Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ПНИПУ)

Электротехнический факультет

Дисциплина: основы алгоритмизации и программирования, 2 семестр

ОТЧЁТ ПО ЛАБОЛАТОРНОЙ РАБОТЕ

Тема: «Методы сортировки в С++: слиянием, подсчетом, блочная»

Выполнил

Студент РИС-24-1б

Конькова С. С.

Проверил

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2025

**Постановка задачи**

Отсортировать массив с помощью трёх методов сортировки.

**Анализ подсчёта**

1. Начинаем с большого шага (например, половина длины массива) и постепенно уменьшаем его до 1.
2. Для каждого шага выполняем сортировку вставками, но сравниваем элементы, находящиеся на расстоянии шага друг от друга.
3. После каждого прохода шаг уменьшается (например, делится на 2).
4. Когда шаг становится равным 1, выполняется обычная сортировка вставками, которая завершает процесс.

**Анализ слияния**

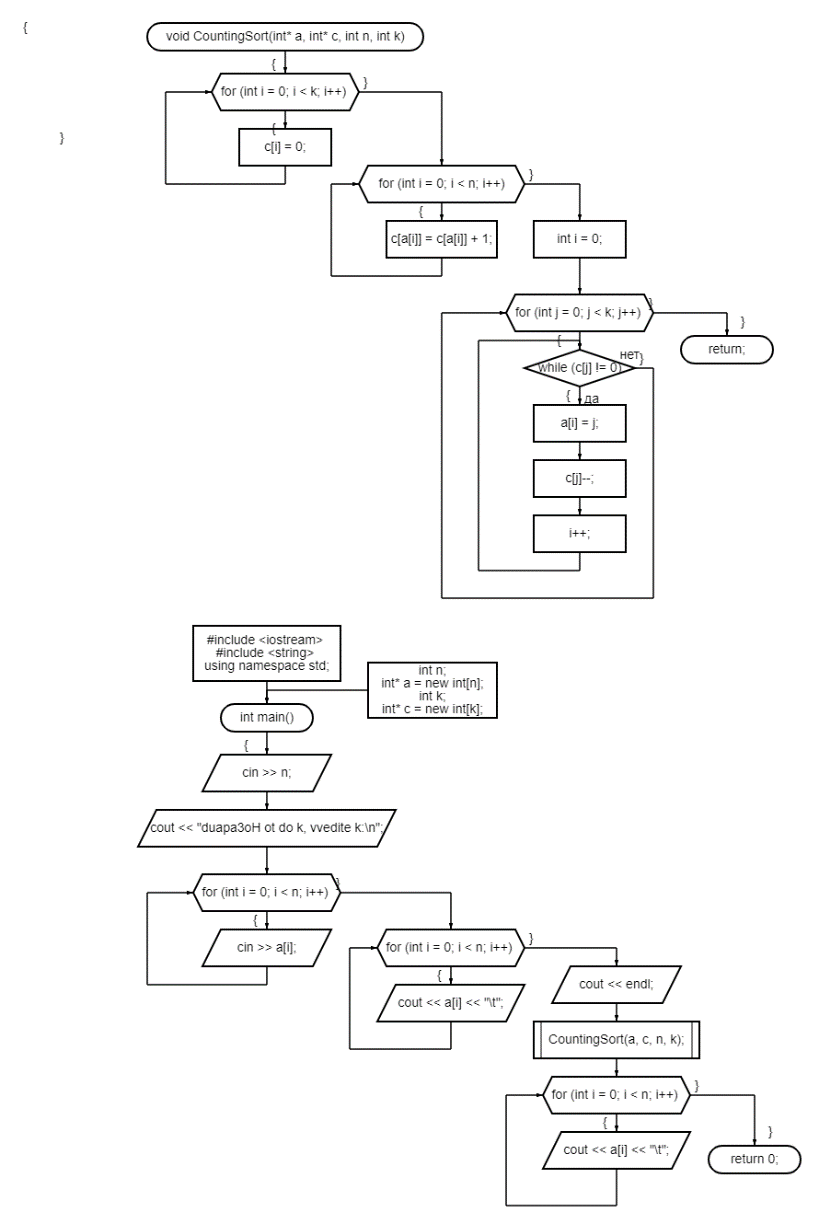
1. Начинаем с большого шага (например, половина длины массива) и постепенно уменьшаем его до 1.
2. Для каждого шага выполняем сортировку вставками, но сравниваем элементы, находящиеся на расстоянии шага друг от друга.
3. После каждого прохода шаг уменьшается (например, делится на 2).
4. Когда шаг становится равным 1, выполняется обычная сортировка вставками, которая завершает процесс.

**Анализ блочной**

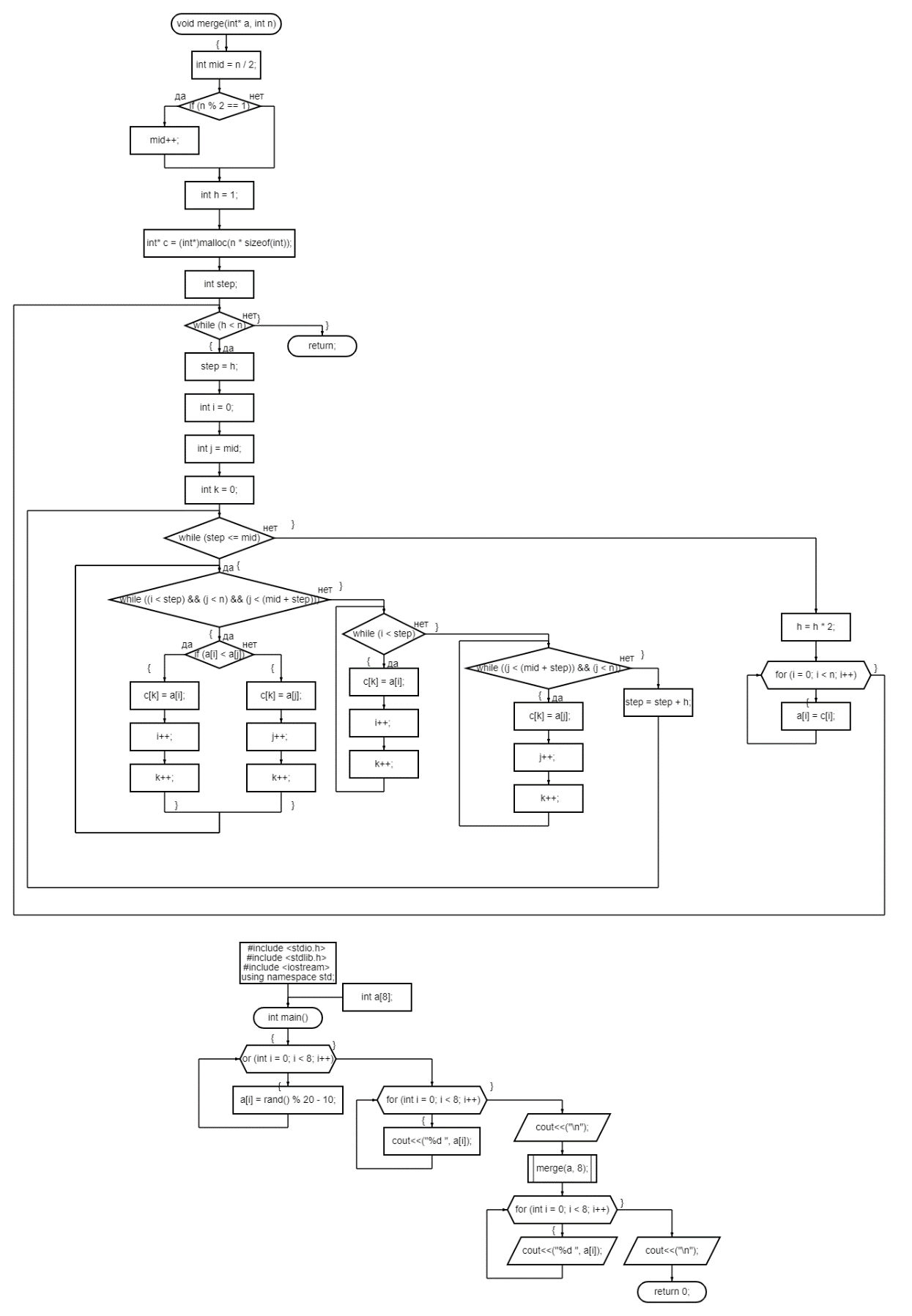
1. Начинаем с большого шага (например, половина длины массива) и постепенно уменьшаем его до 1.
2. Для каждого шага выполняем сортировку вставками, но сравниваем элементы, находящиеся на расстоянии шага друг от друга.
3. После каждого прохода шаг уменьшается (например, делится на 2).
4. Когда шаг становится равным 1, выполняется обычная сортировка вставками, которая завершает процесс.

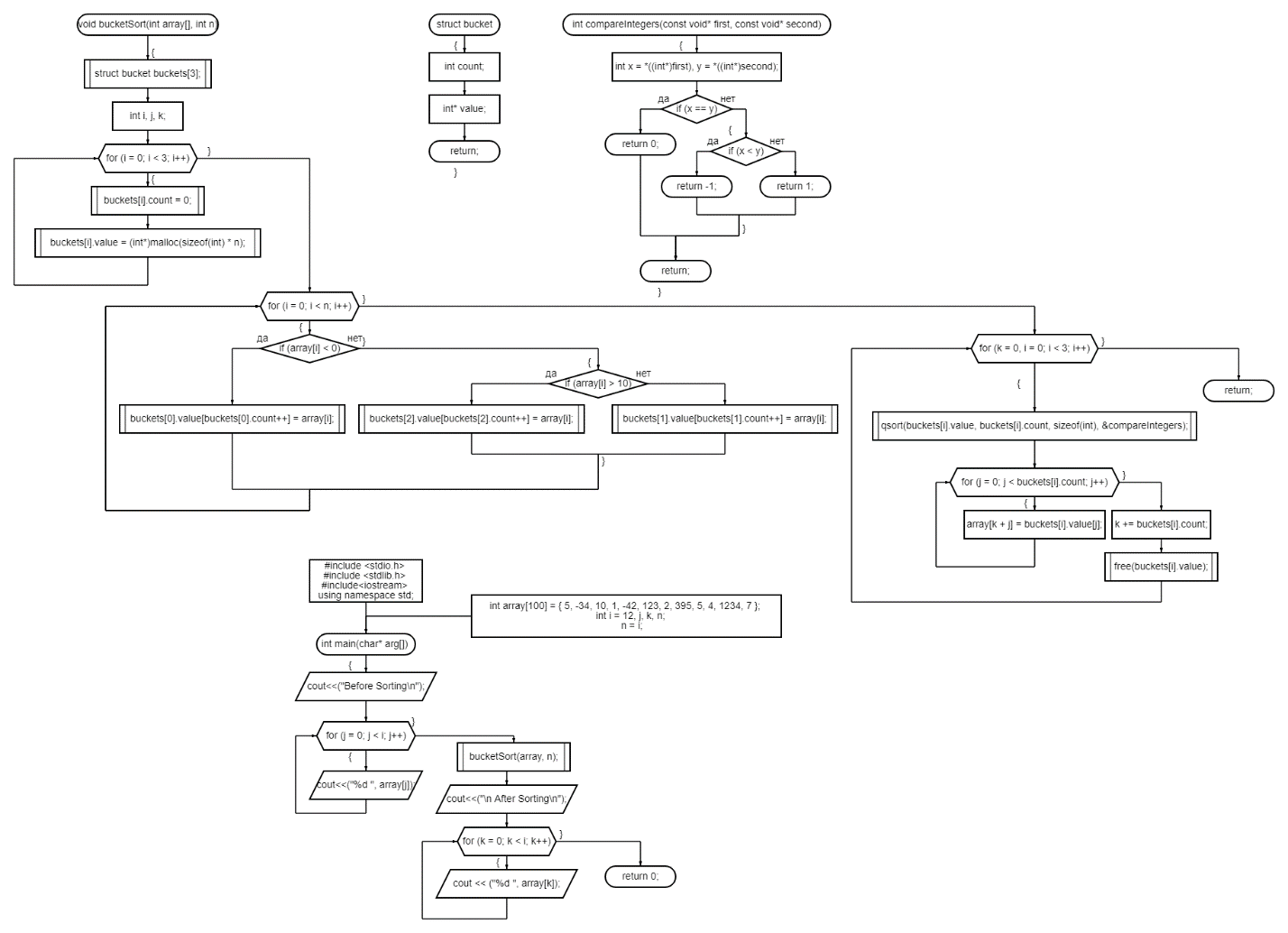
**Блок – схема**

1. Подсчета



1. Слияния



1. Блочная

**Коды программ**

1. Подсчета

#include <iostream>

#include <string>

using namespace std;

void CountingSort(int\* a, int\* c, int n, int k)

{

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

{

c[i] = 0;

}

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

{

c[a[i]] = c[a[i]] + 1;

}

int i = 0;

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

{

while (c[j] != 0)

{

a[i] = j;

c[j]--;

i++;

}

}

}

int main()

{

int n;

cin >> n;

int\* a = new int[n];

int k;

cout << "duapa3oH ot do k, vvedite k:\n";

cin >> k;

int\* c = new int[k];

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

{

cin >> a[i];

}

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

{

cout << a[i] << "\t";

}

cout << endl;

CountingSort(a, c, n, k);

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

{

cout << a[i] << "\t";

}

cout << endl;

system("pause");

return 0;

}

1. Слияния

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

using namespace std;

void merge(int\* a, int n)

{

int mid = n / 2;

if (n % 2 == 1)

mid++;

int h = 1;

int\* c = (int\*)malloc(n \* sizeof(int));

int step;

while (h < n)

{

step = h;

int i = 0;

int j = mid;

int k = 0;

while (step <= mid)

{

while ((i < step) && (j < n) && (j < (mid + step)))

{

if (a[i] < a[j])

{

c[k] = a[i];

i++;

k++;

}

else

{

c[k] = a[j];

j++;

k++;

}

}

while (i < step)

{

c[k] = a[i];

i++;

k++;

}

while ((j < (mid + step)) && (j < n))

{

c[k] = a[j];

j++;

k++;

}

step = step + h;

}

h = h \* 2;

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

{

a[i] = c[i];

}

}

}

int main()

{

int a[8];

for (int i = 0; i < 8; i++)

{

a[i] = rand() % 20 - 10;

}

for (int i = 0; i < 8; i++)

{

cout<<("%d ", a[i]);

}

cout<<("\n");

merge(a, 8);

for (int i = 0; i < 8; i++)

{

cout<<("%d ", a[i]);

}

cout<<("\n");

getchar();

return 0;

}

1. Блочная

#include <stdio.h>

#include <stdlib.h>

#include<iostream>

using namespace std;

struct bucket

{

int count;

int\* value;

};

int compareIntegers(const void\* first, const void\* second)

{

int x = \*((int\*)first), y = \*((int\*)second);

if (x == y)

{

return 0;

}

else if (x < y)

{

return -1;

}

else

{

return 1;

}

}

void bucketSort(int array[], int n)

{

struct bucket buckets[3];

int i, j, k;

for (i = 0; i < 3; i++)

{

buckets[i].count = 0;

buckets[i].value = (int\*)malloc(sizeof(int) \* n);

}

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

{

if (array[i] < 0)

{

buckets[0].value[buckets[0].count++] = array[i];

}

else if (array[i] > 10)

{

buckets[2].value[buckets[2].count++] = array[i];

}

else

{

buckets[1].value[buckets[1].count++] = array[i];

}

}

for (k = 0, i = 0; i < 3; i++)

{

qsort(buckets[i].value, buckets[i].count, sizeof(int), &compareIntegers);

for (j = 0; j < buckets[i].count; j++)

{

array[k + j] = buckets[i].value[j];

}

k += buckets[i].count;

free(buckets[i].value);

}

}

int main(char\* arg[])

{

int array[100] = { 5, -34, 10, 1, -42, 123, 2, 395, 5, 4, 1234, 7 };

int i = 12, j, k, n;

n = i;

cout<<("Before Sorting\n");

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

{

cout<<("%d ", array[j]);

}

bucketSort(array, n);

cout<<("\n After Sorting\n");

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

{

cout << ("%d ", array[k]);

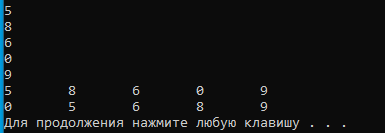
}

return 0;

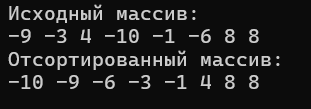
}

**Результаты программы**

1. Подсчета



1. Слияния



1. Блочная

