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. 3341 _____ Рябов М.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Цель работы заключается в разработке программы на языке программирования, которая осуществляет рекурсивный обход иерархии папок и файлов в заданной структуре, анализирует названия текстовых файлов, записывает их полные пути в виде строки в файл.

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.

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. Программа должна вывести правильную цепочку файлов (с путями), которая привела к поимке файла-минотавра. 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. Ваше решение должно находиться в директории /home/box, файл с решением должен называться solution.c. Результат работы программы должен быть записан в файл result.txt. Ваша программа должна обрабатывать директорию, которая называется labyrinth

Выполнение работы

Программа в целом представляет собой поиск файла "file.txt" в структуре каталогов, начиная с указанного пути "./labyrinth". При нахождении файла "file.txt" программа проверяет его содержимое. Если содержимое файла содержит строку "Minotaur", то программа создает файл "result.txt" и записывает в него путь к найденному файлу "file.txt".

Функции программы:

1. returnFileName: Функция извлекает имя файла из строки, соответствующей заданному шаблону (регулярному выражению). Если имя файла найдено, функция возвращает его, иначе возвращает NULL.

2. addPath: Функция добавляет новый путь к списку путей. Если список путей пустой, то создается новый список и добавляется путь. В противном случае новый путь добавляется к существующему списку.

3. isDir: Функция проверяет, является ли указанный элемент директорией (каталогом).

4. writeResult: Функция записывает список путей в указанный файл.

5. recSearchFile: Рекурсивная функция для поиска файла "file.txt" в указанном каталоге и его подкаталогах. При нахождении файла программа проверяет его содержимое и рекурсивно вызывает себя для обработки найденных файлов или перехода в подкаталог. Главная функция main запускает процесс поиска файла "file.txt" в каталоге "./labyrinth".

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии

1.	./labyrinth	`./root/add/add/file.txt`
		`./root/add/mul/add/file4.txt`
		
		`./root/add/mul/file2.txt`
		
		`./root/add/mul/file3.txt`
			Тест с e.moevm

2.	.	`./labyrinth/J0/n2/JU260/q1/r0/file.txt`
		`./file3.txt`
		`./file2.txt`
			Тест, проверяющий, глубину вхождения рекурсии

Выводы

В ходе выполнения данной работы были приобретены навыки эффективного использования рекурсивных методов для обхода дерева файлов, а также работы с файловой системой, анализа данных о файлах и записью информации в файл. Разработка программы, способной автоматически обрабатывать информацию из различных файлов и директорий, позволила улучшить навыки программирования и решения сложных задач.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

char* pattern = "([A-z0-9_]+\\.txt)";

void recSearchFile(char* filename, char* path, char* allPathes);
char* returnFileName(char* buffer);
char* addPath(char* oldPath, char* newPath);

int main(){
    char* allPathes = NULL;
    recSearchFile("file.txt", "./labyrinth", allPathes);
    return 0;
}

char* returnFileName(char* buffer){
    regex_t regexCompiled;
    regcomp(&regexCompiled, pattern, REG_EXTENDED);
    regmatch_t groups[2];

    int j = 0;
    char* filename = (char*)malloc(sizeof(char) * ((groups[1].rm_eo
- groups[1].rm_eo) + 1));
    if(regexec(&regexCompiled, buffer, 2, groups, 0) == 0){
        for(int i = groups[1].rm_so; i < groups[1].rm_eo; i++){
            filename[j++] = buffer[i];
        }
        filename[j] = '\0';
        return filename;
    }else{
        return NULL;
    }
}

char* addPath(char* oldStr, char* newPath){
    char* newStr;
    if (oldStr == NULL){
        newStr = (char*)malloc(sizeof(char)*(strlen(newPath) + 1));
        strcpy(newStr, newPath);
        return newStr;
    }
          newStr   =   (char*)malloc(sizeof(char)*(strlen(oldStr)   +
strlen(newPath) + 2));
    strcpy(newStr, oldStr);
    int size = strlen(oldStr);
    int maxsize = strlen(oldStr) + strlen(newPath) + 1;
    int j = 0;
```

```
        newStr[size] = '\n';
        for(int i = size + 1; i < maxsize + 1; i++){
            newStr[i] = newPath[j++];
        }
        newStr[maxsize] = '\0';
        return newStr;
    }


    void recSearchFile(char* filename, char* path, char* allPathes){
        DIR* fileThread;
        struct dirent* file;
        fileThread = opendir(path);
        if(fileThread == NULL){
            printf("Error, fileThread won't be open");
            exit(1);
        }

        while((file = readdir(fileThread)) != NULL){
             if (strcmp(filename, file->d_name) == 0 && file->d_type ==
DT_REG) //нашли файлик
            {
                                                char*   fullPathFile   =
(char*)malloc(sizeof(char)*(strlen(path)+strlen(filename) + 2));
                sprintf(fullPathFile, "%s/%s", path, file->d_name);

                allPathes = addPath(allPathes, fullPathFile);

                char buffer[256];
                FILE *fp = fopen(fullPathFile, "r");

                if(fp){
                        while((fgets(buffer, 256, fp))!=NULL){ //читаем
найденный файлик

                        if(!strcmp(buffer, "Minotaur")){

                            FILE *nfp = fopen("result.txt", "w");
                            if(nfp)
                            {
                                // записываем строку
                                fputs(allPathes, nfp);
                                fclose(nfp);

                            }

                        }else{
                                                    char*  nameNextFile  =
returnFileName(buffer);
                            if (nameNextFile != NULL){
                                        recSearchFile(nameNextFile, ".",
allPathes);
                            }
                        }
                    }
                    fclose(fp);
                }
            }
```

```
                else if(file->d_type == DT_DIR && strcmp(file->d_name,
"..") && strcmp(file->d_name, ".")){ //опускаемся в подкаталог
                                                        char*    new_dir    =
(char*)malloc(sizeof(char)*(strlen(path)+strlen(file->d_name) + 2));
                    sprintf(new_dir, "%s/%s", path, file->d_name);
                    recSearchFile(filename, new_dir, allPathes);
                }
            }

            if(closedir(fileThread) == -1){
                printf("FileThread don't be close");
                exit(1);
            }
        }
```