**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

Тема: Рекурсивный обход дерева файлов

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Доброхвалов М.О. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель.** Выполнить обход дерева вложенных файлов и папок.

**Задание.**

Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида *<filename>*.txt. В качестве имени файла используется символ латинского алфавита.

На вход программе подается строка. Требуется найти и вывести последовательность полных путей файлов, имена которых образуют эту строку.

**Ход работы.**

1. Объявление структуры данных, в которой храниться список путей, количество элементов списка, максимально возможное количество элементов списка на данной итерации.

typedef struct List

{

char \*\*strings;

int size;

int max;

}List;

1. Описание функции добавления пути в список. Функция принимает путь к папке, в которой осуществляется поиск, указатель на структуру типа List, указатель на структуру папки.

void add(char \*current\_path, List \*list, struct dirent \*de)

{

int len = strlen(current\_path);

int log = 0;

strcat(strcat(current\_path, "/"), de→d\_name);

for (int i = 0; i < list->size; i++)

if ( strcmp(list->strings[i], current\_path) == 0)

log = 1;

if (log != 1 && list->size < list->max)

{

strcpy(list->strings[list->size], current\_path);

list->size++;

log = 0;

}

current\_path[len] = '\0';

}

1. Описание функции, которая печатает список. Функция принимает указатель на структуру списка.

void print(List \*list)

{

for(int i=0; i < list->size; i++)

printf("%s\n", (list->strings)[i]);

}

1. Функция рекурсивного обхода дерева фалов и папок. Функция принимает путь к папке, в которой осуществляется поиск, искомый символ, указатель на структуру списка.

void list\_dir(const char \*startdir, char c, List \*list)

{

char current\_path[500];

strcpy(current\_path,startdir);

DIR \*dir = opendir(current\_path);

struct dirent \*de = readdir(dir);

if(dir)

while(de)

{

if ( de->d\_type == DT\_REG && c == de->d\_name[0] )

{

add(current\_path, list, de);

}

else

if(de->d\_type == DT\_DIR &&

strcmp(".",de->d\_name) && strcmp("..",de->d\_name))

{

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

strcat(strcat(current\_path,"/"),de->d\_name);

list\_dir(current\_path, c, list);

current\_path[path\_len] = '\0';

}

de = readdir(dir);

}

closedir(dir);

}

1. Объявление и инициализация необходимых переменных.

char \*string = (char \*)malloc(SIZE);

List list;

list.strings = (char \*\*)malloc(SIZE\*sizeof(char \*));

for (int i=0; i<SIZE; i++)

(list.strings)[i] = (char \*)malloc(500);

list.size = 0;

list.max = 0;

fscanf(stdin,"%s", string);

1. Выбор папки, из которой начинается поиск, запуск цикла обхода.

if(c==2)

{

for (int i=0; i < strlen(string); i++)

{

list.max = i+1;

list\_dir(v[1], string[i], &list);

}

}

else

{

for (int i=0; i < strlen(string); i++)

{

list.max=i+1;

list\_dir(".", string[i], &list);

}

}

1. Вывод и очищение памяти.

print(&list);

free(string);

for (int i=0; i<20; i++)

free((list.strings)[i]);

1. Файл с лабораторной работой и отчет загружается на github с помощью консоли:

* Создана новая ветка с помощью команды git checkout –b Dobrokhvalov\_sem2\_lab3
* Добавлены файлы загрузки с помощью команды git add Dobrokhvalov\_sem2\_lab3
* Добавлен комментарий коммита командой git commit –m “Dobrokhvalov\_sem2\_lab3”
* Файлы были загружены командой git push origin Dobrokhvalov\_sem2\_lab3

**Вывод:** В ходе работы был выполнен рекурсивный обход дерева файлов и папок, а так же поиск элементов по определенным критериям(обычные файлы и папки, их имена).