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

# ОТЧЕТ

**по лабораторной работе №2**
**по дисциплине «Программирование»**
**Тема: «Обход файловой системы»**

Студент гр. 9383 _____ Ноздрин В.В.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2020

**Цель работы.**

Реализовать программу, выполняющую рекурсинвый обход директории на языке программирования Си.

**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.

Пример

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***.

**Выполнение работы.**

Переменные:

***myDict –*** структура, которая хранит в себе строку – исходное слово, и массив строк – пути к соответствующим файлам. Так же структура хранит длину исходного слова.

Функции:

***isValid –*** функция с помощью регулярного выражения проверяет, что имя файла состоит из одного символа и формат файла - ***txt***

***save –*** функция принимает на вход структуру ***dict*** и путь. Если Файл подходит, в ***dict*** записывает путь для файла с соответствующей буквой

***listDir –*** функция выполняет рекурсивный проход по директории и для каждого ***txt*** файла, имя которого состоит из одного символа, вызывает функцию ***save***

**Тестирование.**

Результаты тестирования совпадают с примером из условия

**Вывод.**

Была изучена структура библиотека ***dirent.h*** в языке программирования Си.

Разработана программа в выполняющая рекурсивный обход директории.

## ПРИЛОЖЕНИЕ А
## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: **main.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <regex.h>
#include <dirent.h>
#include <sys/types.h>

typedef struct {
    char *letters;
    int len;
    char **routes;
} myDict;

int isValid(char *filename);
void save (const char *route, myDict *dict);
void listDir (char *startDir, myDict *dict);

int main (int argc, char **argv, char **env)
{
    if (argc != 2) { printf ("use ./prog <path>\n"); return 0; }

    char *letters = (char*) malloc (NAME_MAX * sizeof(char));
    scanf("%s", letters);
    char **routes = (char**) calloc (strlen(letters), sizeof(char*));
    myDict dict = {letters, strlen(letters), routes};

    listDir (argv[1], &dict);

    int len = strlen(dict.letters);
    for (int i = 0; i < len; i++)
    {
        printf("%s\n", routes[i]? routes[i]: "NULL");
    }
```

```
    return 0;
}

void save (const char *route, myDict *dict)
{
    char c = route[strlen(route)-5];
    //printf("[%c]%s\n", c, route);
    for (int i = 0; i < dict->len; i++)
        if (dict->letters[i] == c)
        {
            dict->routes[i] = malloc ((strlen(route)+1) * sizeof(char));
            strcpy(dict->routes[i], route);
            return;
        }
}

void listDir (char *startDir, myDict *dict)
{
    char *regexp = "^[[:alpha:]]\\.txt$";
    regex_t regexComp;
      if (regcomp (&regexComp, regexp, REG_EXTENDED)) { perror("regexCompile"); return
exit(1); }

    char nextDir[NAME_MAX]={0};
    strcpy(nextDir, startDir);
    DIR *dir = opendir(startDir);
    struct dirent *entry;

    if (!dir) { perror("diropen"); exit(1); }
    while ((entry = readdir (dir)))
    {
        if (!strcmp(entry->d_name, ".")||!strcmp(entry->d_name, ".."))
            continue;

        if (entry->d_type == DT_DIR)
        {
            int len = strlen (nextDir);
```

```
                strcat (nextDir, "/");
                strcat (nextDir, entry->d_name);
                listDir (nextDir, dict);
                nextDir[len] = '\0';
            }

             if (entry->d_type == DT_REG && (regexec(&regexComp, entry->d_name, 0, NULL, 0)
== 0))
            {
                int len = strlen (nextDir);
                strcat (nextDir, "/");
                strcat (nextDir, entry->d_name);
                save (nextDir, dict);
                nextDir[len] = '\0';
            }
        }
        closedir(dir);

        regfree(&regexComp);
    }
```