**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 |
|  | 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;

}