

Лабораторная работа №12

«Обработка текстовых файлов»

1. Цель работы

Целью лабораторной работы является приобретение навыка разработки на языке C++ программ, осуществляющих анализ и преобразование текстовых файлов, записанных в различных форматах.

2. Исходные данные

Рассмотрим некоторые детали стандартной библиотеки языка C++, которые могут понадобиться при выполнении лабораторной работы

2.1. Получение списка файлов в заданном каталоге

В стандартной библиотеке C++, к сожалению, нет средств для получения списка файлов в заданном каталоге. Поэтому нам придётся воспользоваться системными вызовами Linux.

Прежде чем получить список файлов, нужно открыть каталог с помощью функции `opendir`:

```
#include <sys/types.h>
#include <dirent.h>
#include DIR * opendir ( const char * name ) ;
```

Эта функция получает путь к каталогу и возвращает указатель на внутреннюю структуру данных типа `DIR`, через которую в дальнейшем можно работать с открытым каталогом. Если каталог по какой-то причине открыть не удалось, `opendir` возвращает `NULL`.

Чтение содержимого каталога осуществляется последовательными вызовами функции `readdir`:

```
#include <dirent.h>
struct dirent * readdir (DIR * dirp);
```

Эта функция получает указатель на структуру `DIR`, которую вернула функция `opendir`, и возвращает указатель на структуру `dirent`, содержащую описание очередного элемента каталога. В случае, если элементы каталога закончились, `readdir` возвращает `NULL`.

Имя элемента каталога можно получить из поля `d_name` структуры `dirent`. Обратите внимание на то, что имя не содержит пути к каталогу.

Элементы каталога, выдаваемые функцией `readdir`, могут представлять собой как файлы, так и подкаталоги. Чтобы разобраться, с файлом или подкаталогом мы имеем дело, нужно вызвать функцию `lstat`:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
int lstat ( const char * path, struct stat * buf);
```

Функция `lstat` получает путь к нужному файлу или каталогу, а также указатель на переменную типа `stat`, в которую в результате работы функции будет помещена детальная

информация о файле или каталоге.

Структура `stat` имеет поле `st_mode`. Если передать значение этого поля в макрос `S_ISREG`, то этот макрос вернёт 1, если мы имеем дело с обычным файлом, и 0 – если нам попался каталог.

После работы с каталогом, его нужно закрыть, передав указатель на `DIR` в функцию `closedir`:

```
#include <sys/types.h>
#include <dirent.h>
int closedir (DIR * dirp);
```

2.2 Чтение данных из текстового файла

Чтение из текстовых файлов обеспечивает класс `ifstream` стандартной библиотеки C++.

Чтобы воспользоваться этим классом, нужно подключить заголовочный файл `fstream`. Открытие файла и построчное чтение его содержимого можно организовать, объявив переменную типа `ifstream` и вызывая функцию `getline` для чтения каждой строки:

```
ifstream f ("somefile.txt") ;
string line;
while (getline(f,line)){
/* ... */
}
```

Обратите внимание на то, что программу, включающую этот фрагмент кода, нужно компилировать в режиме `c++11`, т.е. в командной строке `g++` нужно указывать ключ `-std=c++11`. В `c++98` данный фрагмент нужно переписать:

```
ifstream f ;
f.open("somefile.txt");
string line;
while (getline(f,line)){
/* ... */
}
```

2.3 Запись данных в текстовый файл

Запись в текстовый файл контролирует класс `ofstream`, объявленный в заголовочном файле `fstream`.

Конструктор этого класса создаёт и открывает на запись файл с указанным именем. Запись данных в файл осуществляется путём применения к объекту класса `ofstream` операции «<<» (аналогично использованию `cout`). Например,

```
ofstream f ("somefile.txt");
f << " Hello, World ! " << endl ;
```

2.4. Закрытие файла

Во избежание утечек файловых дескрипторов, любой файл после работы с ним должен быть закрыт. Для закрытия файла нужно вызвать метод `close` ассоциированного с файлом объекта (класса `ifstream` или `ofstream`).

2.5. Некоторые контейнерные классы

В некоторых вариантах заданий будет удобно воспользоваться контейнерными классами из стандартной библиотеки C++.

Динамические массивы в C++ представляются переменными класса `vector`, объявленного в заголовочном файле `vector`. Использование этого класса можно проиллюстрировать следующим примером (создание вектора целых чисел и заполнение его числами от 0 до 9):

```
vector<int> a ;  
for (int i = 0; i < 10; i++) a.push_back(i);
```

К вектору можно применять операцию индексации. Размер вектора возвращает метод `size`. Поэтому, например, вывести элементы нашего вектора на печать можно следующим образом:

```
for (int i = 0; i < a.size( ); i++) cout << a[i];
```

Вектора, кроме того, предоставляют итераторы для доступа к своему содержимому. Это даёт возможность для перебора элементов вектора использовать специальную форму оператора `for`:

```
for ( int x : a ) cout << x ;
```

Упорядоченный ассоциативный массив, реализованный через дерево, представлен в стандартной библиотеке C++ классом `map` из заголовочного файла `map`.

Следующий фрагмент кода демонстрирует объявление ассоциативного массива, заполнение его значениями и вывод словарных пар на печать:

```
map<string , int> m;  
m["a"] = 1 ;  
m["b"] = 2 ;  
for(auto pair : m) cout << pair.first << ",_" << pair.second << endl ;
```

Обратите внимание на то, что применение ключевого слова `auto` в объявлении переменной `pair` заставляет компилятор выводить её тип автоматически. Эта возможность доступна только в режиме `c++11`.

Упорядоченное множество, также реализованное через дерево, представлено классом `set` из заголовочного файла `set`. Объявление множества, добавление в него элементов и вывод множества на печать можно проиллюстрировать следующим фрагментом кода:

```
set<string> s;  
s.insert("qwerty");  
s.insert("abcd");  
for ( string x : s ) cout << x << endl ;
```

3. Задание

Варианты заданий для программ, которые нужно разработать в ходе выполнения данной лабораторной работы, приведены в таблицах 1–7.

Каждая программа должна принимать через аргумент командной строки путь к каталогу, в котором располагаются подлежащие обработке файлы.

4. Замечание

Ниже приведены таблицы с вариантами заданий.

- 1 Найти все файлы с расширением «txt» в указанном каталоге, разбить текст из каждого файла на слова и сформировать в текущем каталоге два файла: all.txt и shared.txt, содержащие объединение и пересечение множеств слов из найденных файлов. Каждое слово в сформированных файлах должно располагаться в отдельной строке. Слова должны быть отсортированы лексикографически. Работоспособность программы нужно проверить на наборе текстовых файлов, содержащих текст на английском языке.
- 2 Найти все файлы с расширением «html» в указанном каталоге, для каждого файла определить множество используемых в нём тегов HTML и сохранить объединение полученных множеств в файле tags.txt в текущем каталоге. Каждый тег в сформированном файле должен располагаться в отдельной строке. Теги должны быть отсортированы лексикографически. Выявить тег в HTML-файле можно по следующему признаку: тег представляет собой слово, расположенное после знака «<».
- 3 Найти все файлы с расширением «rtf» в указанном каталоге и изменить выделение курсивом в каждом файле на выделение полужирным шрифтом, и наоборот. В RTF-файлах выделение курсивом включается командой «\i», а выделение полужирным шрифтом – командой «\b». Работоспособность программы нужно проверить на наборе созданных в текстовом процессоре RTF-файлов.
- 4 Найти все файлы с расширением «svg» в указанном каталоге и заменить прямоугольники с прямыми углами на прямоугольники с закруглёнными углами, и наоборот. Прямоугольник с прямыми углами в SVG-файле задаётся тегом rect, имеющим вид

```
<rect x="50" y="20" width="150" height="150">
```

Чтобы «закруглить» углы, достаточно добавить атрибуты rx и ry, задающие радиусы закругления. Например,

```
<rect x="50" y="20" width="150" height="150" rx="20" ry="20">
```

Работоспособность программы нужно проверить на наборе SVG-файлов, загруженных из интернета или созданных в векторном графическом редакторе.

- 5 Найти все файлы с расширением «dot» в указанном каталоге, для каждого файла определить, ориентированный или неориентированный граф он

описывает, превратить ориентированный граф в неориентированный и сохранить получившееся описание графа в файле с тем же именем, но в текущем каталоге. Работоспособность программы нужно проверить на наборе предварительно созданных DOT-файлов.

- 6 Найти все файлы с расширением «txt» в указанном каталоге, для каждого файла разбить его текст на слова, случайным образом перемешать буквы в середине каждого слова (оставить на своих местах первую и последнюю буквы) и сохранить полученный текст в файле с тем же именем, но в текущем каталоге. Работоспособность программы нужно проверить на наборе текстовых файлов, содержащих текст на английском языке.
- 7 Найти все файлы с расширением «html» в указанном каталоге и для каждого файла очистить его текст от тегов HTML и сохранить очищенный текст в текущем каталоге в файле с тем же именем, но с расширением «txt». Теги в HTML-файле начинаются с «<» и оканчиваются на «>». В тексте знаки «<» и «>», чтобы не перепутать их с тегам, обозначаются как «<» и «>», поэтому при очистке текста требуется заменять эти обозначения на «<» и «>». Работоспособность программы нужно проверить на наборе HTML-файлов, загруженных из интернета.
- 8 Найти все файлы с расширением «c» в указанном каталоге, найти в них директивы препроцессора «#include» и сформировать в текущем каталоге файл headers.txt с перечислением заголовочных файлов, указанных в этих директивах. Имя каждого заголовочного файла должно располагаться в отдельной строке. Имена должны быть отсортированы лексикографически и не должны повторяться. Работоспособность программы нужно проверить на наборе предварительно подготовленных файлов с исходниками на языке C (можно использовать, например, часть исходных текстов ядра Linux).
- 9 Найти все файлы с расширением «svg» в указанном каталоге, в тексте каждого файла найти описания прямоугольников и сформировать в текущем каталоге файл rectangles.txt, в котором для каждого прямоугольника будет указано имя файла, в котором он найден, номер строки в файле и координаты. Прямоугольник в SVG-файле задаётся тегом rect, имеющим вид

```
<rect x="50" y="20" width="150" height="150">
```

Работоспособность программы нужно проверить на наборе SVG-файлов, загруженных из интернета или

созданных в векторном графическом редакторе.

- 10 Найти все файлы с расширением «dot» в указанном каталоге, для каждого файла определить количество рёбер в описываемом им графе и сформировать в текущем каталоге файл m.txt, в котором будут перечислены количества рёбер в файлах. Каждая строка файла sums.txt должна содержать имя DOT-файла и количество рёбер в нём. Имена файлов должны быть отсортированы лексикографически. Работоспособность программы нужно проверить на наборе предварительно созданных DOT-файлов.
- 11 Найти все файлы с расширением «txt» в указанном каталоге, в тексте каждого файла найти вхождения целых десятичных чисел и сформировать в текущем каталоге файл sums.txt, в котором будут перечислены суммы найденных чисел в каждом файле. Каждая строка файла sums.txt должна содержать имя файла и сумму чисел в нём. Имена файлов должны быть отсортированы лексикографически. Работоспособность программы нужно проверить на наборе предварительно созданных текстовых файлов.
- 12 Найти все файлы с расширением «html» в указанном каталоге, для каждого файла определить множество гиперссылок и сохранить объединение полученных множеств в файле links.txt в текущем каталоге. Каждая гиперссылка в сформированном файле должна располагаться в отдельной строке. Гиперссылки должны быть отсортированы лексикографически.

Гиперссылка в HTML-файле задаётся тегом «a»:

Работоспособность программы нужно проверить на наборе HTML-файлов, загруженных из интернета.

- 13 Найти все файлы с расширением «c» в указанном каталоге, очистить текст каждого файла от комментариев и сохранить в файл с тем же именем в текущем каталоге. Работоспособность программы нужно проверить на наборе предварительно подготовленных файлов с исходниками на языке C (можно использовать, например, часть исходных текстов ядра Linux).
- 14 Найти все файлы с расширением «java» в указанном каталоге, определить для каждого файла номер строки, с которой начинается объявление расположенного в этом файле публичного класса, и сохранить полученную информацию в файле classes.txt в текущем каталоге. Каждая строка файла classes.txt должна содержать имя публичного класса и номер строки. Имена классов должны быть отсортированы лексикографически. Работоспособность программы нужно проверить на

наборе предварительно подготовленных файлов с исходниками на языке Java.

- 15 В указанном каталоге лежат файлы тестов для сервера тестирования системы T-BMSTU. Каждый тест оформлен в виде двух файлов с именами «N» и «N.a», в которых содержатся входные данные для теста и правильный ответ, соответственно. Здесь N – это целое число, обозначающее номер теста. Требуется сгенерировать по набору файлов тестов таблицу в формате Markdown.

Таблица должна иметь следующий вид:

+-----+-----+	
Входные данные	Ответ
+=====+	
abcd	1
+-----+-----+	
qwerty	2
abcd	
+-----+-----+	

Работоспособность программы нужно проверить на наборе тестовых данных, загруженных из системы T-BMSTU.

- 16 Найти все файлы с расширением «txt» в указанном каталоге, для каждого файла разбить его текст на строки и записать их в тот же файл в обратном порядке. Работоспособность программы нужно проверить на наборе текстовых файлов, содержащих текст на английском языке.
- 17 Найти все файлы с расширением «html» в заданном каталоге, для каждого файла очистить его текст от скриптов и сохранить в файл с тем же именем в текущий каталог. Скрипты в HTML-файле записываются внутри тега script.

Например:

```
<script>
document.getElementById("demo").innerHTML = " Hello
JavaScript!";
</script>
```

Работоспособность программы нужно проверить на наборе HTML-файлов, загруженных из интернета.

- 18 Найти все файлы с расширением «md» в указанном каталоге, для каждого файла определить множество выделенных курсивом фраз и сохранить объединение полученных множеств в файле `emph.txt` в текущем каталоге. Каждая фраза в сформированном файле должна располагаться в отдельной строке. Фразы должны быть

отсортированы лексикографически и не должны повторяться. Выделение фразы курсивом в формате Markdown осуществляется путём обрамления этой фразы звёздочками.

Например:

Студентов первого курса называют **первокурами**.

Работоспособность программы нужно проверить на наборе предварительно составленных MD-файлов.

- 19 Найти все файлы с расширением «html» в указанном каталоге, для каждого файла построить три множества заголовков типа h1, h2 и h3 и сохранить объединённые множества заголовков разного типа из всех файлов в файлах h1.txt, h2.txt и h3.txt в текущем каталоге. Каждый заголовок в сформированном файле должен располагаться в отдельной строке, для чего может потребоваться удаление из заголовка символов перевода строки. Кроме того, заголовки должны быть отсортированы лексикографически. Заголовки в HTML-файле задаются тегами «h1», «h2» и «h3».

Например:

```
<h1>Введение</h1>
```

Работоспособность программы нужно проверить на наборе HTML-файлов, загруженных из интернета.

- 20 Найти все файлы с расширением «md» в указанном каталоге, для каждого файла определить множество гиперссылок и сохранить объединение полученных множеств в файле links.txt в текущем каталоге. Каждая гиперссылка в сформированном файле должна располагаться в отдельной строке. Гиперссылки должны быть отсортированы лексикографически. Гиперссылка в формате Markdown записывается в виде:

```
[текст, поясняющий гиперссылку](гиперссылка)
```

Работоспособность программы нужно проверить на наборе предварительно составленных MD-файлов.

- 21 Найти все файлы с расширением «txt» в указанном каталоге, разбить текст каждого файла на абзацы, перевести в формат HTML и сохранить в текущем каталоге в файле с тем же именем, но с расширением «html». Считать, что абзацы отделяются друг от друга пустыми строками. Для формирования HTML-файла можно использовать следующий шаблон:

```
<!DOCTYPE html>  
<html>  
<head>
```

```
<meta charset="UTF-8">
</head>
<body>
<p>Абзац 1</p>
<p>Абзац 2</p>
...
</body>
</html>
```

Работоспособность программы нужно проверить на наборе текстовых файлов, содержащих текст на английском языке.

- 22 Найти все файлы с расширением «html» в указанном каталоге, очистить текст каждого файла от комментариев и сохранить в файл с тем же именем в текущем каталоге. Комментарии в HTML-файлах задаются следующим образом:

```
<!-- Текст комментария -->
```

Работоспособность программы нужно проверить на наборе HTML-файлов, загруженных из интернета.

- 23 Найти все файлы с расширением «c» в указанном каталоге, обнаружить в каждом файле все директивы #define и сформировать в текущем каталоге два файла «constants.txt» и «macros.txt», содержащий имена макроопределений без параметров и имена макроопределений с параметрами, соответственно. Работоспособность программы нужно проверить на наборе предварительно подготовленных файлов с исходниками на языке C (можно использовать, например, часть исходных текстов ядра Linux).
- 24 Найти все файлы с расширением «dot» в указанном каталоге, для каждого файла определить количество элементов (вершин и дуг графа), имеющих красный цвет контура, и сформировать в текущем каталоге файл red.txt, в котором будут перечислены количества красных элементов в файлах. Каждая строка файла sums.txt должна содержать имя DOT-файла и количество элементов в нём. Имена файлов должны быть отсортированы лексикографически. Работоспособность программы нужно проверить на наборе предварительно созданных DOT-файлов.
- 25 Найти все файлы с расширением «java» в указанном каталоге, найти в каждом файле директивы импорта и сохранить полученную информацию в файле imports.txt в текущем каталоге. Каждая строка файла imports.txt должна содержать квалифицированное имя импортируемого класса и количество файлов, в которых он импортируется. Имена классов должны быть

отсортированы лексикографически.

Работоспособность программы нужно проверить на наборе предварительно подготовленных файлов с исходниками на языке Java.

- 26 Найти все файлы с расширением «dot» в указанном каталоге, и убрать в каждом файле раскраску вершин и дуг графа, задаваемую атрибутами «color». Изменённые файлы необходимо сохранить в текущем каталоге. Работоспособность программы нужно проверить на наборе предварительно созданных DOT-файлов.
- 27 Найти все файлы с расширением «txt» в указанном каталоге и перевести их в формат HTML, превратив содержащиеся в файлах адреса web-страниц в гиперссылки. Порождённые HTML-файлы следует сохранить в текущем каталоге в файле с тем же именем, но с расширением «html». Считать, что адреса web-страниц начинаются с «http://». Для формирования HTML-файла можно использовать следующий шаблон:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
...здесь размещается текст...
</body>
</html>
```

Гиперссылка в HTML-файле задаётся тегом «a»:

```
<a href=гиперссылка>поясняющий текст</a>
```

Работоспособность программы нужно проверить на наборе текстовых файлов, содержащих текст на английском языке.

- 28 Найти все файлы с расширением «md» в указанном каталоге, для каждого файла определить множество заголовочных фраз и сохранить объединение полученных множеств в файле headers.txt в текущем каталоге. Каждая фраза в сформированном файле должна располагаться в отдельной строке. Фразы должны быть отсортированы лексикографически и не должны повторяться. Выделение заголовка в формате Markdown осуществляется путём «подчёркивания» его последовательностью знаков «=» или «-».

Например:

Введение

=====

Работоспособность программы нужно проверить на наборе предварительно составленных MD-файлов.

- 29 Найти все файлы с расширением «java» в указанном каталоге, и для каждого файла составить последовательность номеров строк, в которых располагаются заголовки циклов. Полученную информацию нужно сохранить в файле loops.txt в текущем каталоге. Каждая строка файла loops.txt должна содержать имя java-файла и последовательность номеров строк, отсортированную в порядке возрастания. Работоспособность программы нужно проверить на наборе предварительно подготовленных файлов с исходниками на языке Java.
- 30 Найти все файлы с расширением «с» в указанном каталоге, и для каждого файла составить последовательность номеров строк, в которых располагаются заголовки операторов «if». Полученную информацию нужно сохранить в файле ifs.txt в текущем каталоге. Каждая строка файла ifs.txt должна содержать имя с-файла и последовательность номеров строк, отсортированную в порядке возрастания. Работоспособность программы нужно проверить на наборе предварительно подготовленных файлов с исходниками на языке С (можно использовать, например, часть исходных текстов ядра Linux).
- 31 Найти все файлы с расширением «md» в указанном каталоге, и в тексте каждого файла пронумеровать заголовки целыми числами. Изменённые файлы следует сохранить в текущем каталоге. Заголовки в формате Markdown обозначаются с помощью «подчёркивания» последовательностью знаков «=» или «-».
- Например:

Введение
=====

Работоспособность программы нужно проверить на наборе предварительно составленных MD-файлов.

- 32 Найти все файлы с расширением «txt» в указанном каталоге, разбить текст из каждого файла на слова и сформировать в текущем каталоге два файла: all.txt и shared.txt, содержащие объединение и пересечение множеств слов из найденных файлов. Каждое слово в сформированных файлах должно располагаться в отдельной строке. Слова должны быть отсортированы лексикографически. Работоспособность программы нужно проверить на наборе текстовых файлов, содержащих текст на английском языке.
- 33 Найти все файлы с расширением «html» в указанном каталоге, для каждого файла определить множество

используемых в нём тегов HTML и сохранить объединение полученных множеств в файле tags.txt в текущем каталоге. Каждый тег в сформированном файле должен располагаться в отдельной строке. Теги должны быть отсортированы лексикографически. Выявить тег в HTML-файле можно по следующему признаку: тег представляет собой слово, расположенное после знака «<».

Работоспособность программы нужно проверить на наборе HTML-файлов, загруженных из интернета.

- 34 Найти все файлы с расширением «rtf» в указанном каталоге и изменить выделение курсивом в каждом файле на выделение полужирным шрифтом, и наоборот. В RTF-файлах выделение курсивом включается командой «\i», а выделение полужирным шрифтом – командой «\b». Работоспособность программы нужно проверить на наборе созданных в текстовом процессоре RTF-файлов.

- 35 Найти все файлы с расширением «svg» в указанном каталоге и заменить прямоугольники с прямыми углами на прямоугольники с закруглёнными углами, и наоборот. Прямоугольник с прямыми углами в SVG-файле задаётся тегом rect, имеющим вид

```
<rect x="50" y="20" width="150" height="150">
```

Чтобы «закруглить» углы, достаточно добавить атрибуты rx и ry, задающие радиусы закругления. Например,

```
<rect x="50" y="20" width="150" height="150" rx="20" ry="20">
```

Работоспособность программы нужно проверить на наборе SVG-файлов, загруженных из интернета или созданных в векторном графическом редакторе.