**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 |  | Зубов К.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

# Оглавление

[Оглавление 2](#_Toc484819566)

[Цель 2](#_Toc484819568)

[Задание 2](#_Toc484819569)

[Содержание 3](#_Toc484819570)

[1. Функция открытия файла 3](#_Toc484819571)

[2. Обход каталога 3](#_Toc484819572)

[3. Компаратор для функции qsort 4](#_Toc484819573)

[Вывод 5](#_Toc484819574)

[Приложение 5](#_Toc484819575)

Цель

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

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:

1. Функция открытия файла

Функция, которая открывает txt файл и возвращает указатель на первую его строку (если она есть).

В данной функции происходит открытие файла, определение размера файла для корректного выделения памяти, выделение памяти под строку. Первая строка txt файла считывается в выделенную память.Файл закрывается, указатель на строку возвращается.

char\* read\_file(char\* file\_name)

{

FILE\* file = fopen(file\_name, "r"); //Открытие файла

fseek(file,0, SEEK\_END); //Определение размера файла для корректного выделения памяти

int file\_size = ftell(file);

fseek(file,0, SEEK\_SET);

char\*str\_in\_file = (char\*)malloc(sizeof(char)\*(file\_size)); //Выделяется память под строку.

fgets(str\_in\_file, file\_size\*sizeof(char), file); // Первая строка txt файла считывается в выделенную память.Файл закрывается, указатель на строку возвращается.

fclose(file);

return str\_in\_file;

}

1. Обход каталога

Данная функция обходит каталог и при наличии txt файла открывает его с помощью функции read\_file. Полученный указатель помещается в массив из указателей на строки. В данной функции выделяется память под строку, в которую будет помещаться путь к файлам. Используется NAME\_MAX, определяющий макс. длину имени файла. Открывается поток каталога и проверяется на корректное открытие. Далее с помощью функции readdir считывается структура с информацией о первом файле в каталоге.

Происходит цикл, который выполняется, пока в текущей директории будут объекты. К строке, содержащий путь, добавляется имя объекта, проверяется, является ли объект файлом или директорией, если объект - это непустой txt файл,то возвращается указатель на первую строку из него, если объект - это директория не родительская и не текущая, то рекурсивно вызывается функция read\_catalog для прочтения данной директории, после проверки текущего объекта current\_path "обрезается" до первоначального положения. Далее считывается следующий объект из данной директории. После обхода всех объектов в директории, она закрывается.

void read\_catalog(const char\* dir\_name,char\*\*strs\_in\_file, int\* len)

{

char\* current\_path = (char\*)malloc(sizeof(char)\*(strlen(dir\_name)+NAME\_MAX));//Выделяется память под строку, в которую будет помещаться путь к файлам. Используется NAME\_MAX, определяющий макс. длину имени файла.

strcpy(current\_path,dir\_name);

DIR \*current\_dir = opendir(current\_path); //Открывается поток каталога и проверяется на корректное открытие. Далее с помощью функции readdir считывается структура с информацией о первом файле в каталоге.

if(current\_dir == NULL)

return;

dirent\* file\_in\_current\_dir = (dirent\*)readdir(current\_dir);

while(file\_in\_current\_dir) //Цикл выполнятеется, пока в текущей директории будут объекты\*/

{

int path\_len = strlen(current\_path); //К строке, содержащий путь, добавляется имя объекта

strcat(current\_path, "/");

strcat(current\_path, file\_in\_current\_dir->d\_name);

if //Проверяется,является ли объект файлом или директорией

(

file\_in\_current\_dir->d\_type == DT\_REG &&

strstr(file\_in\_current\_dir->d\_name, ".txt")!=NULL

)

{

if((strs\_in\_file[\*len] = (char\*)read\_file(current\_path))!=NULL ) //Если объект - это непустой txt файл,то возвращается указатель на первую строку из него

(\*len)++;

}

if

(

file\_in\_current\_dir->d\_type == DT\_DIR && // Если объект - это директория не родительская и не текущая, то рекурсивно вызывается функция read\_catalog для прочтения данной директории

strcmp(".",file\_in\_current\_dir->d\_name)!=0 &&

strcmp("..",file\_in\_current\_dir->d\_name)!=0

)

{

read\_catalog(current\_path,strs\_in\_file, len);

}

current\_path[path\_len] = '\0'; //После проверки текущего объекта current\_path "обрезается" до первоначального положения. Далее считывается следующий объект из данной директории

file\_in\_current\_dir = (dirent\*)readdir(current\_dir);

}

closedir(current\_dir); //После обхода всех объектов в директории, она закрывается

}

1. Компаратор для функции qsort

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

{

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

}

Вывод

В ходе выполнения работы изучено использование рекурсивной функции для обхода дерева.

Приложение

#include <stdio.h>

#include <sys/types.h>

#include <dirent.h>

#include <string.h>

#include <stdlib.h>

typedef struct dirent dirent;

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

{

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

}

char\* read\_file(char\* file\_name) // Функция, которая открывает txt файл и возвращает указатель на первую его строку (если она есть)

{

FILE\* file = fopen(file\_name, "r"); //Открытие файла

fseek(file,0, SEEK\_END); //Определение размера файла для корректного выделения памяти

int file\_size = ftell(file);

fseek(file,0, SEEK\_SET);

char\*str\_in\_file = (char\*)malloc(sizeof(char)\*(file\_size)); //Выделяется память под строку.

fgets(str\_in\_file, file\_size\*sizeof(char), file); // Первая строка txt файла считывается в выделенную память.Файл закрывается, указатель на строку возвращается.

fclose(file);

return str\_in\_file;

}

void read\_catalog(const char\* dir\_name,char\*\*strs\_in\_file, int\* len) //Данная функция обходит каталог и при наличии txt файла открывает его с помощью функции read\_file. Полученный указатель помещается в массив из указателей на строки

{

char\* current\_path = (char\*)malloc(sizeof(char)\*(strlen(dir\_name)+NAME\_MAX));//Выделяется память под строку, в которую будет помещаться путь к файлам.Использвуется NAME\_MAX, определяющий макс. длину имени файлаа

strcpy(current\_path,dir\_name);

DIR \*current\_dir = opendir(current\_path); //Открывается поток каталога и проверяется на корректное открытие. Далее с помощью функции readdir считывается структура с информацией о первом файле в каталоге.

if(current\_dir == NULL)

return;

dirent\* file\_in\_current\_dir = (dirent\*)readdir(current\_dir);

while(file\_in\_current\_dir) //Цикл выполнятеется, пока в текущей директории будут объекты\*/

{

int path\_len = strlen(current\_path); //К строке, содержащий путь, добавляется имя объекта

strcat(current\_path, "/");

strcat(current\_path, file\_in\_current\_dir->d\_name);

if //Проверяется,является ли объект файлом или директорией

(

file\_in\_current\_dir->d\_type == DT\_REG &&

strstr(file\_in\_current\_dir->d\_name, ".txt")!=NULL

)

{

if((strs\_in\_file[\*len] = (char\*)read\_file(current\_path))!=NULL ) //Если объект - это непустой txt файл,то возвращается указатель на первую строку из него

(\*len)++;

}

if

(

file\_in\_current\_dir->d\_type == DT\_DIR && // Если объект - это директория не родительская и не текущая, то рекурсивно вызывается функция read\_catalog для прочтения данной директории

strcmp(".",file\_in\_current\_dir->d\_name)!=0 &&

strcmp("..",file\_in\_current\_dir->d\_name)!=0

)

{

read\_catalog(current\_path,strs\_in\_file, len);

}

current\_path[path\_len] = '\0'; //После проверки текущего объекта current\_path "обрезается" до первоначального положения. Далее считывается следующий объект из данной директории

file\_in\_current\_dir = (dirent\*)readdir(current\_dir);

}

closedir(current\_dir); //После обхода всех объектов в директории, она закрывается

}

int main()

{

//Выделяется память под массив указателей на строки

char\*\* strs = (char\*\*)malloc(sizeof(char\*)\*50);

int len = 0;

read\_catalog(".",strs, &len); //Специальная функция обходит директории и заполняет массив из строк строками из файлов

int i = 0;

for (i; i < len; i++)

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

qsort(strs, len, sizeof(char\*),compare); //Строки сортируются библиотечной функцией,компаратор для которой сравнивает числа в начале строк.Выводятся уже отсортированные строки.

int j = 0;

for (j; j < len; j++)

printf("\n%s\n", strs[j]);

int k = 0;

for(k; k < len; k++)

free(strs[k]); //Освобождение памяти выделенной под каждую строку и под массив строк

free(strs);

return 0;

}