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»**

Выполнил: студент учебной группы ИСПк- 205-52-00

Ильин Тимофей Анатольевич

Преподаватель:

Сергеева Елизавета Григорьевна

г. Киров

2023

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

**Задание:**

1. Реализовать сортировку данных с помощью алгоритма подсчетом.
2. Реализовать сортировку данных с помощью поразрядного алгоритма.
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.
5. Для демонстрации работы подпрограммы реализаций самостоятельно подготовить варианты входных данных (при этом объем тестовых файлов должен позволять оценить скорость работы программ).

**Описание алгоритма:**

1. Происходит чтение входных данных из файла 'input.txt', в котором первая строка содержит количество элементов массива, а остальные строки содержат сами элементы.
2. Создается динамический массив arr длины n, в котором будут храниться числа из файла.
3. Происходит выбор метода сортировки. Если пользователь выбирает сортировку слиянием (i = 1), то ему дополнительно предлагается выбрать тип сортировки (по возрастанию или убыванию).
4. В соответствии с выбранным типом сортировки, функция Greater или Less назначается переменной comparator, которая является типом функции сравнения (TComparator).
5. Вызывается процедура MergeSort, которая сортирует массив arr с использованием сортировки слиянием и заданным компаратором. Сортировка слиянием разделяет массив на две половины, сортирует их рекурсивно, а затем объединяет полученные отсортированные половины в один отсортированный массив.
6. Если пользователь выбрал сортировку вставками (i = 2), то ему также предлагается выбрать тип сортировки (по возрастанию или убыванию).
7. В соответствии с выбранным типом сортировки, функция Greater или Less назначается переменной comparator.
8. Вызывается процедура InsertionSort, которая сортирует массив arr с использованием сортировки вставками и заданным компаратором. Сортировка вставками проходит по массиву, поочередно вставляя каждый элемент на нужное место в уже отсортированную часть массива.
9. Вызывается процедура WriteDataToFile, которая записывает отсортированный массив в файл 'output.txt'.
10. Выводится сообщение о выполнении сортировки и записи результата в файл 'output.txt'.

**Схемы алгоритмов:**

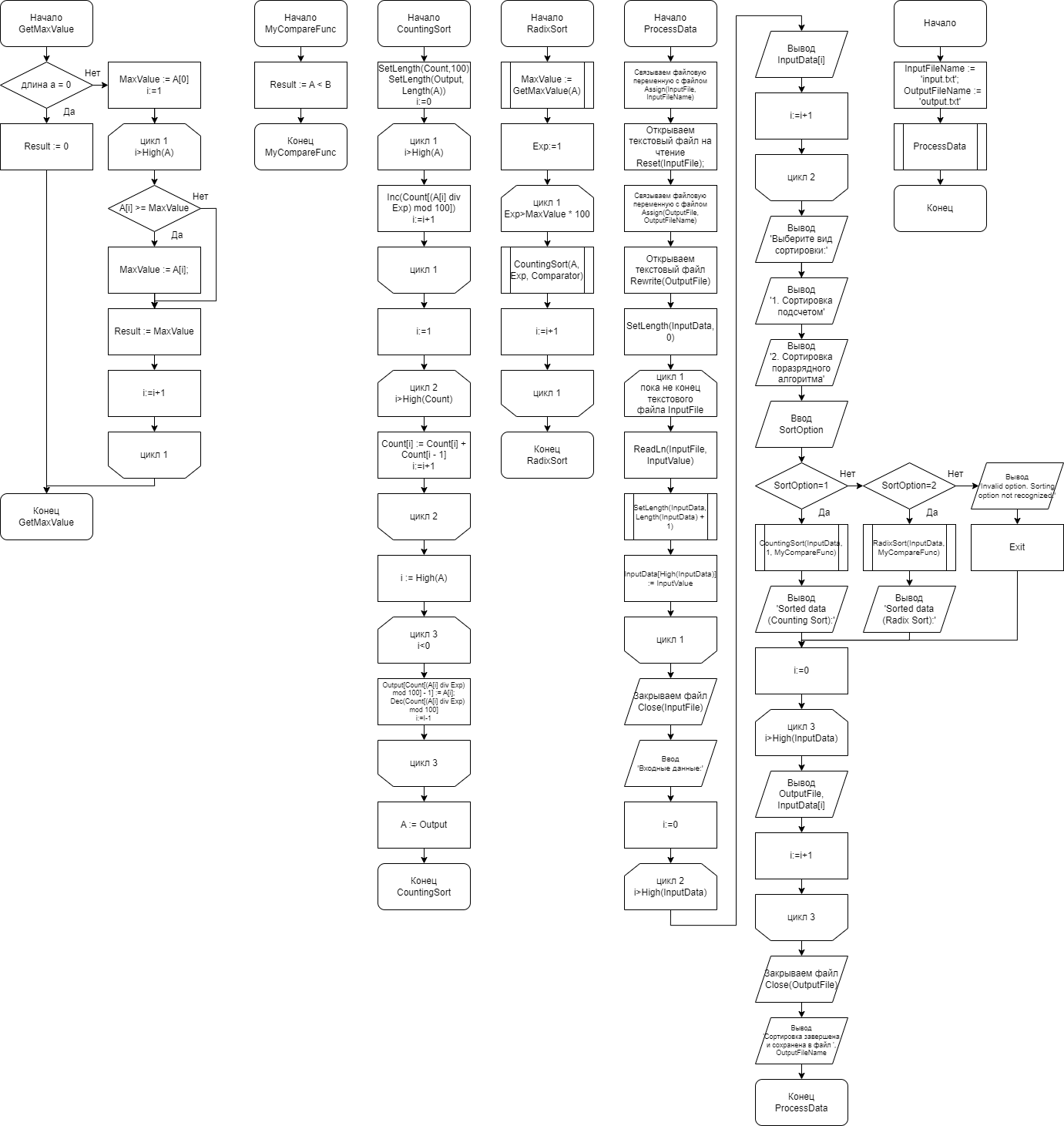


Рисунок 1 – Функция GetMaxValue

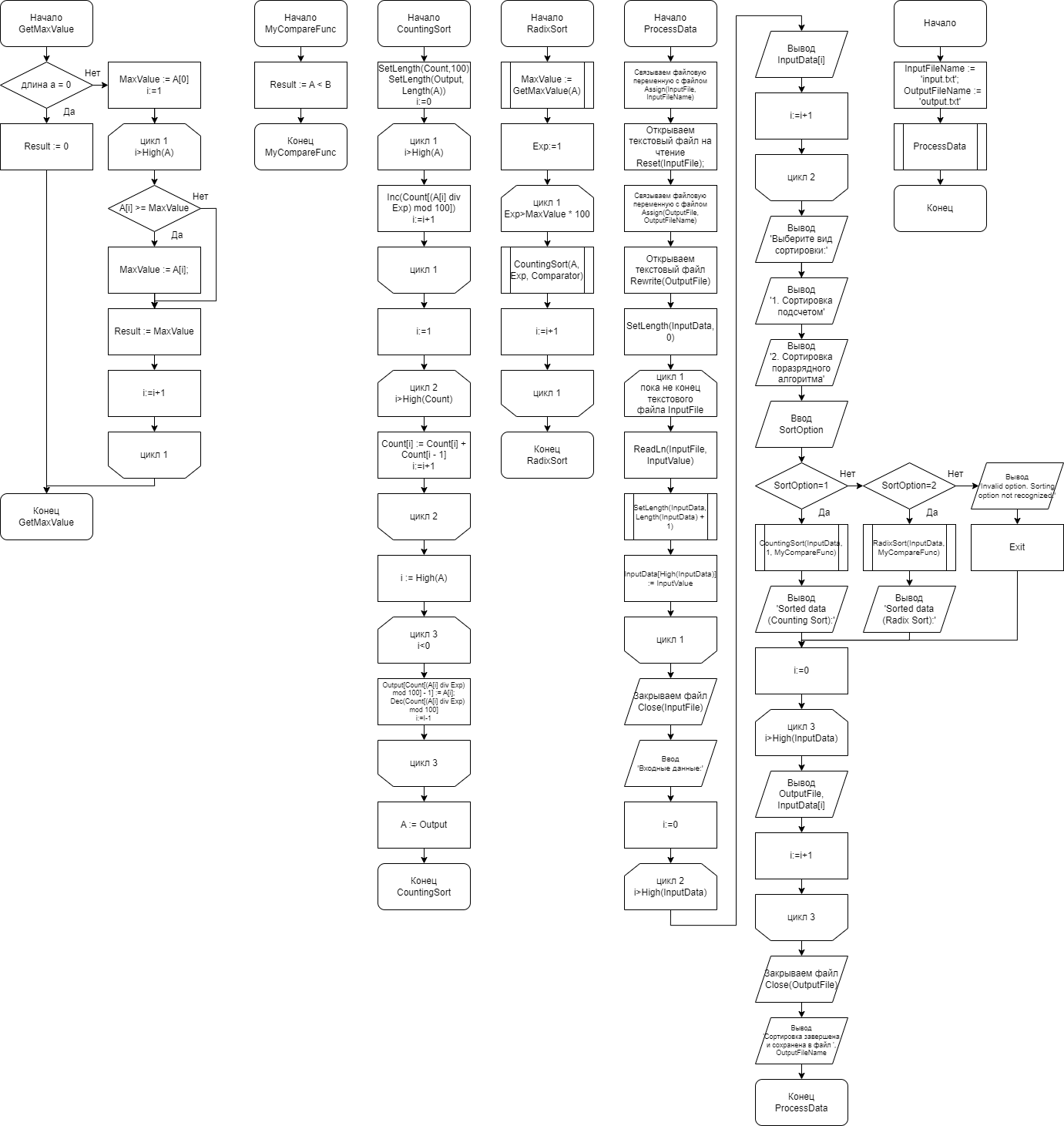


Рисунок 2 – Функция MyCompareFunc

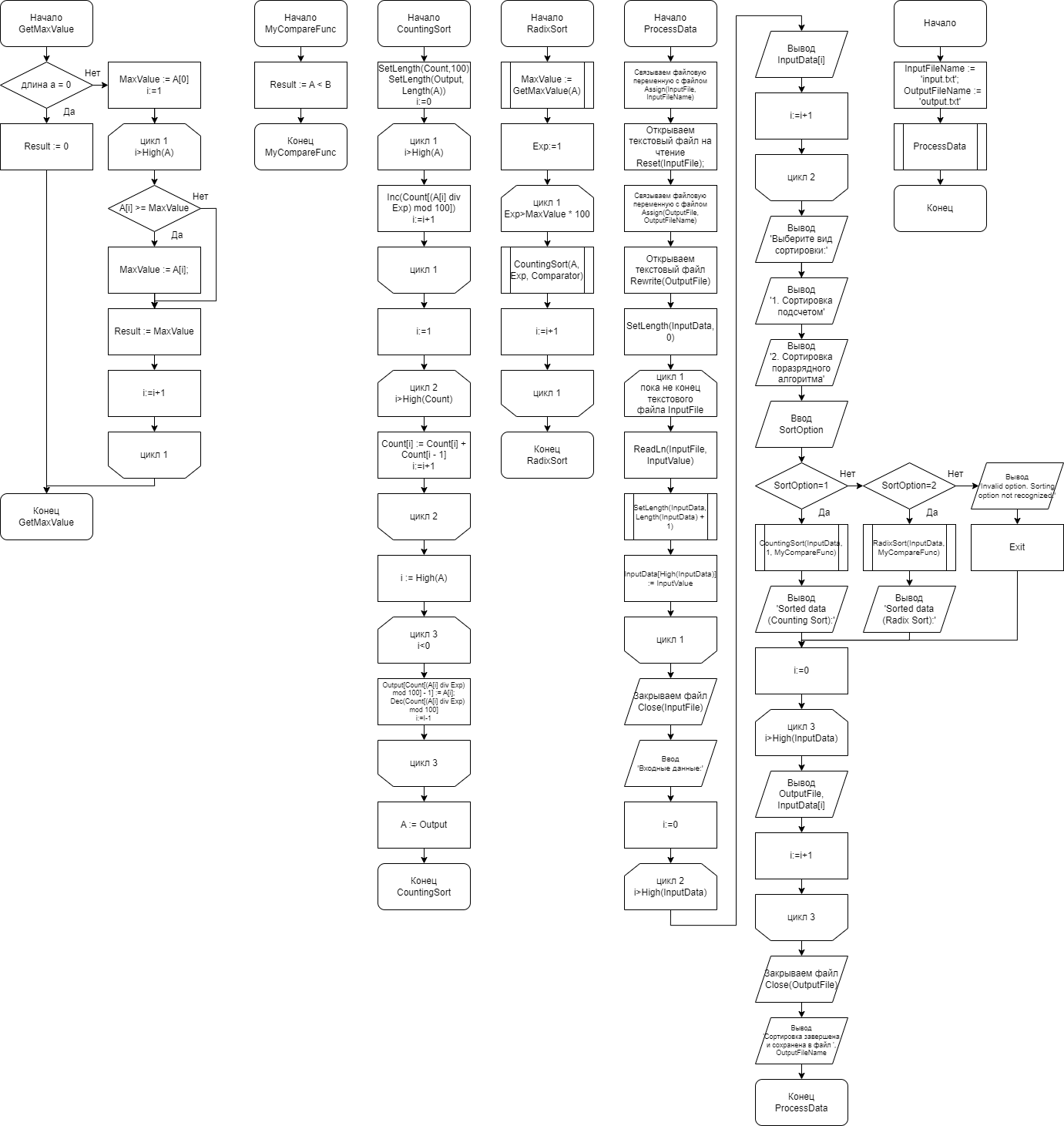


Рисунок 3 – Процедура CountingSort

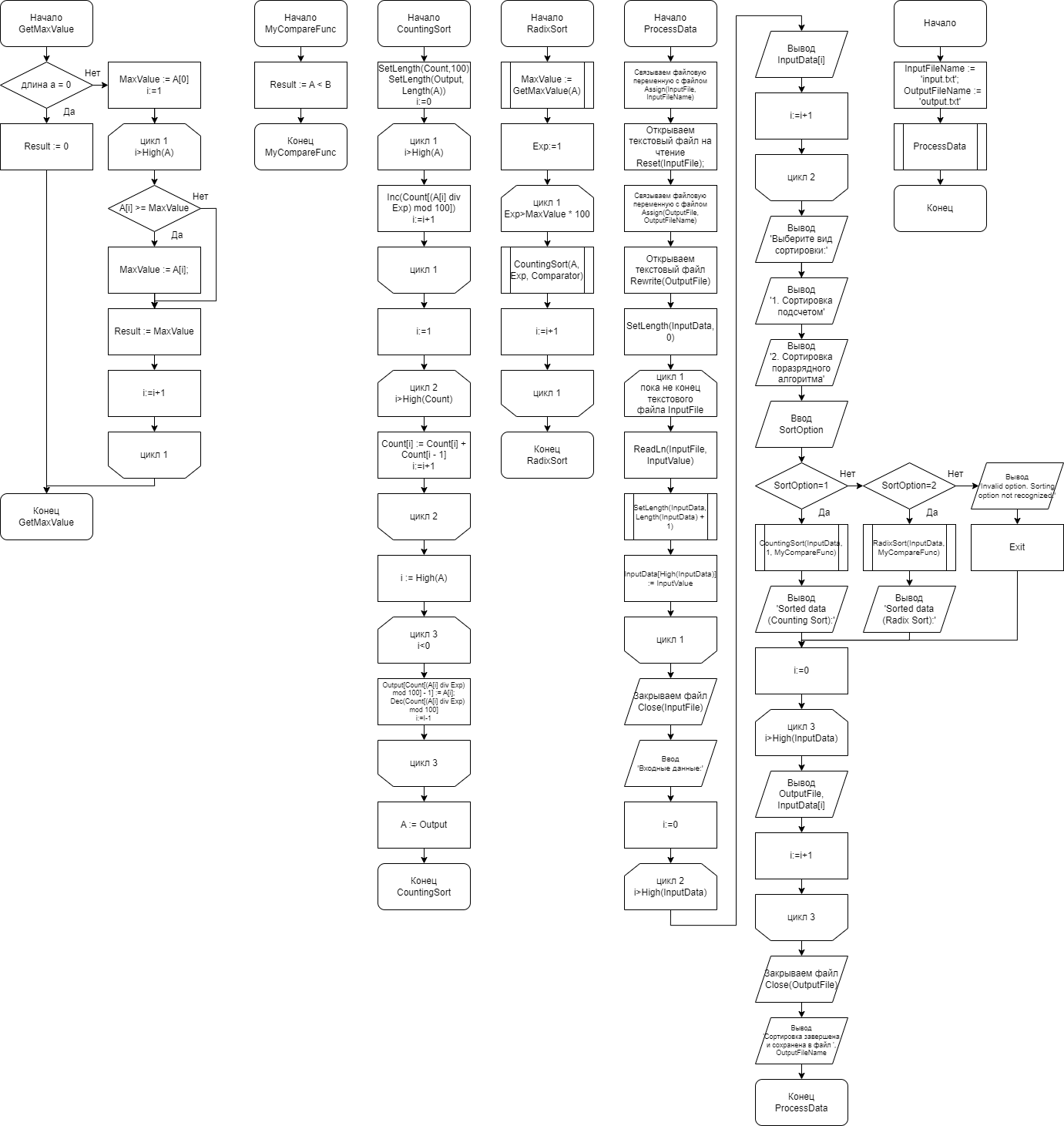


Рисунок 4 – Процедура RadixSort

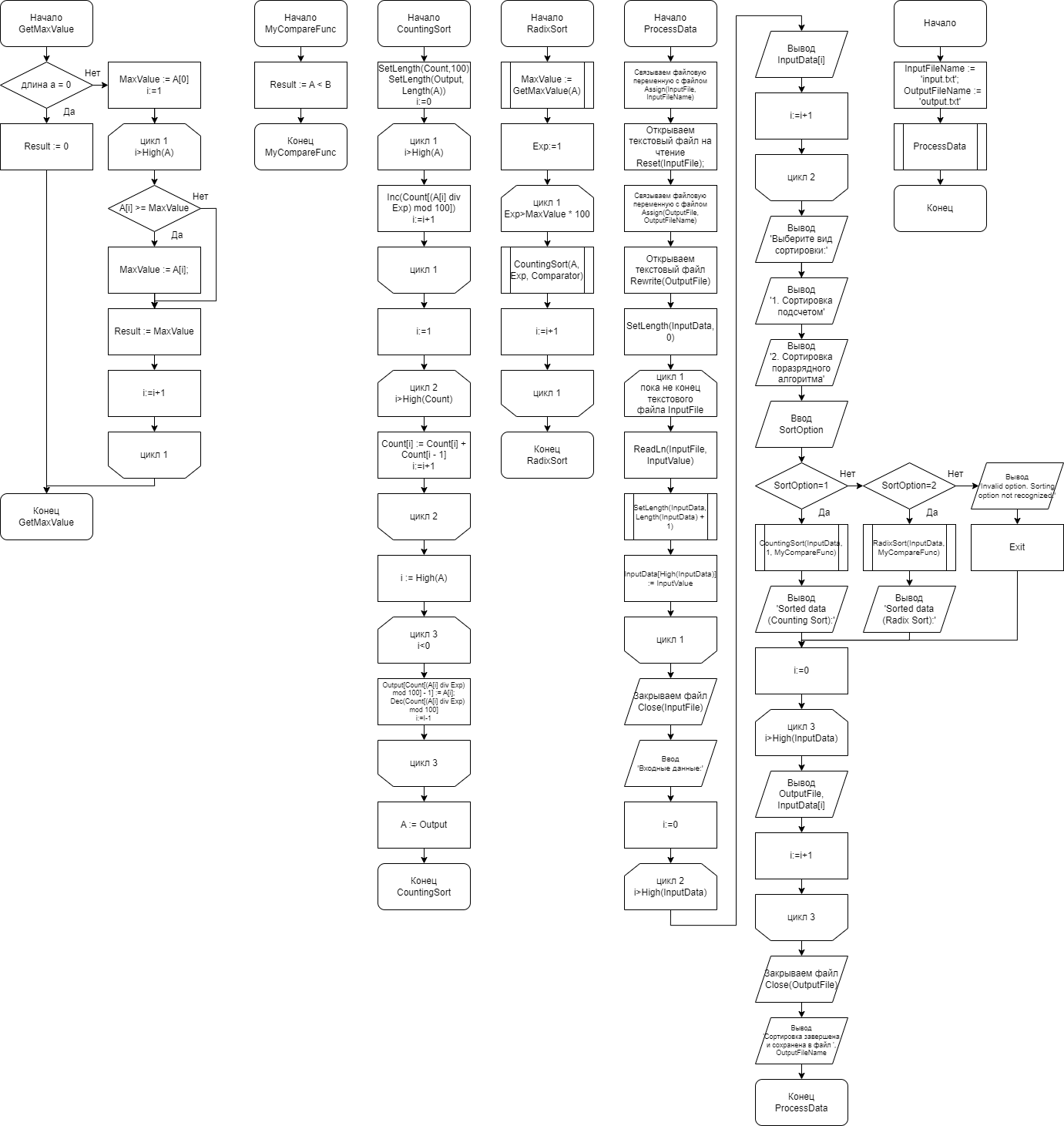


Рисунок 5 – Процедура ProcessData

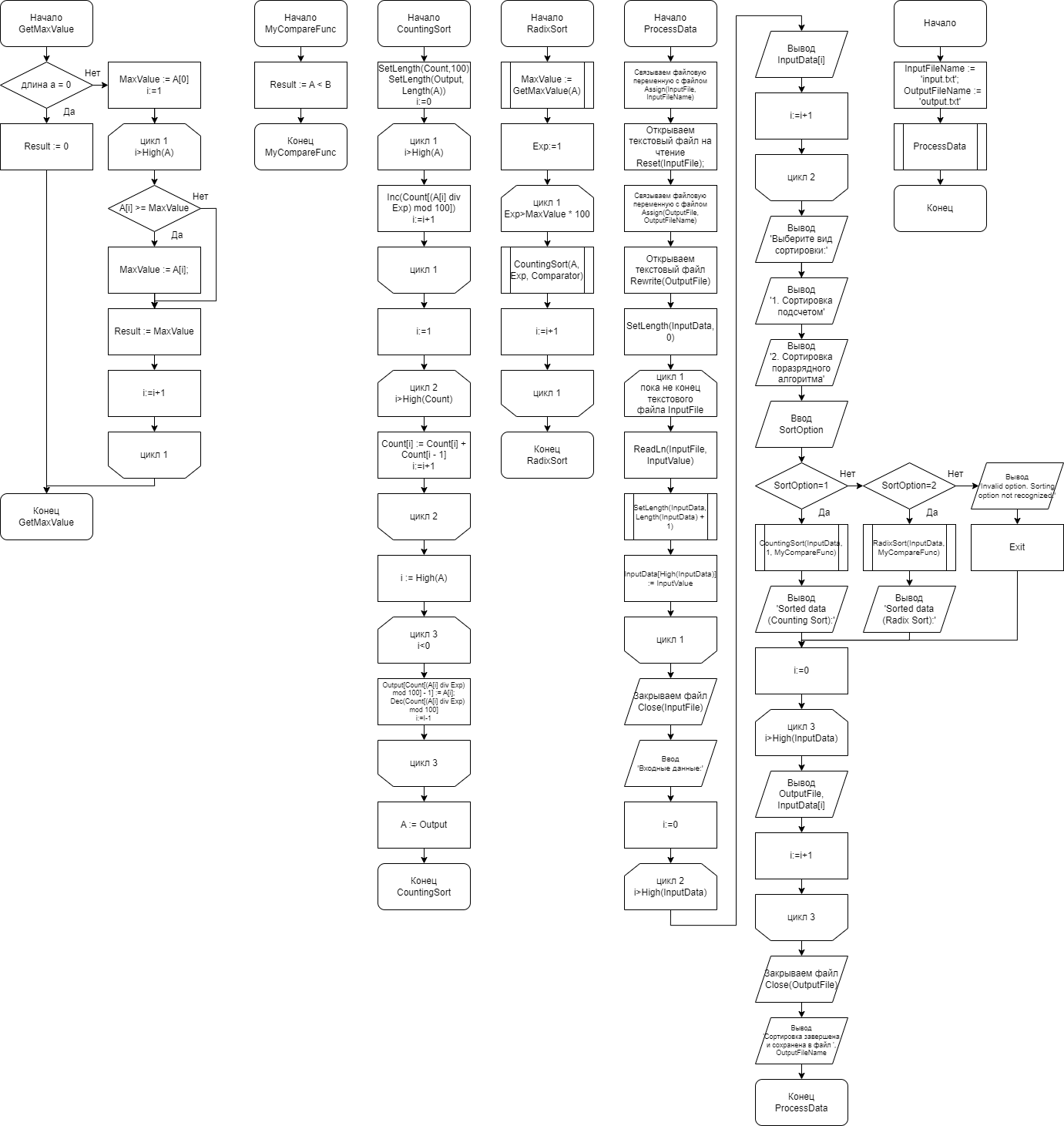


Рисунок 11 – Тело программы

**Код програмы:**

**uses**

ABCObjects;

**const**

MaxArraySize = 10000;

**type**

TArray = **array of** Integer;

**type**

TComparatorFunc = **function**(A, B: Integer): Boolean;

**function** GetMaxValue(**const** A: TArray): Integer;

**var**

i, MaxValue: Integer;

**begin**

**if** Length(A) = 0 **then**

Result := 0

**else**

**begin**

MaxValue := A[0];

**for** i := 1 **to** High(A) **do**

**if** A[i] >= MaxValue **then**

MaxValue := A[i];

Result := MaxValue;

**end**;

**end**;

**function** MyCompareFunc(A, B: Integer): Boolean;

**begin**

// Реализуйте ваше сравнение здесь

Result := A < B;

**end**;

**procedure** CountingSort(**var** A: TArray; Exp: Integer; Comparator: TComparatorFunc);

**var**

Count, Output: **array of** Integer;

i: Integer;

**begin**

SetLength(Count, 100);

SetLength(Output, Length(A));

**for** i := 0 **to** High(A) **do**

Inc(Count[(A[i] **div** Exp) **mod** 100]);

**for** i := 1 **to** High(Count) **do**

Count[i] := Count[i] + Count[i - 1];

**for** i := High(A) **downto** 0 **do**

**begin**

Output[Count[(A[i] **div** Exp) **mod** 100] - 1] := A[i];

Dec(Count[(A[i] **div** Exp) **mod** 100]);

**end**;

A := Output;

**end**;

**procedure** RadixSort(**var** A: TArray; Comparator: TComparatorFunc);

**var**

MaxValue, Exp: Integer;

**begin**

MaxValue := GetMaxValue(A);

**for** Exp := 1 **to** MaxValue \* 100 **do**

CountingSort(A, Exp, Comparator);

**end**;

**var**

InputFileName, OutputFileName: string;

**procedure** ProcessData;

**var**

InputFile, OutputFile: Text;

InputData: TArray;

InputValue: Integer;

i, SortOption: Integer;

**begin**

Assign(InputFile, InputFileName);

Reset(InputFile);

Assign(OutputFile, OutputFileName);

Rewrite(OutputFile);

SetLength(InputData, 0);

**while not** Eof(InputFile) **do**

**begin**

ReadLn(InputFile, InputValue);

SetLength(InputData, Length(InputData) + 1);

InputData[High(InputData)] := InputValue;

**end**;

Close(InputFile);

WriteLn('Входные данные:');

**for** i := 0 **to** High(InputData) **do**

WriteLn(InputData[i]);

WriteLn('Выберите вид сортировки:');

WriteLn('1. Сортировка подсчетом');

WriteLn('2. Сортировка поразрядного алгоритма');

ReadLn(SortOption);

**case** SortOption **of**

1:

**begin**

CountingSort(InputData, 1, MyCompareFunc);

WriteLn('Sorted data (Counting Sort):');

**end**;

2:

**begin**

RadixSort(InputData, MyCompareFunc);

WriteLn('Sorted data (Radix Sort):');

**end**;

**else**

WriteLn('Invalid option. Sorting option not recognized.');

**Exit**;

**end**;

**for** i := 0 **to** High(InputData) **do**

WriteLn(OutputFile, InputData[i]);

Close(OutputFile);

WriteLn('Сортировка завершена и сохранена в файл ', OutputFileName);

**end**;

**begin**

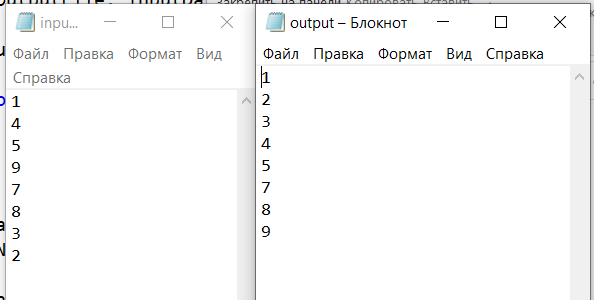
InputFileName := 'input.txt';

OutputFileName := 'output.txt';

ProcessData;

**end**.

1. **Результат выполнения программ:**

****

**Вывод:**

В результате выполнения лабораторной работы были получены базовые знания о наиболее известных алгоритмах сортировки, а также изучены принципы работы с текстовыми файлами. При реализации алгоритмов сортировки с помощью вставок и алгоритма слияния были учтены требования задания, в том числе возможность изменения компаратора путем передачи его в подпрограмму. В ходе работы было отработано считывание и вывод данных из текстового файла, а также подготовлены варианты входных данных для демонстрации работы подпрограммы реализаций.