МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт №8: «Информационные технологии и прикладная математика»

Кафедра 806: «Вычислительная математика и программирование»

**КУРСОВАЯ РАБОТА №9**

По курсу «Вычислительные системы»

**II** семестр

Тема:

**«Сортировка и поиск»**

**Группа:** М80-106Б-22

**Студент:** Ларченко А.О

**Преподаватели:** Дубинин А.В.

**Оценка: \_\_\_\_\_\_\_\_\_ Подпись: \_\_\_\_\_\_\_\_**

**Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Москва, 2023

# Содержание

[**Содержание 1**](#_5bjxas8qpzef)

[**Введение 2**](#_uyifey50bwp5)

[**Программа 3**](#_9jo81si2g25l)

[Задание 3](#_bfqk89p80hab)

[Алгоритм 3](#_parrxz3ovl3i)

[Сценарий 3](#_esz98a66xnsc)

[Описание файлов и функций 3](#_ivl8bapzslky)

[Протокол 3](#_l8vwyeg5fihw)

[**Тестирование 4**](#_7e16my4hesaw)

[**Описание файлов и функций 5**](#_ft3m7yk8cnu8)

[**Заключение 6**](#_be10rxpujcie)

# Введение

9 Курсовой проект направлен на закрепление знаний о различных видах сортировок, с которыми мы познакомились во время семестра, а также напоминает о существование такой прекрасной вещи как бинарный поиск.

# Программа

## Задание

Составить программу на языке Си с использованием процедур и

функций для сортировки ASCII - графики заданным методом и двоичного поиска

по ключу в таблице.

*Тип ключа:* строковый (5 байт).

*Метод сортировки*: шейкер-сортировка.

*Способ хранения:* раздельно.

## Немного теории

***Шейкерная сортировка*** является модифицированной версией пузырьковой сортировки. Основная идея заключается в том, что перебирая элементы также как в пузырьке( при необходимости два соседних элемента меняются местами), шейкерная сортировка двигается в обе стороны, что позволяет после каждого прохода в одном направлении отсеивать самый правый (максимальный элемент, при движении слева направо) и самый левый (минимальный элемент, при движении справа налево) элемент, который вплыл за итерацию. Сложность в худшем случае - О(n^2), в лучшем О(n)(тривиальный случай).

***Бинарный поиск*** (дихотомия или метод деления пополам) - алгоритм поиска элемента в *отсортированном* массиве. Работает по принципу деления массива на 2 части, а затем сравнения серединного элемента и искомого числа, если искомое число больше, то берется правая часть массива, в противном случае левая. Эта итерация повторяется пока не будет найден искомый элемент или не нарушится равенство(l<=r), что будет означать, что такого элемента нет. Сложность - О(log2(n)).

## Алгоритм

Программа получает на вход 2 файла - один содержит ключи, второй ASCII-графику. Первым делом программа считывает 2 файла и делает из них 1 структуру. Также вместе с 2 файлами программе на вход передаётся ключ( в каком виде будет представлена графика: в обычном, перевернутом или рандомном). Вначале массив данных сортируется по ключам, потом в нем осуществляется бинарный поиск, введенного с клавиатуры элемента.

Итоговая сложность - О(n^2), где n- строк.

## Описание файлов и функций

***data.h*** - содержит описание структуры и операции над ней

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#ifndef DATA\_H

#define DATA\_H

typedef struct data{

char key[7];

char str[100];

} data;

int line\_counter(char\* arg);

void text\_print(data origin[], int size);

void merge(FILE\* lines, FILE\* keys, data arr[]);

void reverse\_data(FILE\* lines, FILE\* keys, data arr[], int size);

void random\_data(FILE\* lines, FILE\* keys, data arr[], int size);

#endif

***data.c***

#include "data.h"

int line\_counter(char \*arg){

int cnt=0;

FILE \*tmp=fopen(arg, "r");

while (!feof(tmp)){

fscanf(tmp, "%\*[^\n]%\*c");

cnt++;

}

fclose(tmp);

return cnt;

}

void text\_print(data arr[], int size){

printf("Keys |");

printf(" Strings\n");

for(int i=1; i<size;++i){

printf("%s | %s\n", arr[i].key, arr[i].str);

}

}

void merge(FILE\* lines, FILE\* keys, data arr[]){

fseek(lines, 0, SEEK\_SET);

fseek(keys, 0, SEEK\_SET);

data tmp;

int i=1; //!!!

while (fscanf(lines, "%s", tmp.str)!=EOF && fscanf(keys, "%s", tmp.key)!=EOF){

arr[i]=tmp;

i++;

}

}

void reverse\_data(FILE\* lines, FILE\* keys, data arr[], int size){

fseek(lines, 0, SEEK\_SET);

fseek(keys, 0, SEEK\_SET);

data tmp;

struct data copy[size];

merge(lines, keys, copy);

for(int i=0; i<size; ++i){

arr[i]=copy[size-i];

}

}

void random\_data(FILE\* lines, FILE\* keys, data arr[], int size){

fseek(lines, 0, SEEK\_SET);

fseek(keys, 0, SEEK\_SET);

data tmp;

struct data copy[size];

merge(lines, keys, copy);

int rd\_count[size];

int rd\_order[size];

rd\_order[0]=-1;

rd\_count[0]=1;

for(int i=1; i<size;++i){

rd\_count[i]=0;

}

srand(time(NULL));

for(int i=1; i<size;++i){

int r=1 +rand()%size;

while(1){

if(rd\_count[r]==0){

rd\_count[r]=1;

rd\_order[i]=r;

break;

} else{

r=1 +rand()%size;

}

}

}

// for(int i=1; i<size;++i){

// printf("%d ", rd\_order[i]);

// }

// printf("\n");

// for(int i=0; i<size;++i){

// printf("%d ", rd\_count[i]);

// }

for(int i=1; i<size; ++i){

arr[i]=copy[rd\_order[i]];

}

}

***bin\_find*** - файл с бинарным поиском

#include "bin\_find.h"

int bin\_search(data arr[], char request[7], int size){

int l=1, r=size-1;

int mid;

int answ=-1;

while(l<=r){

mid=(l+r)/2;

int cmp=strcmp(arr[mid].key, request);

if(cmp==0){

answ=mid;

}

if(cmp<0){

l=mid+1;

} else{

r=mid-1;;

}

}

return answ;

}

sorting.c - содержит программы шейкерной сортировки

#include "sorting.h"

typedef struct datas{

char key[3];

char str[2];

}datas;

static void swap(int a, int b, data arr[]){

data tmp;

tmp=arr[a];

arr[a]=arr[b];

arr[b]=tmp;

}

void shaker\_sort(data arr[], int size){

int l=1, r=size-1;

int status=1;

while(l<r && status==1){

status=0;

for(int i=l; i<r; ++i){

if(strcmp(arr[i].key, arr[i+1].key)>0){

swap(i, i+1, arr);

status=1;

}

}

r--;

for(int i=r; i>l; --i){

if(strcmp(arr[i-1].key, arr[i].key)>0){

swap(i, i-1, arr);

status=1;

}

}

l++;

}

}

***term.c*** - отвечает за взаимодействие с пользователем и выполнением всех команд

#include "data.h"

#include "sorting.h"

#include "bin\_find.h"

//Запуск: первый - строки, 2 - ключи 3 - программа

int main(int argc, char\* argv[]){

if (argc != 4){

fprintf(stderr, "Error! Uncorrect input");

return 1;

}

FILE\* lines = fopen(argv[1], "r");

FILE\* key = fopen(argv[2], "r");

if (lines==NULL){

fprintf(stderr, "Error! There is no file with strings");

return 2;

}

if (key==NULL){

fprintf(stderr, "Error! There is no file with keys");

return 3;

}

int size=line\_counter(argv[1]);

if(size<=1){

fprintf(stderr, "Error! File is empty");

return 4;

}

struct data origin[size+1];

if(strcmp( argv[3], "n")==0){ //normal

// struct data origin[size+1];

merge(lines, key, origin);

text\_print(origin, size);

printf("\n");

shaker\_sort(origin,size-1);

text\_print(origin, size);

} else if(strcmp( argv[3], "r")==0){ // reverse

// struct data reverse[size];

reverse\_data(lines, key, origin, size);

text\_print(origin, size);

printf("\n");

shaker\_sort(origin, size);

text\_print(origin, size);

}else if(strcmp( argv[3], "c")==0){ // chaos

// struct data random[size+1];

random\_data(lines, key, origin, size);

text\_print(origin, size);

printf("\n");

shaker\_sort(origin, size);

text\_print(origin, size);

} else{

fprintf(stderr, "Error! uncorrect command");

return 5;

}

printf("\n\n Enter key that you want to find\n");

char request[90];

int c;

c=getchar();

while(c!=EOF){

ungetc(c, stdin);

scanf("%s", request);

printf("\nYour key is '%s' \n", request);

int res\_search=bin\_search(origin, request, size+1);

if(res\_search!=-1){

printf("The result of the search is '%s', string %s\n", origin[res\_search].key, origin[res\_search].str);

}else{

printf("There is no '%s' in this file\n", request);

}

printf("\n\n Enter key that you want to find\n");

c=getchar();

}

scanf("%s", request);

// printf("%d\n",sizeof(request) );

// if(sizeof(request)>7){

// printf("It's to long request");

// }

// printf("\nYour key is '%s' \n", request);

// int res\_search=bin\_search(origin, request, size+1);

// if(res\_search!=-1){

// printf("The result of the search is '%s', string %s\n", origin[res\_search].key, origin[res\_search].str);

// }else{

// printf("There is no '%s' in this file\n", request);

// }

// char trash[200];

// scanf("%s", trash);

}

# Тестирование

Первым делом соберем наши файлы.

arsenii@LarchCompu:~/Documents/prog/projects/ones/Cprog/kp9$ make

gcc -c bin\_find.c -o bin\_find.o

gcc -c data.c -o data.o

gcc -c sorting.c -o sorting.o

gcc -c term.c -o term.o

gcc bin\_find.o data.o sorting.o term.o -o AL\_kp9

Для тестирования программы я буду использовать 2 пары(файл с ключами+файл со значениями) файлов(с 1 набором значений, с тем же набором значений, но дублированный).

Запустив программу с 3 разными ключами, я убедился, что все работает исправно.

Также протестировал программу на одинаковых ключах, программа удовлетворяет требования задания( находится самое первое вхождение).

# 

# 

# Заключение

Благодаря этой работе я закрепил свои знания в области сортировки, а также вспомнил бинарный поиск. Считаю, что с заданием справился успешно.