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 _____ Хангулян С. К.

Преподаватель _____ Глазунов С. А.

Санкт-Петербург

2024

Цель работы

Изучение основ работы рекурсии и создание с их помощью программы, способной работать с вложенными директориями.

Задание

Вариант 3

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

В каждом текстовом файле хранится одна строка, начинающаяся с числа вида: <число><пробел><латинские буквы, цифры, знаки препинания> ("124 string example!")

Требуется написать программу, которая, будучи запущенной в корневой директории, выведет строки из файлов всех поддиректорий в порядке возрастания числа, с которого строки начинаются

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.

Выполнение работы

Объявлен макрос size, участвующий в выделении памяти, и объявлена глобальная переменная count, показывающая количество найденных и записанных в массив и файл строк.

Функция main:

Объявлен двумерный динамический массив solution, в котором будут храниться найденные строки. Вызвана функция find_solution, затем произведена сортировка qsort, использующая comparator. Создан и открыт файл result.txt, с помощью цикла и функции fprintf строки массива записаны в файл. Массив очищен, файл закрыт.

Функция find_solution:

Является рекурсивной функцией. Открыта текущая директория, объявлен путь до файла / директории THE_WAY. Произведена проверка на пустую директорию. В случае непустой директории с помощью цикла while проверяется каждый файл и / или каждая директория в текущей. В случае файла с помощью функций strcpy и strcat «склеивается» путь до файла, после чего путь и массив передаются в функцию add_solution. В случае директории (не являющейся текущей и родительской!) также «склеивается» путь до директории и текущая функция вызывается повторно. В конце функции директория закрывается.

Функция add_solution:

С помощью функции strstr путь проверяется на наличие текстового файла в конце. В случае успеха файл открывается, для строки массива выделяется память и с помощью функции fgets из файла в массив считывается строка. В конце файл закрывается.

Функция comparator:

Разыменовывая указатели на две строки, сравнивает их с помощью функции atoll. Функция atoll переводит строку в число, обрезая все после числа в начале строки.

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

№	Входные данные	Выходные данные	Комментарии
1	root/file.txt: 4 Where am I? root/Newfolder/Newfile.txt: 2 Simple text root/Newfolder/Newfolder/Newfile.txt: 5 So much files! root/Newfolder(1)/Newfile.txt: 3 Wow? Text? root/Newfolder(1)/Newfile1.txt: 1 Small text	1 Small text 2 Simple text 3 Wow? Text? 4 Where am I? 5 So much files!	Корректно

Выводы

Были изучены основы работы с рекурсиями и создана программа, способная находить в «дереве» директорий файлы и сохранять их в отдельный файл.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khangulyan_Sargis_lb3.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#define size 5000
int count = 0;

int comparator(const void* a, const void* b){
    const char* A = *(const char**)a;
    const char* B = *(const char**)b;

    if (atoll(A) > atoll(B)) return 1;
    if (atoll(A) < atoll(B)) return -1;
    return 0;
}

void add_solution(const char* file, char** solution){
    if (strstr(file, ".txt")){
        FILE* o_file = fopen(file, "r");

        if (o_file){
            solution[count] = (char*)malloc(sizeof(char)*size);
            fgets(solution[count++], size, o_file);
        }
        fclose(o_file);
    }
}

void find_solution(const char* dir, char** solution){
    DIR* o_dir = opendir(dir);
    char* THE_WAY = (char*)malloc(sizeof(char)*size);
    if (o_dir == NULL) return;

    if (o_dir){
        struct dirent* temp;
        while ((temp = readdir(o_dir)) != NULL){

            if (temp->d_type == DT_REG){
                strcpy(THE_WAY, dir);
                strcat(THE_WAY, "/");
                strcat(THE_WAY, temp->d_name);
                add_solution(THE_WAY, solution);
            }
              else if (strcmp(temp->d_name, ".") != 0 && strcmp(temp-
>d_name, "..") != 0){
                strcpy(THE_WAY, dir);
                strcat(THE_WAY, "/");
                strcat(THE_WAY, temp->d_name);
                find_solution(THE_WAY, solution);
            }
        }
    }
```

```
    closedir(o_dir);
}

int main(){
    char** solution = (char**)malloc(sizeof(char*)*size);
    find_solution(".", solution);
    qsort(solution, count, sizeof(char*), comparator);

    FILE* result = fopen("./result.txt", "w");
    for (int i = 0; i < count; i++){
        fprintf(result, "%s\n", solution[i]);
        free(solution[i]);
    }
    free(solution);
    fclose(result);

    return 0;
}
```