

**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

Иванов В.С.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2017

## Оглавление

|                                           |   |
|-------------------------------------------|---|
| Цель работы .....                         | 3 |
| Задание .....                             | 3 |
| Содержание (реализация функций).....      | 4 |
| 1.    Функция <i>text_from_file</i> ..... | 4 |
| 2.    Функция <i>list_dir</i> .....       | 4 |
| 3.    Функция <i>main</i> .....           | 6 |
| Вывод .....                               | 7 |
| Приложение .....                          | 8 |

## Цель работы

Создание функции для обхода дерева из папок, реализованной с помощью рекурсии, а также освоение работы с файлами.

## 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*.

Требуется найти файл, который содержит строку "Minotaur" (файл-минотавр).

Файл, с которого следует начинать поиск, всегда называется file.txt (но полный путь к нему неизвестен).

Каждый текстовый файл, кроме искомого, может содержать в себе ссылку на название другого файла (эта ссылка не содержит пути к файлу). Таких ссылок может быть несколько.

## Содержание (реализация функций)

### 1. Функция *text\_from\_file*

Это функция служит для работы с файлом. Она открывает текстовый файл для чтения, затем функций стандартной библиотеки *fseek* и *ftell* определяет его размера. Далее происходит посимвольная запись содержимого файла в строку и в конце возвращает указатель на эту строку.

```
char *text_from_file(const char *current_path, int *file_size){
    FILE *file=fopen(current_path,"rt");
    fseek(file,0, SEEK_END);
    *file_size = ftell(file);
    fseek(file,0, SEEK_SET);

    char *str=(char*)malloc(sizeof(char)*(*file_size));
    int i=0;
    char symb;
    while(fscanf(file,"%c",&symb)>0) str[i++]=symb;
    fclose(file);
    return str;
}
```

### 2. Функция *list\_dir*

Функция производит рекурсивный обход директории. Когда функция находит файл, который нужно найти (его имя *name\_of\_file*), то выполняется функция *text\_from\_file*, представленная выше.

```
int list_dir(const char *newcur_path,const char *startdir, char* name_of_file, char **name_path)
{
    char current_path[10000];
    int height_of_path=0;

    strcpy(current_path,newcur_path);

    DIR *dir = opendir(current_path);
    struct dirent *de = readdir(dir);

    if(dir){
        while(de)
        {
            int path_len = strlen(current_path);

            strcat(current_path,"/");
            strcat(current_path,de->d_name);

            if(de->d_type==DT_REG && !strcmp(de->d_name,name_of_file)){
                int i,j=0,file_size=0;
                char *str=text_from_file(current_path,&file_size);
                char token[199][file_size];
```

Далее происходит разбиение строки, в которую записано содержимое файла на лексемы и производится их анализ:

1. Если лексема *@include*, то рассматривается следующая лексема;
2. Если *Deadlock*, то выход из функции;
3. Если *Minotaur*, то увеличивается счётчик путей, в массив строк записывается путь к файлу и функция возвращает значение счётчика;

```
char *tmp=strtok(str, " \n");
while(tmp){
    strcpy(token[j],tmp);
    j++;
    tmp=strtok(NULL, " \n");
}

for(i=0;i<j;i++){
    if(!strcmp(token[i], "@include"));
    else if(!strcmp(token[i], "Deadlock")) return height_of_path;
    else if(!strcmp(token[i], "Minotaur")){
        name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
        strcpy(name_path[height_of_path],current_path);
        height_of_path++;
        return height_of_path;
    }
}
```

4. В остальных случаях рекурсивно вызывается эта же функция с изменённым именем искомого файла на текущую лексему и по её окончании:

- a) Если *Minotaur* не был найден, продолжается обход;
- b) Если найден, то счётчик путей увеличивается, в массив строк записывается путь к файлу и возвращается значение счётчика.

```
    }
    else{
        height_of_path=list_dir(startdir, startdir, token[i], name_path);
        if(height_of_path!=0){
            name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
            strcpy(name_path[height_of_path],current_path);
            height_of_path++;
            return height_of_path;
        }
    }
}
```

Если же на пути встречается папка (не текущая или родительская), то рекурсивно вызывается эта же функция, но в качестве текущего пути берётся путь к этой папке, а если в ней был найден *Minotaur*, то обход прекращается.

```
if( de->d_type == DT_DIR && 0!=strcmp(".",de->d_name) && 0!=strcmp("../",de->d_name))
{
    height_of_path=list_dir(current_path, startdir, name_of_file, name_path);
}
if (height_of_path!=0){
    return height_of_path;
}
```

Если по завершению обхода в текущей папке отсутствуют объекты, то функция заканчивает работу и возвращает значение счётчика путей.

```

        current_path[path_len] = '\\0';
        de = readdir(dir);
    }
    closedir(dir);
}
return height_of_path;
}

```

### 3. Функция *main*

```

int main()
{
    char **name_path=(char**)malloc(200*sizeof(char*)); //Массив путей
    int height_of_path=0,i;
    char *name_of_file="file.txt";
    height_of_path=list_dir(".", ".", name_of_file, name_path);
    for(i=height_of_path-1; i>=0; i--){ //Вывод путей в обратном порядке
        printf("%s\n", name_path[i]);
        free(name_path[i]);
    }
    free(name_path);
    return 0;
}

```

## **Вывод**

В результате выполнений данной лабораторной работы, была написана программа для рекурсивного обхода дерева из папок, которая использовалась для чтения и поиска файлов, находящихся в подпапках, что помогло изучить работу с файловой системой.

## Приложение

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <string.h>
#include <stdlib.h>

char *text_from_file(const char *current_path, int *file_size){
    FILE *file=fopen(current_path,"rt");
    fseek(file,0, SEEK_END);
    *file_size = ftell(file);
    fseek(file,0, SEEK_SET);

    char *str=(char*)malloc(sizeof(char)*(*file_size));
    int i=0;
    char symb;
    while(fscanf(file,"%c",&symb)>0) str[i++]=symb;
    fclose(file);
    return str;
}

int list_dir(const char *newcur_path,const char *startdir, char* name_of_file, char
**name_path)
{
    char current_path[10000];
    int height_of_path=0;

    strcpy(current_path,newcur_path);

    DIR *dir = opendir(current_path);
    struct dirent *de = readdir(dir);

    if(dir){
        while(de)
        {
            int path_len = strlen(current_path);

            strcat(current_path,"/");
            strcat(current_path,de->d_name);

            if(de->d_type==DT_REG && !strcmp(de->d_name,name_of_file)){
                int i,j=0,file_size=0;
                char *str=text_from_file(current_path,&file_size);
                char token[199][file_size];

                char *tmp=strtok(str," \n");
                while(tmp){
                    strcpy(token[j],tmp);
                    j++;
                    tmp=strtok(NULL," \n");
                }

                for(i=0;i<j;i++){
```



```

        if(!strcmp(token[i], "@include"));
        else if(!strcmp(token[i], "Deadlock")) return height_of_path;
        else if(!strcmp(token[i], "Minotaur")){
            name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
            strcpy(name_path[height_of_path], current_path);
            height_of_path=1;
            return height_of_path;
        }
        else{
            height_of_path=list_dir(startdir, startdir, token[i], name_path);
            if(height_of_path!=0){
                name_path[height_of_path]=(char*)malloc(sizeof(char)*1000);
                strcpy(name_path[height_of_path], current_path);
                height_of_path++;
                return height_of_path;
            }
        }
    }
    if( de->d_type == DT_DIR && 0!=strcmp(".", de->d_name) &&
    0!=strcmp("../", de->d_name))
    {
        height_of_path=list_dir(current_path, startdir, name_of_file, name_path);
    }
    if (height_of_path!=0){
        return height_of_path;
    }
    current_path[path_len] = '\0';
    de = readdir(dir);
}
closedir(dir);
}
return height_of_path;
}

int main()
{
    char **name_path=(char**)malloc(200*sizeof(char*)); //Массив путей
    int height_of_path=0, i;
    char *name_of_file="file.txt";
    height_of_path=list_dir(".", ".", name_of_file, name_path);
    for(i=height_of_path-1; i>=0; i--){ //Вывод путей в обратном порядке
        printf("%s\n", name_path[i]);
        free(name_path[i]);
    }
    free(name_path);
    return 0;
}

```