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

Студент гр. 3344 _____ Щербак М.С.

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

Используемые библиотеки

- stdio.h для работы с вводом/выводом.

- stdlib.h для работы с динамической памятью.

- string.h для работы со строками.

- dirent.h для работы с директориями.

Функция inDirect()

- Описание: Рекурсивный поиск файла в директориях для нахождения полного пути к переданному файлу.

- Параметры: Название директории и файла.

- Возвращаемое значение: Полный путь к найденному файлу.

Функция searchWord()

- Описание: Поиск ключевых слов "Minotaur" и "Deadlock" в файле, рекурсивный вызов для обработки вложенных файлов.

- Параметры: Имя файла и массив строк для записи результатов.

- Действия: Считывание информации с файла, поиск ключевых слов, запись путей к файлам с ключевым словом "Minotaur".

Функция main()

- Действия: Вызов функции searchWord() для начала поиска ключевых слов, запись результатов в файл "result.txt".

Результаты тестирования

Результаты тестирования программы показали успешное выполнение поиска ключевого слова "Minotaur" в файлах и запись путей к найденным файлам в файл "result.txt".

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 ./labyrinth/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 <stdlib.h>
#include <string.h>
#include <dirent.h>

#define SIZE 256
#define MAXFILES 3000

int flag = 0;
int count = 0;

char *inDirect(const char *directName, const char *fileName)
{
    DIR *dir;
    struct dirent *entry;
    char *full_path_file = NULL;
    dir = opendir(directName);
    if (dir)
    {
        entry = readdir(dir);
        while (entry)
        {
                 if (entry->d_type == DT_REG && !strcmp(entry->d_name,
fileName))
            {
                 int resPathLen = strlen(directName) + strlen(fileName) +
2;
                full_path_file = malloc(resPathLen * sizeof(char));
                sprintf(full_path_file, "%s/%s", directName, fileName);
                break;
            }
               else if (entry->d_type == DT_DIR && strcmp(entry->d_name,
".") != 0 && strcmp(entry->d_name, "..") != 0)
            {
                        char *newDirect = malloc((strlen(directName) +
strlen(entry->d_name) + 2) * sizeof(char));
                sprintf(newDirect, "%s/%s", directName, entry->d_name);
                full_path_file = inDirect(newDirect, fileName);
                free(newDirect);
                if (full_path_file)
                {
                    break;
                }
            }
            entry = readdir(dir);
        }
        closedir(dir);
    }
    return full_path_file;
}

void searchWord(char *filename, char **result)
{
    char *file_path = inDirect(".", filename);
```

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

    char data[SIZE];
    while (fgets(data, SIZE, file) != NULL && flag == 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);
            searchWord(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);

    searchWord("file.txt", result);

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

    for (int i = count - 1; i >= 0; --i)
    {
        fprintf(result_file, "%s\n", result[i]);
    }

    fclose(result_file);

    for (int i = 0; i < count; i++)
```

```
    {
        free(result[i]);
    }

    free(result);

    return 0;
}
```