**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: РЕКУРСИЯ, ЦИКЛЫ, РЕКУРСИВНЫЙ ОБХОД

ФАЙЛОВОЙ СИСТЕМЫ

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Моисеева А.Е. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Цель данной работы заключается в изучении и применении рекурсивных функций, а также в освоении работы с файловой системой, включая ее рекурсивное исследование. Для успешного выполнения поставленной цели необходимо выполнить ряд задач:

-Познакомиться с концепцией рекурсии;

-Научиться создавать рекурсивные функции на языке Си;

-Изучить методы взаимодействия с файловой системой на языке Си;

-Разработать программу для рекурсивного просмотра файлов в директории, включая поиск в поддиректориях.

## Задание

Вариант 4

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 качестве имени файла используется символ латинского алфавита.

На вход программе подается строка. Требуется найти и вывести последовательность полных путей файлов, имена которых образуют эту строку.

Входная строка:

HeLlO

Правильный ответ:

hello\_world\_test/asdfgh/mkoipu/H.txt

hello\_world\_test/qwerty/e.txt

hello\_world\_test/qwerty/qwert/L.txt

hello\_world\_test/asdfgh/l.txt

hello\_world\_test/asdfgh/O.txt

! Регистрозависимость

! Могут встречаться файлы, в имени которых есть несколько букв и эти файлы использовать нельзя.

! Одна буква может встречаться один раз.

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

1. Язык программирования C предоставляет мощные и гибкие средства для работы с файлами и каталогами, что позволяет программистам создавать сложные приложения, включая те, что требуют обхода файловой системы. Этот процесс зачастую включает в себя рекурсию, метод, при котором функция вызывает саму себя для выполнения задачи. Работа с файловой иерархией в C требует понимания базовых операций с файлами и каталогами, а также умения применять рекурсивные алгоритмы для их эффективного обхода.

2. Файловая система организована в виде иерархии, которая включает в себя каталоги (или папки) и файлы. Каталоги могут содержать другие каталоги и файлы, образуя древовидную структуру. Для работы с этой структурой программа должна быть способна открывать каталоги, читать их содержимое, а также выполнять операции с файлами.

3. Рекурсия — это техника в программировании, при которой функция вызывает саму себя. Она идеально подходит для обработки структур, которые имеют вложенную, иерархическую организацию, как, например, файловая система. Рекурсия позволяет эффективно обходить все каталоги и файлы, начиная с заданной точки, и выполнять необходимые операции (например, поиск, копирование или перемещение).

## Выполнение работы

1. Функция void search\_letter(char, char\*, FILE\*):

принимает на вход букву, поиск которой будет осуществляться в названиях файлов, путь к директории, где будет происходить поиск и файловый поток для записи результата. Сначала открывается переданная директория, считывается её содержимое, проверяется каждый элемент. Если он является директорией (за исключением текущей и родительской), функция вызывает сама себя рекурсивно для этой директории. Таким образом, осуществляется рекурсивный спуск по директориям. Если элемент не является директорией, проверяется соответствие имени файла критериям (длина имени равна 5 и первый символ совпадает с *letter*) и, в случае совпадения, записывается путь к файлу в файл с результатом.

1. Функция main:

Считывается слово, посимвольный поиск которого будет осуществляться. Инициализируется путь к директории и файл для записи. Далее перебираются все буквы введенного слова, вызывается *search\_letter* для каждой из них для поиска файлов, соответствующих условию. В конце закрывается файл с результатами и завершается программа.

## 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 |

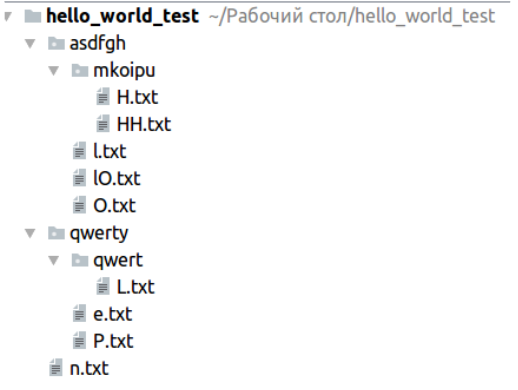


Рисунок 1- Файловое дерево для тестирования №1

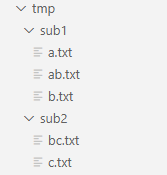


Рисунок 2 – Файловое дерево для тестирования №2

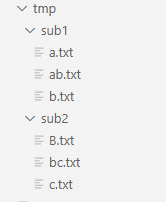


Рисунок 3 – Файловое дерево для тестирования №3

## Выводы

Поставленная цель достигнута, освоена работа с файловой системой, изучены и применены в программе рекурсивные функции для обхода файловой системы. В результате разработана программа для рекурсивного просмотра файлов в директории, включая поиск в поддиректориях, выбирающая файлы с определёнными названиями.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdio.h>

#include <string.h>

#include <dirent.h>

#include <stdbool.h>

#include <ctype.h>

#define BUF 1024

#define SEP\_PATH "/"

#define END\_LINE '\0'

#define CURRENT\_DIRECTORY "."

#define PARENT\_DIRECTORY ".."

bool is\_valid\_directory(struct dirent \*entity) {

return (entity->d\_type == DT\_DIR && strcmp(entity->d\_name, CURRENT\_DIRECTORY) != 0 && strcmp(entity->d\_name, PARENT\_DIRECTORY) != 0);

}

void print\_in\_file(FILE\* file, const char \*path, char \*line){

fprintf(file, "%s%s%s\n", path, SEP\_PATH, line);

}

void search\_letter(char letter, const char \*path, FILE \*file) {

DIR\* directory = opendir(path);

if (directory != NULL) {

struct dirent\* entity;

while ((entity = readdir(directory)) != NULL) {

if (is\_valid\_directory(entity)) {

char new\_path[strlen(path) + strlen(entity->d\_name) + 1];

new\_path[0] = END\_LINE;

strcat(new\_path, path);

strcat(new\_path, SEP\_PATH);

strcat(new\_path, entity->d\_name);

search\_letter(letter, new\_path, file);

}

else if (strlen(entity->d\_name) == 5 && entity->d\_name[0] == letter){

print\_in\_file(file, path, entity->d\_name);

}

}

closedir(directory);

}

}

void work\_with\_file(const char \*word, const char \*path, const char \*filename) {

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

if (file != NULL) {

for (int i = 0; i < strlen(word); i++) {

search\_letter(word[i], path, file);

}

}

fclose(file);

}

int main() {

char word[BUF];

scanf("%s", word);

const char \*path = "./tmp";

const char \*output\_filename = "result.txt";

work\_with\_file(word, path, output\_filename);

return 0;

}