

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

Студентка гр. 3344

Коняева М.В.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

## **Цель работы**

Целью работы является освоение работы с рекурсией в языке Си на примере использующей ее программы.

## Задание

Вариант 1. 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 *.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.

Цепочка, приводящая к файлу-минотавру может быть только одна. Общее количество файлов в каталоге не может быть больше 3000. Циклических зависимостей быть не может. Файлы не могут иметь одинаковые имена. Программа должна вывести правильную цепочку файлов (с путями), которая привела к поимке файла-минотавра.

## Выполнение работы

Подключим стандартные библиотеки *stdio.h* для работы с вводом, *string.h* для работы со строками, *stdlib.h* для работы с памятью, а также *dirent.h* для работы с директориями.

Функция *char \*pathcat()* получает на вход две строки, возвращает отформатированную строку массива символов. Нужна для конкатенации имен родительской и вложенной директорий.

Функция *char \*find\_file()* получает на вход название директории и файла, рекурсивно проходит по директориям для нахождения полного пути к переданному файлу.

Функция *void process()* получает на вход имя файла и массив строк, нужный для записи конечного результата. Происходит считывание информации с файла. Если строка «*Minotaur*», то переменная *flag* приравнивается к единице, путь к файлу-минотавру записывается в массив строк. Если строка «*Deadlock*», то функция прекращает работу с этим файлом, возвращая пустоту. Если строка содержит ссылку на другой файл, то рекурсивно вызывается эта же функция для того, чтобы сделать те же действия. Функция будет идти вглубь, пока не встретит файл с записью «*Minotaur*» и не изменит *flag*. После рекурсивного вызова и нахождения дороги к файлу, верный путь в обратном порядке записывается в массив строк.

В функции *main()* происходит изначальный вызов функции *void process()*, а также открытие файла для записи конечного результата.

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	File.txt: @include file1.txt @include file4.txt @include file5.txt file1.txt: Deadlock file2.txt: @include file3.txt file3.txt: Minotaur file4.txt: @include file2.txt file5.txt: Deadlock  ./labyrinth/file.txt ./labyrinth/file1.txt ./labyrinth/mul/file2.txt ./labyrinth/mul/file3.txt ./laby- rinth/mul/mul/file4.txt	./labyrinth/file.txt ./labyrinth/mul/mul/file4.txt ./labyrinth/mul/file2.txt ./labyrinth/mul/file3.txt	Данные обработаны корректно.

## **Выводы**

Были изучена работа с рекурсией. Также была создана программа, в которой реализован рекурсивный обход дерева для нахождения верного пути к файлу.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

```
#include <stdio.h>
#include <string.h>
#include <string.h>
#include <stdlib.h>
#include <dirent.h>

#define SIZE 256
#define MAXFILES 3000

int flag = 0;
int count = 0;

char *pathcat(const char *path1, const char *path2)
{
    int res_path_len = strlen(path1) + strlen(path2) + 2;

    char *res_path = malloc(res_path_len * sizeof(char));

    sprintf(res_path, "%s/%s", path1, path2);

    return res_path;
}

char *find_file(const char *dir_name, const char *filename)
{
    char *full_path_file = NULL;
    DIR *dir = opendir(dir_name);
    if (dir)
    {
        struct dirent *de = readdir(dir);
        while (de)
        {
            if (de->d_type == DT_REG && !strcmp(de->d_name, filename))
            {
                full_path_file = pathcat(dir_name, filename);
            }
            else if (de->d_type == DT_DIR && strcmp(de->d_name, ".") != 0
&& strcmp(de->d_name, "..") != 0)
            {
                char *new_dir = pathcat(dir_name, de->d_name);
                full_path_file = find_file(new_dir, filename);
                free(new_dir);
            }
            if (full_path_file)
                break;
            de = readdir(dir);
        }
        closedir(dir);
    }
    else
        printf("Failed to open %s directory\n", dir_name);
}
```

```

    return full_path_file;
}

void process(char* filename, char** result)
{
    char* file_path = find_file(".", filename);

    FILE* file = fopen(file_path, "r");
    if (!file)
        return;

    char data[SIZE];
    while (fgets(data, SIZE, file) != NULL && flag==0)
    {
        //data[strlen(data)-1] = '\0';

        if (strstr(data, "Deadlock") )
        {
            return;
        }
        else if (strstr(data, "Minotaur"))
        {
            flag = 1;
            result[count] = malloc(SIZE * sizeof(char));
            strcpy(result[count++], file_path);
        }
        else if (strncmp(data, "@include ", 9) == 0 && data[strlen(data)
- 1] == '\n')
        {
            data[strlen(data)-1] = '\0';
            memmove(&data[0], &data[9], sizeof(char) * SIZE);
            process(data, result);
            if (flag)
            {
                result[count] = malloc(SIZE * sizeof(char));
                strcpy(result[count++], file_path);
            }
        }
    }

    fclose(file);
    return;
}

int main()
{
    char** result = (char**)malloc(sizeof(char*) * MAXFILES);

    process("file.txt", result);

    FILE *fp = fopen("result.txt", "w");
    if (fp == NULL)
    {
        return 1;
    }
    int i;

```



```
for (i = count - 1; i >= 0; --i)
{
    fprintf(fp, "%s\n", result[i]);
}
fclose(fp);
return 0;
}
```