Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №5**

**«ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ»**

**ПО ДИСЦИПЛИНЕ**

**«ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнила: студент учебной группы

ИСПк-202-52-00

Долгополов Ярослав Максимович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы**: получить базовые сведения о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами.

**Формулировка задания. Вариант 8**

**Постановка задачи**

* 1. Реализовать сортировку данных с помощью алгоритма подсчетом.
  2. Реализовать сортировку данных с помощью алгоритма слияния.
  3. В обоих случаях необходимо предусмотреть возможность изменения компаратора (реализация компаратора в виде передаваемой в подпрограмму функции).
  4. Считывание и вывод данных необходимо производить из текстового файла.

1. **Описание алгоритма**
   1. Считываем файлы **f1** и **f2**.
   2. Открываем **f1** для чтения и все числа оттуда записываем в массив **arrinput**. После этого закрываем файл **f1**.
   3. Вводим символ «+» для сортировки по возрастанию и «-» для сортировки по убыванию.
   4. Дальше применяем методы сортировки:
      1. Для алгоритма подсчёта применяем процедуру, которая подсчитывает каждое значение элементов в массиве и заносит их количество в массив **count** на позицию, равному значению элемента (напр.: если в массиве 3 элемента, равных 5, то число 3 заносим на позицию 5). После этого процедура перебирает массив **count** с параметром **i** (с начала до конца, если до этого был введён «+» или с конца до начала, был введён «-») и заносим в массив **arroutput** число **i** в количестве **count[i]**.
      2. Для алгоритма слияния применяем процедуру, которая посредством рекурсии разделяет, а затем соединяет части массива в порядке возрастания элементов.
   5. Открываем файл **f2** для записи. Записываем элементы отсортированного массива в файл **f2**. После этого закрываем файл **f2**.
2. **Схема алгоритма с комментариями**

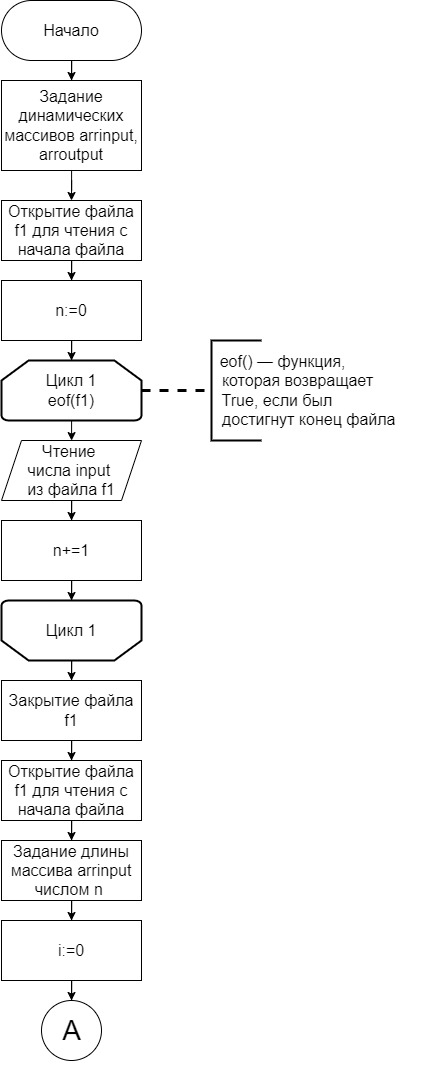
****

Рисунок 1 - Алгоритм решения задачи. Сортировка подсчётом. Часть 1

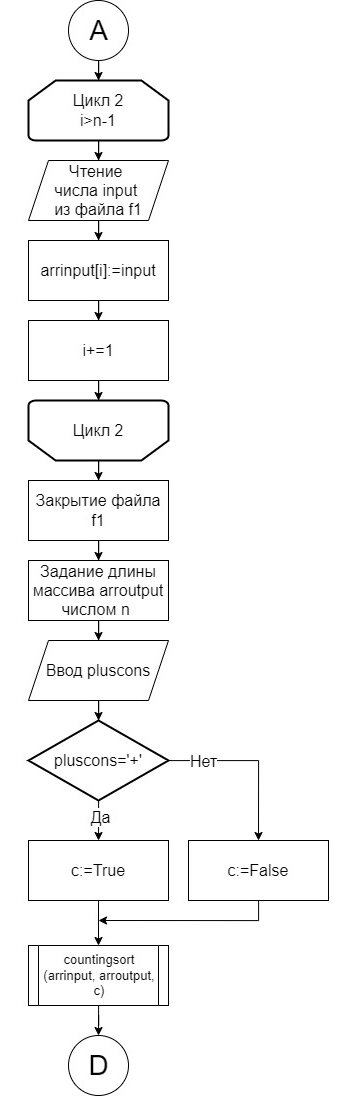
****

Рисунок 2 - Алгоритм решения задачи. Сортировка подсчётом. Часть 2

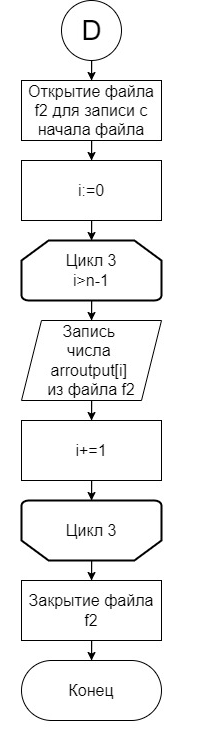


Рисунок 3 - Алгоритм решения задачи. Сортировка подсчётом. Часть 3

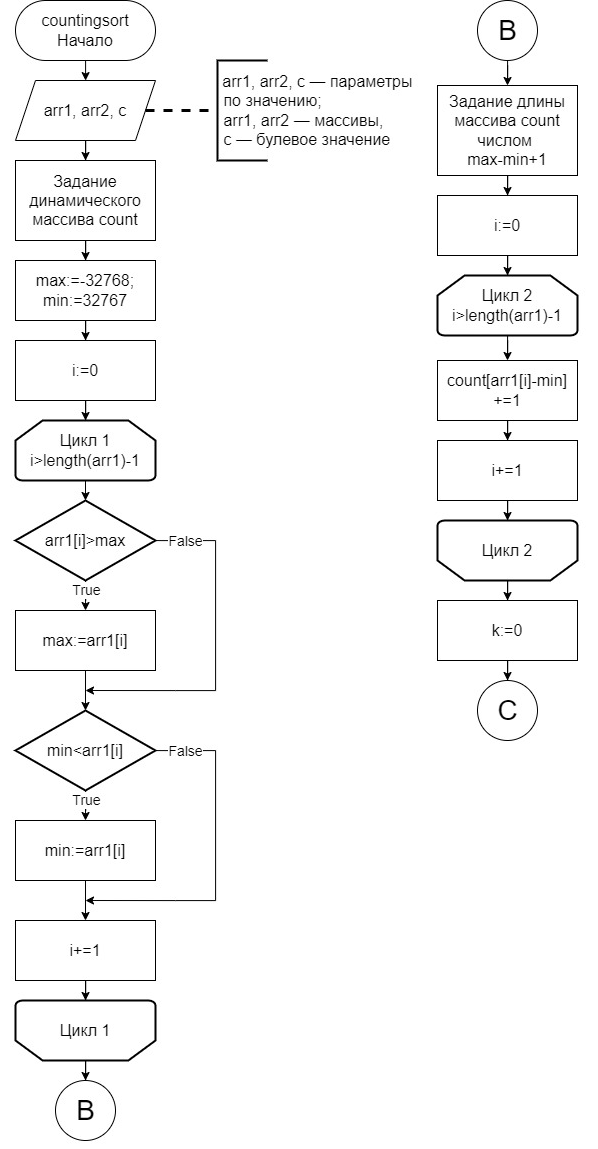
****

Рисунок 4 - Алгоритм процедуры countingsort. Часть 1

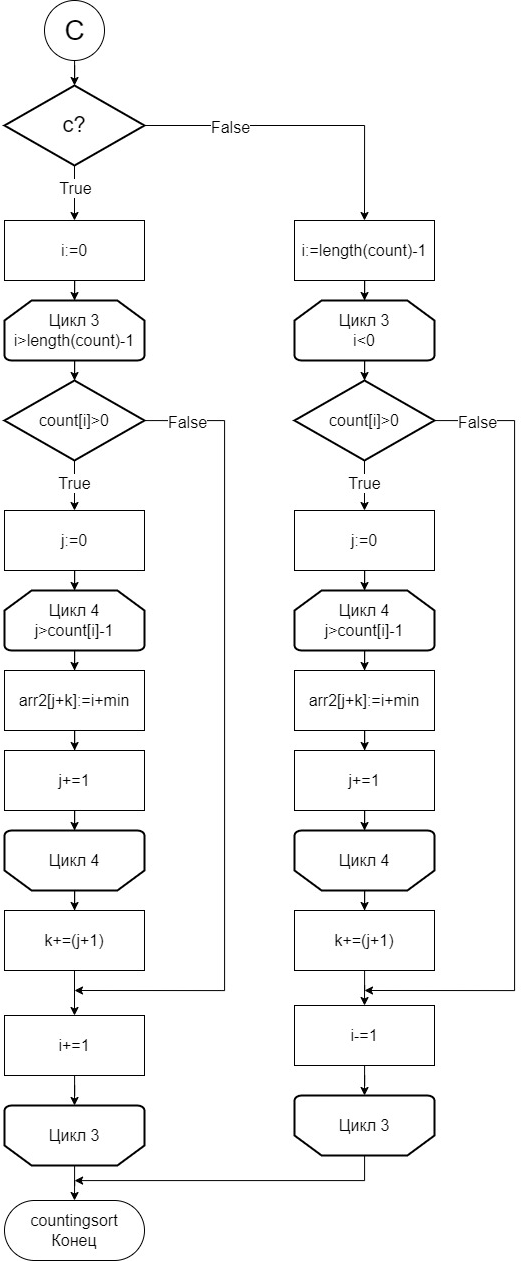


Рисунок 5 - Алгоритм процедуры countingsort. Часть 2



Рисунок 6 - Алгоритм решения задачи. Сортировка слиянием. Часть 1

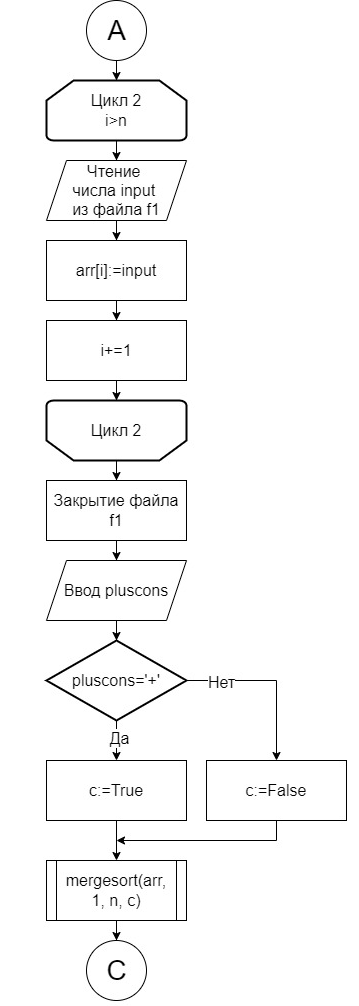


Рисунок 7 - Алгоритм решения задачи. Сортировка слиянием. Часть 2

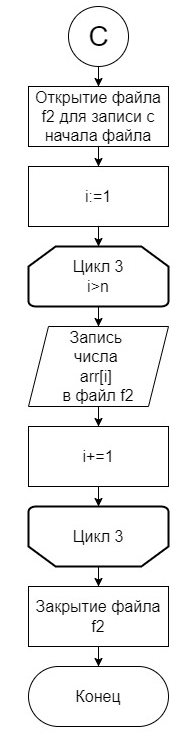
****

Рисунок 8 - Алгоритм решения задачи. Сортировка слиянием. Часть 3

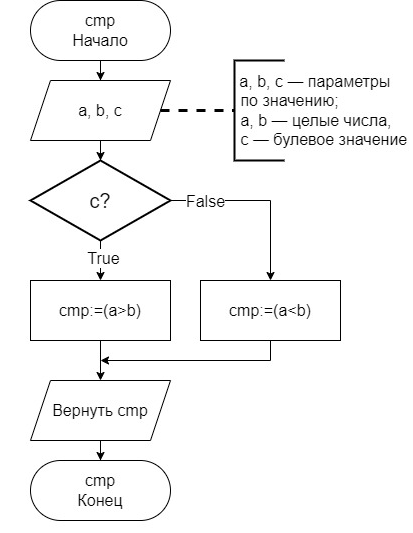
****

Рисунок 9 - Алгоритм процедуры cmp.

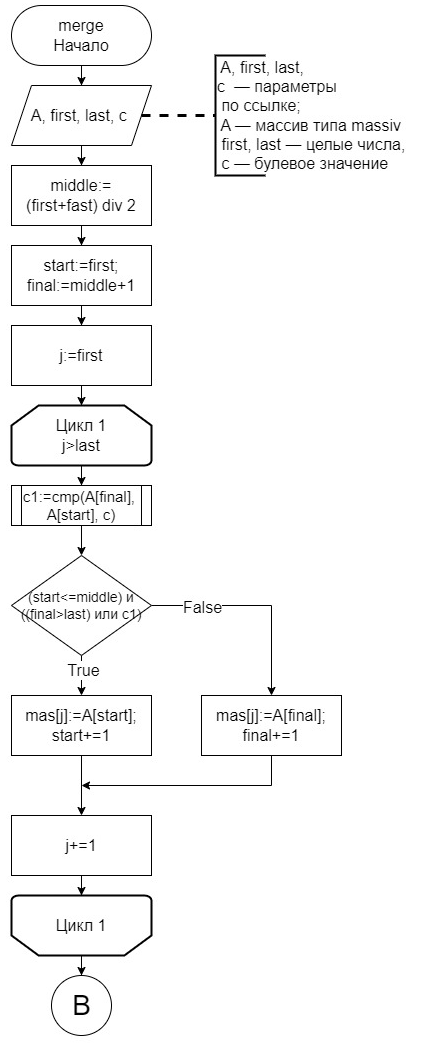
****

Рисунок 10 - Алгоритм процедуры merge. Часть 1

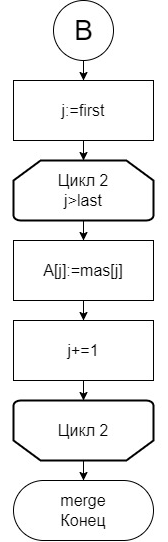
****

Рисунок 11 - Алгоритм процедуры merge. Часть 2

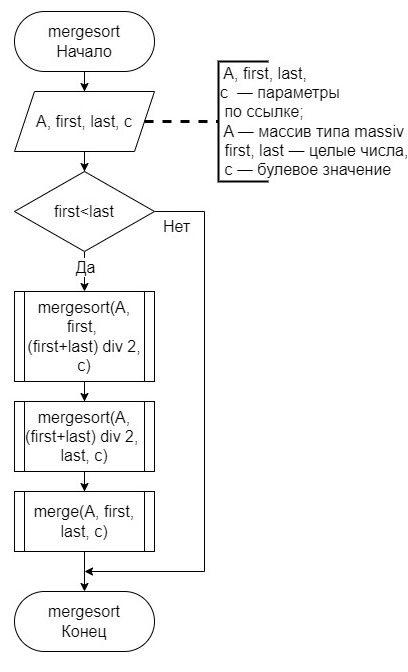
****

Рисунок 12 - Алгоритм процедуры mergesort.

1. **Коды программ**
   1. **Сортировка подсчётом**

**procedure** countingsort(arr1, arr2: **array of** integer; c: boolean);

**var**

max, min, i, j, k: integer;

count: **array of** integer;

**begin**

max:=-32768;

min:=32767;

**for** i:=0 **to** (length(arr1)-1) **do**

**begin**

**if** arr1[i]>max **then** max:=arr1[i];

**if** min>arr1[i] **then** min:=arr1[i];

**end**;

setlength(count, max-min+1);

**for** i:=0 **to** (length(arr1)-1) **do**

**begin**

inc(count[arr1[i]-min])

**end**;

k:=0;

**if** c **then**

**for** i:=0 **to** (length(count)-1) **do**

**begin**

**if** count[i]>0 **then**

**begin**

**for** j:=0 **to** (count[i]-1) **do** arr2[j+k]:=i+min;

k+=(j+1)

**end**

**end**

**else**

**for** i:=(length(count)-1) **downto** 0 **do**

**begin**

**if** count[i]>0 **then**

**begin**

**for** j:=0 **to** (count[i]-1) **do** arr2[j+k]:=i+min;

k+=(j+1)

**end**

**end**

**end**;

**var**

f1, f2: text;

pluscons: char;

c: boolean;

n, a, b, i, input: integer;

arrinput, arroutput: **array of** integer;

**begin**

assign(f1, 'C:\Users\User\Documents\Колледж\ОАиП\ДКР5\Input.txt');

assign(f2, 'C:\Users\User\Documents\Колледж\ОАиП\ДКР5\Output1.txt');

reset(f1);

n:=0;

**while not** eof(f1) **do**

**begin**

readln(f1, input);

inc(n)

**end**;

close(f1);

reset(f1);

setlength(arrinput, n);

**for** i:=0 **to** n-1 **do**

**begin**

readln(f1, input);

arrinput[i]:=input;

**end**;

close(f1);

setlength(arroutput, n);

readln(pluscons);

**if** pluscons='+' **then** c:=True **else** c:=False;

countingsort(arrinput, arroutput, c);

rewrite(f2);

**for** i:=0 **to** n-1 **do**

**begin**

writeln(f2, arroutput[i])

**end**;

close(f2)

**end**.

* 1. **Сортировка слиянием**

**type** massiv=**array**[1..100000] **of** integer;

**function** cmp(a, b: integer; c: boolean ): boolean;

**begin**

**if** c **then** cmp:=a>b **else** cmp:=a<b

**end**;

**procedure** merge(**var** A: massiv; first, last: integer; c: boolean);

**var**

middle, start, final , j: integer;

mas: massiv;

**begin**

middle:=(first+last) **div** 2;

start:=first;

final:=middle+1;

**for** j:=first **to** last **do**

**if** (start<=middle) **and** ((final>last) **or** cmp(A[final], A[start], c)) **then**

**begin**

mas[j]:=A[start];

inc(start);

**end**

**else**

**begin**

mas[j]:=A[final];

inc(final);

**end**;

**for** j:=first **to** last **do** A[j]:=mas[j];

**end**;

**procedure** mergesort(**var** A: massiv; first, last: integer; c: boolean);

**begin**

**if** first<last **then**

**begin**

MergeSort(A, first, (first+last) **div** 2, c); {сортировка левой части}

MergeSort(A, (first+last) **div** 2+1, last, c); {сортировка правой части}

Merge(A, first, last, c); {слияние двух частей}

**end**;

**end**;

**var**

f1, f2: text;

n, i, input: integer;

pluscons: char;

c: boolean;

arr: massiv;

**begin**

assign(f1, 'C:\Users\User\Documents\Колледж\ОАиП\ДКР5\Input.txt');

assign(f2, 'C:\Users\User\Documents\Колледж\ОАиП\ДКР5\Output2.txt');

reset(f1);

n:=0;

**while not** eof(f1) **do**

**begin**

readln(f1, input);

inc(n)

**end**;

close(f1);

reset(f1);

**for** i:=1 **to** n **do**

**begin**

readln(f1, input);

arr[i]:=input;

**end**;

close(f1);

readln(pluscons);

**if** pluscons='+' **then** c:=True **else** c:=False;

mergesort(arr, 1, n, c);

rewrite(f2);

**for** i:=1 **to** n **do**

**begin**

writeln(f2, arr[i])

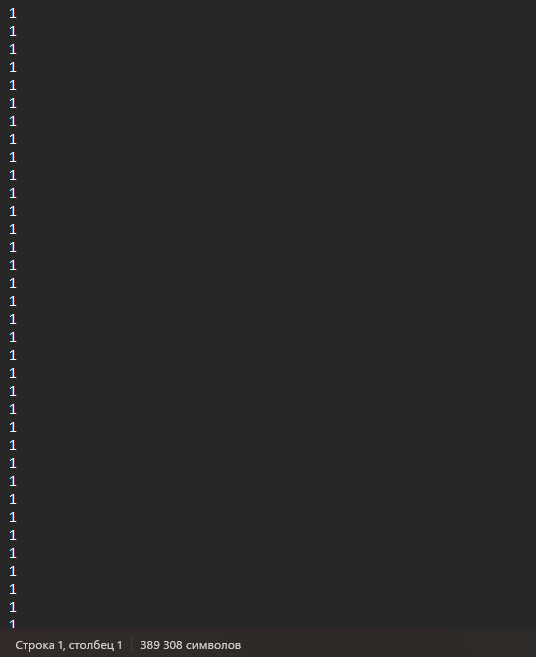
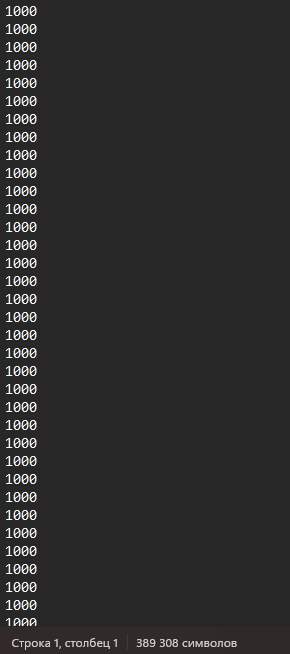
**end**;

close(f2)

**end**.

1. **Результат выполнения программы**

Сортировка подсчётом (по возрастанию/по убыванию):

Сортировка слиянием (по возрастанию/по убыванию):

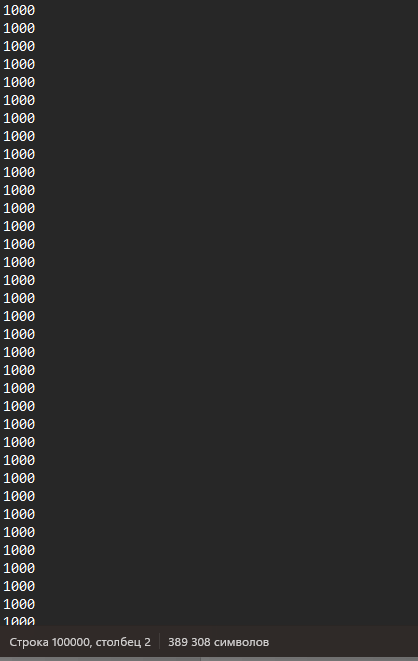
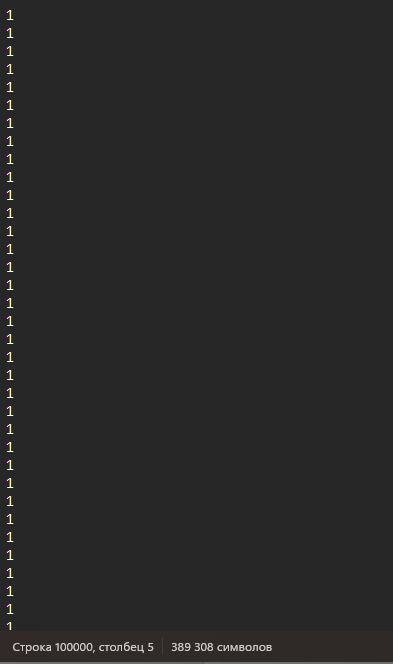


Рисунок 13 - Результаты выполнения программы**Вывод**

В процессе работы над домашней контрольной мы изучили принципы осуществления сортировки разными методами и работы с текстовыми файлами.

Для создания алгоритмических схем был использован draw.io — бесплатный онлайн-инструмент для создания и совместного редактирования диаграмм и схем.

В ходе работы мы столкнулись с некоторыми трудностями. Например, нельзя было оценить скорость работы каждого метода сортировки. Это было решено посредством разделения кода на разные файлы для разных методов сортировки.

В итоге, благодаря полученным знаниям и устранённым ошибкам, мы смогли успешно решить задачу, применив код с процедурами сортировки чисел и взаимодействием с текстовыми файлами

.