**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 |  | Пчелкин Н.И. |
| 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 содержимое текстовых файлов, выполняет математические операции в соответствии с правилами задания и выводит на экран итоговый результат вычислений, основанный на содержимом файлов и вложенных папок.

## Задание

Вариант 2

Задана иерархия папок и файлов по следующим правилам:

название папок может быть только "add" или "mul"

В папках могут находиться другие вложенные папки и/или текстовые файлы

Текстовые файлы имеют произвольное имя с расширением .txt

Содержимое текстовых файлов представляет собой строку, в которой через пробел записано некоторое количество целых чисел

Требуется написать программу, которая, запускается в корневой директории, содержащей одну папку с именем "add" или "mul" и вычисляет и выводит на экран результат выражения состоящего из чисел в поддиректориях по следующим правилам:

Если в папке находится один или несколько текстовых файлов, то математическая операция определяемая названием папки (add = сложение, mul = умножение) применяется ко всем числам всех файлов в этой папке

Если в папке находится еще одна или несколько папок, то сначала вычисляются значения выражений, определяемые ими, а после используются уже эти значения

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 tmp.

## Основные теоретические положения

Основные теоретические положения для работы с файловой иерархией в C и использования рекурсии:

1. Работа с файловой иерархией: файловая иерархия представляет собой структуру, в которой файлы и директории организованы в виде дерева. Взаимодействие с файловой иерархией включает операции чтения, записи, создания, удаления файлов и директорий.

2. Работа с файлами и директориями: для работы с файловой иерархией в языке программирования C используются функции стандартной библиотеки языка, такие как fopen(), fclose(), fread(), fwrite(), opendir(), readdir(), closedir() и другие. Эти функции позволяют осуществлять доступ к файлам и директориям, выполнять чтение и запись данных.

3. Рекурсия: рекурсия в программировании — это прием, при котором функция вызывает саму себя. При работе с файловой иерархией рекурсия позволяет обходить все уровни директорий и файлов вложенных структур. Это особенно полезно при неопределенном количестве уровней вложенности или при необходимости выполнить однотипную операцию на каждом уровне.

4. Рекурсивное обход директорий: в рамках обработки файловой иерархии рекурсия часто используется для обхода всех элементов директории, включая поддиректории. Это позволяет пройти по всем уровням вложенности и обработать каждый файл или директорию в структуре.

5. Базовый и рекурсивный случаи: в рекурсивной функции для обхода директорий важно определить базовый случай, при котором рекурсия завершится, и рекурсивный случай, в котором функция вызывает саму себя для обработки следующего уровня директории.

6. Управление памятью: при работе с файловой иерархией и использовании рекурсии важно правильно управлять памятью. Необходимо освобождать ресурсы, выделенные для открытия файлов и директорий, чтобы избежать утечек памяти и повысить производительность программы.

## Выполнение работы

Были реализованы следующие структуры:

Структура Line, означающая содержащуюся в файле строку:

long long num – число в начале файла;

char\* text – последующий текст;

Были реализованы следующие функции:

char\* pathcat – принимает на вход текущий путь и путь к файлу или директории. Создает путь до указанного файла или директории.

void line\_finder – функция обработки файла. Заполняет соответствующие поля структуры Line, исходя из данных файла.

void file\_searcher – рекурсивная функция обхождения всей директории.

void output – функция, выводящая результат программы в файл.

int cmp – функция-компаратор. Сравнивает две структуры Line по полю num.

Работа программы заключается в прохождении всей директории, нахождению необходимых файлов и считывания с них информации в массив структур Line\*\*. Затем этот массив сортируется по полю num и отсортированный массив выводится в файл.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | HeLlO | ./tmp/asdfgh/mkoipu/H.txt  ./tmp/qwerty/e.txt  ./tmp/qwerty/qwert/L.txt  ./tmp/asdfgh/l.txt  ./tmp/asdfgh/O.txt7 | Программа обрабатывает директорию и выбирает файлы из одной буквы, чьи названия образуют заданную последовательность |
|  | abc | ./tmp/sub1/a.txt  ./tmp/sub1/b.txt  ./tmp/sub2/c.txt | Ищется заданная последовательность, притом игнорируются файлы с названием из нескольких букв |
| 1. hf | aBc | ./tmp/sub1/a.txt  ./tmp/sub2/B.txt  ./tmp/sub2/c.txt | Ищется заданная последовательность, притом игнорируются файлы с названием из нескольких букв и файлы с нужными буквами неподходящего регистра |

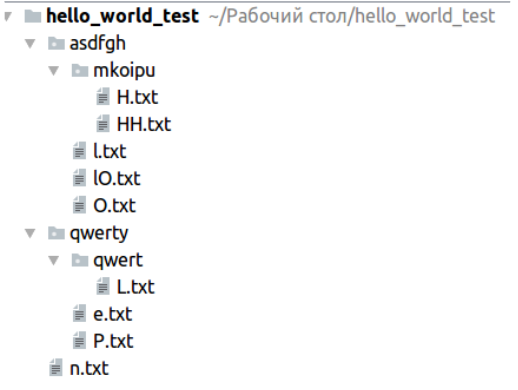


Рисунок 1- Файловое дерево для тестирования №1

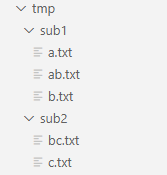


Рисунок 2 – Файловое дерево для тестирования №2

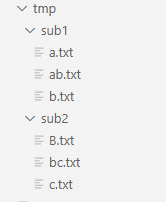


Рисунок 3 – Файловое дерево для тестирования №3

## Выводы

В ходе выполнения данной работы были приобретены навыки эффективного использования рекурсивных методов для обхода сложных структур данных, а также работы с файловой системой, анализа содержимого текстовых файлов и выполнения математических операций в соответствии с заданными правилами. 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: solution.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <dirent.h>

#define MAX\_STRING 1000

#define BLOCK 1000

#define RESET\_DIR ".."

#define PREV\_DIR "."

#define CUR\_DIR "./"

#define TEXT\_FOR ".txt"

typedef struct Line{

long long num;

char\* text;

} Line;

char\* pathcat(char\* cur\_path, char\* new\_dir){

char\* new\_path = malloc((strlen(cur\_path) + strlen(new\_dir) + 2) \* sizeof(char));

sprintf(new\_path, "%s/%s", cur\_path, new\_dir);

return new\_path;

}

void line\_finder(char\* fpath, Line\*\* line\_list, long long\* number\_of\_lines){

FILE\* file = fopen(fpath, "r");

if(file){

fscanf(file, "%lld ", &((\*line\_list)[\*number\_of\_lines].num));

int line\_size = 0;

int capacity = 1;

char \*new\_line = (char \*)calloc(1, BLOCK \* sizeof(char));

char new\_char;

while((new\_char = fgetc(file)) != EOF){

new\_line[line\_size] = new\_char;

line\_size++;

if(line\_size - 1 >= capacity \* BLOCK){

capacity \*= 2;

new\_line = (char \*)realloc(new\_line, capacity \* BLOCK \* sizeof(char));

}

}

(\*line\_list)[\*number\_of\_lines].text = new\_line;

(\*number\_of\_lines)++;

}

fclose(file);

}

void file\_searcher(char\* cur\_path, Line\*\* line\_list, long long\* number\_of\_lines, int\* capacity){

DIR\* cur\_dir = opendir(cur\_path);

if(cur\_dir){

struct dirent\* dir = readdir(cur\_dir);

while(dir){

char\* new\_path = pathcat(cur\_path, dir->d\_name);

if(dir->d\_type == DT\_REG && strstr(dir->d\_name, TEXT\_FOR) != NULL){

if((\*capacity)\*BLOCK < (\*number\_of\_lines) + 1){

(\*capacity)++;

\*line\_list = (Line\*)realloc(\*line\_list, (\*capacity) \* BLOCK \* sizeof(Line));

}

line\_finder(new\_path, line\_list, number\_of\_lines);

} else if(dir->d\_type == DT\_DIR && strcmp(dir->d\_name, PREV\_DIR) != 0 && strcmp(dir->d\_name, RESET\_DIR) != 0){

file\_searcher(new\_path, line\_list, number\_of\_lines, capacity);

}

dir = readdir(cur\_dir);

}

closedir(cur\_dir);

}

}

void output(Line\*\* line\_list, long long number\_of\_lines, char\* file\_name){

FILE \*file = fopen(file\_name, "w");

if(file){

for(long long i = 0; i < number\_of\_lines; i++){

fprintf(file, "%lld %s", (\*line\_list)[i].num, (\*line\_list)[i].text);

if(i < number\_of\_lines - 1)

fprintf(file, "\n");

}

}

fclose(file);

}

int cmp(const void\* a, const void\* b){

return ((Line\*)a)->num > ((Line\*)b)->num;

}

int main(){

Line\* line\_list = (Line\*)calloc(BLOCK, sizeof(Line));

long long number\_of\_lines = 0;

int capacity = 1;

file\_searcher(CUR\_DIR, &line\_list, &number\_of\_lines, &capacity);

qsort(line\_list, number\_of\_lines, sizeof(Line), cmp);

output(&line\_list, number\_of\_lines, "./result.txt");

return 0;

}