

**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. 0382

Осинкин Е. А.

Преподаватель

Берленко Т. А.

Санкт-Петербург

2021

## **Цель работы.**

Научиться работать с файловой системой с помощью функций языка Си.

## **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, запускаясь 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

## **Основные теоретические положения.**

Для работы с деревом файловой системы используется библиотека *dirent.h*.

Рассмотрим основные функции.

Для того, чтобы получить доступ к содержимому некоторой директории можно использовать функцию:

```
DIR *opendir(const char *dirname);
```

Которая возвращает указатель на объект типа `DIR` с помощью которого можно из программы работать с заданной директорией. Тип `DIR` представляет собой поток содержимого директории.

Для того, чтобы получить очередной элемент этого потока, используется функция:

```
struct dirent *readdir(DIR *dirp);
```

Она возвращает указатель на объект структуры *dirent*, в котором хранится информация о файле. Название файла содержит поле *d\_name*.

После завершения работы с содержимым директории, необходимо вызвать функцию:

```
int closedir(DIR *dirp);
```

Передавая ей полученный функцией *readdir()* ранее дескриптор.

### **Выполнение работы.**

Функция *int main()*:

Открывается директория *tmp* и проверяется, папка с каким названием там лежит. Далее в зависимости от названия вызывается функция *mul* или *add*, и результат её работы записывается в файл *result.txt*.

Функция *long long add(char \*rootPath)*:

Рекурсивная функция, принимающая на вход путь к текущей папке. Перебирает все файлы, которые есть в текущей директории. Если встречает папку с названием *add* или *mul*, вызывает соответствующую функцию. Если найден файл, то с помощью *fscanf()* считывает из него числа и складывает с результатом *res*.

Функция *long long mul(char \*rootPath)*:

функция работает аналогично *add*, только все действия над *res* заменены на умножение.

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
1.	 <p>file.txt: 1</p> <p>file1.txt: 1</p> <p>file2.txt: 2 2</p> <p>file3.txt: 7</p> <p>file4.txt: 1 2 3</p> <p>file5.txt: 3 -1</p>	226	Программа работает верно

### Выводы.

Была разработана программа на языке Си для работы с файловой системой.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

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

long long add(char *rootPath);

long long mul(char *rootPath);

long long add(char *rootPath) {
    FILE* file;
    DIR* dirList = opendir(rootPath);
    long long res = 0;
    long long n;
    char c;
    if (dirList) {
        struct dirent* dir = readdir(dirList);
        while (dir) {
            if (dir->d_name[0] == '.') {
                dir = readdir(dirList);
                continue;
            }
            if (strcmp(dir->d_name, "add") == 0) {
                strcat(rootPath, "/add");
                res = res + add(rootPath);
                rootPath[strlen(rootPath) - 4] = '\\0';
            }
            else if (strcmp(dir->d_name, "mul") == 0) {
                strcat(rootPath, "/mul");
                res = res + mul(rootPath);
                rootPath[strlen(rootPath) - 4] = '\\0';
            }
            else {
                strcat(rootPath, "/");
                strcat(rootPath, dir->d_name);
                file = fopen(rootPath, "r");
                fscanf(file, "%lld", &n);
                c = fgetc(file);
                res = res + n;
                while ((c != EOF) && (c != '\\n') && (fscanf(file,
"%lld", &n) != EOF)) {
                    c = fgetc(file);
                    res = res + n;
                }
                fclose(file);
                rootPath[strlen(rootPath) - strlen(dir->d_name) - 1]
= '\\0';
            }
            dir = readdir(dirList);
        }
        closedir(dirList);
        return res;
    }

long long mul(char *rootPath) {
```

```

FILE* file;
DIR* dirList = opendir(rootPath);
long long res = 1;
long long n;
long long flag = 0;
char c;
char cl;
if (dirList) {
    struct dirent* dir = readdir(dirList);
    while (dir) {
        if (dir->d_name[0] == '.') {
            dir = readdir(dirList);
            continue;
        }
        if (strcmp(dir->d_name, "add") == 0) {
            strcat(rootPath, "/add");
            res = res * add(rootPath);
            rootPath[strlen(rootPath) - 4] = '\\0';
        }
        else if (strcmp(dir->d_name, "mul") == 0) {
            strcat(rootPath, "/mul");
            res = res * mul(rootPath);
            rootPath[strlen(rootPath) - 4] = '\\0';
        }
        else {
            strcat(rootPath, "/");
            strcat(rootPath, dir->d_name);
            file = fopen(rootPath, "r");
            fscanf(file, "%lld", &n);
            c = fgetc(file);
            res = res * n;
            while ((c != EOF) && (c != '\\n') && (fscanf(file,
"%lld", &n) != EOF)) {
                c = fgetc(file);
                res = res * n;
            }
            fclose(file);
            rootPath[strlen(rootPath) - strlen(dir->d_name) - 1]
= '\\0';
        }
        dir = readdir(dirList);
    }
}
closedir(dirList);
return res;
}

int main() {
    char path[10000];
    FILE* result = fopen("./result.txt", "w");
    strcat(path, "./tmp");
    DIR* dirList = opendir(path);
    struct dirent* dir = readdir(dirList);
    if (dirList) {
        while (dir) {
            if (dir->d_name[0] == '.') {
                dir = readdir(dirList);
                continue;
            }
            if (strcmp(dir->d_name, "add")) {
                fprintf(result, "%lld", add(path));
                dir = readdir(dirList);
            }
        }
    }
}

```

```
        continue;
    }
    if (strcmp(dir->d_name, "mul")) {
        fprintf(result, "%lld", mul(path));
        dir = readdir(dirList);
        continue;
    }
}
return 0;
}
```