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 алгоритма сортировки»**

**ПО ДИСЦИПЛИНЕ «Основы алгоритмизации и программирования»**

Выполнил: студент учебной группы

ИСПк-101-51-00

Клыков Денис Борисович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

В отчете должны отображаться:

1. Цель работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма и ответы на вопросы
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод

Цель работы: получить базовые сведенья о наиболее известных алгоритмах сортировки изучить принципы работы с текстовыми файлами

1.Задание Реализовать сортировку данных с помощь алгоритма выбором

**program** SelectionSortFromFile;

**const**

InputFileName = 'дкр5.txt'; // Имя входного файла

OutputFileName = 'sorted\_дкр5.txt'; // Имя выходного файла

**var**

numbers: **array of** Int64; // Массив для хранения чисел типа Int64

i, j, n, minIndex: Integer;

temp: Int64;

inputFile, outputFile: TextFile;

**begin**

// Открываем входной файл для чтения

AssignFile(inputFile, InputFileName);

Reset(inputFile);

// Считываем количество чисел

n := 0;

**while not** Eof(inputFile) **do**

**begin**

SetLength(numbers, n + 1);

ReadLn(inputFile, numbers[n]);

Inc(n);

**end**;

CloseFile(inputFile);

// Реализация сортировки выбором

**for** i := 0 **to** n - 2 **do**

**begin**

minIndex := i;

**for** j := i + 1 **to** n - 1 **do**

**begin**

**if** numbers[j] < numbers[minIndex] **then**

minIndex := j;

**end**;

// Обмен значений

**if** minIndex <> i **then**

**begin**

temp := numbers[i];

numbers[i] := numbers[minIndex];

numbers[minIndex] := temp;

**end**;

**end**;

// Открываем выходной файл для записи

AssignFile(outputFile, OutputFileName);

Rewrite(outputFile);

// Записываем отсортированные данные в выходной файл

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

**begin**

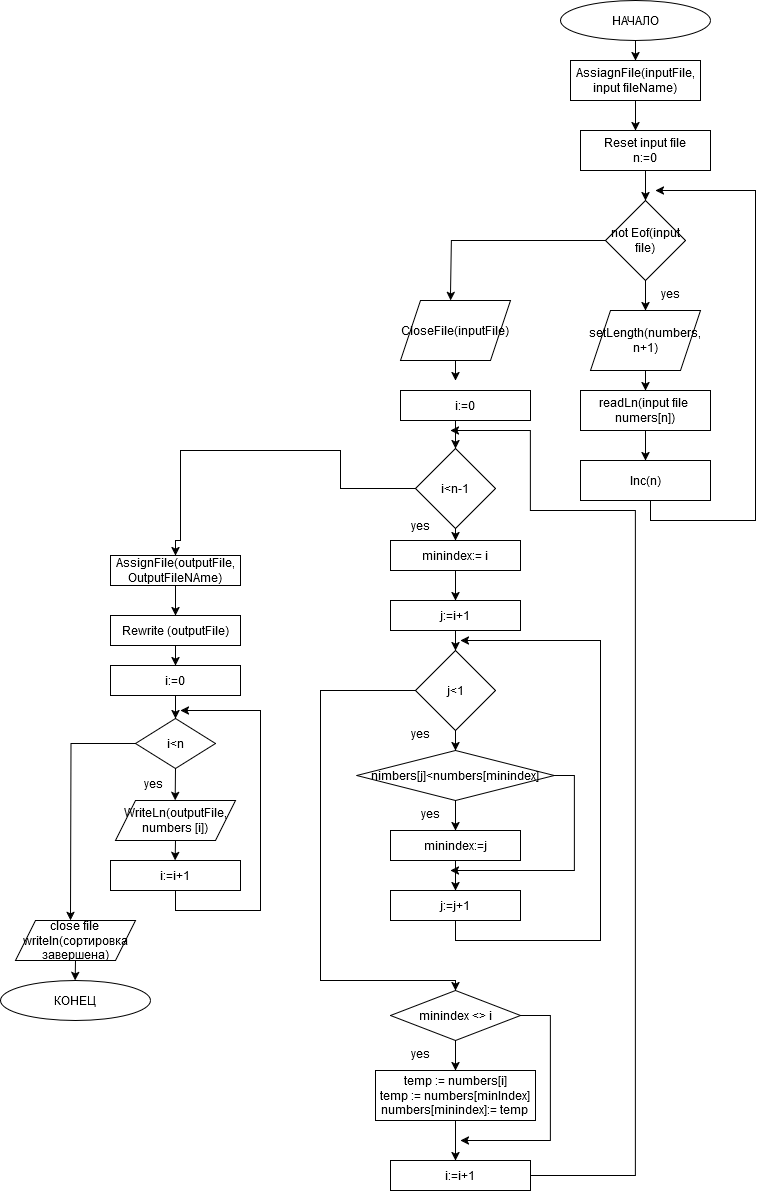
WriteLn(outputFile, numbers[i]);

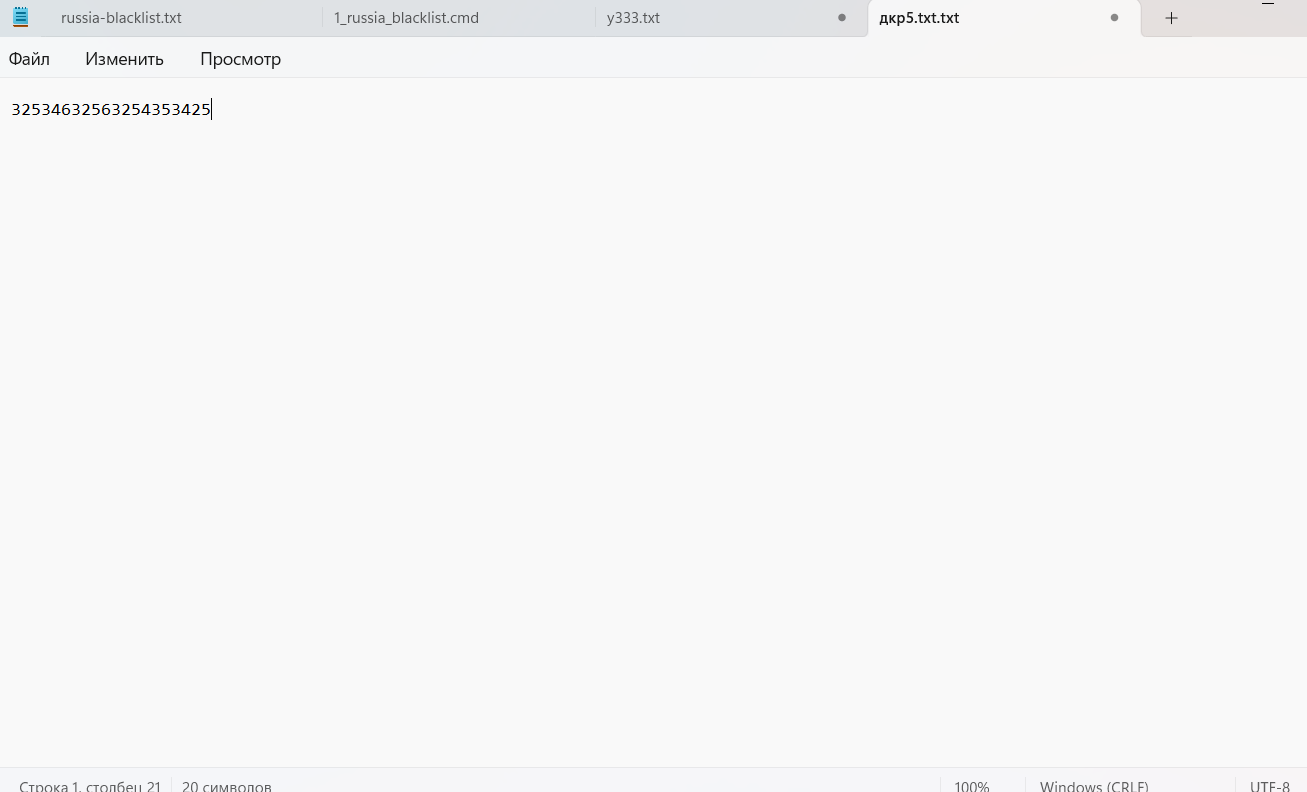
**end**;

CloseFile(outputFile);

WriteLn('Сортировка завершена. Результаты записаны в файл ', OutputFileName);

**end**.





2. Задание реализовать сортировку чисел с помощю поразрядного алгоритма в файл дкр5.txt

**program** RadixSortToFile;

**uses** System.IO;

**const**

FILENAME = 'дкр5.txt';

RADIX = 10; // Основание системы счисления (десятичная)

**type**

TIntegerArray = **array of** Integer;

**function** GetDigit(number, position: Integer): Integer;

**begin**

// Получаем цифру в заданной позиции (справа налево, начиная с 0)

Result := (number **div** Round(Power(RADIX, position))) **mod** RADIX;

**end**;

**procedure** RadixSort(**var** arr: TIntegerArray);

**var**

i, j, maxVal, numDigits, digit, pass: Integer;

buckets: **array of** TIntegerArray;

bucketCount: **array of** Integer;

**begin**

**if** Length(arr) <= 1 **then Exit**; // Если массив пуст или состоит из одного элемента - сортировать не нужно

// 1. Находим максимальный элемент, чтобы определить количество разрядов

maxVal := arr[0];

**for** i := 1 **to** Length(arr) - 1 **do**

**if** arr[i] > maxVal **then**

maxVal := arr[i];

// 2. Вычисляем количество разрядов в максимальном числе

numDigits := 0;

**while** maxVal > 0 **do**

**begin**

maxVal := maxVal **div** RADIX;

Inc(numDigits);

**end**;

// 3. Инициализируем массив "карманов" (buckets)

SetLength(buckets, RADIX);

SetLength(bucketCount, RADIX);

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

SetLength(buckets[i], 0);

// 4. Основной цикл сортировки по разрядам

**for** pass := 0 **to** numDigits - 1 **do**

**begin**

// a. Очищаем карманы

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

SetLength(buckets[i], 0);

// FillChar(bucketCount, SizeOf(bucketCount), 0); // Обнуляем счетчики

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

bucketCount[i] := 0; // Альтернативный способ обнуления

// b. Распределяем числа по карманам в зависимости от текущего разряда

**for** i := 0 **to** Length(arr) - 1 **do**

**begin**

digit := GetDigit(arr[i], pass);

SetLength(buckets[digit], bucketCount[digit]+1);

buckets[digit][bucketCount[digit]] := arr[i];

Inc(bucketCount[digit]);

**end**;

// c. Собираем числа обратно в исходный массив

j := 0;

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

**for var** k := 0 **to** Length(buckets[i]) - 1 **do**

**begin**

arr[j] := buckets[i][k];

Inc(j);

**end**;

**end**;

// Освобождаем память

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

SetLength(buckets[i], 0);

SetLength(buckets, 0);

SetLength(bucketCount, 0);

**end**;

**procedure** SaveArrayToFile(arr: TIntegerArray; filename: string);

**var**

i: Integer;

writer: TextFile;

**begin**

AssignFile(writer, filename);

Rewrite(writer);

**try**

**for** i := 0 **to** Length(arr) - 1 **do**

**begin**

Writeln(writer, arr[i]);

**end**;

**finally**

CloseFile(writer);

**end**;

**end**;

**function** LoadArrayFromFile(filename: string): TIntegerArray;

**var**

reader: TextFile;

line: string;

number: Integer;

arr: TIntegerArray;

i: Integer;

**begin**

SetLength(arr, 0);

AssignFile(reader, filename);

Reset(reader);

**try**

i := 0;

**while not** Eof(reader) **do**

**begin**

Readln(reader, line);

**if** TryStrToInt(line, number) **then**

**begin**

SetLength(arr, i + 1);

arr[i] := number;

Inc(i);

**end**;

**end**;

**finally**

CloseFile(reader);

**end**;

Result := arr;

**end**;

**var**

numbers: TIntegerArray;

**begin**

// Пример использования:

// 1. Загружаем числа из файла

numbers := LoadArrayFromFile(FILENAME);

// 2. Сортируем массив

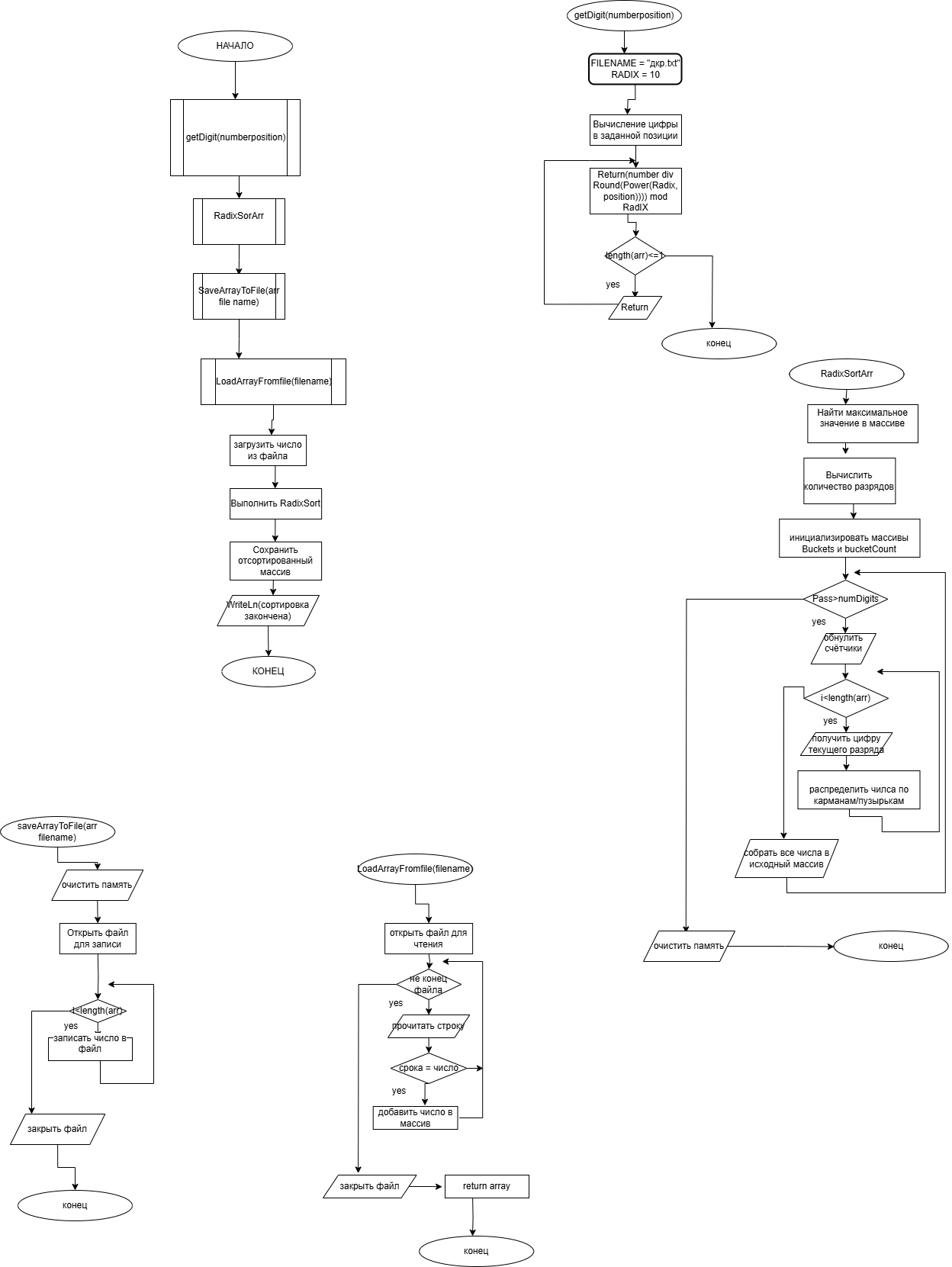
RadixSort(numbers);

// 3. Сохраняем отсортированный массив в файл

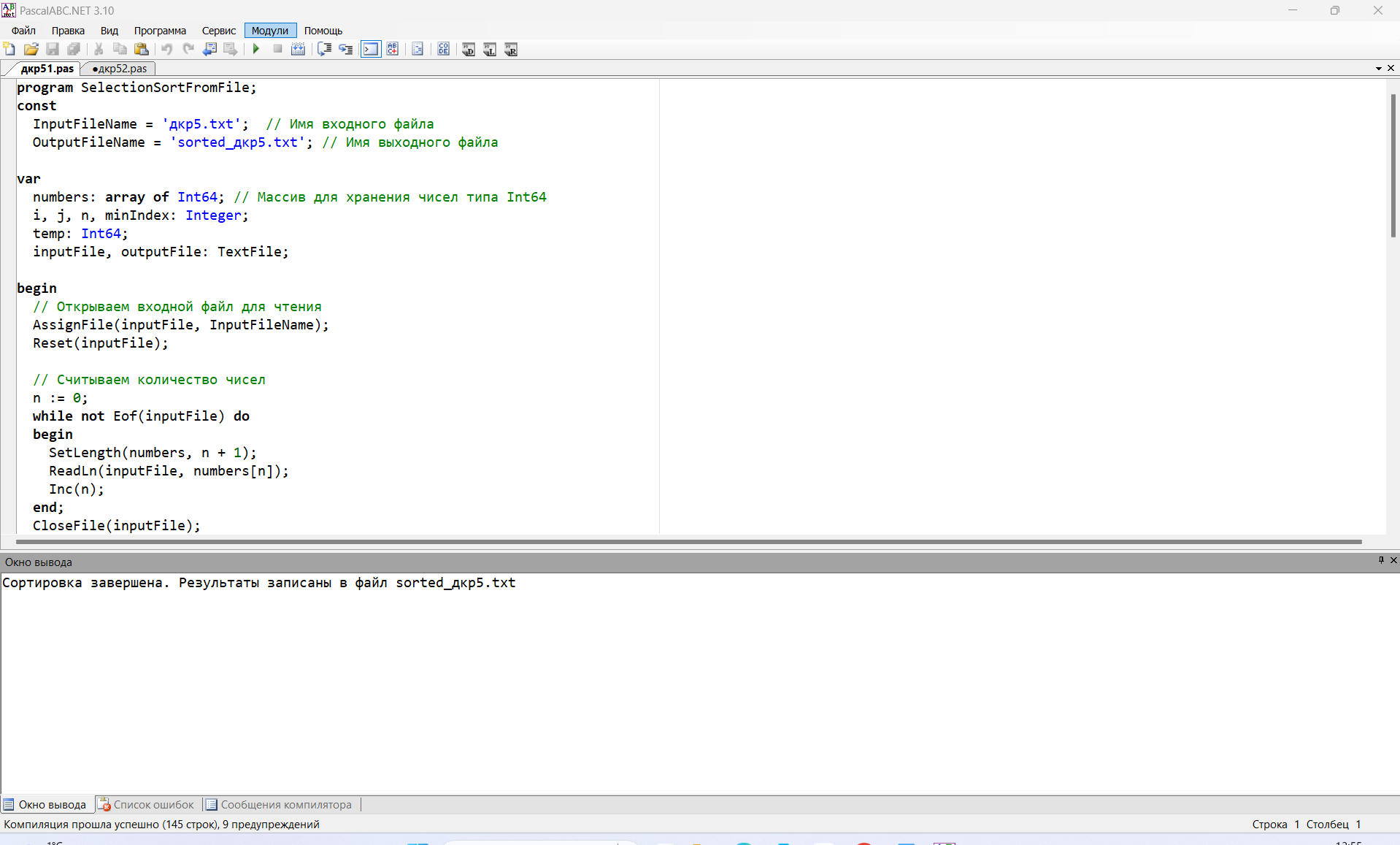
SaveArrayToFile(numbers, FILENAME);

Writeln('Числа отсортированы и сохранены в файл ', FILENAME);

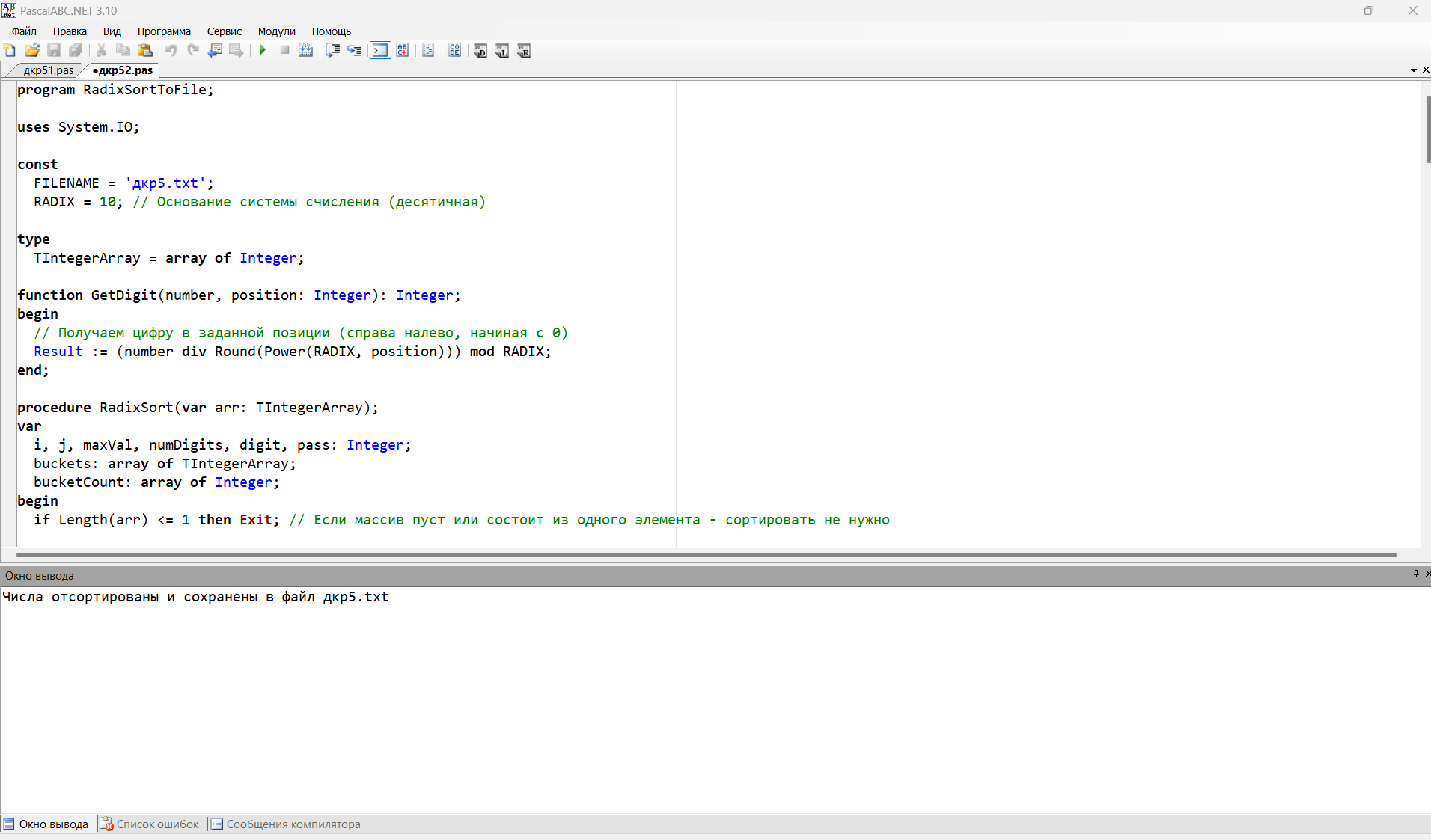
**end**.



Задание 1



Задание 2



Выводы:

Мы использовали процедуры для открытия файла, считывания данных и их сохранения в массив. Это позволяет нам загружать данные для сортировки из внешнего источника, что делает программу более гибкой и удобной.

После сортировки мы записывали отсортированные данные в новый текстовый файл. Это позволяет сохранять результаты работы программы для дальнейшего анализа или использования.

В ходе лабораторной работы мы получили базовые сведения о наиболее известных алгоритмах сортировки и изучили принципы работы с текстовыми файлами. Мы научились считывать данные из текстового файла, сортировать их с помощью различных алгоритмов и записывать отсортированные данные обратно в файл.