МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра МОЭВМ

отчет

по лабораторной работе №3

по дисциплине «Программирование»

Тема: [Строки. Рекурсия, циклы, обход дерева](https://stepik.org/course/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-(%D0%9B%D0%AD%D0%A2%D0%98)-1096/syllabus?module=12)

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6303 |  | Горбунова А.П. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель**: Написать программу, которая находит и выводит последовательность полных путей файлов, имена которых образуют строку.

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.

**Алгоритм работы программы**

ОТКРЫТИЕ ДИРЕКТОРИИ

СОЗДАНИЕ СТРУКТУРЫ dirrent ДЛЯ КАЖДОГО ЭЛЕМЕНТА ДИРЕКТОРИИ

ЭЛЕМЕНТ ДИРЕКТОРИИ

ФАЙЛ ДИРЕКТОРИЯ

ПРОВЕРКА НА .txt ФАЙЛ текущая директория

+ПРОВЕРКА НАЗВАНИЯ ФАЙЛА + «/»

+имя директории

ДА НЕТ

текущая директория пропуск рекурсивный вызов функции для директории

+ «/»

+имя файла

сравнение с другими путями

путь использовался путь не использовался

return 0; добавление в массив использованных путей

+

продолжение поиска вывод пути в консоль

return 1;

закрытие всех рекурсивно вызванных функций

**Ход работы:**

* Для выполнения данной работы необходимо подключить пять библиотек: «*stdio.h*», «*string.h*» (в которой содержится функции для работы со строками) и «stdlib.h» (в которой содержатся функция для работы с памятью) и #include <sys/types.h> и #include <dirent.h>

для работы с директориями и файлами

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <sys/types.h>  #include <dirent.h>  #include <stdlib.h> |

* Также до функции main интерпретируется структура, в которой содержится информация об использованных путях, а именно: динамический массив указателей, в котором содержатся строки – пути и переменная size - размер этого массива

typedef struct ListOfWays

{

char \*\*ways;

int size;

} ListOfWays;

* Функция main.

int main ()

{

ListOfWays listofways;

//создание структуры для хранения использованных путей к файлам

// в структуре: динамический массив указателей, его размер

//выделение памяти

listofways.ways = (char \*\*)malloc(SIZE\_OF\_LIST\*sizeof(char \*));

for (int i=0; i<SIZE\_OF\_LIST; i++)

(listofways.ways)[i] = (char \*)malloc(SIZE\_OF\_ONE\_WAY);

//начальный размер массива = 0

listofways.size = 0;

char str[SIZE\_OF\_STR];

//считывание строки

fgets (str, SIZE\_OF\_STR, stdin);

// вызов функции поиска путей для каждой буквы

for (int i = 0; i < (strlen(str)-1); i++)

{

FoundWay(".", str[i], &listofways);

}

//очищение памяти

for (int i=0; i<SIZE\_OF\_LIST; i++)

free((listofways.ways)[i]);

free(listofways.ways);

return 0;

}

* Функция compare

int compare (char\* way, ListOfWays \*listofways)

{

int foreach = 0;

for (int j = 0; j < (listofways->size - 1); j++)

{

if (strcmp (listofways->ways[j], way) == 0)

foreach++;

}

return foreach;

}

* Функция FoundWay

Функция ищет путь к файлу, название которого совпадает с данной буквой

Функция принимает начальную директорию, букву для поиска, структуру списка использованных путей; возвращает 0 - если файл не найден, 1 - если найден.

int FoundWay (char\* startDir, char letter, ListOfWays \*listofways)

{

//строка для записи текущего пути

char way[SIZE\_OF\_ONE\_WAY];

//копируем в строку текущую начальную директорию

strcpy (way, startDir);

//открытие директории

DIR \*dir = opendir (way);

if (dir)

//создание структуры dirrent для каждого встречающегося элемента директории

struct dirent {

ino\_t d\_ino; /\* Inode number \*/

off\_t d\_off; /\* Not an offset; see below \*/

unsigned short d\_reclen; /\* Length of this record \*/

unsigned char d\_type; /\* Type of file; not supported

by all filesystem types \*/

char d\_name[256]; /\* Null-terminated filename \*/

};

struct dirent \*de = readdir (dir);

while (de)

{

//если это файл && текстовый файл && название файла совпадает с данной буквой

if (de->d\_type == DT\_REG && strstr (de->d\_name, ".txt") && letter == de->d\_name[0])

{

//запись в переменную текущей длины строки пути

int waylen = strlen (way);

strcat (way, "/");

strcat (way, de->d\_name);

//добавление в строку слеш+имя файла

strcpy (listofways->ways[listofways->size], way);

listofways->size++;

//добавление в массив использованных путей текущего пути

if (!compare (way, listofways))

//вызов функции compare – проверка на совпадение с уже использованными путями

//если путь ранее не использовался – вывод в консоль

//иначе – продолжение поиска

{

printf ("%s\n", way);

closedir (dir);

return 1;

}//если нужный путь найден – вернуть 1, иначе - 0

//обрезка имени файла из строки//закрытие директории

way[waylen] = '\0';

closedir(dir);

return 0;

}

else if (de->d\_type == DT\_DIR && strcmp(".", de->d\_name)!=0 && strcmp("..", de->d\_name)!=0 )

//если элемент- новая директория

{

int waylen = strlen (way);

strcat (way, "/");

strcat (way, de->d\_name);

//рекурсивный вызов функции для текущей директории

if (FoundWay (way, letter, listofways)==1)

//если при этом вызове функции найден файл, вернуть – 1, иначе – 0

продолжение поиска до тех пор, пока функция не вернет 1.

{

closedir(dir);

return 1;

}

way[waylen] = '\0';

}

de = readdir(dir);

}

}

closedir(dir);

return 0;

}

ИСХОДНЫЙ КОД

#include <stdio.h>

#include <string.h>

#include <sys/types.h>

#include <dirent.h>

#include <stdlib.h>

#define SIZE\_OF\_LIST 10

#define SIZE\_OF\_ONE\_WAY 1000

#define SIZE\_OF\_STR 100

typedef struct ListOfWays

{

char \*\*ways;

int size;

} ListOfWays;

int compare (char\* way, ListOfWays \*listofways)

{

int foreach = 0;

for (int j = 0; j < (listofways->size - 1); j++)

{

if (strcmp (listofways->ways[j], way) == 0)

foreach++;

}

return foreach;

}

//функция поиска пути, принимает начальную директорию, букву для поиска,структуру списка использованных путей

//возвращает 0 - если файл не найден, 1 - если найден

int FoundWay(char\* startDir, char letter, ListOfWays \*listofways)

{

char way[SIZE\_OF\_ONE\_WAY];

strcpy (way, startDir);

DIR \*dir = opendir (way);

if (dir)

{

struct dirent \*de = readdir (dir);

while (de)

{

if (de->d\_type == DT\_REG && strstr (de->d\_name, ".txt") && letter == de->d\_name[0])

{

int waylen = strlen (way);

strcat (way, "/");

strcat (way, de->d\_name);

strcpy (listofways->ways[listofways->size], way);

listofways->size++;

if (!compare (way, listofways))

{

printf ("%s\n", way);

closedir (dir);

return 1;

}

way[waylen] = '\0';

closedir(dir);

return 0;

}

else if (de->d\_type == DT\_DIR && strcmp(".", de->d\_name)!=0 && strcmp("..", de->d\_name)!=0 )

{

int waylen = strlen (way);

strcat (way, "/");

strcat (way, de->d\_name);

if (FoundWay (way, letter, listofways)==1)

{

closedir(dir);

return 1;

}

way[waylen] = '\0';

}

de = readdir(dir);

}

}

closedir(dir);

return 0;

}

int main ()

{

ListOfWays listofways;

//создание структуры для хранения использованных путей к файлам ()в структуре: динамический массив указателей, его размер)

//выделение памяти

listofways.ways = (char \*\*)malloc(SIZE\_OF\_LIST\*sizeof(char \*));

for (int i=0; i<SIZE\_OF\_LIST; i++)

(listofways.ways)[i] = (char \*)malloc(SIZE\_OF\_ONE\_WAY);

listofways.size = 0;

char str[SIZE\_OF\_STR];

fgets (str, SIZE\_OF\_STR, stdin);

//считывание строки и вызов функции поиска путей для каждой буквы

for (int i = 0; i < (strlen(str)-1); i++)

{

FoundWay(".", str[i], &listofways);

}

//очищение памяти

for (int i=0; i<SIZE\_OF\_LIST; i++)

free((listofways.ways)[i]);

free(listofways.ways);

return 0;

}

**Пример работы программы**

**1)**

**Input:**

name

**Output:**

./ngfg/asdasd/n.txt

./aslkmda/reew/asdad/a.txt

./dklsam/m.txt

./aslkmda/dfs/e.txt

**2)**

**Input:**

kdsjld

**Output:**

./dklsam/msdsa/k.txt

./aslkmda/d.txt

./dklsam/ndsksn/s.txt

./dnksdsnkf/j.txt

./dklsam/l.txt

./dnksdsnkf/dkwdw/d.txt

**3)**

**Input:**

nnamme

**Output:**

./ngfg/asdasd/n.txt

./ngfg/dfsd/n.txt

./aslkmda/reew/asdad/a.txt

./dklsam/m.txt

./dnksdsnkf/m.txt

./aslkmda/dfs/e.txt

**Добавление файлов на Github**

### Добавление файлов на Github было произведено с помощью консоли

### Переход на ветку master и создания + переход в новую ветку Gorbunova\_Alina\_sem2\_lr3 :

git checkout master && git pull origin master

git checkout -b Gorbunova\_Alina\_sem2\_lr3

* Cозданиe папки :

mkdir Gorbunova\_Alina\_sem2\_lr3

* Добавлениe файлов, создание коммита и отправка коммитов на GitHub :

git add Gorbunova\_Alina\_sem2\_lr3/lab3.c

git commit -m "Gorbunova LR №3 done"

git push origin Gorbunova\_Alina\_sem2\_lr3

**Вывод:** в ходе лабораторной работы получены навыки использования различных функций стандартных библиотек, изучены методы работы с директориями и файлами. В результате лабораторной работы была написана программа, которая находит и выводит последовательность полных путей файлов, имена которых образуют строку.