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 _____ Кузнецов Р.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Знакомство и работа с рекурсией на языке программирования Си при помощи программы, использующей ее.

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.

Выполнение работы

Были подключены такие библиотеки как *<stdio.h>*, *<stdlib.h>*, *<string.h>* и *<dirent.h>*, последняя библиотека необходима для работы с каталогами и файлами. Определена структура *Line*, которая состоит из двух переменных: числа и текста. Она используется для хранения строк из файлов. Инициализированы такие глобальные переменные как: *lines* - указатель на массив структур *Line*, который изначально установлен в *NULL*, *count*, *capacity* это переменные, которые отслеживают текущее количество файлов и заполненность массива *lines*. Функция *int comparation(const void* a, const void* b)* используется для сортировки строк на основе их числовых значений. *void read_txt(const char* name_of_file, FILE* result)* это функция для чтения строк из файла. Она открывает файл, считывает строку, и сохраняет в массив *lines*. *void find_txt(const char* dirname, FILE* result)* - рекурсивная функция для обхода файлов и каталогов. Функция открывает директорию, проверяет файлы на их тип и в зависимости от этого либо вызывает *read_txt* для каждого текстового файла в каталоге, либо вызывает саму себя для каждого подкаталога. В функции *int main()* выделяется память для массива *lines*, открывается файл для записи результатов *result.txt*, вызывается *find_txt* для текущего каталога, сортируются строки в массиве *lines* с использованием быстрой сортировки *qsort* и записываются отсортированные строки в *result.txt*.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	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! result.txt	-

Выводы

Было проведено знакомство и работа с рекурсией на языке программирования Си при помощи программы, использующей ее.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

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

typedef struct {
    long int number;
    char text[256];
} Line;

Line *lines = NULL;
int count = 0;
int capacity = 0;

int comparation(const void* a, const void* b) {
    const Line* line_a = (const Line*)a;
    const Line* line_b = (const Line*)b;

    if (line_a->number < line_b->number) return -1;
    if (line_a->number > line_b->number) return 1;
    return 0;
}

void read_txt(const char* name_of_file, FILE* result) {
    FILE* f = fopen(name_of_file, "r");
    if (f == NULL) {
        perror(name_of_file);
        return;
    }

    if(fscanf(f, "%ld %[^\n]\n", &lines[count].number, lines[count].text)
== 2) {
        count++;
        if (count >= capacity) {
            capacity = (capacity == 0) ? 1 : capacity * 2;
            lines = realloc(lines, capacity * sizeof(Line));
            if (lines == NULL) {
                fprintf(stderr, "Memory allocation error\n");
                exit(1);
            }
        }
    }

    fclose(f);
}

void find_txt(const char* dirname, FILE* result) {
    DIR* dir = opendir(dirname);
    if (dir == NULL) {
        perror(dirname);
```

```
        return;
    }

    struct dirent* entry;

    while ((entry = readdir(dir)) != NULL) {

        if (entry->d_type == DT_REG) {
             char path_to_file[strlen(entry->d_name) + strlen(dirname) +
2];

            strcpy(path_to_file, dirname);
            strcat(path_to_file, "/");
            strcat(path_to_file, entry->d_name);
            read_txt(path_to_file, result);

          } else if (entry->d_type == DT_DIR && strcmp(entry->d_name,
".") != 0 && strcmp(entry->d_name, "..") != 0) {
             char path_to_dir[strlen(entry->d_name) + strlen(dirname) +
2];

            strcpy(path_to_dir, dirname);
            strcat(path_to_dir, "/");
            strcat(path_to_dir, entry->d_name);
            find_txt(path_to_dir, result);
        }
    }

    closedir(dir);
}

int main() {
    lines = malloc(1000 * sizeof(Line));
    if (lines == NULL) {
        fprintf(stderr, "Memory allocation error\n");
        return 1;
    }

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

    find_txt(".", result);
    qsort(lines, count, sizeof(Line), comparation);

    for (int i = 0; i < count; i++) {
        fprintf(result, "%ld %s\n", lines[i].number, lines[i].text);
    }

    fclose(result);
    free(lines);
    return 0;
}
```