Práctica 5
Buscador de información con memoria dinámica

Fecha de entrega: 28 de mayo de 2017

1 Eficiencia en el uso de la memoria

A pesar de que los equipos informáticos de usuario actuales disponen de recursos hardware razonablemente por encima de lo que generalmente necesitamos, es importante usar la memoria de forma eficiente y no desperdiciar ni replicar contenido. Por esto, vamos a modificar la práctica 4 para que gestione su memoria con memoria dinámica y punteros, de forma que la implementación el cálculo del PageRank sea más eficiente.

2 Modificaciones a la práctica

2.1 Módulo de listas de cadenas

Este módulo tendrá ahora *listas dinámicas*. En decir, las listas ya no serán de tamaño fijo, sino que serán representadas *internamente* por un puntero que apunte a una región en el montículo. Tendrás que modificar la estructura de datos para que contenga el puntero a los datos, el tamaño actual, y la capacidad actual.

Cuando, durante la inserción, se detecta que ya no caben más elementos, se ampliará la lista automáticamente, de manera transparente al usuario del módulo, según la siguiente función de tamaño:

$$nuevaCapacidad = (capacidadAntigua * 3) / 2 + 1$$

Sabiendo el tamaño nuevo de la lista, deberás crear una nueva, copiar el contenido de la antigua en ella, y actualizar los punteros en la estructura de datos. No olvides también borrar aquello que quede en desuso.

Por lo demás, observa que la interfaz del módulo **no cambia**. Esto quiere decir que habremos hecho una versión de la lista que, siendo más eficiente, no afecta al resto del programa.

2.2 Módulo de tablas clave-valor y módulo creador de índices

Para este módulo, que básicamente contiene también una lista de estructuras de datos, deberás hacer un proceso análogo al explicado en la sección 2.1, en el que la tabla sea de tamaño dinámico (con punteros y usando el montículo) y que se redimensione automáticamente cuando es necesario más espacio. Aplica la misma fórmula para el tamaño que en la sección 2.1. Además, ahora el valor no será una lista estática, sino un puntero a una lista creada en el montículo de forma dinámica. Recuerda inicializar y borrar el contenido de la memoria cuando sea necesario.

2.3 Búsqueda binaria recursiva

Como modificación adicional, tienes que reemplazar las búsquedas binarias que se realizan en la práctica 4 por sus correspondientes versiones recursivas. Al menos tienes que hacerlo en el módulo de índice de palabras, pero si has aplicado la búsqueda binaria en más sitios (porque la información esté ordenada, que es un requisito), también debes adaptarla a su versión recursiva.

El interfaz de la función (el nombre y los parámetros) no deben cambiar. Así, el resto del programa queda igual. Para que esto pueda cumplirse, es posible que necesites crear alguna función auxiliar.

3 Memoria dinámica

Para que al terminar la ejecución del programa se muestre información sobre la basura que ha podido dejar, añade al inicio de la función `main` el comando:

```
_CrtSetDbgFlag ( _CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF );
```

(También se puede mostrar la basura con el comando: `_CrtDumpMemoryLeaks()`)

Además, para que la información muestre el nombre del módulo y la línea de código, añade al proyecto un archivo "`checkML.h`" con las siguientes directivas de VS, e inclúyelo en los archivos de código fuente (archivos `.cpp`) del proyecto.

```
// archivo checkML.h
#ifdef _DEBUG
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define new DBG_NEW
#endif
#endif
```

4 Entrega de la práctica

La práctica se entregará a través del Campus Virtual, en la tarea Entrega de la Práctica 5 donde debes subir el archivo practica5.zip con todos los archivos `.h` y `.cpp` del proyecto.

Asegúrate de poner el **nombre** de los miembros del grupo en un comentario al principio de cada archivo.

Recuerda:

- La aplicación no debe dejar basura.
- No se permite el uso de variables globales, ni de instrucciones de salto, salvo un `return` como última instrucción de las funciones y break en los casos del `switch`.

Fin del plazo de entrega: 28 de mayo de 2017.