k] > temp) {

buckets[i][k + 1] = buckets[i][k];

k--;

}

buckets[i][k + 1] = temp;

}

}

int index = 0;

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

for (int j = 0; j < bucketSizes[i]; j++) {

arr[index++] = buckets[i][j];

}

}

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

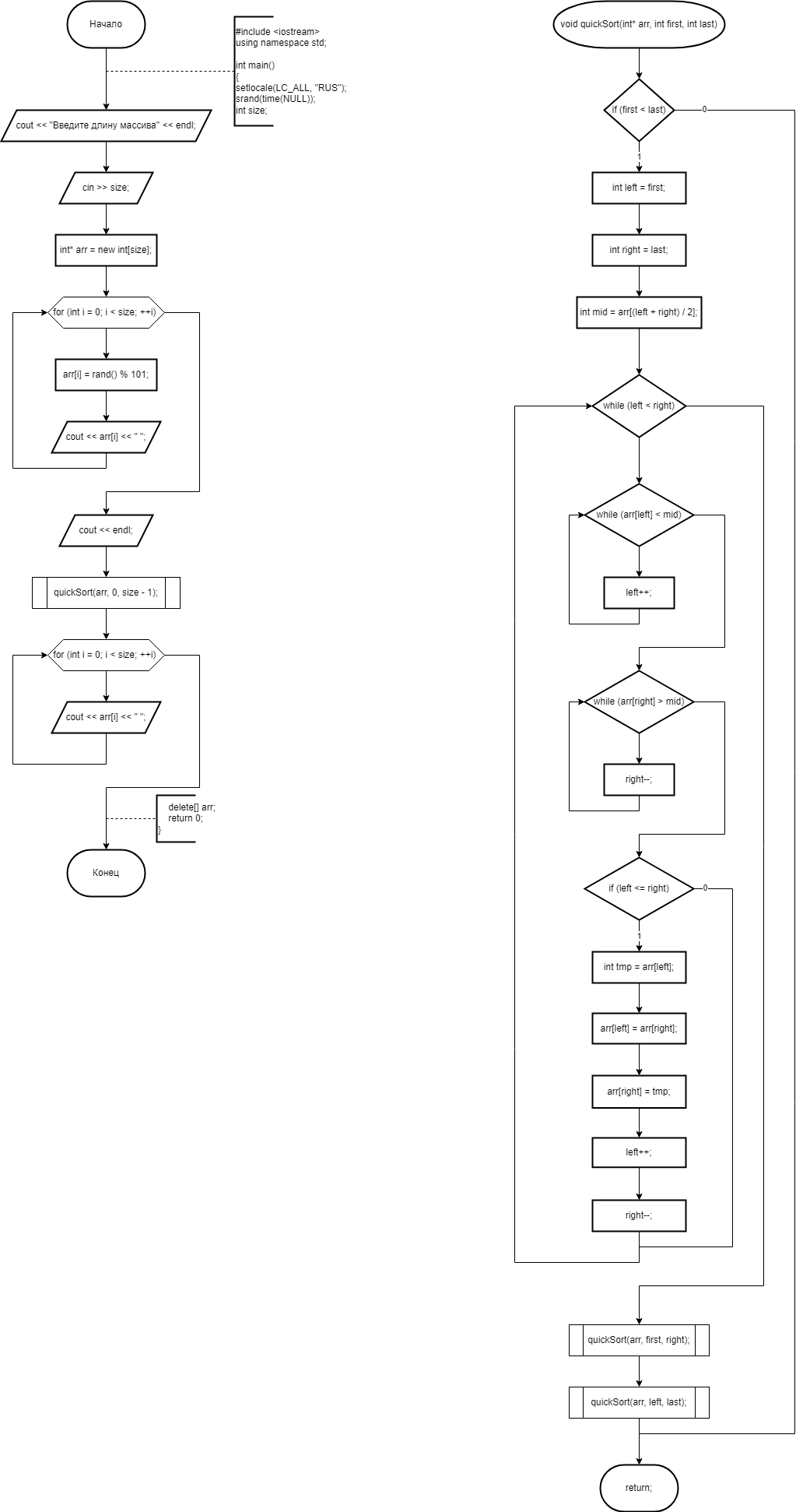
cout << arr[i] << " ";

}

cout << endl;

}

Быстрая сортировка:  
  
Блок схема:



Код:

void hoara() {

int a, n;

cout << "Введите количество элементов в массиве: ";

cin >> n;

int\* arr = new int[n];

cout << "Введите элементы массива" << endl;

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

cin >> a;

arr[i] = a;

}

int i = 0, j = n-1, sr = 0, tmp;

bool f = false;

while (sr < n) {

if (arr[i] >= arr[sr]) {

j = n - 1;

while (arr[j] >= arr[sr] && j > sr) {

j--;

}

if (i != j) {

tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

}

i++;

}

if (arr[i] < arr[sr]) {

i++;

}

for (int y = sr; y < n; y++) {

if (arr[y] < arr[sr]){

f = true;

}

}

if (!f || i > n) {

i = 0;

sr++;

}

}

cout << "Массив отсортированный методом Хоара(быстрой сортировкой)\n";

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

cout << arr[iu] << " ";

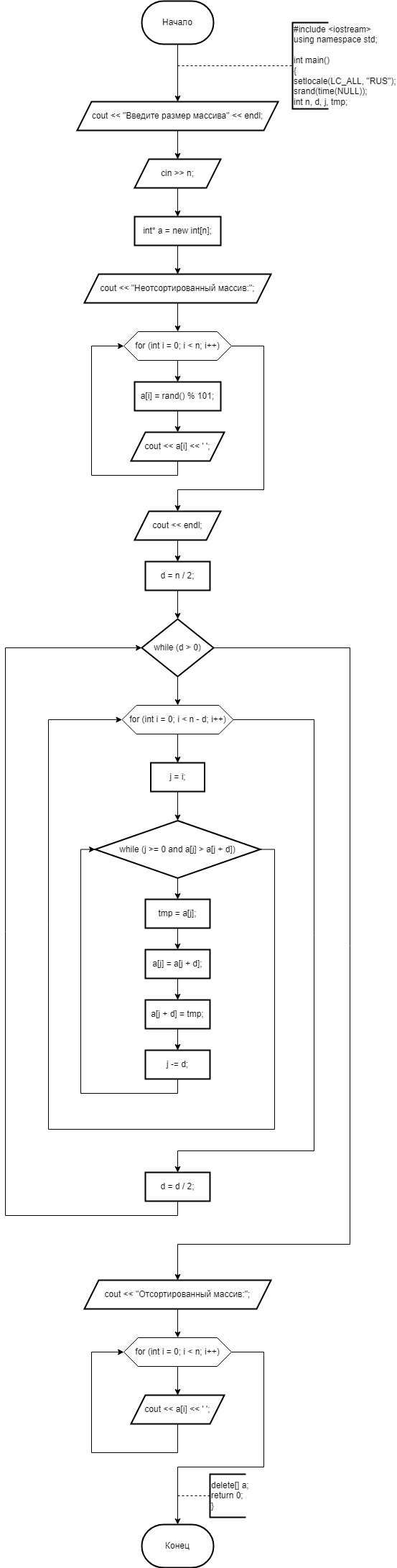
}

cout << endl;

}

Шелл:

Блок схема:



Код:

void shell() {

int n, d, j, tmp;

cout << "Введите размер массива: ";

cin >> n;

int\* a = new int[n];

cout << "Введите элементы массива:\n";

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

cin >> a[i];

}

d = n / 2;

while (d > 0) {

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

j = i;

while (j >= 0 and a[j] > a[j + d]) {

tmp = a[j];

a[j] = a[j + d];

a[j + d] = tmp;

j -= d;

}

}

d = d / 2;

}

cout << "Отсортированный массив: ";

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

cout << a[i] << ' ';

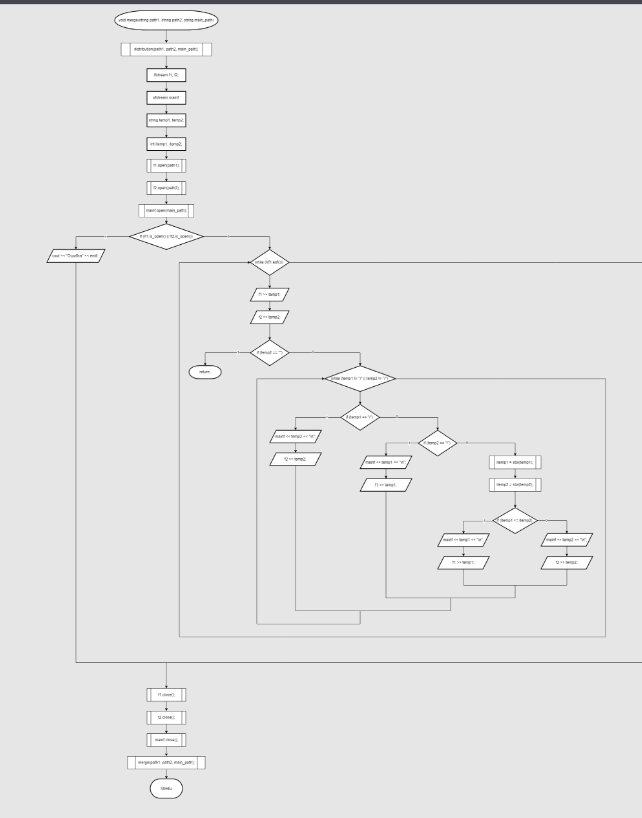
}

delete[] a;

}

Естественная:

Блок схема:

Код:

void distribution(string path1, string path2, string main\_path)

{

ofstream f1, f2;

ifstream mainf;

string temp, cur;

int itemp, icur;

f1.open(path1);

f2.open(path2);

mainf.open(main\_path);

if (!f1.is\_open() || !f2.is\_open())

{

cout << "Не удалось открыть файл" << endl;

}

else

{

bool firstfile = true;

mainf >> cur;

while (!mainf.eof())

{

if (firstfile)

{

f1 << cur << "\n";

temp = cur;

mainf >> cur;

if (mainf.eof())

{

f1 << "/" << "\n";

return;

}

itemp = stoi(temp);

icur = stoi(cur);

while (itemp <= icur)

{

if (mainf.eof())

{

f1 << "/" << "\n";

return;

}

f1 << cur << "\n";

temp = cur;

mainf >> cur;

itemp = stoi(temp);

icur = stoi(cur);

}

f1 << "/" << "\n";

firstfile = false;

}

else

{

f2 << cur << "\n";

temp = cur;

mainf >> cur;

if (mainf.eof())

{

f2 << "/" << "\n";

return;

}

itemp = stoi(temp);

icur = stoi(cur);

while (itemp <= icur)

{

if (mainf.eof())

{

f2 << "/" << "\n";

return;

}

f2 << cur << "\n";

temp = cur;

mainf >> cur;

itemp = stoi(temp);

icur = stoi(cur);

}

f2 << "/" << "\n";

firstfile = true;

}

}

}

f1.close();

f2.close();

mainf.close();

}

void merge(string path1, string path2, string main\_path)

{

distribution(path1, path2, main\_path);

ifstream f1, f2;

ofstream mainf;

string temp1, temp2;

int itemp1, itemp2;

f1.open(path1);

f2.open(path2);

mainf.open(main\_path);

if (!f1.is\_open() || !f2.is\_open())

{

cout << "Не удалось открыть файл" << endl;

}

else

{

while (!(f1.eof()))

{

f1 >> temp1;

f2 >> temp2;

if (temp2 == "")

{

return;

}

while (temp1 != "/" || temp2 != "/")

{

if (temp1 == "/")

{

mainf << temp2 << "\n";

f2 >> temp2;

}

else if (temp2 == "/")

{

mainf << temp1 << "\n";

f1 >> temp1;

}

else

{

itemp1 = stoi(temp1);

itemp2 = stoi(temp2);

if (itemp1 <= itemp2)

{

mainf << temp1 << "\n";

f1 >> temp1;

}

else

{

mainf << temp2 << "\n";

f2 >> temp2;

}

}

}

}

}

f1.close();

f2.close();

mainf.close();

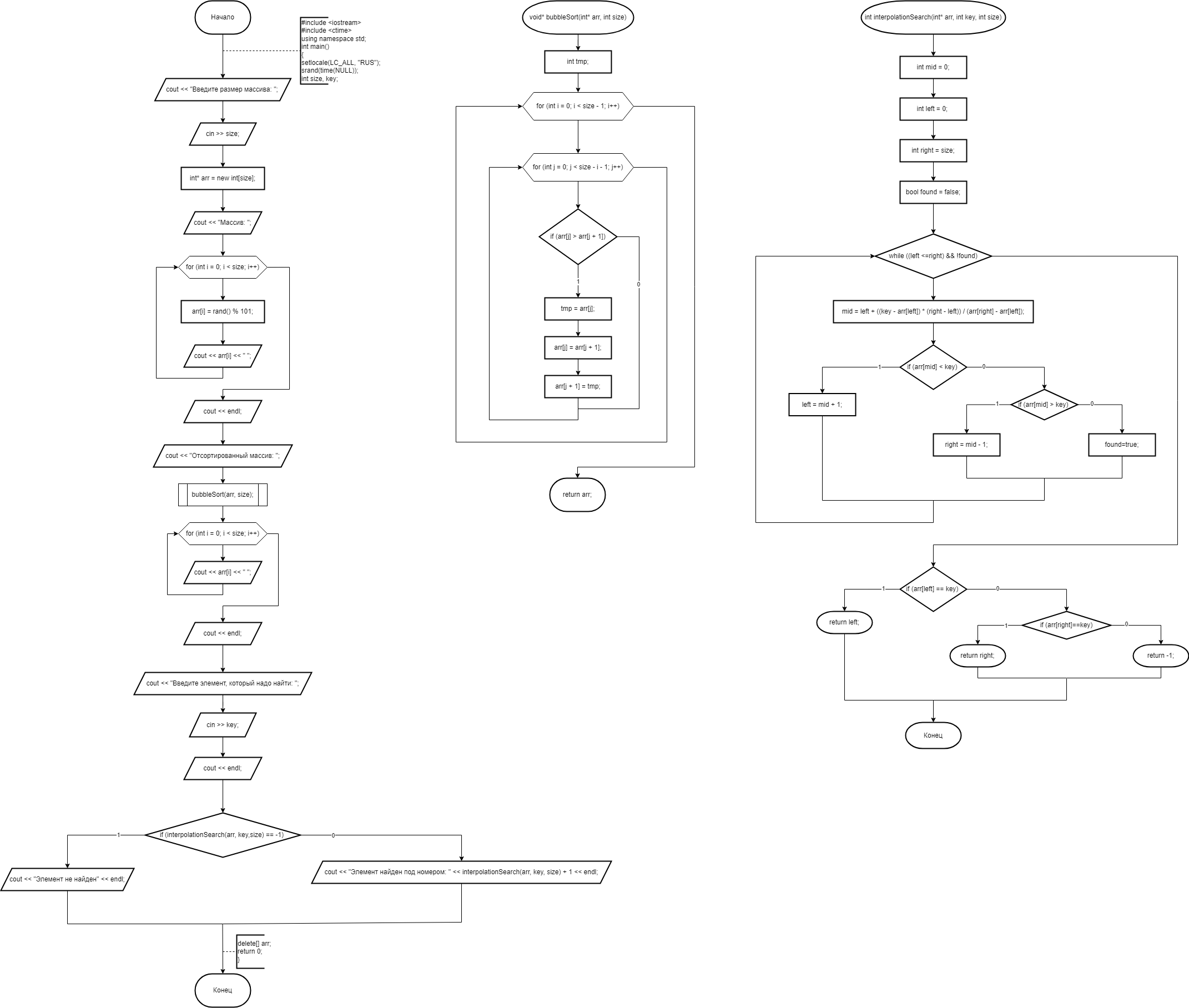
merge(path1, path2, main\_path);

}

Поиски:

Интерполяционный:

Блок схема:



Код:

void interpol() {

int n;

int a, maxi = 0;

cout << "Введите количество элементов в массиве: ";

cin >> n;

int\* arr = new int[n];

cout << "Введите элементы массива" << endl;

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

cin >> a;

if (a > maxi) maxi = a;

arr[i] = a;

}

int\* sort\_arr = new int[maxi];

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

sort\_arr[i] = 0;

}

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

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

}

int y = 0;

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

if (sort\_arr[i] > 0) {

for (int j = 0; j < sort\_arr[i]; j++) {

arr[y] = i;

y++;

}

}

}

cout << "Введите элемент, который нужно найти: ";

int find = 0;

cin >> find;

bool flag = false;

int left = 0, right = n-1;

int fin = left + ((find - arr[left]) \* (right - left) / (arr[right] - arr[left]));

cout << "Отсортированный массив: ";

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

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

}

cout << endl;

while (!flag && left < right - 1)

{

if (arr[fin] != find)

{

if (arr[fin] > find)

{

right = fin;

fin = left + ((find - arr[left]) \* (right - left) / (arr[right] - arr[left]));

}

else

{

left = fin+1;

fin = left + ((find - arr[left]) \* (right - left) / (arr[right] - arr[left]));

}

}

else

{

flag = true;

}

}

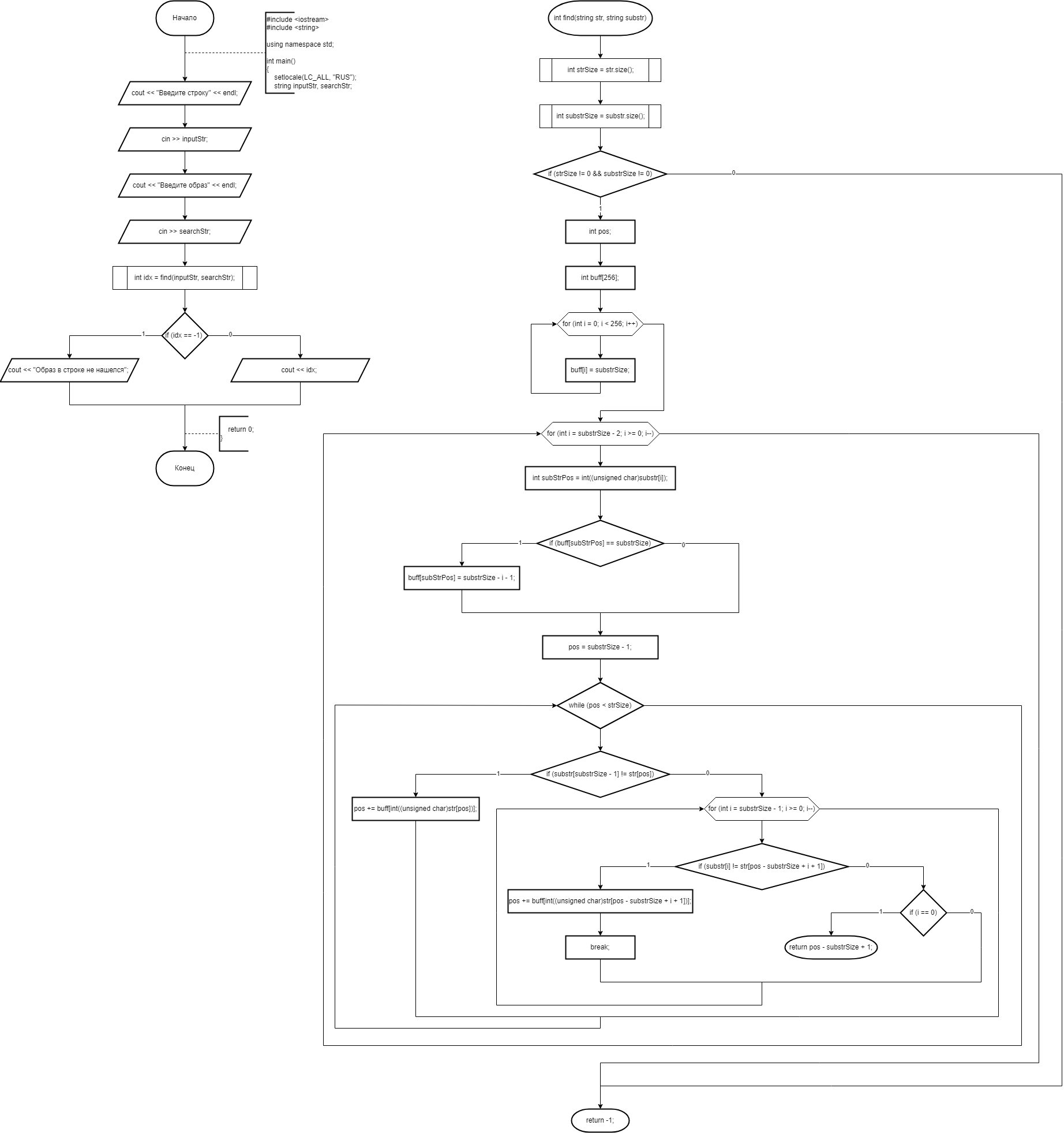
if (flag) cout << "Индекс искомого элемента: " << fin << endl;

else cout << "Элемент не найден!" << endl;

}

Поиск Бойера Мура:

Блок схема:



Код:

void bol\_mur() {

map <char, int> mp;

string str, substr;

cout << "Введите строку: ";

cin.get();

getline(cin, str);

cout << "Введите подстроку: ";

getline(cin, substr);

int len = substr.length(), len\_str = str.length();

int k = len-1;

int i = len\_str-1;

while (i > -1 && k > -1) {

if (str[i] == substr[k]) {

mp[substr[k]] = len\_str - i - 1;

k--;

i--;

}

else i--;

}

mp[substr[len - 1]] = len;

bool flag = false;

int y1 = len-1, y2 = y1, smech = 0;

while (y1 <= y2 && y1 < len && y2 < len\_str && y1 > -1 && y2 > -1) {

if (substr[y1] != str[y2]) {

for (int z = 0; z < len; z++) {

if (substr[z] == str[y2]) {

flag = true;

}

}

if (flag) {

smech += mp[str[y2]];

y2 = smech + len - 1;

}

else {

smech += len;

y2 = smech + len - 1;

}

y1 = len-1;

}

else {

y1--;

y2--;

}

}

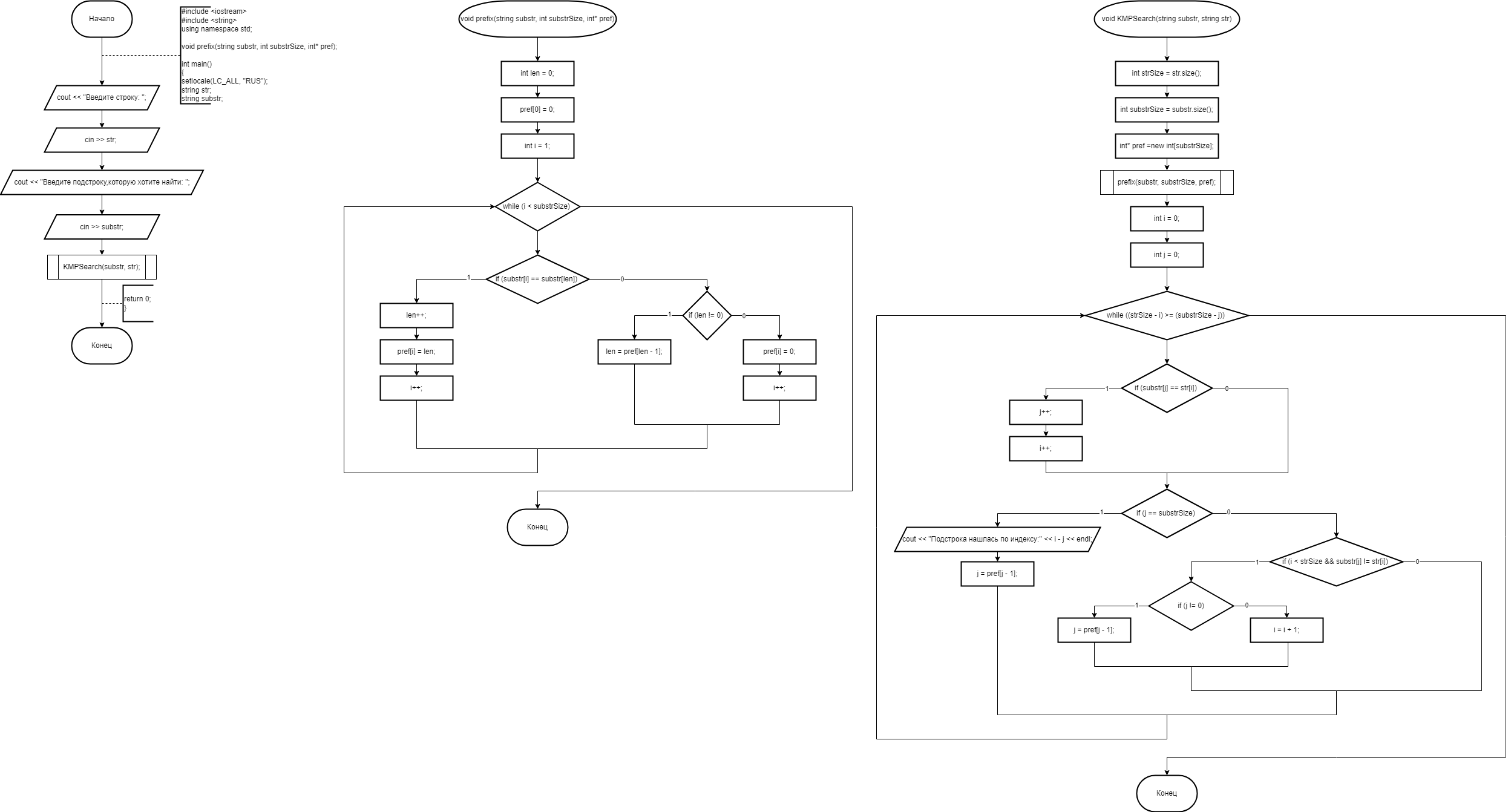
if (y1 == -1) cout << "Подстрока смещена относительно начала на " << smech+1 << " символов.\n";

else cout << "Подстрока не найдена!";

}

Поиск Кнута Мориса Пратта:

Блок схема:



Код:

void prefix(string substr, int len\_sub, int\* pref)

{

int len = 0;

pref[0] = 0;

int i = 1;

while (i < len\_sub)

{

if (substr[i] == substr[len])

{

len++;

pref[i] = len;

i++;

}

else

{

if (len != 0)

{

len = pref[len - 1];

}

else

{

pref[i] = 0;

i++;

}

}

}

}

void knut() {

string str;

string substr;

bool f = false;

cout << "Введите строку: ";

cin.get();

getline(cin, str);

cout << "Введите подстроку: ";

getline(cin, substr);

int len = str.length();

int len\_sub = substr.length();

int\* pref = new int[len\_sub];

prefix(substr, len\_sub, pref);

int i = 0;

int j = 0;

while ((len - i) >= (len\_sub - j))

{

if (substr[j] == str[i])

{

j++;

i++;

}

if (j == len\_sub)

{

cout << "Индекс первого символа подстроки в строке:" << i - j << endl;

j = pref[j - 1];

f = true;

}

else if (i < len && substr[j] != str[i])

{

if (j != 0)

{

j = pref[j - 1];

}

else

{

i = i + 1;

}

}

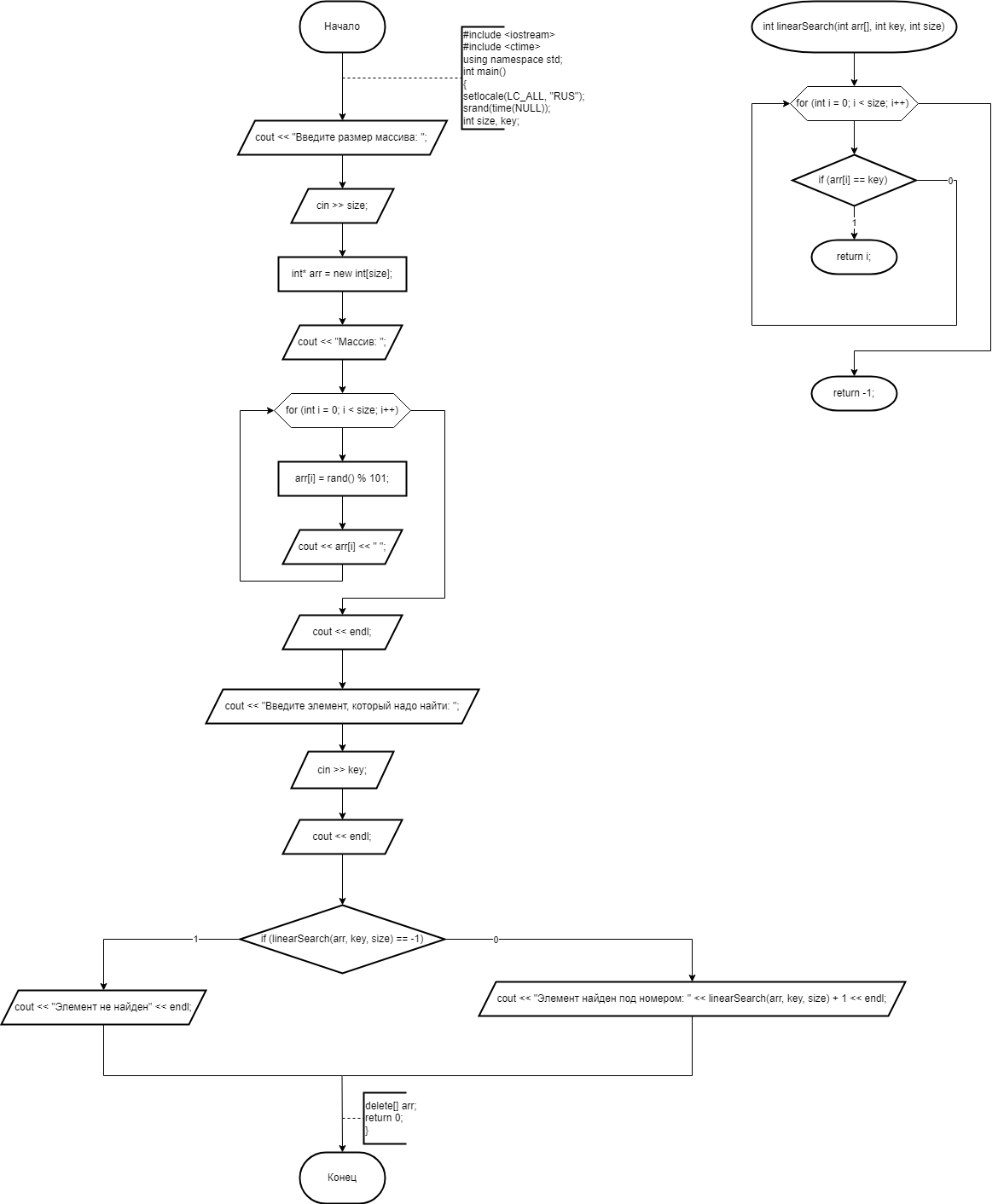
}

if (!f) cout << "Подстрока не найдена\n";

}

Прямой поиск:

Блок схема:



Код:

void straight\_sea() {

string str1, str2;

cout << "Введите строку: ";

cin.get();

getline(cin, str1);

cout << "Введите подстроку, которую надо найти: ";

getline(cin, str2);

int n = str1.length(), m = str2.length(), index = 0;

bool f = false;

int t = 0;

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

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

if (str1[i + j] == str2[j]) {

t++;

}

}

if (t == m and !f) {

index = i;

f = true;

}

t = 0;

}

if (index != 0) cout << "Подстрока смещена относительно начала на " << ++index << " символов." << endl;

else cout << "Подстрока не найдена!\n";

}