Tema 3: Gestión Dinámica de Memoria

UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y CC. Computación

PRÁCTICA DE LABORATORIO/EVALUABLE

Programación II
ST y TT

1. Desarrollar un TAD para gestionar la red social `Twitter`. 4 pts.

```
/*****************************************************************************
 * Clase Twitter
 *
 * Esta clase implementa toda la operativa relativa a una red social similar
 * a Twitter, con una lista de usuarios que publican tweets, se siguen
 * y se mencionan entre ellos, etc.
 *
 * Autor:
 * Fecha: Mon Mar 29 09:50:18 2016
 *****************************************************************************/

#ifndef __TWITTER__
#define __TWITTER__

#include "usuario_twitter.hpp"
#include <string>

namespace bblProgII {

class Twitter {
public:
    // Constructor por defecto
    // Inicializa la red como red vacía, sin usuarios y sin
    // sin ningún usuario conectado
    Twitter();

    // Constructor de copia de la clase
    Twitter(const Twitter &otro_twitter);

    // Operador de asignación (opcional)
    Twitter & operator=(const Twitter &otro_twitter);

    // Destructor de la clase.
    ~Twitter();

    // Devuelve el número de usuarios de la red social
    unsigned num_usuarios() const;

    // Devuelve el id del usuario actualmente conectado. Si no hay usuario
    // conectado, devuelve una cadena vacía
    std::string usuario_actual() const;

    //-----------------------------------------------------------------
    // PARA EL USUARIO ACTUAL. |
```

```
// ------------------------|
//
// Devuelve los seguidores del usuario conectado
// Si no hay usuario conectado, los métodos
// devuelven false a través de 'conectado'. Si no, devuelven true.
// Devuelve la lista de seguidores
void obtener_seguidores(Usuarios &lista_seg, bool &conectado) const;

// Devuelve la lista de usuarios a los que se sigue el usuario conectado
void obtener_siguiendo(Usuarios &lista_sig, bool &conectado) const;

// Devuelve la lista de tweets del usuario conectado
void obtener_tweets(Tweets &lista_tweets, bool &conectado) const;

// Indica si un determinado usuario es seguidor del usuario conectado
void me_sigue(const std::string &otro_usuario, bool &mesigue, bool &conectado) const;

// Indica si este usuario conectado está siguiendo a otro
void estoy_siguiendo(const std::string &otro_usuario, bool &sigo, bool &conectado) const;

// Devuelve el número de seguidores del usuario conectado
void num_seguidores(unsigned &n_mesiguen, bool &conectado) const;

// Devuelve el número de usuarios a los que sigue el usuario conectado
void num_siguiendo(unsigned &n_siguiendo, bool &conectado) const;

// Devuelve el número de tweets del usuario
void num_tweets(unsigned &n_tweets, bool &conectado) const;

// Imprime por pantalla la lista de seguidores
// Si num_imprime == 0, imprime todos los seguidores.
// PRECONDICIÓN: num_imprime <= num_seguidores
void imprimir_seguidores(unsigned num_imprime, bool &conectado) const;

// Imprime por pantalla la lista de usuarios a los que sigue
// Si num_imprime == 0, imprime todos los usuarios a los que sigue.
// PRECONDICIÓN: num_imprime <= num_siguiendo
void imprimir_siguiendo(unsigned num_imprime, bool &conectado) const;

// Imprime por pantalla la lista de tweets
// Si num_imprime == 0, imprime todos los tweets del usuario.
// PRECONDICIÓN: num_imprime <= num_tweets
void imprimir_tweets(unsigned num_imprime, bool &conectado) const;

// Modifica el identificador del usuario actualmente conectado
// Devuelve 'true' si ha sido posible modificar el usuario
// actualmente conectado.
// Deben modificarse la lista de seguidores y siguiendo de los
// demás usuarios de la red para actualizarlas con el nuevo
// identificador del usuario conectado
void establecer_id(const std::string &nuevo_id, bool &conectado);

// Inserta un usuario en la lista de usuarios a los que sigue-
// El usuario debe estar conectado y el usuario al que se quiere
// seguir debe existir. Si no hay usuario conectado actualmente,
// se devuelve 'false' a través de 'conectado'. Si el usuario que
// se quiere incluir en la lista de usuarios seguidos no existe en la
// red, se devuelve 'NO_EXISTE' a través de 'res'.
// Si ambos usuarios existen, se actualizan las listas de seguidores
```

```
// y siguiendo. Si la lista de seguidores o siguiendo están
// llenas, se devuelve 'LISTA_LLENA' a través de 'res'.
void nuevo_siguiendo(const std::string &nuevo, Resultado &res, bool &conectado);

// Inserta un nuevo tweet al final de la lista de tweets. Si la lista de
// tweets está llena, se devuelve 'LISTA_LLENA' a través de 'res'. Si no,
// se devuelve 'OK'. La longitud máxima
void nuevo_tweet(const Tweet &nuevo, Resultado &res, bool &conectado);

// Elimina un usuario en la lista de usuarios a los que sigue el usuario.
// El usuario debe estar conectado y el usuario al que se quiere
// dejar de seguir debe existir. Si no hay usuario conectado actualmente,
// se devuelve 'false' a través de 'conectado'. Si el usuario que
// se quiere eliminar de la lista de usuarios seguidos no existe en la
// red, se devuelve 'NO_EXISTE' a través de 'res'. Si ambos usuarios
// existen, se actualizan las lista se de seguidores y siguiendo,
// eliminando los nombres de los usuarios correspondientes.
void eliminar_siguiendo(const std::string &usuario, Resultado &res, bool &conectado);

//-----------------------------------------------------------------

//-----------------------------------------------------------------
// MÉTODOS DE LA RED SOCIAL

// Imprime por pantalla los usuarios actuales de la red social
void imprimir_usuarios() const;

// Cambia el usuario actualmente conectado a la red. Si el usuario
// existe, se cambia el usuario y se devuelve 'OK' a través del
// parámetro 'res'. Si el usuario no existe, se mantiene el usuario
// actual y se devuelve 'NO_EXISTE' a través del parámetro 'res'
void cambiar_usuario_actual(const std::string &nuevo_id_actual, Resultado &res);

// Insertar un nuevo usuario en la red social, a partir de su
// id de usuario, con todos sus datos vacíos (lista de seguidores,
// lista de tweets, etc.). Si el id del nuevo usuario no existe,
// se inserta en la red y se devuelve 'OK' a través del parámetro
// 'res'. Si el usuario ya existía, se devuelve 'YA_EXISTE' a
// través de 'res'.
void insertar_usuario(const std::string &nuevo_usuario, Resultado &res);

// Insertar un nuevo usuario en la red social.
// Si el id del nuevo usuario no existe,
// se inserta en la red y se devuelve 'OK' a través del parámetro
// 'res'. Si el usuario ya existía, se devuelve 'YA_EXISTE' a
// través de 'res'.
void insertar_usuario(const UsuarioTwitter &nuevo_usuario, Resultado &res);

// Elimina un usuario en la red social, a partir de su
// id de usuario. Si el id del nuevo usuario existe,
// se elimina de la red y se devuelve 'OK' a través del parámetro
// 'res'. Deben actualizarse la lista de seguidores y siguiendo
// en las listas de los demás usuarios de la red.
// Si el usuario no existía, se devuelve 'NO_EXISTE' a
// través de 'res'.
void eliminar_usuario(const std::string &usuario, Resultado &res);

// Recibe como parámetro el nombre del fichero con la lista de usuarios
// de la red. A partir de ahí, carga desde disco todos los seguidores,
```

```
    // usuarios seguidos, tweets e menciones de cada uno de los
    // usuarios de la red.
    // Si el fichero de la lista de usuario existe, se procede a la carga
    // de la red desde disco y se devuelve 'true' a través del parámetro
    // 'red_cargada'. Si no, la red permanece vacía y se devuelve
    // 'false' a través del parámetro.
    // Los ficheros de datos de cada usuario tienen como nombre el
    // nombre del usuario y como extensión '.seg' para los seguidores,
    // '.sig' para los usuarios seguidos, '.twt' para los tweets
    // y (opcional) '.men' para las menciones (véase la clase UsuarioTwitter).
    void cargar_red(const std::string &fic_lista_usuarios, bool &red_cargada);


    // Guarda toda la red social en disco. El nombre de fichero con los nombres de
    // usuario se pasa como parámetro. Los datos de cada usuario se guardan
    // según se indica en la clase UsuarioTwitter
    // Los ficheros de datos de cada usuario tienen como nombre el
    // nombre del usuario y como extensión '.seg' para los seguidores,
    // '.sig' para los usuarios seguidos, '.twt' para los tweets
    // y (opcional) '.men' para las menciones (véase la clase UsuarioTwitter).
    void guardar_todo(const std::string &nom_fic_usuario) const;
    //------------------------------------------------------------------

private:
    //------------------------------------------------------------------
    // TIPOS PRIVADOS:
    struct NodoUsuario {
        UsuarioTwitter usuario;
        NodoUsuario *sig;
    };
    typedef NodoUsuario *PtrUsuario;
    //------------------------------------------------------------------

    //------------------------------------------------------------------
    // ATRIBUTOS:
    PtrUsuario usuarios; // Lista de todos los usuarios de la red
    PtrUsuario usuario_conectado;  // Usuario actualmente conectado
    // a la red social.
    //------------------------------------------------------------------

    //------------------------------------------------------------------
    // MÉTODOS PRIVADOS:

    // Busca un usuario en la lista de usuarios. Si lo encuentra,
    // devuelve un puntero al nodo de la lista que contiene el usuario.
    // Si no lo encuentra, devuelve nullptr
    PtrUsuario buscar_usuario(const std::string &id_usuario) const;

    // Eliminar toda la lista de usuarios
    void borrar_todo();

    // Insertar un nuevo usuario al final de la lista de usuarios
    void insertar_usuario(const UsuarioTwitter &nuevo_usuario);

    // Eliminar un usuario de la lista de usuarios
    // PRECONDICIÓN: el usuario existe
    void borrar_usuario(const std::string &id_usuario);
    //------------------------------------------------------------------
};
```

```
}
#endif
```

2. Completar el siguiente programa principal para probar el funcionamiento correcto de los métodos de la clase `Twitter`: 0,5 pts.

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include "twitter.hpp"
#include "usuario_twitter.hpp"


using namespace std;
using namespace bblProgII;

// Escribe el menú de opciones, lee la opción por teclado y la devuelve
char menu();

// Pulsar una tecla para continuar
void Seguir();

// Escribir resultado de operación por pantalla
void escribir_resultado(const Resultado res);


int main() {
  /* ALUMNO:
    Declaraciones de datos: variables, objetos, etc.*/

  time_t tSac; // instante actual
  tm tms;

  system("clear"); //// En Windows: system("cls");
  bool respuesta;

  /* ALUMNO: cargar toda la red social desde disco y obtener 'respuesta'*/

  if (respuesta) {
    cout << "LA RED SE HA CARGADO DE DISCO CORRECTAMENTE..." << endl;
  } else {
    cout << "NO SE HA PODIDO CARGAR LA RED DESDE DISCO..." << endl;
  }
  Seguir();

  do {
    opcion = menu();
    switch (opcion) {
    case 'a': /* ALUMNO: completar */
      Seguir();
      break;
    case 'b':
      Seguir();
      break;
    case 'c': /* ALUMNO: completar */
      Seguir();
      break;
    case 'd': /* ALUMNO: completar */
      Seguir();
      break;
    case 'e': /* ALUMNO: completar */
```

```
      Seguir();
      break;
    case 'f': /* ALUMNO: completar */
      Seguir();
      break;
    case 'g': /* ALUMNO: completar solo si se implementan las menciones*/
      Seguir();
      break;
    case 'h': /* ALUMNO: completar solo si se implementan las menciones*/
      Seguir();
      break;
    case 'i': /* ALUMNO: completar */
      Seguir();
      break;
    case 'j': /* ALUMNO: completar */
      Seguir();
      break;
    case 'k': /* ALUMNO: completar */
      Seguir();
      break;
    case 'l': /* ALUMNO: completar */
      Seguir();
      break;
    case 'm': /* ALUMNO: completar */
      escribir_resultado(res);
      Seguir();
      break;
    case 'n': /* ALUMNO: completar */
      escribir_resultado(res);
      Seguir();
      break;
    case 'o': /* ALUMNO: completar */
      escribir_resultado(res);
      Seguir();
      break;
    case 'p': /* ALUMNO: completar */
      Seguir();
      break;
    case 'x': /* ALUMNO: completar */
    }
  } while (opcion != 'x');
}

char menu() {
  char opcion;

  system("clear"); // En Windows: system("cls");

  do {
    cout << "MENÚ DE OPCIONES (elija una opción y pulse <enter>):" << endl << endl;
    cout << "a -> Obtener id del usuario actualmente conectado" << endl;
    cout << "b -> Consultar si un usuario me sigue" << endl;
    cout << "c -> Consultar si estoy siguiendo a un usuario" << endl;
    cout << "d -> Imprimir mis estadísticas (seguidores, siguiendo, etc.)" << endl;
    cout << "e -> Imprimir mi lista de seguidores y usuarios a los que sigo" << endl;
    cout << "f -> Imprimir mi lista de tweets" << endl;
    /* OPCIONAL:
    cout << "g -> Imprimir mi lista de menciones por otros usuarios" << endl;
    cout << "h -> Actualizar mi lista de menciones que me han hecho otros usuarios" << endl;
```

```
    */
    cout << "i -> Modificar el id de mi usuario" << endl;
    cout << "j -> Insertar un nuevo tweet" << endl;
    cout << "k -> Insertar un nuevo usuario en mi lista de usuarios a los que sigo" << endl;
    cout << "l -> Eliminar un usuario de mi lista de usuarios a los que sigo" << endl;
    cout << "m -> Cambiar el usuario actualmente conectado" << endl;
    cout << "n -> Insertar un nuevo usuario en red" << endl;
    cout << "o -> Eliminar un usuario de la red" << endl;
    cout << "p -> Imprimir la lista de usuarios de la red social" << endl;
    cout << "x -> SALIR DEL PROGRAMA" << endl;

    cin.get(opcion); cin.ignore(); opcion = char(tolower(int(opcion)));
  } while ((opcion < 'a' || opcion > 'p') && (opcion != 'x'));

  return opcion;
}

void Seguir() {
  string enter;

  cout << endl << "Pulse <enter> para continuar..." << endl;
  getline(cin, enter);
}


void escribir_resultado(const Resultado res) {
  switch (res) {
  case OK: cout << "Operación realizada correctamente." << std::endl;
    break;
  case LISTA_LLENA: cout << "Operación NO realizada: lista llena." << std::endl;
    break;
  case YA_EXISTE: cout << "Operación NO realizada: el elemento ya existe" << std::endl;
    break;
  case NO_EXISTE: cout << "Operación NO realizada: el elemento no existe" << std::endl;
    break;
  case FIC_ERROR: cout << "Error en la apertura de fichero" << std::endl;
  }
}
```

OPCIONAL Desarrollar los siguientes métodos públicos relacionados con la gestión de menciones a otros usuarios en los tweets de los usuarios de la red social: 0,5 pts.

```
// Devuelve el número de menciones
void num_menciones(unsigned &n_menciones, bool &conectado) const;

// Devuelve la lista de menciones al usuario conectado
void obtener_menciones(Menciones &lista_int, bool &conectado) const;

// Devuelve el número de menciones
void num_menciones(unsigned &n, bool &conectado) const;

// Imprime por pantalla la lista de menciones
// Si num_imprime == 0, imprime todas las menciones.
// PRECONDICIÓN: num_imprime <= num_menciones
void imprimir_menciones(unsigned num_imprime, bool &conectado) const;

// Se obtiene la lista de menciones que los demás usuarios de
// la red tienen con el usuario actualmente conectado y se
// almacenan en la lista de menciones del usuario actualmente
// conectado.
// Las menciones comienzan todas con el carácter ampsersand ('@').
void obtener_menciones(bool &conectado);
```