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-203-52-00

Вершинин Григорий Сергеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

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

**Формулировка задания:**

1. Реализовать сортировку данных с помощью «пузырькового» алгоритма.
2. Реализовать сортировку данных с помощью алгоритма слияния.
3. В обоих случаях необходимо предусматривать 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 подпрограммных реализаций самостоятельно подготовить варианты входных данных (при этом объем текстовых файлов должен позволить оценить скорость работы программы).

**Описание алгоритма:**

1. Пузырьковая сортировка (Bubble Sort)

Пузырьковая сортировка — это простой алгоритм сортировки, который многократно проходит по списку, сравнивает соседние элементы и меняет их местами, если они находятся в неправильном порядке. Проходы по списку повторяются до тех пор, пока не будет сделан проход без обменов, что означает, что список отсортирован.

Как работает код:

• В процедуре BubbleSort используется два вложенных цикла:

• Внешний цикл проходит по всем элементам массива.

• Внутренний цикл сравнивает каждый элемент с его соседом. Если текущий элемент больше следующего (по сравнению с функцией comparator), они меняются местами.

• После завершения всех проходов массив будет отсортирован.

2. Сортировка слиянием (Merge Sort)

Сортировка слиянием — это более эффективный алгоритм сортировки, который использует принцип «разделяй и властвуй». Он рекурсивно делит массив на две половины, сортирует каждую половину и затем объединяет отсортированные половины в один отсортированный массив.

Вывод результатов

В конце программы выводится время, затраченное на каждую из сортировок, а также сообщение о том, где можно найти результаты.

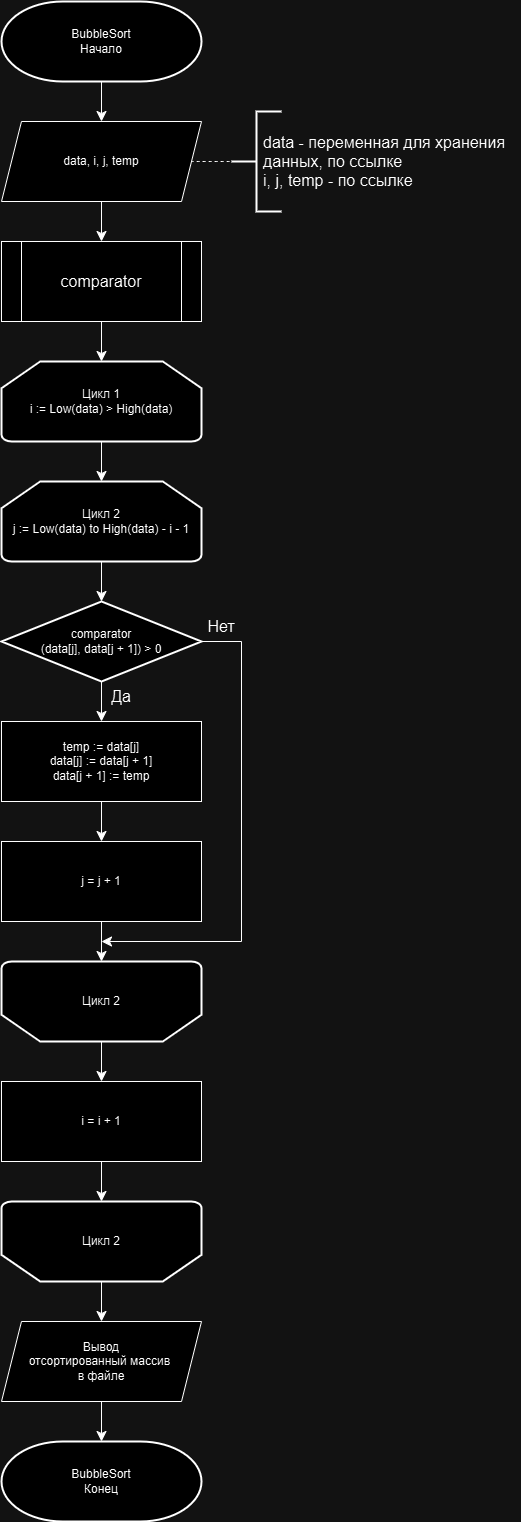


Рисунок 1 – схема алгоритма функции BubbleSort

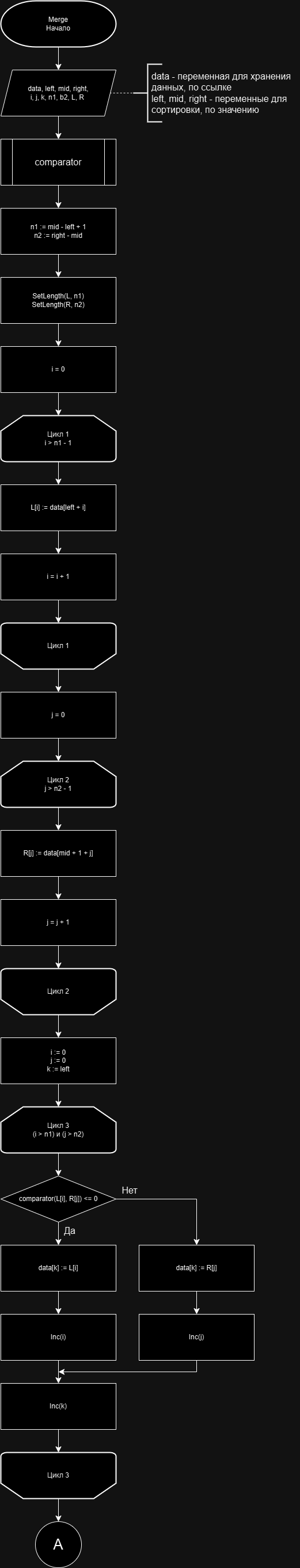


Рисунок 2.1 – схема алгоритма функции Merge

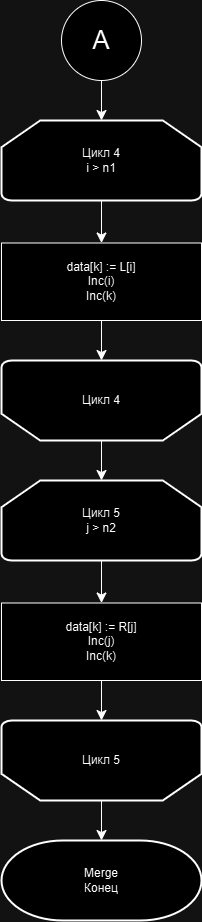


Рисунок 2.2 – схема алгоритма функции Merge

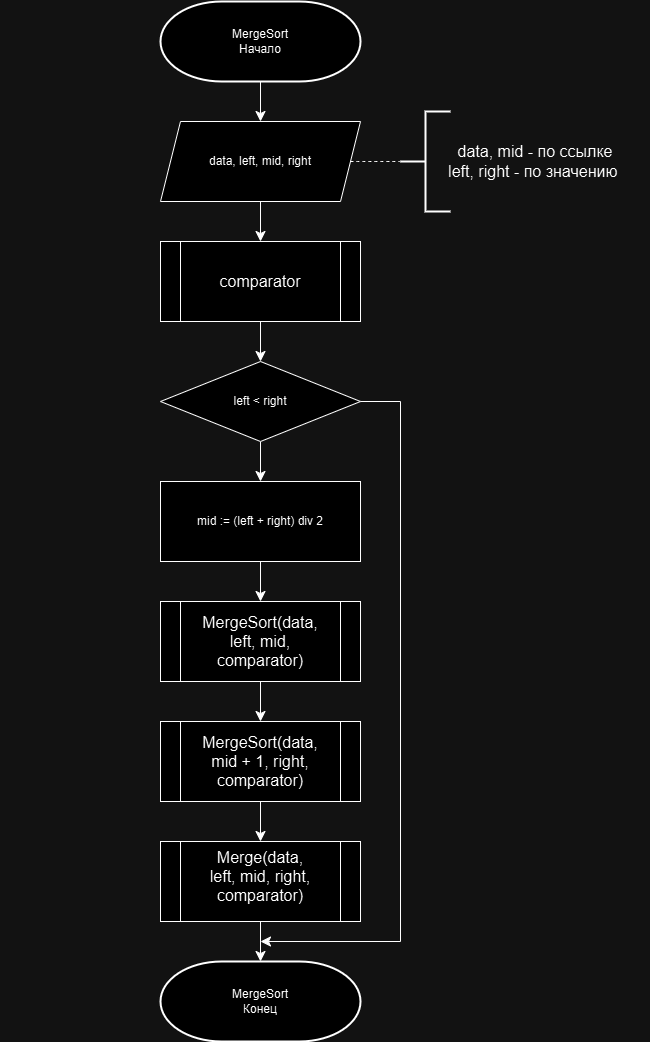


Рисунок 3 – схема алгоритма функции MergeSort

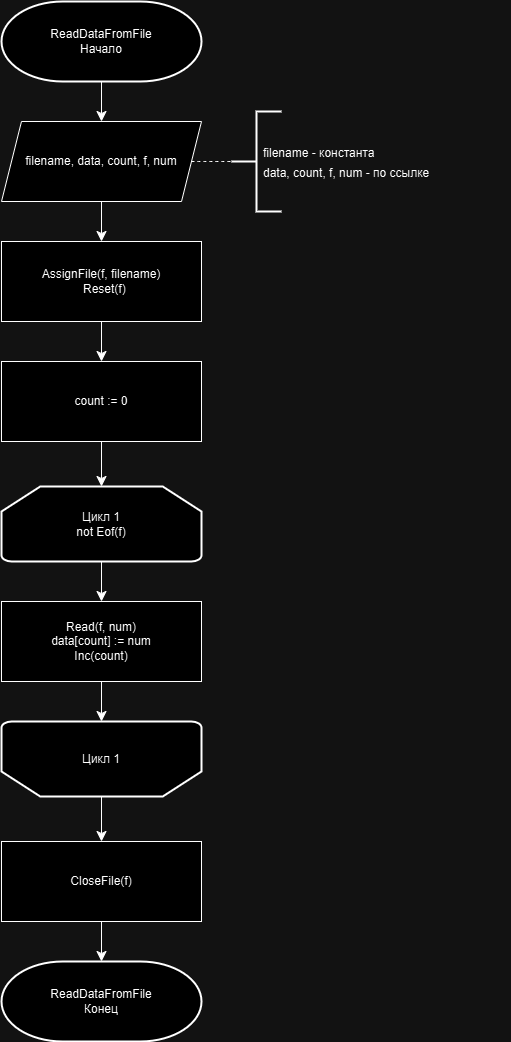


Рисунок 4 – схема алгоритма функции ReadDataFromFile

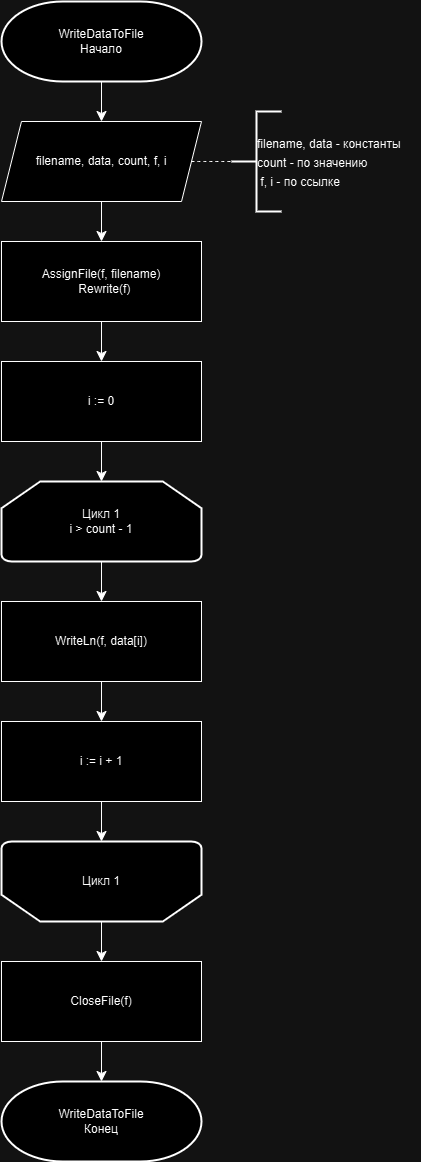


Рисунок 5 – схема алгоритма функции WriteDataToFile

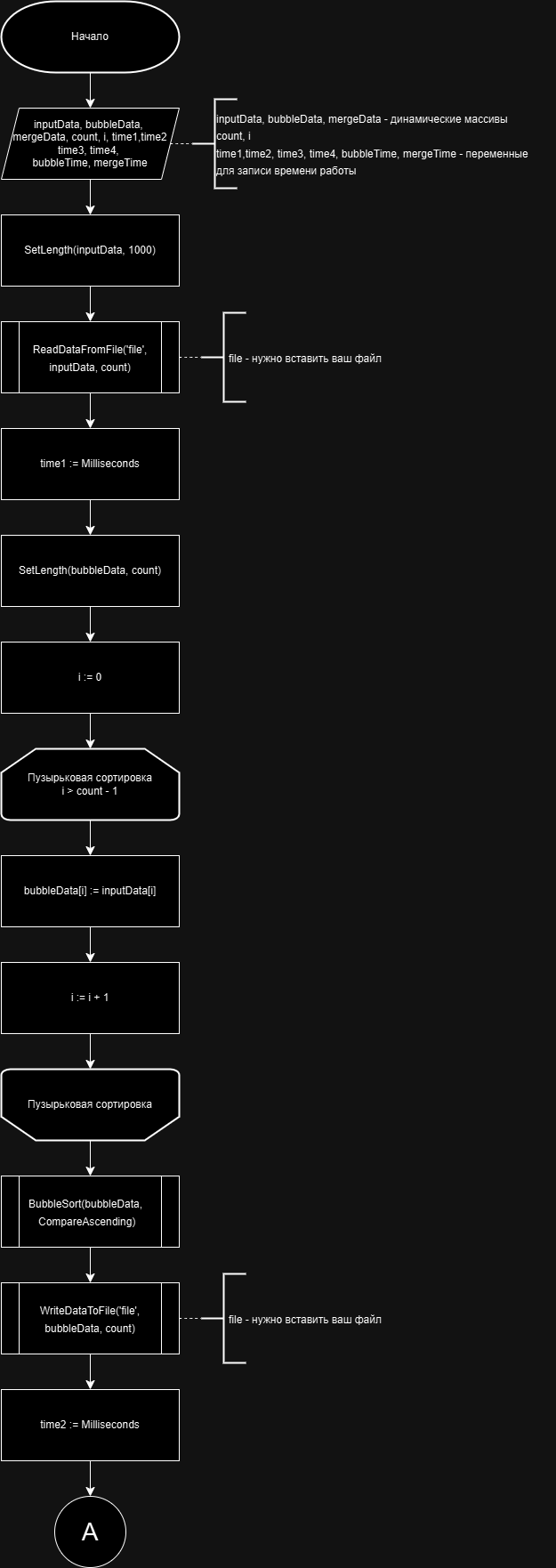


Рисунок 6.1 – схема алгоритма основной программы

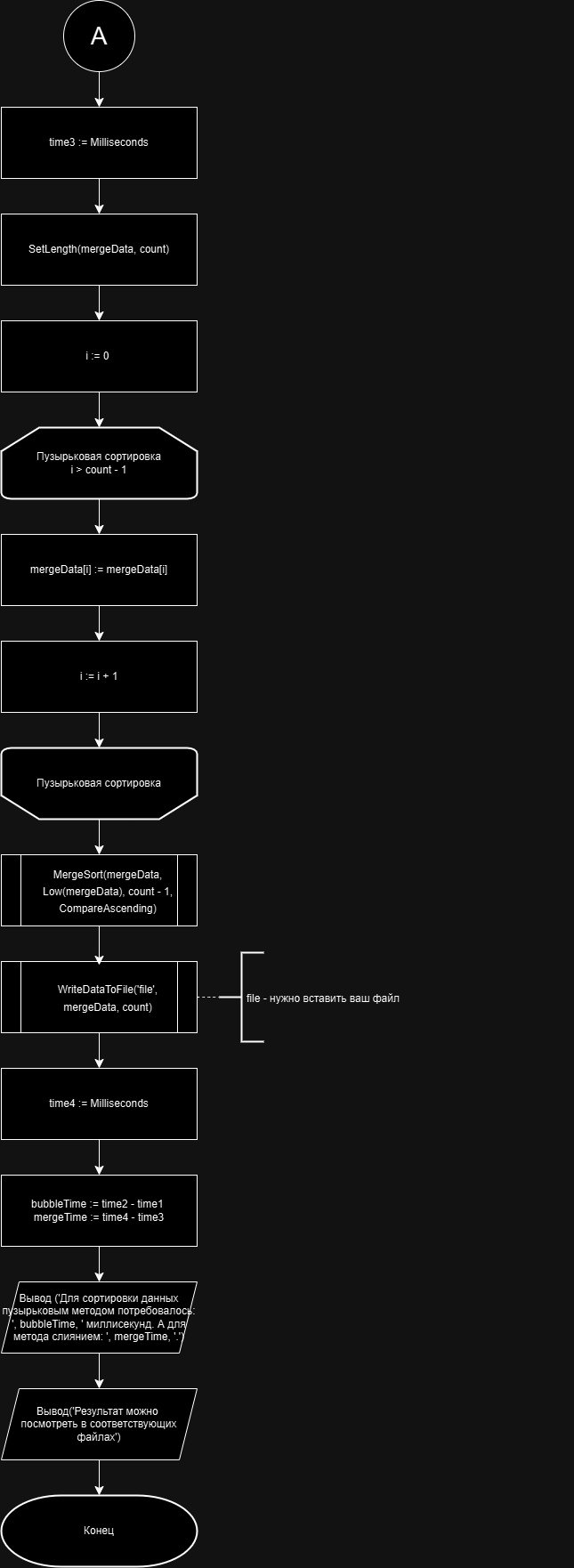


Рисунок 6.2 – схема алгоритма основной функции

**Код программы:**

procedure BubbleSort(var data: array of Integer; comparator: function(a, b: Integer): Integer);

var

i, j, temp: Integer;

begin

for i := Low(data) to High(data) do

for j := Low(data) to High(data) - i - 1 do

if comparator(data[j], data[j + 1]) > 0 then

begin

temp := data[j];

data[j] := data[j + 1];

data[j + 1] := temp;

end;

end;

procedure Merge(var data: array of Integer; left, mid, right: Integer; comparator: function(a, b: Integer): Integer);

var

i, j, k: Integer;

n1, n2: Integer;

L, R: array of Integer;

begin

n1 := mid - left + 1;

n2 := right - mid;

SetLength(L, n1);

SetLength(R, n2);

for i := 0 to n1 - 1 do

L[i] := data[left + i];

for j := 0 to n2 - 1 do

R[j] := data[mid + 1 + j];

i := 0;

j := 0;

k := left;

while (i < n1) and (j < n2) do

begin

if comparator(L[i], R[j]) <= 0 then

begin

data[k] := L[i];

Inc(i);

end

else

begin

data[k] := R[j];

Inc(j);

end;

Inc(k);

end;

while i < n1 do

begin

data[k] := L[i];

Inc(i);

Inc(k);

end;

while j < n2 do

begin

data[k] := R[j];

Inc(j);

Inc(k);

end;

end;

procedure MergeSort(var data: array of Integer; left, right: Integer; comparator: function(a, b: Integer): Integer);

var

mid: Integer;

begin

if left < right then

begin

mid := (left + right) div 2;

MergeSort(data, left, mid, comparator);

MergeSort(data, mid + 1, right, comparator);

Merge(data, left, mid, right, comparator);

end;

end;

function CompareAscending(a, b: Integer): Integer;

begin

Result := a - b;

end;

procedure ReadDataFromFile(const filename: string; var data: array of Integer; var count: Integer);

var

f: TextFile;

num: Integer;

begin

AssignFile(f, filename);

Reset(f);

count := 0;

while not Eof(f) do

begin

Read(f, num);

data[count] := num;

Inc(count);

end;

CloseFile(f);

end;

procedure WriteDataToFile(const filename: string; const data: array of Integer; count: Integer);

var

f: TextFile;

i: Integer;

begin

AssignFile(f, filename);

Rewrite(f);

for i := 0 to count - 1 do

WriteLn(f, data[i]);

CloseFile(f);

end;

var

inputData: array of Integer;

bubbleData: array of Integer;

mergeData: array of Integer;

count, i, time1, time2, time3, time4, bubbleTime, mergeTime: Integer;

begin

SetLength(inputData, 1000);

ReadDataFromFile('C:\Users\lolko\OneDrive\Рабочий стол\ДКР5.Ввод1.txt', inputData, count);

time1 := Milliseconds;

SetLength(bubbleData, count); // Устанавливаем размер для пузырьковой сортировки.

for i := 0 to count - 1 do

bubbleData[i] := inputData[i];

BubbleSort(bubbleData, CompareAscending);

WriteDataToFile('C:\Users\lolko\OneDrive\Рабочий стол\ДКР5.Вывод.Пузырьком.txt', bubbleData, count);

time2 := Milliseconds;

time3 := Milliseconds;

SetLength(mergeData, count); // Устанавливаем размер для сортировки слиянием.

for i := 0 to count - 1 do

mergeData[i] := inputData[i];

MergeSort(mergeData, Low(mergeData), count - 1, CompareAscending);

WriteDataToFile('C:\Users\lolko\OneDrive\Рабочий стол\ДКР5.Вывод.Слиянием.txt', mergeData, count);

time4 := Milliseconds;

bubbleTime := time2 - time1;

mergeTime := time4 - time3;

WriteLn('Для сортировки данных пузырьковым методом потребовалось: ', bubbleTime, ' миллисекунд. А для метода слиянием: ', mergeTime, '.');

WriteLn('Результат можно посмотреть в соответствующих файлах.');

end.

**Вывод:** данная работа помогла изучить и сравнить некоторые алгоритмы сортировок, в моем случае, пузырьковая сортировка и сортировка слиянием. Работа помогла более подробно изучить синтаксис ЯП PascalABC 3.9.