МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: Рекурсивный обход файловой системы

Студент гр. 9304 Мохаммед А.А

Преподаватель Чайка К.В.

Санкт-Петербург

2020

Цель работы.

Научиться составлять и использовать рекурсивные алгоритмы, понимать, в каких случаях предпочтительнее рекурсивный алгоритм; научится работать с функциями из библиотеки dirent.

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 корневойдиректории, 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, томатематическая 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 вычисляютсязначения выражений, определяемые ими, а после используются уже этизначения.

Основные теоретические положения.

Рекурсия — в общем смысле включение неким алгоритмом самого себя. В частности, самый простой пример рекурсии в программировании — рекурсивная (вызывающая себя) функция. Необходимой составляющей рекурсивного алгоритма является база — условие, при котором алгоритм должен завершиться.

Заголовочный файл dirent.h включает:

— Определение типа DIR — директорного потока;
— Определение структуры dirent с полями:

- ino_t d_ino
- char d_name[256]
- off_t d_off
- unsigned char d_type
- unsigned short d_reclen

Структура dirent позволяет узнать информацию о файле или каталоге, в том числе имя файла и его тип (например, обычный файл или каталог).

— Определение функций:

- DIR *opendir(const char *); — «открыть» директорию;
- struct dirent *readdir(DIR *); — получить следующий элемент директории;
- int closedir(DIR *); — «закрыть» директорию.

Выполнение работы.

Разработанный алгоритм решает поставленную задачу следующим образом:

1) Каждый текстовый файл в директории поочерёдно открывается начтение , инадвсе мисчитанными из файла числами производится операция, соответствующая названию папки. Результат вычислений сохраняется в счётчик;

2) Для каждой вложенной папки к счётчику прибавляется или умножается на него результат применения этого же алгоритма к вложеннойпапки;
3) Возвращается результат работы (счётчик). Полный кодпрограммы см. в Приложении А.

Выводы.

Были изучены рекурсивные алгоритмы. Была написана программа, использующая рекурсивный алгоритм для обхода файловой системы и вычисления арифметического выражения. Для работы с файловой системой использовались функции библиотеки dirent.h.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: solution.c

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

```
#include<dirent.h>
#include<string.h>
#include<stdlib.h>
#include<sys/types.h>
#define MAX_LENGTH 2000

int is_add(char* folder){
    return (strcmp(folder, "add")==0)?1:0;
}

void error(){
    perror("error\n");
    exit(EXIT_FAILURE);
}

char* line;
char* copy_line;
int access_file(char* path, int is_add_operation){
    int acc = 1;
    if (is_add_operation) acc=0;
    FILE *in_file;
    line[0]='\0';

    if ((in_file=fopen(path, "r")) == NULL) error();
    fgets(line, MAX_LENGTH, in_file);
```

```
    strcpy(copy_line, line);
    line[0]='\0';
    char* word = strtok(copy_line, " \n");
    while (word!=NULL) {
        if (is_add_operation) acc+=atoi(word);
        else acc*=atoi(word);
        word = strtok(NULL, " \n");
    }
    fclose(in_file);
    return acc;
}

int access_dir(char* root, int is_add_operation){

  int acc = 1;
  if (is_add_operation) acc=0;
  DIR* dir = opendir(root);
  if (dir == NULL){
    error();
    return acc;
  }
  int is_add_operation_temp = 1;
  struct dirent *de;
  char* path = (char*)malloc(MAX_LENGTH* sizeof (char));
```

```
  while ((de = readdir(dir))!=NULL){
      path[0] = '\0';
      strcpy(path, root);
      strcat(path, "/");
      strcat(path, de->d_name);
      path[strlen(path)] = '\0';
    if (de->d_type==DT_DIR && strcmp(de->d_name,".")!=0
     && strcmp(de->d_name,"..")!=0
    && (strcmp(de->d_name, "add")==0 || strcmp(de->d_name, "mul")==0)){
        if (is_add(de->d_name)) is_add_operation_temp = 1;
        else is_add_operation_temp = 0;
        if (is_add_operation)
            acc += access_dir(path, is_add_operation_temp);
        else
            acc *= access_dir(path, is_add_operation_temp);
    }else if (de->d_type==DT_REG){
      if (is_add_operation)
          acc += access_file(path, is_add_operation);
      else
          acc *= access_file(path, is_add_operation);
    }
}
free(path);
free(de);
closedir(dir);
```

```
return acc;
}



int main(void) {
    char l[] ="tmp";
    line = (char*)malloc(MAX_LENGTH*sizeof (char));
    copy_line = strdup(line);
    int value = access_dir(l, 1);
    FILE* file;
    file = fopen("result.txt", "w");
    if (file==NULL) error();
    char* str = (char*)malloc(18*sizeof (char));
    snprintf(str, 16, "%d", value);
    fputs(str, file);
    fclose(file);
    return 0;
}
```