**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **Динамические структуры данных.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2017

Цель:

Требуется написать программу, которая, будучи запущенной в корневой дирректории, выведет строки из файлов всех поддиректорий в порядке возрастания числа, с которого строки начинаются.

Задание:

Дана некоторая корневая дирректория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида <filename>.txt

В каждом текстовом файле хранится одна строка, начинающаяся с числа вида:

<число><пробел><латинские буквы, цифры, знаки препинания> ("124 string example!")

Ход работы:

* Прохождение дерева папок для доступа к файлам

void pathSearch(char\* path, char\*\* origin, int\* str\_counter) {

DIR \*dir = opendir(path);

if (dir != NULL) {

struct dirent\* s\_dir=(struct dirent\*)malloc(sizeof(struct dirent));

s\_dir =readdir(dir);

while(s\_dir!=NULL) {

int close = strlen(path);

strcat(path, "/");

strcat(path, s\_dir->d\_name);

if(s\_dir->d\_type == DT\_REG &&(strcmp(s\_dir->d\_name, "a.out")!=0)&&(strcmp(s\_dir->d\_name, "main.c")!=0)) {

if(scanData(path)!=NULL) {

origin[\*str\_counter]=(char\*)malloc(1000\*sizeof(char));

strcpy(origin[\*str\_counter], scanData(path));

(\*str\_counter)++;

}

}

if (s\_dir->d\_type == DT\_DIR && strcmp(s\_dir->d\_name, ".") != 0 && strcmp(s\_dir->d\_name, "..") != 0) {

pathSearch(path,origin,str\_counter);

}

path[close] = '\0';

s\_dir = readdir(dir);

}

closedir(dir);

} else return;

}

* Считывание текста из читаемых файлов

char\* scanData(char\* path) {

FILE\* file=fopen(path, "r");

if(file==NULL){

//printf("Corrupted file\n");

fclose(file);

return NULL;

}

fseek(file, 0, SEEK\_END);

int file\_size=ftell(file);

fseek(file, 0, SEEK\_SET);

if(file\_size==0){

//printf("Empty file\n");

fclose(file);

return NULL;

}

char\* data=(char\*)malloc(1000\*sizeof(char));

fgets(data, 1000, file);

fclose(file);

return(data);

}

* Сортировка массива строк

int compare(const void\* a, const void\* b) {

return atoi(\*(char\*\*)a) - atoi(\*(char\*\*)b);

}

qsort(origin, str\_counter, sizeof(char\*), compare);

Вывод:

В данной лабораторной работе мы применили и закрепили ранее изученные правила использования рекурсивных функций, а также функций сортировки.

Работа с репозиторием

1. Копируем репозиторий группы 6304

Git clone https://github.com/moevm/pr1-2016-6304.git

1. От ветки master создаем локальную ветку с названием работы и именем:

Git checkout –b “”

1. В папку с клонированным репозиторием скопируем файлы с работой с помощью команды cp –r
2. Добавляем папку с файлом на ветку командой

git add

1. Коммитим изменения:

Git commit –m “”

1. Отправляем изменения в репозиторий командой

Git push origin

# На сайте github создаем pull request в ветку master.

ПРИЛОЖЕНИЕ 1

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

//#include <unistd.h>

#include <sys/types.h>

char\* scanData(char\* path) {

FILE\* file=fopen(path, "r");

if(file==NULL){

//printf("Corrupted file\n");

fclose(file);

return NULL;

}

fseek(file, 0, SEEK\_END);

int file\_size=ftell(file);

fseek(file, 0, SEEK\_SET);

if(file\_size==0){

//printf("Empty file\n");

fclose(file);

return NULL;

}

char\* data=(char\*)malloc(1000\*sizeof(char));

fgets(data, 1000, file);

fclose(file);

return(data);

}

void pathSearch(char\* path, char\*\* origin, int\* str\_counter) {

DIR \*dir = opendir(path);

if (dir != NULL) {

struct dirent\* s\_dir=(struct dirent\*)malloc(sizeof(struct dirent));

s\_dir =readdir(dir);

while(s\_dir!=NULL) {

int close = strlen(path);

strcat(path, "/");

strcat(path, s\_dir->d\_name);

if(s\_dir->d\_type == DT\_REG &&(strcmp(s\_dir->d\_name, "a.out")!=0)&&(strcmp(s\_dir->d\_name, "main.c")!=0)) {

if(scanData(path)!=NULL) {

origin[\*str\_counter]=(char\*)malloc(1000\*sizeof(char));

strcpy(origin[\*str\_counter], scanData(path));

(\*str\_counter)++;

}

}

if (s\_dir->d\_type == DT\_DIR && strcmp(s\_dir->d\_name, ".") != 0 && strcmp(s\_dir->d\_name, "..") != 0) {

pathSearch(path,origin,str\_counter);

}

path[close] = '\0';

s\_dir = readdir(dir);

}

closedir(dir);

} else return;

}

int compare(const void\* a, const void\* b) {

return atoi(\*(char\*\*)a) - atoi(\*(char\*\*)b);

}

int main() {

char\*\* origin=(char\*\*)malloc(1000\*sizeof(char\*));

char path[1000]=".";

int i=0;

int str\_counter=0;

pathSearch(path, origin, &str\_counter);

qsort(origin, str\_counter, sizeof(char\*), compare);

for (i=0; i<str\_counter; i++) {

printf("%s\n", origin[i]);

}

for (i=0; i<str\_counter; i++) {

free(origin[i]);

}

free(origin);

return 0;

}