Laboratorio 7 – Motor de búsqueda web basado en el TAD Árbol Binario de Búsqueda

GUIÓN DEL LABORATORIO

1.- Objetivos del laboratorio

- Diseño de clases en C++
- Comprensión y uso del TAD Árbol Binario de Búsqueda para resolver un problema
- Análisis de los requisitos de un problema y diseño de las estructuras de datos más adecuadas para su resolución

2.- Antes de asistir al laboratorio

Antes de asistir al laboratorio debes realizar las siguientes tareas:

1. Leer los apuntes de clase sobre el TAD Árbol, y más específicamente sobre el tipo Árbol Binario de Búsqueda
2. Revisar el guión del laboratorio (este documento)
3. Contestar a las cuestiones 1 a 3 de la Actividad 0 del Prerequisito.

Recuerda que es preciso entregar las respuestas a esta actividad antes de la fecha prevista. En caso contrario, no podrás acceder al laboratorio ni entregar de ninguna manera la solución a esta práctica.

3.- Actividades a realizar

Actividad 1: Implementación del TAD ABB (Árbol Binario de Búsqueda) utilizando estructuras dinámicas.

1. Implementar el TAD ABB utilizando estructuras dinámicas (los ficheros deben denominarse ABB.h y ABB.cpp). El campo clave de cada nodo será de tipo carácter, mientras que el campo dato será un entero.
2. Crear un nuevo proyecto en *DevC++* y añadir los ficheros ABB.h, ABB.cpp y testABB.cpp (éste último se suministra como material de la práctica).
3. Compilar el programa "`testABB.cpp`" y verificar que no hay errores.
4. Ejecutar el programa y completar la Actividad 1 de tu hoja de trabajo.

Actividad 2: Uso del TAD ABB para la resolución de un problema

La principal característica de un ABB es su potencia a la hora de realizar búsquedas basadas en el campo clave del árbol. Su estructura y organización permite la localización de cualquier nodo en el árbol con un coste $O(log(n))$, siendo n el número de nodos (claves diferentes) del árbol. Este coste teórico se produce siempre que el árbol se encuentre perfectamente equilibrado, situación que, aunque se va a aproximar mucho, no vamos a tener exactamente en nuestro ABB.

Utilizaremos la eficiencia de los ABB para la creación de un motor de búsqueda web que sea capaz de indexar todas las palabras contenidas en las páginas de un dominio web. El motor realizará una carga de todas las páginas del dominio guardando en el ABB las palabras que encuentre. Cada palabra encontrada, que será el campo clave de un nodo del árbol, tendrá como dato asociado una lista con las direcciones de todas las páginas en las que ha sido encontrada. De esta manera se podrá consultar posteriormente en que páginas se encuentran una palabra determinada.

Además del árbol de búsqueda, utilizaremos una lista auxiliar donde se almacenarán todas las páginas que se van a indexar. En la lista se agregaran aquellos enlaces a nuevas páginas que se van encontrando durante la indexación de las páginas. La carga del ABB finalizará cuando se alcance el final de esta lista auxiliar (esto significará que se han indexado todas las páginas del dominio).

Para eliminar repeticiones en las listas de páginas de cada nodo (en el caso de que una palabra se encuentre más de una vez en una misma página) y, sobre todo, para evitar bucles infinitos en la indexación que se podrían producir si se permitieran duplicidades en la lista auxiliar de páginas a indexar (se darían en el caso de que una página tuviera un vínculo a una segunda y ésta otra a su vez un vínculo a la primera), se debe implementar la función *bool Agregar (ValorLista)* en la clase *Lista*, que únicamente insertará el valor en la lista en el caso de que dicho valor NO se encuentre previamente en la lista.

Para facilitar el desarrollo del programa, el motor búsqueda se encontrará en la misma máquina que el servidor web, con lo cual las páginas a indexar serán ficheros locales. En el material de la práctica se incorporá un directorio, *www*, donde se encuentra un pequeño portal web de prueba que se utilizará durante el desarrollo del programa y para responder a las cuestiones. El dominio que hay que indicarle al programa para la carga del ABB será el path completo de la página inicial del portal ubicado en este directorio.

La estructura del programa debe ser la siguiente:

1. Indicar el path completo de la página inicial del portal y almacenarla en la lista de páginas a indexar, que estará vacía.

2. Consultar la dirección de la siguiente página web de la lista de páginas a indexar, abrir el fichero correspondiente y cargar cada una de las palabras en el ABB según el siguiente criterio:
    - Si la palabra no se encontraba en el árbol generar un nuevo nodo en el lugar apropiado del árbol y agregar a la lista de páginas del nodo la página actual
    - Si la palabra se encontraba en el árbol sólo hay que agregar a la lista de páginas del nodo la página actual
    - Analizar la palabra para ver si contiene la referencia *"href="*. En caso afirmativo se trata de un enlace a una nueva página, por lo que ésta debe agregarse a la lista de páginas a indexar.
        - Ejemplo: *English**
          En este caso, la dirección que hay que guardar es *ingles/index.htm*

3. Repetir el paso 2 mientras no se alcance el final de la lista de páginas a indexar.

4. Permitir consultar el ABB con una palabra para conocer las páginas donde ésta se encuentra.

Las clases que se deben implementar son las siguientes:

```
class Trovator
{
   public:
     // Constructor de la clase en el que se va a solicitar la página inicial del
dominio,
     // se inicializará el atributo dominio a partir de esta página, y se cargará el ABB
     // con todas las páginas que se encuentren en el dominio
     Trovator ();

     // Busca en el ABB todas las palabras introducidas y construye una
     // lista donde aparezcan todas ellas
     Lista BuscarPalabras ( string );

   private:
     ABB ar;         // ABB donde se guardararán las palabras con su lista de páginas
                     // asociada
     Lista paginas;  // Lista de páginas que debe recorrer e indexar en el ABB
     string dominio; // Dominio al que pertenece la página inicial
                     // (p.e de http://www.uv.es/index.html será http://www.uv.es) y que
                     // utilizaremos para excluir las búsquedas en páginas ajenas al
                     // dominio

     // Indexa la página (fichero) y carga sus palabras en el ABB
     bool LeerFichero ( string );


     // Extraigo una palabra valida del fichero. De esta manera aislo en un
procedimiento
     // el tratamiento de la captura de palabras, pudiendo excluir aquellas palabras
     // reservadas o eliminar caracteres extraños (extremo que no se va a exigir en la
     // práctica). Al leer la palabra, se identificara si es una referencia a otra
página
     // y se apilará si no ha sido indexada anteriormente.
     bool LeerPalabra ( ifstream &, string & );
};
```

```
class ABB
{
    public:
        typedef string TipoClave;
        typedef Lista TipoDato;
        struct Valor
        {
           TipoClave clave;
           TipoDato dato;
        };

        ABB ();
        ABB (const ABB&);
        ~ABB ();
        const ABB& operator= (const ABB&);

        bool ABBVacio ();

        bool Buscar (TipoClave, TipoDato&);
        bool Insertar (Valor);
        bool Eliminar (TipoClave);

        void ImprimirABB ();

   private:
        typedef ABB* PunteroABB;

        Valor info;
        PunteroABB izdo;
        PunteroABB dcho;
        bool esVacio;

        void Copiar (const ABB&);
        void Vaciar ();
        PunteroABB BuscarPtr (TipoClave);

};
```

Contesta a la pregunta 5 de la hoja de trabajo, indicando el código mediante el cual se ha implementado la función *Agregar* de la clase *Lista*.

Contesta a las preguntas 6 a 7 de la hoja de trabajo.

Actividad 3: Extiende la capacidad de consulta sobre el ABB introduciendo la posibilidad de realizar la consulta sobre más de una palabra, con lo que la lista de páginas resultantes serán aquellas en las que aparezcan todas las palabras consultadas. Para ello habrá que modificar de nuevo la clase *Lista*, incorporando la función *void Interseccion(Lista&)*.

Ejemplo: Si se realiza la consulta sobre las palabras “Algoritmos”, “Estructuras” y “Datos”, el resultado debe ser las páginas en las que aparecen las tres palabras, es decir, la intersección de la lista de páginas web donde aparece “Algoritmos” con la lista de páginas donde aparece “Estructuras” y con la lista de páginas donde aparece “Datos”.

Contesta a la pregunta 8 de la hoja de trabajo, indicando el código mediante el cual se ha implementado la función *Interseccion* de la clase *Lista*.

Contesta a la pregunta 9 de la hoja de trabajo.

Actividad Extra: Esta actividad es voluntaria y se calificará de forma extraordinaria a la evaluación de la práctica. Para la completa realización de la práctica es suficiente con la realización de las tres actividades enunciadas anteriormente.

En esta actividad se pretende dotar al buscador de la capacidad de realizar indexaciones de portales remotos, no limitándose a la búsqueda en páginas locales como habíamos visto hasta ahora. Para ello utilizaremos la utilidad *wget* (que se proporciona en el material de la práctica), cuyo ejecutable debe residir en el mismo directorio desde donde se ejecute nuestro buscador.

Modificaremos la carga del ABB para que en el análisis de cada página compruebe previamente si se trata de una página remota. En caso afirmativo se descargará la página a local antes de proceder a su análisis. Con estos requisitos el código de la carga del ABB debe de ser parecido a lo siguiente:

```
paginas.IrInicio(); //paginas es la Lista de páginas web a indexar
while ( ! paginas.FinalLista() )
{
     paginas.Consultar(p);

       // Si el fichero es remoto, compruebo si se encuentra en el dominio, y en caso
       // afirmativo, lo descargo previamente a local, lo indexo y despues lo elimino
       if ( p.substr(0,4) == "http" && ( p.find(dominio, 0) != string::npos ) )
       {
             comando = "wget.exe " + p;
             system(comando.c_str());
             nomf = p.substr(p.rfind("/")+1,p.length()-p.rfind("/"));
             LeerFichero(p,nomf);
             comando = "del " + nomf;
             system(comando.c_str());
       }
       else
             LeerFichero(p,p);

       paginas.Avanzar();
}
```

Como se advierte en el anterior código, se debe modificar también la implementación de *LeerFichero* para que acepte dos parámetros, un primero con el nombre completo de la página y el segundo con el nombre de la página local que se debe analizar. De esta manera seremos capaces de guardar la referencia completa de la página en la lista y no sólo el nombre en local.

Los archivos con los programas correspondientes a las actividades 1, 2 y 3 (y de la actividad extra si éste fuera el caso) se deberán entregar, a través del aula virtual, al finalizar la sesión de prácticas, junto con la Hoja de Trabajo del estudiante.

4.- Después de asistir al laboratorio

Una vez finalizada la práctica contesta las cuestiones correspondientes a la Actividad 4 que aparecen en el Cuestionario Posterior al laboratorio.