МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

**«Вятский государственный университет»**

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Исследование фракталов.

Отчет по лабораторной работе №7 дисциплины

«Программирование»

Выполнил студент группы ИВТб-11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Бурдукова А.А./

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Чистяков Г.А./

Киров 2019

**Цель работы:** получение навыков реализации алгоритмов с рекурсивными вычислениями, знакомство с фракталами.

**Задание:**

**1. Написать программу для визуализации фрактала «Кривая Госпера».**

**2. Предусмотреть возможности масштабирования, изменения глубины прорисовки и перемещения полученной фигуры.**

**3. Построение множества ломанных, образующих фрактал, должно осуществляться в модуле.**

**Схема алгоритмов:**

**Изображение выглядит как текст, карта

Автоматически созданное описание**

**Исходный текст программы:**

1. **Сортировка вставками:** 
   1. **По возрастанию:**

**type** func = **function**(a,b:longint) : boolean;

**var** a:**array** [1..1000000] **of** int64;

n,i:longint;

t, f : text;

{$F+}

**function** compare(a, b : longint) : boolean;

**begin**

**if** (a > b) **then** compare := true **else** compare := false;

**end**;

{$F-}

**procedure** Insert\_Sort(n : longint; comp : func);

**var** temp, i, numb : longint;

**begin**

**for** i := 2 **to** n **do begin**

numb := i;

temp := a[i];

**while** ((numb > 1) **and** (comp(a[numb - 1], temp))) **do begin**

a[numb] := a[numb - 1];

dec(numb);

**end**;

a[numb] := temp;

**end**;

**end**;

**begin**

assign(f,'input.txt');

reset(f);

readln(f, n);

**for** i :=1 **to** n **do** read(f, a[i]);

Insert\_Sort(n, compare);

assign(t, 'output.txt');

rewrite(t);

**for** i :=1 **to** n **do** write(t, a[i],' ');

close(t);

close(f);

**end**.

**По убыванию:**

**type** func = **function**(a,b:longint) : boolean;

**var** a:**array** [1..1000000] **of** longint;

n,i:longint;

t, f : text;

{$F+}

**function** compare(a, b : longint) : boolean;

**begin**

**if** (a < b) **then** compare := true **else** compare := false;

**end**;

{$F-}

**procedure** Insert\_Sort(n : longint; comp : func);

**var** temp, i, numb : longint;

**begin**

**for** i := 2 **to** n **do begin**

numb := i;

temp := a[i];

**while** ((numb > 1) **and** (comp(a[numb - 1], temp))) **do begin**

a[numb] := a[numb - 1];

dec(numb);

**end**;

a[numb] := temp;

**end**;

**end**;

**begin**

assign(f,'input.txt');

reset(f);

readln(f, n);

**for** i :=1 **to** n **do** read(f, a[i]);

Insert\_Sort(n, compare);

assign(t, 'output.txt');

rewrite(t);

**for** i :=1 **to** n **do** write(t, a[i],' ');

close(t);

close(f);

**end**.

1. **Быстрая сортировка:**

**2.1) По возрастанию:**

**type** func = **function**(a,b:int64) : boolean;

**var** a:**array** [1..100000000] **of** int64;

n,i:longint;

t, f : text;

{$F+}

**function** compare(a, b : int64) : boolean;

**begin**

**if** (a > b) **then** compare := true **else** compare := false;

**end**;

{$F-}

**procedure** Quick\_Sort(L, R : int64; comp : func);

**var** i, j, x, t : int64;

**begin**

i := L;

j := R;

x := a[(i + j)**div** 2];

**repeat**

**while** (**not** comp(a[i], x)) **and** (a[i] <> x) **do** inc(i);

**while** (comp(a[j], x)) **do** dec(j);

**if** (i <= j) **then begin**

t := a[i];

a[i] := a[j];

a[j] := t;

inc(i);

dec(j);

**end**;

**until** (i > j);

**if** (i < R) **then** Quick\_Sort(i, R, comp);

**if** (j > L) **then** Quick\_Sort(L, j, comp);

**end**;

**begin**

assign(f,'input.txt');

reset(f);

readln(f, n);

**for** i :=1 **to** n **do** read(f, a[i]);

Quick\_Sort(1, n, compare);

assign(t, 'output.txt');

rewrite(t);

**for** i :=1 **to** n **do** write(t, a[i],' ');

close(t);

close(f);

**end**.

**2.2) По убыванию:**

**type** func = **function**(a,b:int64) : boolean;

**var** a:**array** [1..100000000] **of** int64;

n,i:longint;

t, f : text;

{$F+}

**function** compare(a, b : int64) : boolean;

**begin**

**if** (a < b) **then** compare := true **else** compare := false;

**end**;

{$F-}

**procedure** Quick\_Sort(L, R : int64; comp : func);

**var** i, j, x, t : int64;

**begin**

i := L;

j := R;

x := a[(i + j)**div** 2];

**repeat**

**while** (**not** comp(a[i], x)) **and** (a[i] <> x) **do** inc(i);

**while** (comp(a[j], x)) **do** dec(j);

**if** (i <= j) **then begin**

t := a[i];

a[i] := a[j];

a[j] := t;

inc(i);

dec(j);

**end**;

**until** (i > j);

**if** (i < R) **then** Quick\_Sort(i, R, comp);

**if** (j > L) **then** Quick\_Sort(L, j, comp);

**end**;

**begin**

assign(f,'input.txt');

reset(f);

readln(f, n);

**for** i :=1 **to** n **do** read(f, a[i]);

Quick\_Sort(1, n, compare);

assign(t, 'output.txt');

rewrite(t);

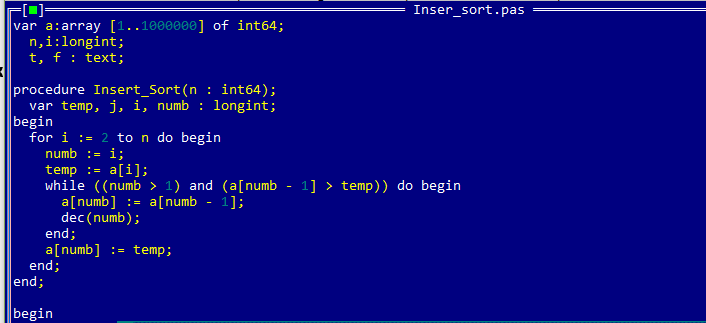
**for** i :=1 **to** n **do** write(t, a[i],' ');

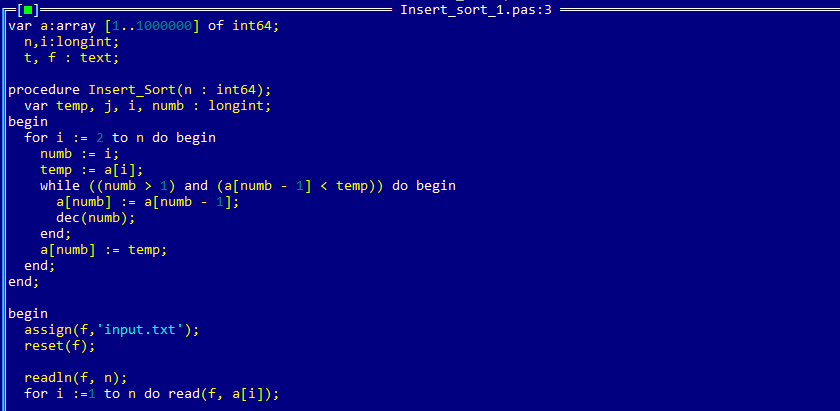
close(t);

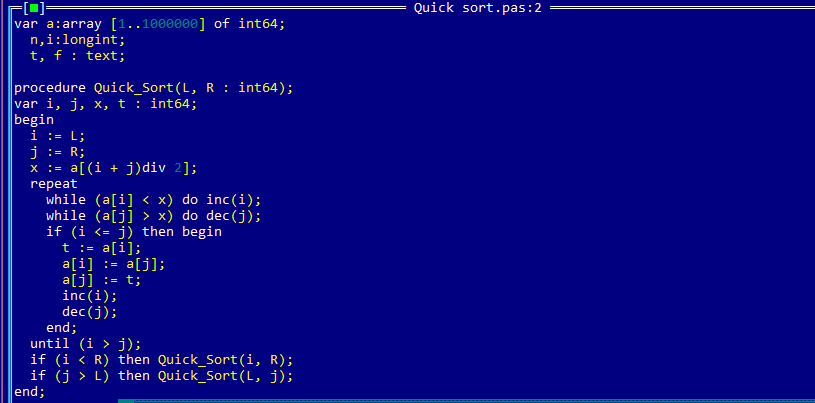
close(f);

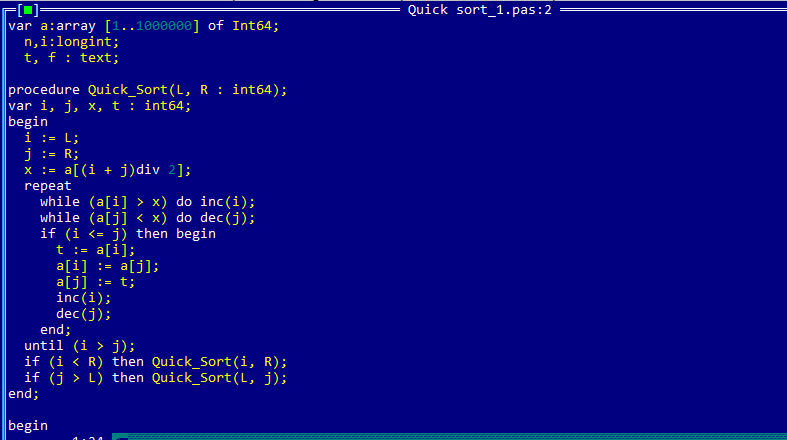
**end**.

**Скриншоты выполнения программы**

****

****

****

****

**Вывод**

В ходе данной лабораторной работы были получены знания для работы с текстовыми файлами. Также были получены базовые сведения об основных алгоритмах сортировки. Например, были изучены такие алгоритмы сортировки как сортировка “вставками” и быстрая сортировка. Каждая сортировка была проверена на определённом количестве входных данных, в частности на количестве элементов массива. При большом количестве чисел быстрая сортировка показывает намного лучшие результаты. Например, при 100 000 рандомных чисел быстрая сортировка выполняется за 300 миллисекунд, а “вставками”-22 секунды. При миллионе элементов: быстрая-1,5 секунды, “вставками”-почти час. Входные данных находились в текстовом файле input.txt, исходный отсортированный массив находится в выходном текстовом файле output.txt.