Министерство науки и высшего образования РФ
ФГАОУ ВО «Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина»
ИРИТ-РТФ
Центр ускоренного обучения

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по дисциплине «Прикладное программирование»

Тема: ***Освоение основных методов упорядочения числовых данных,знакомство с реализацией различных алгоритмов сортировки средствами языка C++***

Студент группы РИЗ-200028у: И. С. Арсентьев
Преподаватель: О. Л. Чагаева,
ст. преподаватель

Екатеринбург 2022

СОДЕРЖАНИЕ

1 Постановка задачи

Цель:

1. Разобраться,как работает предложенная программа сортировки масива методом «мини-макса».
2. Написать программы,сортирующие одномерный массив методами «пузырька» и «быстрой сортировки».
3. Для массива целых значений выполнить сортировки по возрастаниючётных и по убыванию нечётных

2 Описание работы

2.1. Выполнение поставленных задач.

1. Сортировка методом «мини-макса»

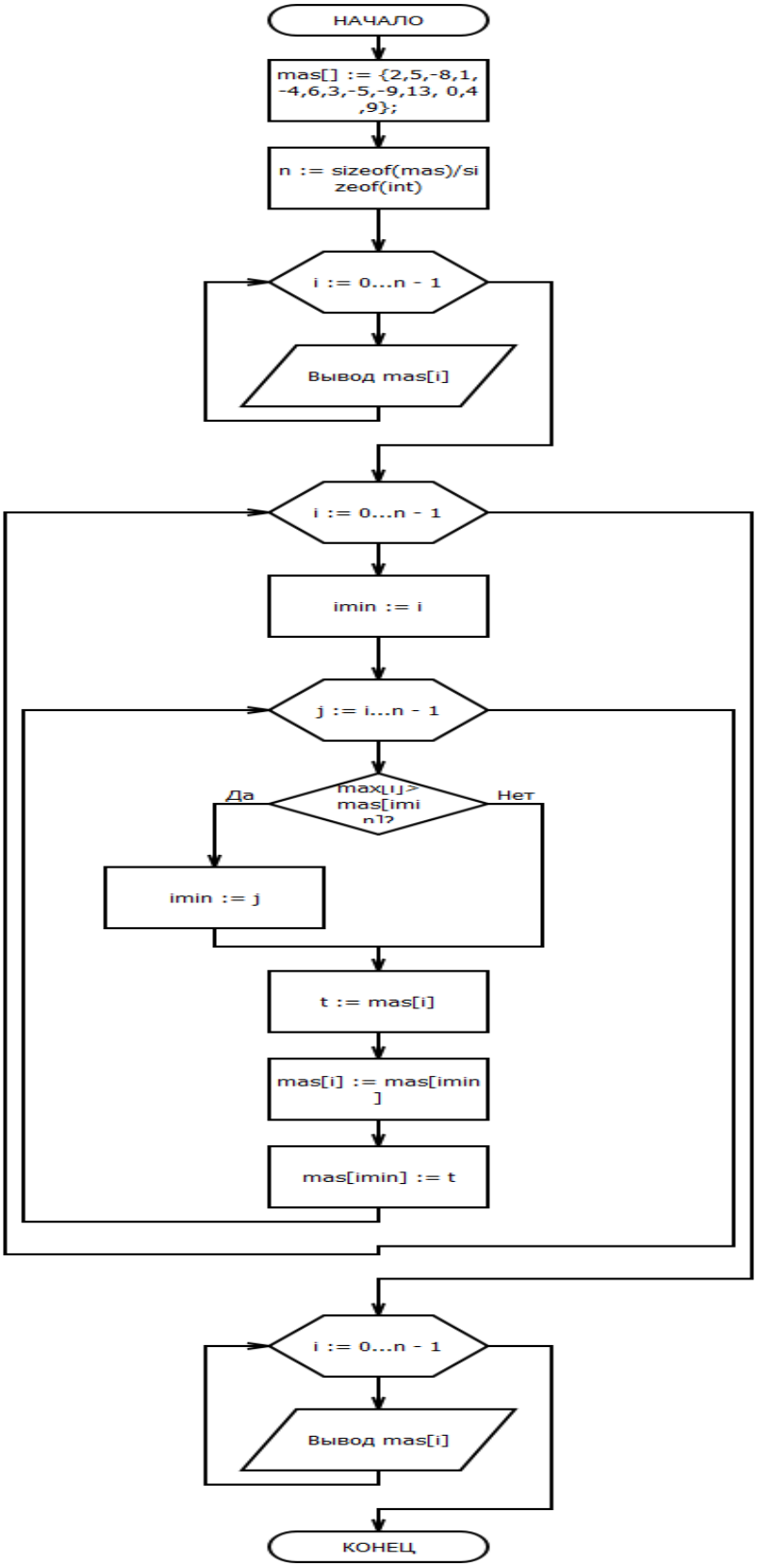

Рисунок 1 – Блок схема к решению задачи 1

```
//Сортировка методом минимакса
#include <iostream>
#include <conio.h>
using namespace std;
void main() //точка старта
{     //вводим тип данных и значения переменных
int mas[] = { 2,1,5,7,-4,56,6,5,-7,-9 };
//текущие
int imin, imax;
int n = sizeof(mas) / sizeof(int); //Использование оператора sizeof для
нахождения размера массива
int i;
imin = i = 0;
imax = i = 0;
      //вывод полученного массива
for (i = 0; i < n; i++)
{
cout << mas[i] << " ";
cout << endl;
}// реализация сортировки методом минимакса
for (i = 0; i < n - 1; i++)
{imin = i;
for (int j = i + 1; j < n; j++)
{if (mas[j] < mas[imin])
{imin = j;
}
int t = mas[i];
mas[i] = mas[imin];
mas[imin] = t;
}}//вывод полученного результата в виде массива
for (i = 0; i < n; i++)
{
cout << mas[i] << " ";
}
cout << endl;
      }
```

2. Сортировка методом «пузырька»

```
#include <iostream>
#include <conio.h>
using namespace std;
void main()
{
      setlocale(LC_ALL, "ru"); // русская кодировка в командной строке
      //ввод массива создание переменных типов данных и присваивания им
//значений.
      int mas[] = { 2,5,-83 - 41,6,3, 10, 23, 324,34, 34,34, 87,3,-90,4,9 };
      int imin, imax;
      int n = sizeof(mas) / sizeof(int);
      int i, temp;
      //вывод полученного результата
      cout << "Массив до сортировки:";
            for (i = 0; i < n; i++)
      {cout << mas[i] << " ";
      }
      cout << endl;
      cout << "Массив после сортировки:";
            for (i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++)
            {if (mas[j] < mas[i]) {
                        temp = mas[i];
                        mas[i] = mas[j];
                        mas[j] = temp;
                  }}}
      for (i = 0; i < n; i++)
      {
            cout << mas[i] << " ";
      }
      cout << endl;
}
```

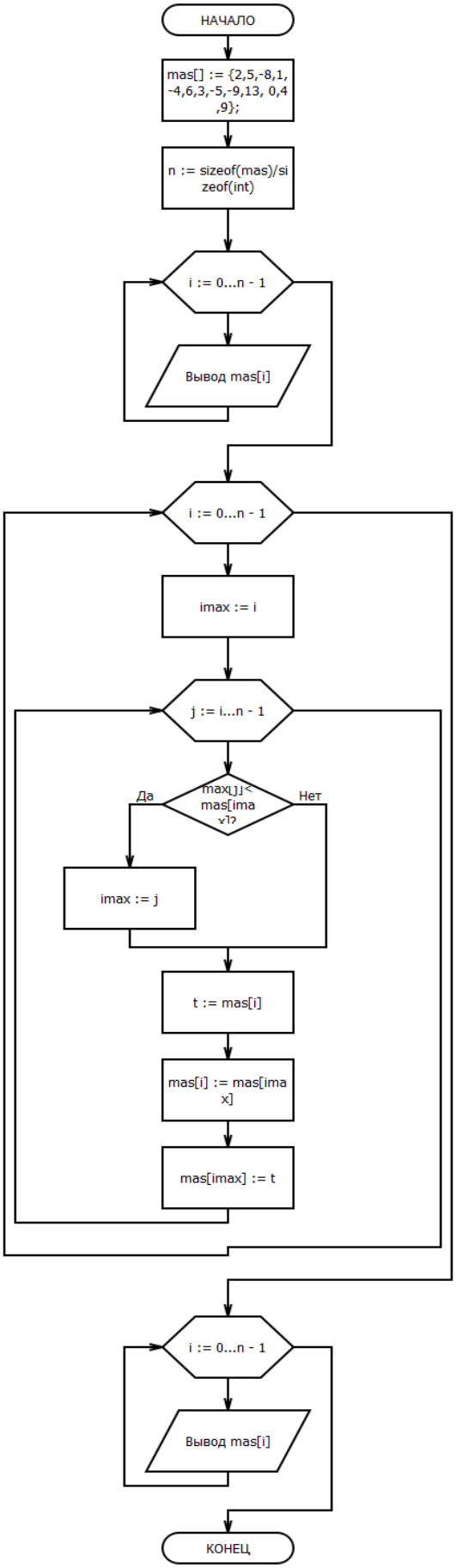

Рисунок 2 – Блок схема к решению задачи 2

3. Сортировка методом «быстрой сортировки»

```
#include <iostream>
#include <iomanip>
#include <ctime>
#include <algorithm>
#include <vector>
using namespace std;
const int n = 10;
int first, last;
int i;
//функция сортировки
void quicksort(int* mas, int first, int last)
{
    int mid, count;
    int f = first, l = last;
    mid = mas[(f + l) / 2]; //вычисление опорного элемента
    do
    {
        while (mas[f] < mid) f++;
        while (mas[l] > mid) l--;
        if (f <= l) //перестановка элементов
        {
            count = mas[f];
            mas[f] = mas[l];
            mas[l] = count;
            f++;
            l--;
        }
    } while (f < l);
    if (first < l) quicksort(mas, first, l);
    if (f < last) quicksort(mas, f, last);
}
int main() {
    srand(time(NULL));
    int k = 0, m = 0;
    int** B = new int* [k]; int** C = new int* [m];
        setlocale(LC_ALL, "Rus");
    int* A = new int[n];
    cout << "Исходный массив: ";

    for (int i = 0; i < n; i++)//наполнение массива случайными числами
    {
        A[i] = rand() % 100;
        cout << A[i] << " ";
    }

    first = 0; last = n - 1;
    quicksort(A, first, last);
        cout << endl << "Сортированный исходный массив: ";//сортировка
исходного массива
    for (int i = 0; i < n; i++) cout << A[i] << " ";
    cout << endl;
    cout << ("чётные элементы, отсортированные по возрастанию: ") << "
";//сортировка чётных элементов по возрастанию
    for (int i = 0; i < n; i++) {
        if (A[i] % 2 == 0)
        {
            cout << A[i] << " ";
        }
    }
    cout << endl;
```

```
    cout << ("Нечётные элементы, отсортированные по убыванию: ") << "
";//сортировка нечётных элементов по убыванию
    for (int i = 1; i < n; ++i)
    {
        for (int r = 0; r < n - i; r++)
        {
            if (A[r] < A[r + 1])
            {
                // Обмен местами
                int temp = A[r];
                A[r] = A[r + 1];
                A[r + 1] = temp;
            }
        }
    }
    for (int i = 0; i < n; i++) {
        if (A[i] % 2 != 0)
        {
            cout << A[i] << " ";
        }
    } system("pause>>void");
    return 0;
    cout << endl;

}
```

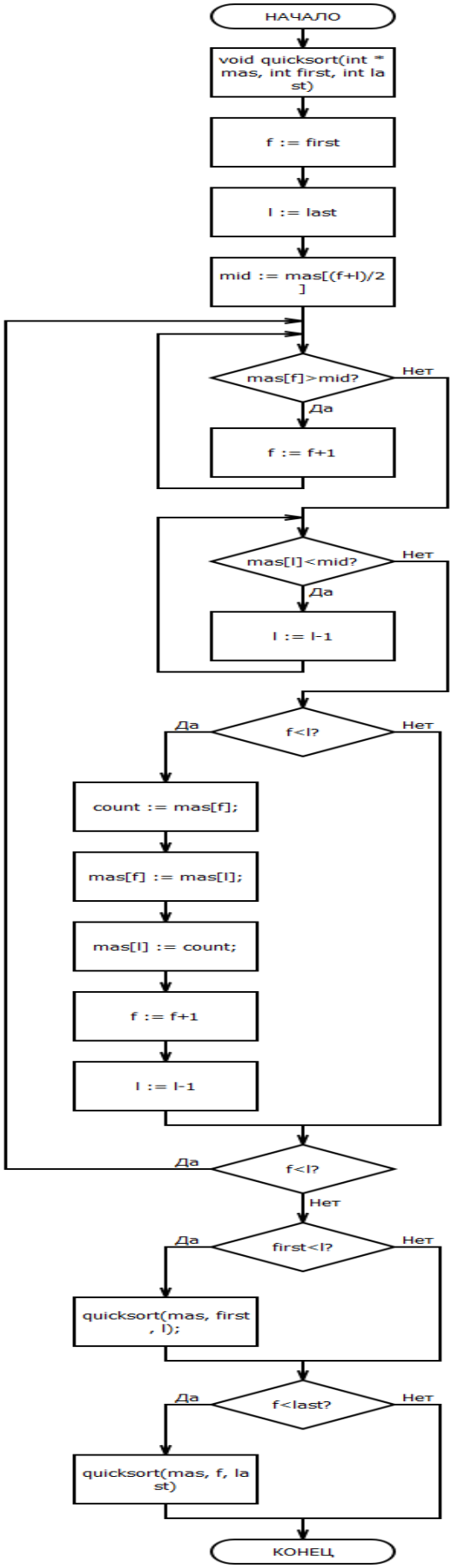

Рисунок 3 – Блок-схема к 3 задаче

4. Отсортировать массив по возрастанию на интервале от индекса N2 до N1

```
#include <iostream>
using namespace std;
//рекурсивный метод быстрой сортировки
void quicksort(int* mas, int first, int last)
{
    int mid, count;
    int f = first, l = last;
    mid = mas[(f + l) / 2]; //вычисление опорного элемента
    do
    {
        while (mas[f] < mid) f++;
        while (mas[l] > mid) l--;
        if (f <= l) //перестановка элементов
        {
            count = mas[f];
            mas[f] = mas[l];
            mas[l] = count;
            f++;
            l--;
        }
    } while (f < l);
    if (first < l) quicksort(mas, first, l);
    if (f < last) quicksort(mas, f, last);
}
int main()
{    setlocale(LC_ALL, "Rus");
    int mas[] = { 2,5,-8,1,-4,6,3,-5,-9,13, 0,4,9 };
    //вычисление n - количества элементов
    int n = sizeof(mas) / sizeof(int);
    int i;
        for (i = 0; i < n; i++)
        cout << mas[i] << " ";
    cout << endl;
    int n1, n2;
    cout << "Введите индекс n1>>"; cin >> n1;
    cout << "Введите индекс n2>>"; cin >> n2;
    if (n2 > n1)
        quicksort(mas, (n2 - 1), (n1 - 1));
    if (n2 < n1)
        quicksort(mas, (n2-1), (n1-1));
    for (i = (n1 - 1); i <=(n2-1); i++)
        cout << mas[i] << " ";
    cout << endl;
    system("pause>>void");
}
```

5. Отсортировать массив по убыванию на интервале от индекса N2 до N1

```
#include <iostream>
using namespace std;//рекурсивный метод быстрой сортировки
void quicksort(int* mas, int first, int last)
{    int mid, count;
    int f = first, l = last;
    mid = mas[(f + l) / 2]; //вычисление опорного элемента
    do { while (mas[f] > mid) f++;
        while (mas[l] < mid) l--;
        if (f <= l) //перестановка элементов
        {count = mas[f];
            mas[f] = mas[l];
            mas[l] = count;
            f++; l--;
        }} while (f < l);
    if (first < l) quicksort(mas, first, l);
    if (f < last) quicksort(mas, f, last);
}int main()
{  setlocale(LC_ALL, "Rus");
    int mas[] = { 2,5,-8,1,-4,6,3,-5,-9,13, 0,4,9 };
    //вычисление n - количества элементов
    int n = sizeof(mas) / sizeof(int);
    int i;for (i = 0; i < n; i++)
        cout << mas[i] << " ";
    cout << endl;
    int n1, n2;
    cout << "Введите индекс n1:"; cin >> n1;
    cout << "Введите индекс n2:"; cin >> n2;
    if (n2 > n1){quicksort(mas, (n1 - 1), (n2 - 1));
        for (i = (n1 - 1); i < (n2 - 1); i++)
            cout << mas[i] << " ";
        cout << mas[i] << " ";
    }if (n2 < n1)
    { quicksort(mas, (n2 - 1), (n1 - 1));
        for (i = (n2 - 1); i < (n1 - 1); i++)
            cout << mas[i] << " ";
        cout << endl;
    }
}
```

3 Выводы по лабораторной работе

Алгоритм сортировки методом пузырька — это довольно простой в реализации алгоритм для сортировки массивов. Можно встретить и другие названия: пузырьковая сортировка, BubbleSort или сортировка простыми обменами — но все они будут обозначать один и тот же алгоритм.

Назван алгоритм так, потому что большее или меньшее значение «всплывает» (сдвигается) к краю массива после каждой итерации,

Отличительной особенностью быстрой сортировки является операция разбиения массива на две части относительно опорного элемента. Например, если последовательность требуется упорядочить по возрастанию, то в левую часть будут помещены все элементы, значения которых меньше значения опорного элемента, а в правую элементы, чьи значения больше или равны опорному.