Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

ИРИТ-РТФ

Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ N 3**

по дисциплине «Современные языки программирования»

Студент группы РИВ-210938у: Вахрушев Н.А.

Преподаватель: Архипов Н.А.

Екатеринбург 2023

Цель

Необходимо освоить основные методы упорядочения числовых данных, а также познакомиться с реализацией различных алгоритмов сортировки средствами языка C++.

Ход работы

Ниже представлены блок схемы разработанных алгоритмов:

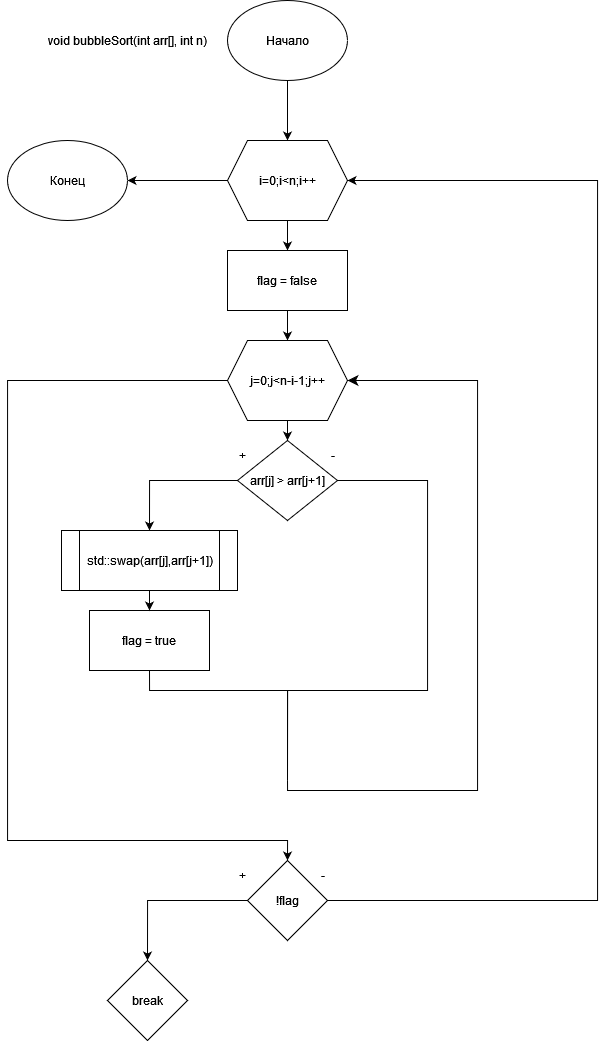


Рисунок 1 – Сортировка пузырьком

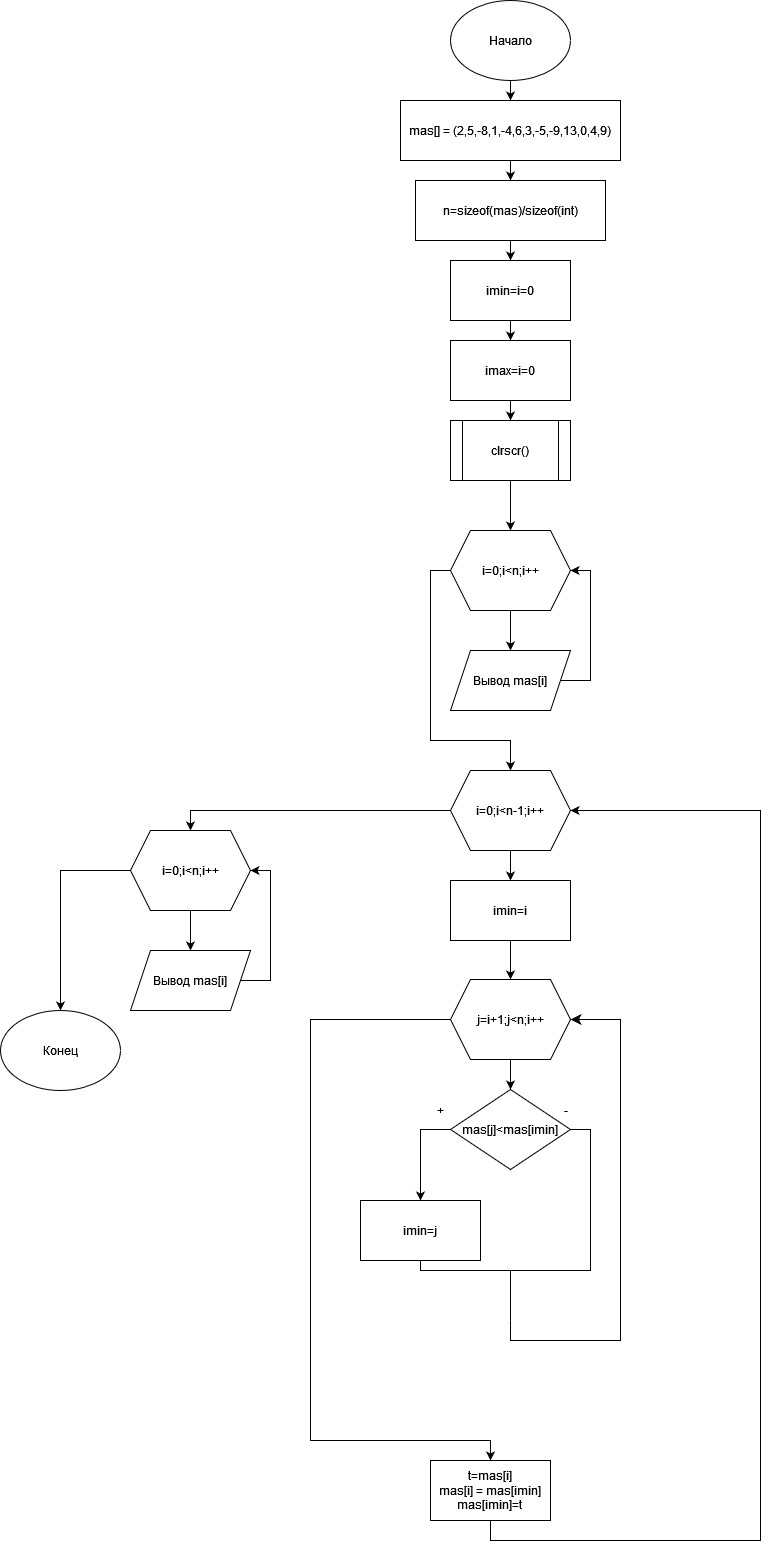


Рисунок 2 – Сортировка min-max

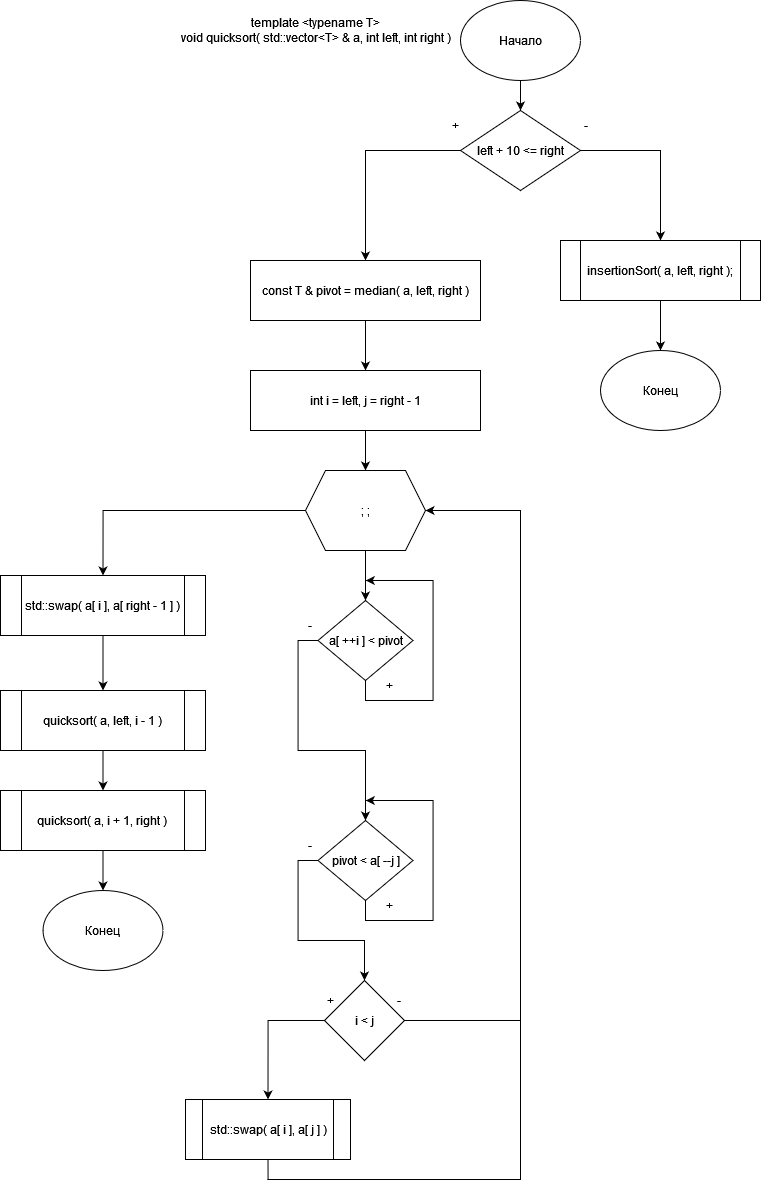


Рисунок 3 – Основная функция сортировки qsort

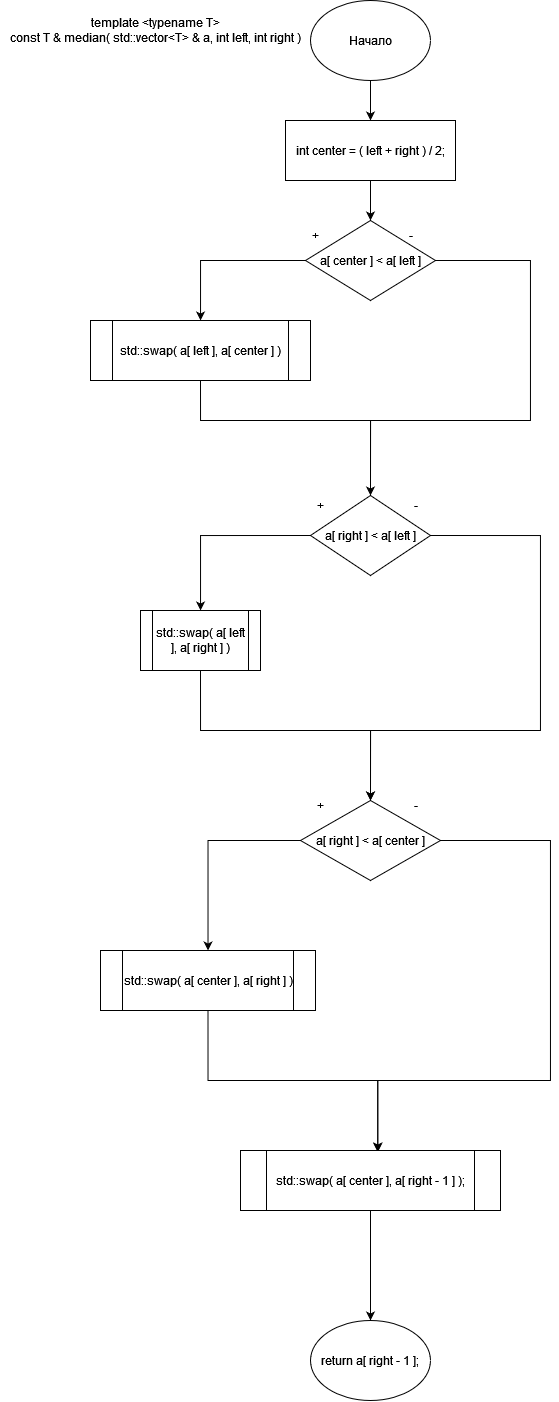


Рисунок 4 – Функция median

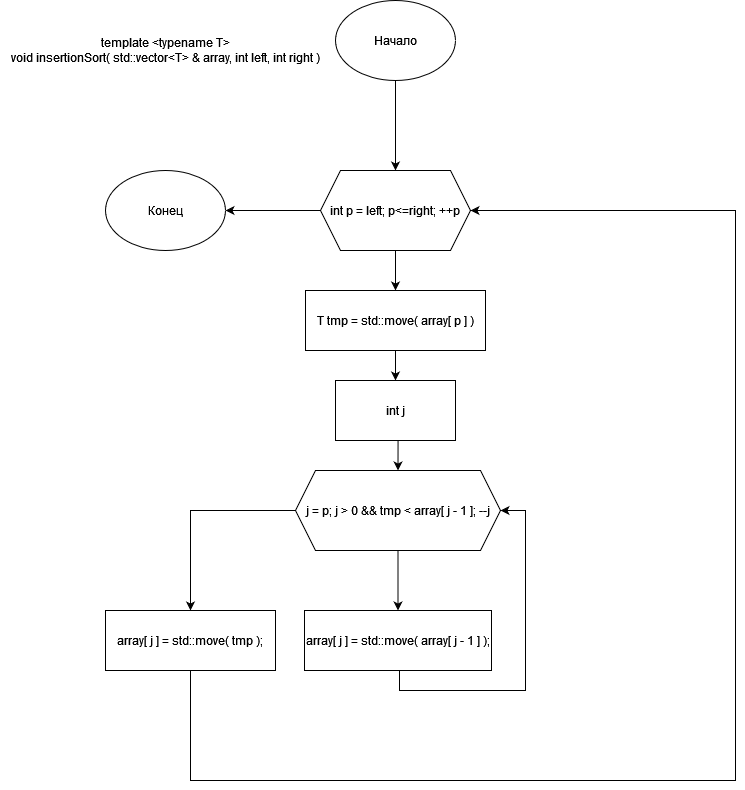


Рисунок 5 – Функция insertionSort

Ниже представлен листинг пузырьковой сортировки:

#include <iostream>

void bubbleSort(int arr[], int n)

{

int i, j;

bool flag;

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

{

flag = false;

for(j = 0; j < n-i-1; j++)

{

if( arr[j] > arr[j+1])

{

std::swap(arr[j],arr[j+1]);

flag = true;

}

}

if(!flag)

{

break;

}

}

}

Ниже представлен листинг qsort:

#include <vector>

#include <iostream>

#include <random>

#include <chrono>

#include <algorithm>

using namespace std::chrono;

#include <vector>

#include <iostream>

template <typename T>

void insertionSort( std::vector<T> & array, int left, int right )

{

for( int p = left; p<=right; ++p )

{

T tmp = std::move( array[ p ] );

int j;

for( j = p; j > 0 && tmp < array[ j - 1 ]; --j )

array[ j ] = std::move( array[ j - 1 ] );

array[ j ] = std::move( tmp );

}

}

template <typename T>

const T & median( std::vector<T> & a, int left, int right )

{

int center = ( left + right ) / 2;

if( a[ center ] < a[ left ] )

std::swap( a[ left ], a[ center ] );

if( a[ right ] < a[ left ] )

std::swap( a[ left ], a[ right ] );

if( a[ right ] < a[ center ] )

std::swap( a[ center ], a[ right ] );

std::swap( a[ center ], a[ right - 1 ] );

return a[ right - 1 ];

}

template <typename T>

void quicksort( std::vector<T> & a, int left, int right )

{

if( left + 10 <= right )

{

const T & pivot = median( a, left, right );

int i = left, j = right - 1;

for( ; ; )

{

while( a[ ++i ] < pivot ) { }

while( pivot < a[ --j ] ) { }

if( i < j )

std::swap( a[ i ], a[ j ] );

else

break;

}

std::swap( a[ i ], a[ right - 1 ] );

quicksort( a, left, i - 1 );

quicksort( a, i + 1, right );

}

else

insertionSort( a, left, right );

}

template <typename T>

void sort( std::vector<T> & array )

{

quicksort( array, 0, array.size( ) - 1 );

}

Ниже представлен листинг программы, которая сортирует по возрастанию четные числа и по убыванию нечетные:

#include <vector>

#include <iostream>

using namespace std;

void sort(vector<int>& array)

{

vector<int> odd;

vector<int> even;

for(const auto& h:array)

{

if(h%2) odd.push\_back(h);

else even.push\_back(h);

}

for(int i=0;i<odd.size();i++)

{

int temp = std::move(odd[i]);

int j;

for( j = i; j > 0 && temp > odd[ j - 1 ]; --j )

odd[ j ] = std::move( odd[ j - 1 ] );

odd[ j ] = std::move( temp );

}

for(int i=0;i<even.size();i++)

{

int temp = std::move(even[i]);

int j;

for( j = i; j > 0 && temp < even[ j - 1 ]; --j )

even[ j ] = std::move( even[ j - 1 ] );

even[ j ] = std::move( temp );

}

even.insert(end(even),begin(odd),end(odd));

array.swap(even);

}

int main()

{

vector <int> arr {1,7,5,4,9,12,17,12,18,16,23,12,175,12,171};

sort(arr);

for(const auto& h:arr)

cout<<h<<" ";

}

Ниже представлен листинг программы, которая может отсортировать массив по возрастанию или убыванию на любом интервале:

#include <vector>

#include <iostream>

#include <random>

#include <chrono>

#include <algorithm>

using namespace std;

#include <vector>

#include <iostream>

bool bigger(const int lhs,const int rhs)

{

return lhs>rhs;

}

bool lesser(const int lhs,const int rhs)

{

return lhs<rhs;

}

void insertionSort( std::vector<int> & array, int left, int right, bool(\*func)(const int,const int))

{

for( int p = left; p<=right; ++p )

{

int tmp = std::move( array[ p ] );

int j;

for( j = p; j > 0 && func(tmp,array[ j - 1 ]); --j )

array[ j ] = std::move( array[ j - 1 ] );

array[ j ] = std::move( tmp );

}

}

int main()

{

vector<int> arr {1,7,2,15,543,254,13,84,93,59,234};

insertionSort(arr,0,arr.size()-1,lesser);

for(const auto& h: arr)

cout<<h<<" ";

cout<<endl;

insertionSort(arr,0,arr.size()-1,bigger);

for(const auto& h: arr)

cout<<h<<" ";

}

В процессе работы также возник вопрос, можно ли использовать вышеописанные способы сортировки для двумерного массива. Если использовать вспомогательные структуры данных (например одномерный массив, в который просто будут помещены все элементы из двумерного массива), то вполне можно использовать данные способы сортировки.

Вывод

В результате работы были приобретены навыки работы основными методами упорядочения числовых данных, а также познакомился с различными алгоритмами сортировки.