**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: Рекурсия

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Путьков Д.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Оглавление**

[**Цель работы** 3](#_Toc482904877)

[**Задание** 3](#_Toc482904878)

[**Содержание** 5](#_Toc482904879)

[**Вывод** 9](#_Toc482904880)

# **Цель работы**

Реализация программы, использующей рекурсию для прохождения по директориям и нахождения файла с нужной информацией.

# **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).

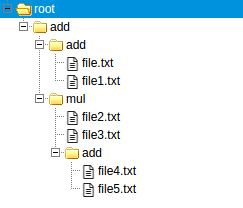
Каждый текстовый файл, кроме искомого, может содержать в себе ссылку на название другого файла (эта ссылка не содержит пути к файлу). Таких ссылок может быть несколько.

А также файл может содержать тупик:

Deadlock

Программа должна вывести правильную цепочку файлов (с путями), которая привела к поимке файла-минотавра.

**Пример работы программы :**

****

**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**

**@include file1.txt**

**file5.txt:**

**Deadlock**

*Правильный ответ:*

root/add/add/file.txt

root/add/mul/add/file4.txt

root/add/mul/file2.txt

root/add/mul/file3.txt

**Информация для выполнения задания**

1. Цепочка, приводящая к файлу-минотавру может быть только одна.
2. Общее количество файлов в каталоге не может быть больше 200.
3. Циклических зависимостей быть не может.
4. Файлы не могут иметь одинаковые имена.

**Перечень функций**

* int list\_dir(const char \*newcur\_path,const char \*rootdir, char\* nameOfFile, char \*\*nameOfPath);

Функция, с помощью рекурсии проходящаяся по папкам вплоть до папки с искомым файлом.

* void printPath();

Функция, которая вызывает list\_dir, возвращает глубину рекурсии и выводит на экран путь к искомому файлу.

* int findTheSize(char \*current\_path);

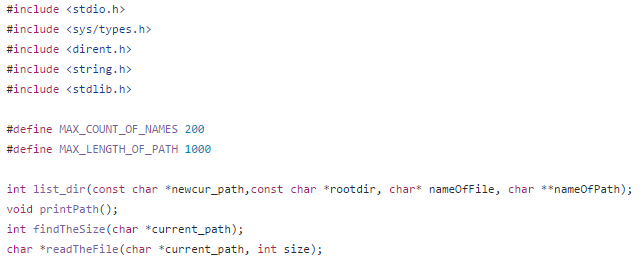
Находит размер файла.

* char \*readTheFile(char \*current\_path, int size);

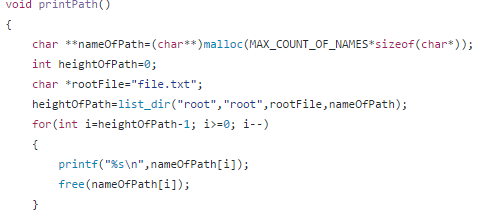
Считывает данные из файла.

# **Содержание**

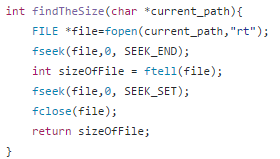
Подлючение библиотек и определение максимальной длины пути и количества файлов в директории. Объявление функций, использованных в программе.



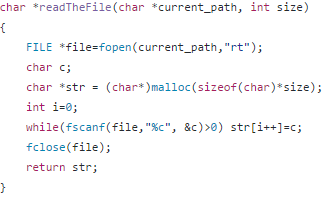
В функции выделяется память под массив, в котором будут храниться пути к файлам, вызывается функция *list\_dir,* возвращающая глубину рекурсии, выводится на экран путь к нужному файлу, освобождается память массива адресов.



Функция, принимающая полный путь к файлу, функция *fseek* устанавливает внутренний указатель на конец файла, функция *ftell* возвращает его текущее положение, затем устанавливаем внутренний указатель обратно на начало строки и закрываем файл.  
Функция возвращает размер файла в байтах.



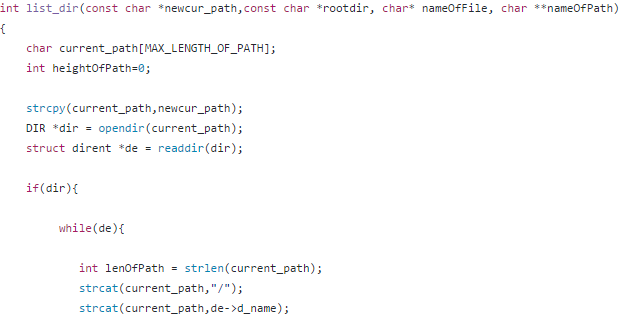
Функция посимвольно считывает данные из файла, возвращает строку, в которую все это записывалось.



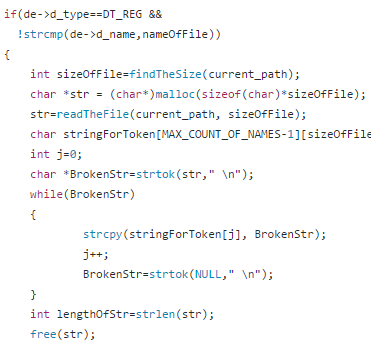
Функция принимает путь текущей директории, корневую папку, имя файла и массив, в который будет записываться путь.

Статически выделяется память полный путь к файлу, локально значение глубины рекурсии 0.

В путь к текущей директории копируется путь, который уже был пройден до этого момента, далее открывается эта директория и проверяется, существуют ли в ней папки. Затем, пока существуют объекты в папке, присваиваем текущему пути имя файла.



Проверка, не является ли файл директорией, после чего вызывается функция *findTheSize*, выделяется память под массив, в который будут записаны данные из файла при помощи вызова функции *readTheFile*. Затем создаем массив указателей, в который при помощи функции *strtok* будем записывать данные, разделенные пробелами либо символом перевода на новую строку. Освобождаем память строки с данными из файла.



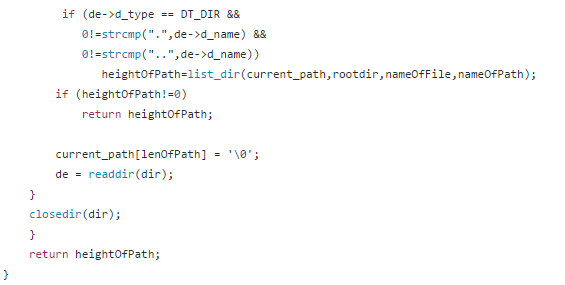
Теперь проверяем каждое слово из получившихся токенов: пропускаем весь мусор;

если *deadlock*-выходим из функции; если *Minotaur*-записываем в массив путей путь к файлу и начинаем от этого файла считать глубину рекурсии;

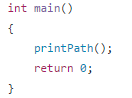
Если ни одно из условий не выполнилось, значит, данные являются путем к следующему файлу. В таком случае мы вызываем функцию *list\_dir*, в которую в качестве имени файла передаем токен, и начинаем искать этот файл с корневой директории. После того, как функция вернет значение глубины рекурсии, записываем в массив пути путь к этому файлу, увеличиваем глубину рекурсии на 1 и возвращаем значение.



Если же файл является директорией, причем не текущей и не родительской, то опять вызываем функцию *list\_dir;* Если глубина не ноль, то возвращаем ее значение, иначе продолжаем читать файлы в текущей директории. Когда файлы закончатся, выходим из папки и возвращаем глубину рекурсии.



Функция main вызывает функцию *printPath.*



# **Вывод**

Выполнив данную лабораторную работу, было закреплено на практике использование рекурсии в программе; также было освоена работа с директориями и файлами при помощи библиотек *«sys\types.h»* и *«dirent.h».*