UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA

LENGUAJE Y MÉTODOS DE PROGRAMACIÓN

BUSCADOR EN C

Profesor: Roberto González
Alumno: Roberto Orellana

Fecha de Entrega: 14/09/2017

Santiago de Chile

15 - 10 – 2017

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCION

El siguiente informe trata de explicar cómo llegar a la solución del laboratorio N°1 de Lenguaje y métodos de Programación.

1.1 ANTECEDENTES Y MOTIVACIÓN

El laboratorio consta de crear un Buscador en cuatro lenguajes distintos de programación, siendo este el primero en Lenguaje C, lo cual me agrada porque es uno de los que más conozco hasta ahora.

1.2 DESCRIPCIÓN DEL PROBLEMA

El Problema consiste en crear un sistema de recuperación de Información en el lenguaje Imperativo Procedural C el cual consiste en cargar Stopwords en memoria, crear un índice y almacenarlo en memoria, guardar un Índice en un archivo de texto, cargar un Índice en un archivo de texto, hacer una consulta de alguna palabra o frase completa, mostrar el resultado de los documentos de mayor a menor según las palabras encontrada.

1.3 SOLUCIÓN PROPUESTA

Para obtener una solución real al problema que se propone, se estudió cuidadosamente los requerimientos funcionales y no funcionales pedidos en el enunciado, de esta forma, se logra detallar las funcionalidades que este contendrá de forma correcta.

Almacenar Stopwords
Debido a que se trata de un buscado, la primera problemática que presenta el proyecto es la eliminación de las palabras comunes que se encuentran en el documento y en las palabras que consultara el usuario. Teniendo en cuenta que la eliminación de estas palabras permite una búsqueda mucho más eficiente de las palabras.

Crear Índice.
Para la creación del índice la representación elegida fue la de matriz dispersa en el cual almacena un contador cada vez que encuentra una incidencia de una palabra en cada documento, un arreglo de caracteres donde se almacena cada una de las palabras sin repetirse, un arreglo en el cual se guarda todo el documento, la cantidad de archivos y la línea de inicio y final de cada documento.

Guarda y Cargar Índice.
Se debe guardar y cargar un índice en un archivo de texto.

Consultar y Mostrar Resultados.
La creación de consultas es muy parecida a la de la creación del índice donde elimina las palabras comunes que se encuentran en las Stopwords, se consultan las palabras y se retorna la cantidad de veces que se repiten en cada documento gracias a la matriz de incidencia. Los resultados son ordenados en un ranking de mayor a menor por la cantidad de palabras encontradas en cada documento, luego una función básica que imprime los resultados.

1.4 OBJETIVOS Y ALCANCES DEL PROYECTO

1.4.1 Objetivo General

El Objetivo general del proyecto consta en lograr el correcto funcionamiento del programa en el lenguaje C, utilizando lo aprendido en clases y laboratorio sobre paradigma de programación imperativo Procedural.
Conocer el correcto uso de cada código utilizado, la forma de desacoplar los archivos en .c y .h y la forma en que estos deben estar enlazados y su funcionalidad en las plataformas Windows y Linux.

1.4.2 Objetivos Específicos

Lograr que el usuario pueda interactuar con el buscador, cargar Palabras comunes para que sean ignoradas por el buscador, crear un índice limpio libre de Stopwords y caracteres que compliquen la búsqueda, almacenar un índice creado en un archivo de texto, cargar un índice creado en un archivo de texto con su Id, fecha y hora de creación, hacer una consulta de una frase compuesta o palabra, y mostrar el resultado de la búsqueda en un ranking de mayor a menor según la cantidad de palabras encontradas en cada documento

1.4.3 Alcances

Obtener un correcto funcionamiento del programa gracias a los objetivos y problemas que se plantearon anteriormente.

1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

Las herramientas y técnicas utilizadas para la solución de final del problema constan de lo aprendido en clases, investigación propia del alumno y las metodologías enseñadas por el profesor, analizando el problema y teniendo una abstracción de cómo resolverlo.
La aplicación utilizada en mi caso fue la de Dev C++ que gracias a su fácil compilación y ejecución hace que sea una interfaz más amigable para el programador.

CAPITULO 2. FUNDAMENTOS TEORICOS

2.1 definiciones

- **Paradigma de programación imperativa**: La programación imperativa es uno de los paradigmas de programación de computadoras más generales, que describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador como realizar una tarea.

- **Funcion:** una función es un grupo de instrucciones con un objetivo en particular y que se ejecuta al ser llamada desde otra función o procedimiento. Una función puede llamarse múltiples veces e incluso llamarse a sí misma (función recurrente).

- **Punteros**: un apuntador o puntero es una variable que hace referencia (apunta) a una región de memoria. Al trabajar con punteros se manipulan directamente las direcciones de memorias en las cuales están los datos.

- **Paso por valor:** Técnica que entrega como parámetro a la función, una copia del dato que se está ofreciendo

- **Paso por referencia:** Técnica que entrega como parámetro actual a la función, la dirección de memoria del dato que se desea modificar o trabajar.

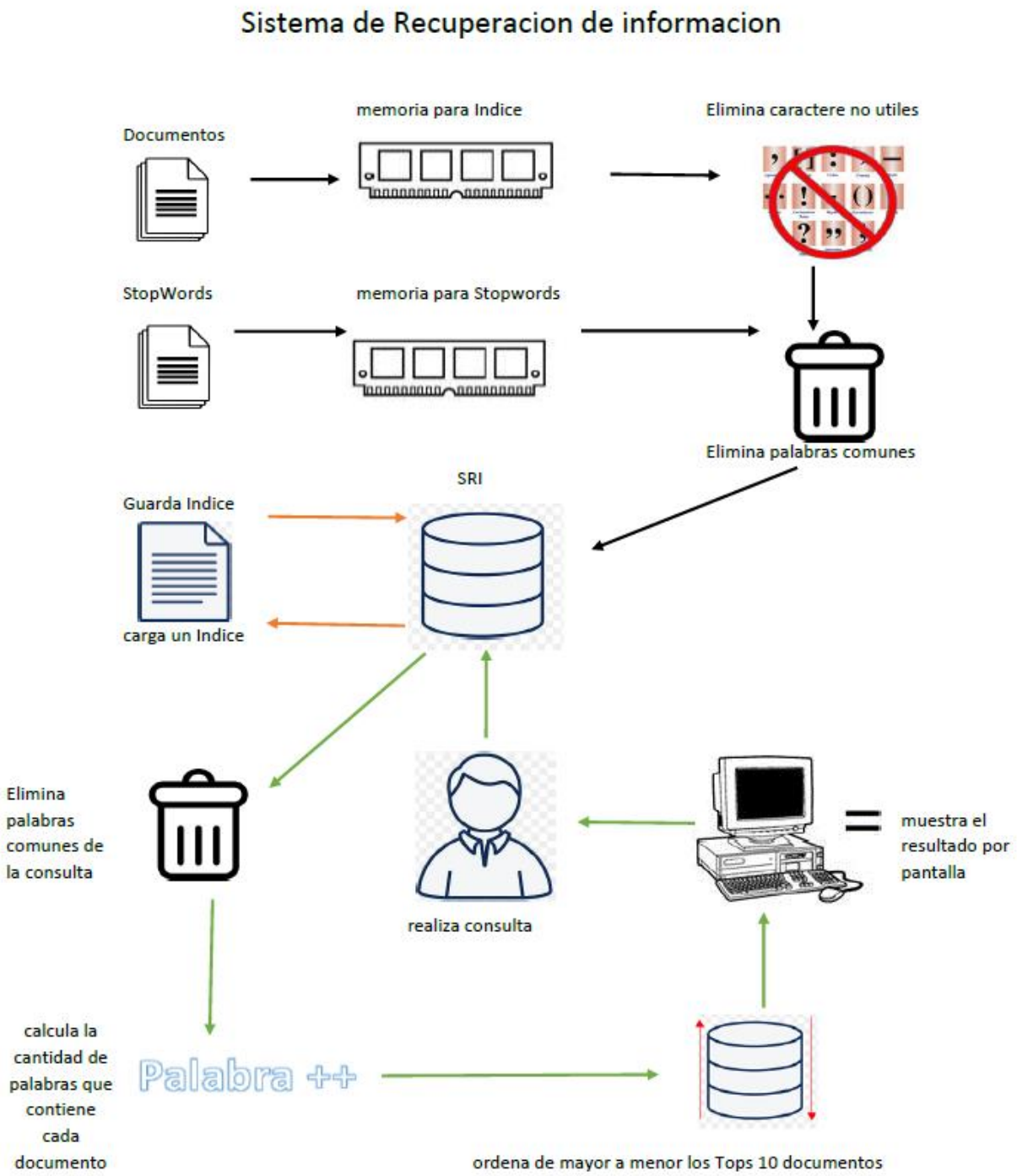
- **Vector Dinámico:** es un array de elementos que crece o mengua dinámicamente conforme los elementos se agregan o se eliminan.

- **Dirección de memoria:** la memoria principal de un ordenador es como una colección de celdas que almacenan datos e instrucciones. Cada celda está identificada unívocamente por un número o dirección de memoria.

- **StopWords:** (palabras comunes) son términos que los buscadores ignoran cuando rastrean un texto.

- **Índice:** el Índice me permite encontrar información en documentos y colecciones de documentos.

2.2 Diagrama SRI



2.3 Métodos Utilizados

Los métodos usados dentro de la organización de un algoritmo que realice un procedimiento correctamente son diversos.

Si el algoritmo es demasiado extenso para ser abordado en su totalidad, se divide en sub problemas que permitan tener soluciones parciales, en donde se aplica una especie de principio de superposición, o sea, la suma de las soluciones, resultando la solución general del algoritmo.

CAPITULO 3, DESARROLLO DE LA SOLUCION

3.1 Análisis

En el análisis la deducción principal fue la creación de funciones que leerán y almacenaran las palabras de las StopWords y el Índice Invertido, que gracias a estos se podrá trabajar fácilmente sobre las otras funciones del programa.

3.2 Diseño

Está diseñado básicamente con un menú de opciones que permite ingresar a diferentes funciones, por ejemplo:

1) Cargar StopWords (LoadStopWords).

```
//funcion que permite cargar los StopWords de un documento
StopWords* loadStopWords(char* pathStopWordsFile, code *statusCode)
{
    char temp[20];
    int  i, largo, cont;
    StopWords *words = (StopWords*)malloc(sizeof(StopWords));
    FILE *f;
    f = fopen(pathStopWordsFile,"r");//lee el archivo de text
```

2) Crear Índice (createIndex).

```
//crea indice para buscar las palabras. la representacion consta de un arreglo dinamico
Index* createIndex(char* pathDocumentsFile, StopWords *sw, code *statusCode)
{
    int i,  cont = 0,cont2 = 0, j = 0;
    char temp[80],aux;
    Index *ind = (Index*)malloc(sizeof(Index));

    FILE *f;
    f = fopen(pathDocumentsFile,"r");//lee el archivo de text
```

3) Guardar Índice (saveIndex)

```
//guarda el Indice en un Documento
void saveIndex(Index *i,int *id, code *statusCode)
{
    FILE* salida;
    char nombre[40];
    int j;

    printf("Guarde el archivo en formato nombre.txt (ejemplo.txt): ");/
```

4) Cargar Índice (loadIndex)

```
//carga el Indice desde un Documento
Index* loadIndex(int id, code *statusCode)
{
    Index *ind = (Index*)malloc(sizeof(Index));
    FILE* entrada;
    char *nombre = (char*)malloc(sizeof(char)*40);
    printf("Cargue el archivo con extension nombre.txt (ejemplo.txt):
```

5) Hacer Consulta de Palabra(s) (query)

```
//funcion que ingresa una palabra, busca referencias en el indice y ordena p
Ranking* query(Index *ind, StopWords *sw, char* text, code *statusCode)
{
    Ranking *Rk = (Ranking*)malloc(sizeof(Ranking));
    Rk->result = (int*)calloc(ind->contInd,sizeof(int));//se asigna memoria
    Rk->doc = ind->contInd;
    int i,j,l,cont = 0,cont2;

    for(i = 0; i < strlen(text)+1; i++)
```

6) Mostrar Resultados por Ranking(displayResults)

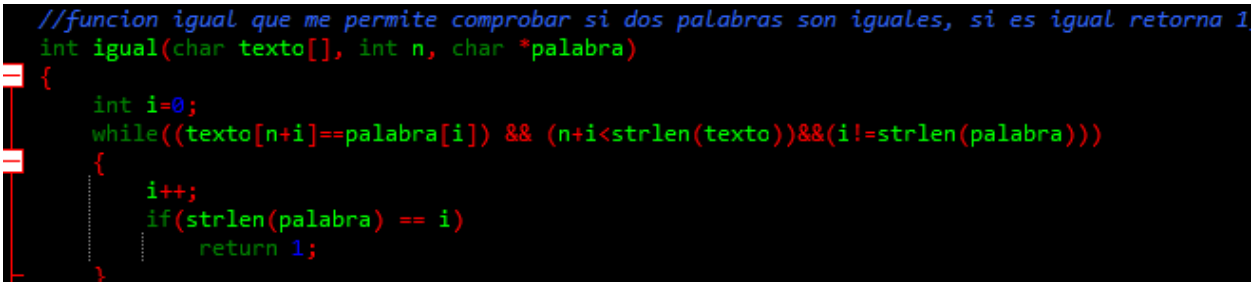
```
//muestra una lista en orden de mayor a menor segun la palabra ingresada
void displayResults(Ranking *r, int TopK, code *statusCode)
{
    int i,j;
    for (i=0;i<10 && i < r->doc;i++)
        printf("Ranking N-%d el Doc %d con la catidad de %d palabras enco
```

7) Mostrar Documentos por Ranking(displayDoc)

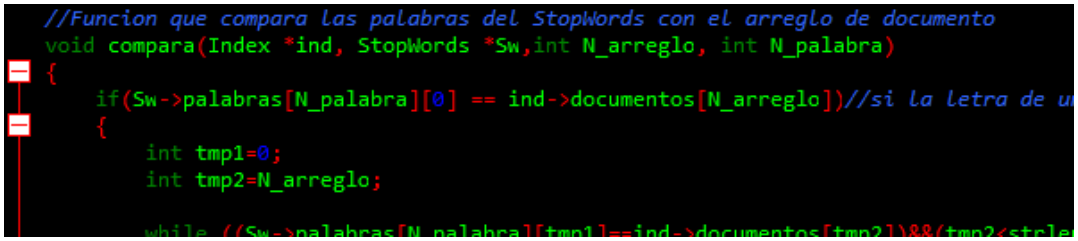
```
//muestra los doc en orden de mayor a menor segun la palabra ingresada
void displayDoc(Ranking *r, Index *ind, code *statusCode)
{
    int i,j;

    int cont = 0;
    for(i = 0; i < 10 && i < r->doc; i++)
```

Dentro de cada función existen subfunciones (subrutinas) que me permiten realizar pequeñas tareas que muchas veces pueden ser utilizadas por una o más funciones, por ejemplo:

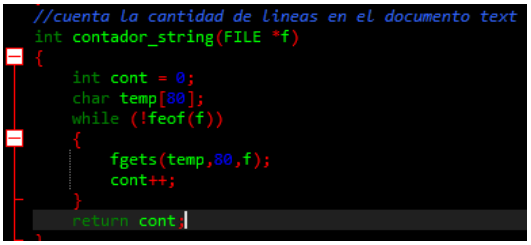
Función que me permite comparar dos palabras, retorna 1 si son iguales, 0 si no son iguales



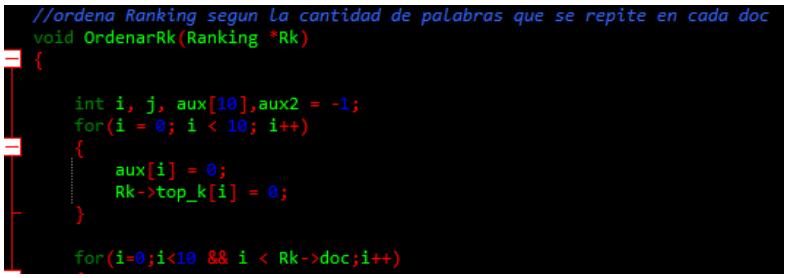
Función que compara las palabras (StopWords) con las palabras del Índice o de la consulta.



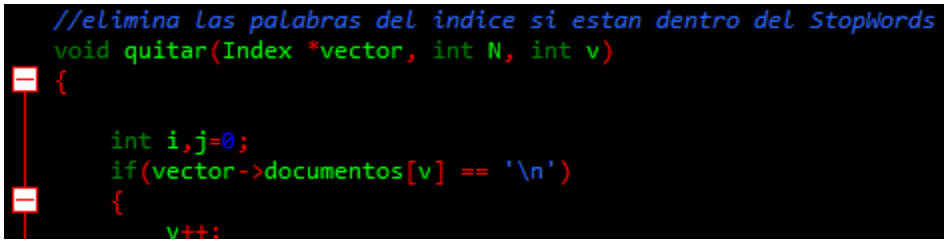
contador de string que me permite ver cuantos string contiene un documento



Función que me permite ordenar el ranking de mayor a menor dentro de los top_k (10).



Función que elimina las palabras si están contenidas dentro de la StopWords



```
//elimina las palabras del indice si estan dentro del StopWords
void quitar(Index *vector, int N, int v)
{

    int i,j=0;
    if(vector->documentos[v] == '\n')
    {
        v++;
```

Entre otras funciones.

3.3 Estructura del proyecto

La estructura del proyecto consta de 3 archivos, 1 archivo de cabecera (.h) llamado funciones.h y dos archivos de código fuente (.c) llamados funciones.c y buscados.c
Inicia con un menú principal, se indica al usuario las opciones que tiene para utilizar el programa y se puede usar por separado a cada una de las funciones.
Para el uso de la mayoría de las funciones, antes se debe cargar el archivo con las StopWords y luego la Creación del Índice
Podra guardar y cargar un índice, consultar por una frase y mostrar por pantalla en orden de mayo a menor los tops 10 archivos que más contengan esa frase
Se utilizan las librerías estándares de c: stdio.h, stdlib.h, string.h y time.h

CAPITULO 4. RESULTADOS

Se da cumplimiento a los objetivos del proyecto, creando las funciones requeridas; leer y almacenar las Stopwords, crear un índice a partir de un archivo de entrada y limpiar el archivo eliminando las palabras comunes, Almacenar Índice en un archivo de texto plano y autogenerar un Id también se logró almacenar la fecha y la hora en el archivo de texto plano, Cargar Indice desde un archivo de texto plano, compara el Id ingresado por el usuario con la del índice cargado, realizar consultas de frases y eliminar las palabras comunes de la consulta, ordenar top k documentos de mayor a menor y mostrarlos los resultados top 10 por pantalla.

CAPITULO 5. CONCLUSION

Gracias a este laboratorio logre adquirir más conocimientos sobre este lenguaje imperativo procedural. en cuanto a punteros, lectura y escritura desde un archivo, estructuras, arreglos dinámicos y la forma en que estos se almacenan en memoria, binding y scope.
La forma en que las subrutinas se pueden volver a utilizar desde distintas funciones, el almacenamiento dinámico que se puede redimensionar la memoria y la forma en que puedo obtener valores de otras funciones a través del paso por referencia, y su cambio de estado en proceso de ejecución.
Es un leguaje y paradigma muy agradable de trabaja que nos permite hacer muchas cosas solo con entregarle instrucciones a la máquina.

CAPITULO 6. REFERENCIAS

https://www.youtube.com/watch?v=LZoS_iKLi-c&list=PLik-Tt9YjWT1t2_ayXZWuyQ5G1UboSHzx
https://kesquivel.files.wordpress.com/2011/08/ficherosc_2011.pdf
http://c.conclase.net/ficheros/index.php?cap=002#inicio
http://www.tutorialesprogramacionya.com/cya/
http://www.udesantiagovirtual.cl/moodle2/course/view.php?id=9365