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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование»
Тема: Обход дерева

Студент гр. 6304 _____ Васильев А.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

Оглавление

Цель работы 3
Формулировка задачи 3
Реализация функций 4
1. **Функция open_txt_file** 4
2. **Функция read_catalog** 4
3. **Функция main** 6
4. **Функция-компаатор для qsort** 6

Разбиение на файлы 7
1. **Заголовочный файл** 7
2. **Makefile** 7

Тестирование 8
Вывод 9
Приложение 1 10

Цель работы

Создание функций для обхода дерева из папок и поиска объектов с заданными параметрами среди них.

Формулировка задачи

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 дирректории, выведет строки из файлов всех поддиректорий в порядке возрастания числа, с которого строки начинаются

Реализация функций

1. Функция open_txt_file

Для начала требуется создать функцию, которая будет выполнять указанные действия с уже найденным txt файлом. В ней требуется учесть две исключительные ситуации: файл поврежден, файл является пустым. В этих случаях из файлов ничего не выводится.

```
char* open_txt_file(char* file_name)
{
/*Открытие файла. Значение NULL предусмотрено в функции "read_catalog",
которая вызывает данную функцию.*/
	FILE* txt_file = fopen(file_name, "r");
	if(txt_file == NULL)
	{
		printf("error!\n");
		return NULL;
	}
/*Определение размера файла для корректного выделения памяти*/
	fseek(txt_file,0, SEEK_END);
	int file_size = ftell(txt_file);
	fseek(txt_file,0, SEEK_SET);


/*Если размер равен нулю, то сообщается, что файл пуст.
Если размер ненулевой, то выделяется память под строку.*/
	if(file_size == 0)
	{
		fclose(txt_file);
		printf("<!file %s is empty!>\n", file_name);
		return NULL;
	}
	char* string_in_file = (char*)malloc(sizeof(char)*(file_size+100));


/*Первая строка txt файла считывается в выделенную память.
Файл закрывается, указатель на строку возвращается.*/
	fgets(string_in_file, file_size*sizeof(char), txt_file);
	fclose(txt_file);
	return string_in_file;
}
```

2. Функция read_catalog

Теперь можно реализовать функцию для непосредственного обхода дерева папок.

```
void read_catalog(const char* dir_name,char** strings_in_txt, int* len)
{/*Выделяется память под строку, в которую будет помещаться путь к файлам.
Используется макрос NAME_MAX, определяющий максимально возможную длину имени файла*/
	char* current_path = (char*)malloc(sizeof(char)*(strlen(dir_name)+NAME_MAX));
	strcpy(current_path,dir_name);
/*Открывается поток каталога и проверяется на корректное открытие.
Далее с помощью функции readdir считывается структура с информацией
о первом файле в каталоге.*/
	DIR *current_dir = opendir(current_path);
	if(current_dir == NULL)
		return;
	dirent* file_in_current_dir = (dirent*)readdir(current_dir);
/*Цикл выполнятеется, пока в текущей директории будут объекты*/
		while(file_in_current_dir)
		{/*К строке, содержащий путь, добавляется имя объекта*/
			int path_len = strlen(current_path);
			strcat(current_path, "/");
			strcat(current_path, file_in_current_dir->d_name);
			printf("[%s]\n", current_path); //Печать для контроля (можно
отключить).
/*С помощью спец макросов проверяется, является ли объект файлом или директорией.*/
			if ( file_in_current_dir->d_type == DT_REG &&
			strstr(file_in_current_dir->d_name, ".txt")!=NULL)
			{
/*Если объект - это неповрежденный и непустой (данные условия проверяются в
open_txt_file) txt файл, то возвращается указатель на первую строку из него. (*/
				if((strings_in_txt[*len] =
				(char*)open_txt_file(current_path))!=NULL )
					(*len)++;
			}
			if(file_in_current_dir->d_type == DT_DIR &&
			strcmp(".",file_in_current_dir->d_name)!=0 &&
			strcmp("..",file_in_current_dir->d_name)!=0)
			{
/*Если объект - это директория (причем не родительская и не текущая), то рекурсивно
вызывается функция read_catalog для прочтения данной директории*/
				read_catalog(current_path,strings_in_txt, len);
			}
/*После проверки текущего объекта current_path "обрезается" до первоначального
положения.
Далее считывается следующий объект из данной директории*/
			current_path[path_len] = '\0';
			file_in_current_dir = (dirent*)readdir(current_dir);
		}
/*После обхода всех объектов в директории, она закрывается*/
	closedir(current_dir);
}
```

3. Функция main

```
#define MEM_BLOCK 50

int main()
{
/*Выделяется память под массив указателей на строки*/
	char** strings = (char**)malloc(sizeof(char*)*MEM_BLOCK);
	int len = 0;

/*Специальная функция обходит директории и заполняет
массив из строк строками из файлов(реализация в "functions.c").
Далее выводится несортированные строки (для контроля).*/
	read_catalog(".",strings, &len);
	printf("\n\n");
	for (int i = 0; i < len; i++)
		printf("%s\n", strings[i]);

/*Строки сортируются библиотечной функцией,
компаратор для которой сравнивает числа в начале строк.
Выводится уже отсортированные строки.*/
	qsort(strings, len, sizeof(char*),compare);
	printf("\n\n");
	for (int i = 0; i < len; i++)
		printf("%s\n", strings[i]);

/*Освобождение памяти, выделенную под каждую строку.
Освобождение памяти под массив строк*/
	for(int i = 0; i < len; i++)
		free(strings[i]);
	free(strings);
	return 0;
}
```

4. Функция-компаратор для qsort

Функция-компаратор для qsort. Так как известно, что строка начинается с числа, то с помощью библиотечной функции можно вычислить разность между числами двух строк.

```
int compare(const void* a, const void* b)
{
	return atoi(*(char**)a) - atoi(*(char**)b);
}
```

Разбиение на файлы

1. Заголовочный файл

Для удобства возможной дальнейшей модификации программы можно разбить её на отдельные файлы, для сборки которых написать makefile.

Ниже приведен заголовочный файл “functions.h”, в котором содержатся все объявления функций, а также инструкция typedef, для удобства использования структур.

```
char* open_txt_file(char*);
void read_catalog(const char*, char**, int*);
int compare(const void*, const void*);
typedef struct dirent dirent;
```

Реализация самих функций помещена в файл “functions.c”. main помещена в отдельный файл “main.c”.

2. Makefile

Ниже приведен makefile для сборки программы

```
Lab3: main.o functions.o
	g++ main.o -o read_catalog.out functions.o
	rm *.o
main.o: main.c functions.h
	g++ -c main.c
functions.o: functions.c functions.h
	g++ -c functions.c
```

Тестирование

Для тестирования создадим систему из вложенных папок, в которых случайным образом помещены txt файлы со строчками из начала поэмы «Двенадцать» А. Блока.

Переместим собранную программу в корень и вызовем её.

Полученный результат:

```
4 На ногах не стоит человек.
6 На всем божьем свете!
3 Ветер, ветер!
5 Ветер, ветер -
7 Завивает ветер
2 Белый снег.
1 Черный вечер.
12 Скользит - ах, бедняжка!
10 Скользко, тяжко,
13              А. А. Блок - Двенадцать
11 Всякий ходок
9 Под снежком - ледок
8 Белый снежок.


1 Черный вечер.
2 Белый снег.
3 Ветер, ветер!
4 На ногах не стоит человек.
5 Ветер, ветер -
6 На всем божьем свете!
7 Завивает ветер
8 Белый снежок.
9 Под снежком - ледок
10 Скользко, тяжко,
11 Всякий ходок
12 Скользит - ах, бедняжка!
13              А. А. Блок - Двенадцать
andrey@andrey:~/Proga/2sem/Vasilev_Andrey_Lab3/root$
```

Программа отработала корректно.

Вывод

В ходе выполнения данной лабораторной работы была создана программа для обхода дерева из папок и функция для работы с файлами и подпапками. Также была реализована возможность добавления дополнительных функций, путем разбиения программы на отдельные файлы.

Приложение 1

Файл main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "functions.h"
#define MEM_BLOCK 50

int main()
{
/*Выделяется память под массив указателей на строки*/
	char** strings = (char**)malloc(sizeof(char*)*MEM_BLOCK);
	int len = 0;

/*Специальная функция обходит директории и заполняет
массив из строк строками из файлов(реализация в "functions.c").
Далее выводится несортированные строки (для контроля).*/
	read_catalog(".",strings, &len);
	printf("\n\n");
	for (int i = 0; i < len; i++)
		printf("%s\n", strings[i]);

/*Строки сортируются библиотечной функцией,
компаратор для которой сравнивает числа в начале строк.
Выводится уже отсортированные строки.*/
	qsort(strings, len, sizeof(char*),compare);
	printf("\n\n");
	for (int i = 0; i < len; i++)
		printf("%s\n", strings[i]);

/*Освобождение памяти, выделенную под каждую строку.
Освобождение памяти под массив строк*/
	for(int i = 0; i < len; i++)
		free(strings[i]);
	free(strings);
	return 0;
}
```

Файл functions.c

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <string.h>
#include <stdlib.h>
#include "functions.h"
```

```
char* open_txt_file(char* file_name)
{
/*Открытие файла. Значение NULL предусмотрено в функции "read_catalog",
которая вызывает данную функцию.*/
	FILE* txt_file = fopen(file_name, "r");
	if(txt_file == NULL)
	{
		printf("error!\n");
		return NULL;
	}
/*Определение размера файла для корректного выделения памяти*/
	fseek(txt_file,0, SEEK_END);
	int file_size = ftell(txt_file);
	fseek(txt_file,0, SEEK_SET);


/*Если размер равен нулю, то сообщается, что файл пуст.
Если размер ненулевой, то выделяется память под строку.*/
	if(file_size == 0)
	{
		fclose(txt_file);
		printf("<!file %s is empty!>\n", file_name);
		return NULL;
	}
	char* string_in_file = (char*)malloc(sizeof(char)*(file_size+100));


/*Первая строка txt файла считывается в выделенную память.
Файл закрывается, указатель на строку возвращается.*/
	fgets(string_in_file, file_size*sizeof(char), txt_file);
	fclose(txt_file);
	return string_in_file;
}

void read_catalog(const char* dir_name,char** strings_in_txt, int* len)
{
/*Выделяется память под строку, в которую будет помещаться путь к файлам.
Используется макрос NAME_MAX, определяющий максимально возможную длину имени файла*/
	char* current_path = (char*)malloc(sizeof(char)*(strlen(dir_name)+NAME_MAX));
	strcpy(current_path,dir_name);
/*Открывается поток каталога и проверяется на корректное открытие.
Далее с помощью функции readdir считывается структура с информацией
о первом файле в каталоге.*/
	DIR *current_dir = opendir(current_path);
	if(current_dir == NULL)
		return;
	dirent* file_in_current_dir = (dirent*)readdir(current_dir);
```

```
/*Цикл выполнятеется, пока в текущей директории будут объекты*/
            while(file_in_current_dir)
            {/*К строке, содержащий путь, добавляется имя объекта*/
                  int path_len = strlen(current_path);
                  strcat(current_path, "/");
                  strcat(current_path, file_in_current_dir->d_name);
                  printf("[%s]\n", current_path); //Печать для контроля (можно
отключить).
/*С помощью спец макросов проверяется, является ли объект файлом или директорией.*/
                  if ( file_in_current_dir->d_type == DT_REG &&
                  strstr(file_in_current_dir->d_name, ".txt")!=NULL)
                  {
/*Если объект - это неповрежденный и непустой (данные условия проверяются в
open_txt_file) txt файл, то возвращается указатель на первую строку из него. (*/
                        if((strings_in_txt[*len] =
                        (char*)open_txt_file(current_path))!=NULL )
                              (*len)++;
                  }
                  if(file_in_current_dir->d_type == DT_DIR &&
                  strcmp(".",file_in_current_dir->d_name)!=0 &&
                  strcmp("..",file_in_current_dir->d_name)!=0)
                  {
/*Если объект - это директория (причем не родительская и не текущая), то рекурсивно
вызывается функция read_catalog для прочтения данной директории*/
                        read_catalog(current_path,strings_in_txt, len);
                  }
/*После проверки текущего объекта current_path "обрезается" до первоначального
положения.
Далее считывается следующий объект из данной директории*/
                  current_path[path_len] = '\0';
                  file_in_current_dir = (dirent*)readdir(current_dir);
            }
/*После обхода всех объектов в директории, она закрывается*/
      closedir(current_dir);
}

int compare(const void* a, const void* b)
{
      return atoi(*(char**)a) - atoi(*(char**)b);
}
```

Файл functions.h

```
char* open_txt_file(char*);
void read_catalog(const char*, char**, int*);
int compare(const void*, const void*);
typedef struct dirent dirent;
```

Файл makefile

```
Lab3: main.o functions.o
	g++ main.o -o read_catalog.out functions.o
	rm *.o
main.o: main.c functions.h
	g++ -c main.c
functions.o: functions.c functions.h
	g++ -c functions.c
```