**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

Тема: **Строки. Рекурсия, циклы, обход дерева.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Тимофеев А.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

# Оглавление

Оглавление

[Цель 3](#_Toc484079853)

[Задание 3](#_Toc484079855)

[Содержание 4](#_Toc484079859)

[1.Функция открытия файла 4](#_Toc484079861)

[2.Обход каталога 4](#_Toc484079880)

[3.Компаратор 5](#_Toc484079901)

[Код программы 5](#_Toc484079905)

[Работа с репозиторием 8](#_Toc484079971)

[Вывод 8](#_Toc484079972)

# Цель

Написание 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!").

# Содержание

# В ходе выполнения данной работы были использованы следующие функции:

# Функция открытия файла

# char\* txtOpen(char\* txtName){ //функция открытия файла

# FILE\* txt = fopen(txtName, "r");

# if (txt == NULL){ //проверяем можно ли открыть файл

# printf("ERROR\n");

# return NULL;

# }

# fseek(txt,0,SEEK\_END); //перемещаем указатель на последнюю позицию потока

# int txtSize = ftell(txt); //узнаем размер файла

# fseek(txt,0,SEEK\_SET); //перемещаем указатель обратно на первую позицию потока

# if(txtSize == 0){ //если размер файла равен нулю, закрываем его

# fclose(txt);

# return NULL;

# }

# char\* txtStr = (char\*)malloc(sizeof(char)\*10000); //в противном случае выделяем память для строки

# fgets(txtStr, txtSize\*sizeof(char),txt); //считываем строку из файла

# fclose(txt);

# return txtStr; //возвращаем считанную строку

# }

# Обход каталога

# void list\_dir(const char\* dirName, char\*\* strsTxt, int\* len){ //обход каталога

# char current\_path[10000]; //выделяем память для строки, где будет путь к объекту

# strcpy(current\_path, dirName);

# DIR \*current\_dir = opendir(current\_path);

# if (current\_dir == NULL) return; //проверяем корректно ли открылся каталог

# struct dirent\* current\_dir\_file =readdir(current\_dir); //считываем структуру с информацией об объекте в первом каталоге

# while(current\_dir\_file){ //пока в текущей дирректории есть объекты

# int path\_len = strlen(current\_path);

# strcat(current\_path, "/");

# strcat(current\_path, current\_dir\_file->d\_name); //добавляем имя объекта к строке

# if(current\_dir\_file->d\_type == DT\_REG && strstr(current\_dir\_file->d\_name, ".txt") != NULL){ //является ли объект нужным файлом?

# if((strsTxt[\*len] = (char\*)txtOpen(current\_path)) != NULL)

# (\*len)++; //если файл не пустой и не поврежден, возвращаем указатель на первую строку в нем

# }

# if (current\_dir\_file->d\_type == DT\_DIR && strcmp(".", current\_dir\_file->d\_name) != 0 && strcmp("..",current\_dir\_file->d\_name) != 0) list\_dir(current\_path,strsTxt,len); //если объект - дирректория(не родительская и не текущая), вызываем(рекурсивно) функцию обхода каталога

# current\_path[path\_len] = '\0'; //возвращаемся к первоначальному положению

# current\_dir\_file = readdir(current\_dir); //считываем следующий объект

# }

# closedir(current\_dir); //закрываем дирректорию после обхода всех ее объектов

# }

# Компаратор

# int compare(const void\* a, const void\* b){ //компаратор для qsort-a

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

# }

# Код программы

# #include <stdio.h>

# #include <stdlib.h>

# #include <string.h>

# #include <sys/types.h>

# #include <dirent.h>

# char\* txtOpen(char\* txtName){ //функция открытия файла

# FILE\* txt = fopen(txtName, "r");

# if (txt == NULL){ //проверяем можно ли открыть файл

# printf("ERROR\n");

# return NULL;

# }

# fseek(txt,0,SEEK\_END); //перемещаем указатель на последнюю позицию потока

# int txtSize = ftell(txt); //узнаем размер файла

# fseek(txt,0,SEEK\_SET); //перемещаем указатель обратно на первую позицию потока

# if(txtSize == 0){ //если размер файла равен нулю, закрываем его

# fclose(txt);

# return NULL;

# }

# char\* txtStr = (char\*)malloc(sizeof(char)\*10000); //в противном случае выделяем память для строки

# fgets(txtStr, txtSize\*sizeof(char),txt); //считываем строку из файла

# fclose(txt);

# return txtStr; //возвращаем считанную строку

# }

# void list\_dir(const char\* dirName, char\*\* strsTxt, int\* len){ //обход каталога

# char current\_path[10000]; //выделяем память для строки, где будет путь к объекту

# strcpy(current\_path, dirName);

# DIR \*current\_dir = opendir(current\_path);

# if (current\_dir == NULL) return; //проверяем корректно ли открылся каталог

# struct dirent\* current\_dir\_file =readdir(current\_dir); //считываем структуру с информацией об объекте в первом каталоге

# while(current\_dir\_file){ //пока в текущей дирректории есть объекты

# int path\_len = strlen(current\_path);

# strcat(current\_path, "/");

# strcat(current\_path, current\_dir\_file->d\_name); //добавляем имя объекта к строке

# if(current\_dir\_file->d\_type == DT\_REG && strstr(current\_dir\_file->d\_name, ".txt") != NULL){ //является ли объект нужным файлом?

# if((strsTxt[\*len] = (char\*)txtOpen(current\_path)) != NULL)

# (\*len)++; //если файл не пустой и не поврежден, возвращаем указатель на первую строку в нем

# }

# if (current\_dir\_file->d\_type == DT\_DIR && strcmp(".", current\_dir\_file->d\_name) != 0 && strcmp("..",current\_dir\_file->d\_name) != 0) list\_dir(current\_path,strsTxt,len); //если объект - дирректория(не родительская и не текущая), вызываем(рекурсивно) функцию обхода каталога

# current\_path[path\_len] = '\0'; //возвращаемся к первоначальному положению

# current\_dir\_file = readdir(current\_dir); //считываем следующий объект

# }

# closedir(current\_dir); //закрываем дирректорию после обхода всех ее объектов

# }

# int compare(const void\* a, const void\* b){ //компаратор для qsort-a

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

# }

# int main(){

# char\*\* strs = (char\*\*)malloc(sizeof(char\*)\*1000); //выделяем память под массив указателей на строки

# int len = 0;

# int i=0;

# char str[1000];

# FILE\* fo;

# fo=fopen("fo","w"); //открываем файл для записи

# list\_dir(".",strs,&len); //заполняем массив строками из файлов

# qsort(strs, len, sizeof(char\*), compare); //сортируем строки

# for (i=0;i<len;i++) fprintf(fo,"%s\n", strs[i]); //выводим отсортированные строки

# fclose(fo); //закрываем

# fo=fopen("fo","r"); //открываем для чтения

# while (fgets(str, 1000, fo))

# printf("%s", str);

# fclose(fo); //закрываем

# for (i=0;i<len;i++) free(strs[i]); //освобождаем память

# free(strs);

# return 0;

# }

# 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 “[TIMOFEE\_ANDREY\_LAB3\_SEM2](https://github.com/moevm/pr1-2016-6304/tree/TIMOFEE_ANDREY_LAB3_SEM2) ”

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 Kurswork

1. Коммитим изменения:

Git commit –m “#Zalil lab3”

1. Отправляем изменения в репозиторий командой

Git push origin [TIMOFEE\_ANDREY\_LAB3\_SEM2](https://github.com/moevm/pr1-2016-6304/tree/TIMOFEE_ANDREY_LAB3_SEM2) На сайте github создаем pull request в ветку master.

# Вывод

В ходе выполнения данной работы было закреплено использование рекурсивной функции для обхода дерева.