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

Студентка гр. 3341 _____ Мильхерт А.С.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Для освоения методов работы с текстовыми строками в языке программирования C, рекурсивных алгоритмов и циклических структур данных, а также для углубленного понимания процессов обхода деревьев каталогов файловой системы следует выполнить такие шаги:

1. Изучить представление и операции над строками в языке C.
2. Освоить динамическое выделение и освобождение памяти для работы со строками.
3. Применить полученные знания для чтения и обработки текстовых файлов.
4. Реализовать рекурсивную функцию для обхода дерева каталогов.
5. Использовать циклические структуры для выполнения задачи сортировки строк.
6. Изучить и применить регулярные выражения для фильтрации файлов по расширению.
7. Создать структурированное решение для записи результатов обработки в выходной файл.

Задание

Вариант 3

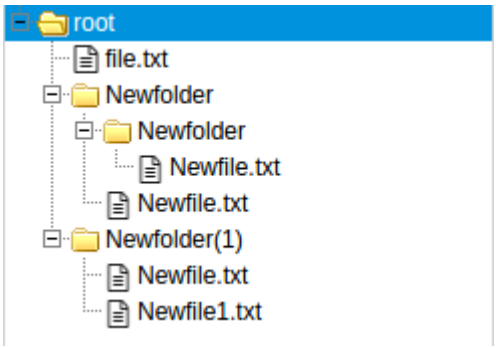
Дана некоторая корневая директория, в которой может находиться некоторое количество папок, в том числе вложенных. В этих папках хранятся некоторые текстовые файлы, имеющие имя вида <*filename*>.txt

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

Пример



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

Решение:

1 Small text

2 Simple text

3 Wow? Text?

4 Where am I?

5 So much files!

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

Был создан односвязный список для хранения сообщений из текстовых файлов. Struct Node имеет поля text – само сообщение, next – указатель на следующий элемент. Функция cr_Node – принимает на вход строку – сообщение и возвращает указатель на созданный ей объект Node, поле text которого равно входной строке, а поле next равно NULL. Функция add_Node позволяет сортировать список на вводе, т.е. принимая на вход строку, котору требуется вставить и указатель на указатель на голову списка, функция сначала создаёт структуру Node с соответствующим сообщением, а затем, проходя по списку, вставляет его либо после последнего элемента, либо после элемента, после которого следует элемент со строкой, начинающейся с большего, чем во входном тексте первым числом. Если список пуст (head == NULL), то присваивает head значение добавляемого элемента. Если число меньше чем в head, то делает элемент новой головой списка. Prnt_Node – получает на вход указатель на голову списка и печатает по порядку сообщения узлов (эта функция использовалась для тестирования программы). Free_Node – освобождает память, занимаемую списком.

Функция dir_func принимает на вход строку – путь до директории относительно той директории, в которой была запущена программа, и указатель на указатель на голову списка, в который будут сохранены сообщения. Функция с помощью библиотечной функции readdir проходит по элементам директории и, если он является текстовым файлом, с помощью функции add_Node добавляет сообщение в список или, если он является поддиректорием, вызывает саму себя от пути данной поддиректории и того же указателя на указатель на голову списка. Т.е. функция рекуррентно проходит по всем поддиректориям.

Функция main осуществляет вызов функции dir_func от текущей директории (строки «.») и от пустого списка (head = NULL), далее, с помощью цикла while содержимое списка записывается в файл result.txt, после чего память, занимаемая списком освобождается.

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!	OK
2.	root/notATxtFileAtAll: 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.c: 1 Small text	2 Simple text 3 Wow? Text? 5 So much files!	OK

Выводы

В ходе выполнения данной работы были углублены знания и практические навыки в области программирования на языке C, особенно в части работы со строками, рекурсией и циклическими алгоритмами. Было достигнуто понимание механизмов работы с файловой системой, а также разработаны методы для рекурсивного обхода дерева каталогов с целью поиска, чтения и обработки текстовых файлов. Реализация сортировки данных, извлеченных из файлов, демонстрирует важность алгоритмов сортировки в обработке и структурировании информации.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

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

typedef struct Node{
    char* text;
    struct Node* next;
}Node;

Node* cr_Node(char* txt){
    Node* tmp = malloc(sizeof(Node));
    tmp->text = malloc(sizeof(char)*(strlen(txt)+1));
    strcpy(tmp->text, txt);
    tmp->next = NULL;
    return tmp;
}

void add_Node(Node** head, char* txt){
    Node* tmp = cr_Node(txt);
    if(!*head){
        tmp->next = *head;
        *head = tmp;
    }else if(atoi(tmp->text) < atoi((*head)->text)){
        tmp->next = *head;
        *head = tmp;
    }else{
        Node* cur = *head;
        while(cur->next){
            if(atoi(cur->next->text) > atoi(tmp->text)){
                break;
            }else {
                cur = cur->next;
            }
        }
        Node* t = cur->next;
        cur->next = tmp;
        tmp->next = t;
    }
}

void prnt_Node(Node* head){
    Node* cur = head;
    while(cur){
        printf("%s\n", cur->text);
        cur = cur->next;
    }
}

void dir_func(const char* dir_name, Node** head){
    DIR* dir = opendir(dir_name);
```

```
    struct dirent* de;
    FILE* fp;
    while(de = readdir(dir)){
        if(de->d_type == 8 && strstr(de->d_name, ".txt")){

            char str[120];
            strcpy(str, dir_name);
            strcat(str, "/");
            strcat(str, de->d_name);
            fp = fopen(str, "r");

            char* ans = calloc(100, sizeof(char));
            fgets(ans, 99, fp);
            ans[strlen(ans)] = '\0';
            add_Node(head, ans);
            free(ans);

            fclose(fp);
        }
        if(de->d_type == 4 && strcmp(de->d_name, ".") && strcmp(de->d_name,
"..")){
            char* str = calloc(120, sizeof(char));
            strcpy(str, dir_name);
            strcat(str, "/");
            strcat(str, de->d_name);
            dir_func(str, head);
            free(str);
        }
    }
    closedir(dir);
}

void free_Node(Node* head){
    Node* tmp;
    while(head){
        tmp = head->next;
        free(head->text);
        free(head);
        head = tmp;
    }
}

int main() {
    char dir_name[5] = ".";
    Node* head = NULL;
    dir_func(dir_name, &head);
    FILE* result = fopen("result.txt", "w");
    Node* cur = head;
    while(cur){
        fprintf(result, "%s\n", cur->text);
        cur = cur->next;
    }
    free_Node(head);
    return 0;
};
```