Tema 2: Tipos Abstractos de Datos

UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y CC. Computación

PRÁCTICA DE LABORATORIO/EVALUABLE

Programación II
ST y TT

1. Dearrollar un TAD para gestionar un usuario de la red social `Twitter`. El usuario debe poder mantener una lista de usuarios (nombres de usuario) a los que sigue en la red social, una lista de usuarios que son sus seguidores y, por último, una lista de tweets. 4 pts.

   Un tweet comienza con una serie de datos numéricos, correspondientes al **día**, **mes**, **año**, **hora**, **minuto** y **segundo** en el que se escribió el tweet, y continúa con un cadena, con un máximo de 140 caracteres, correspondiente al `tweet`. Ejemplos de tweets, del usuario `ada_lovelace` podrían ser los siguientes:

   ```
   5 2 2014 9 10 1 Me encanta la maquina de @carlos_babbage, es mi favorita con diferencia

   5 2 2014 9 11 2 Creo que puedo escribir algun algoritmo para la maquina de
   @carlos_babbage, no me parece dificil @blas_pascal

   5 2 2014 19 1 10 He escrito un algoritmo para sumar numeros complejos con la maquina
   de @carlos_babbage, creo que me hare famosa
   ```

   Una vez desarrollado el TAD `UsuarioTwitter` debe implementarse un programa principal, lo más sencillo posible, para probar su correcto funcionamiento.

   La interfaz del TAD `UsuarioTwitter` es la siguiente:

   ```
   /*****************************************************************************
    * Clase UsuarioTwitter
    *
    * Esta clase implementa toda la operativa relativa a un usuario de la red
    * social 'Twitter', permitiendo la gestión de la lista de tweets,
    * usuarios seguidos (following) y usuarios seguidores (followers).
    *
    * Autor:
    * Fecha: Fri Mar 10 12:08:18 2017
    *****************************************************************************/

   #ifndef __USUARIO__TWITTER__
   #define __USUARIO__TWITTER__
   #include <string>
   #include <array>

   namespace {
       const unsigned MAX_USUARIOS = 100; // Máximo número de seguidores o siguiendo
       const unsigned MAX_TWEETS = 10000; // Máximo número de tweets del usuario
   }


   namespace bblProgII {
   //---------------------------------------------------------------------------
   // TIPOS PÚBLICOS
   ```

```
//
// Lista de usuarios
typedef std::array <std::string, MAX_USUARIOS> ListaUsuarios;
struct Usuarios {
    unsigned num_usuarios;
    ListaUsuarios listado;
};
//------------------------------------------------------------------------
// Lista de tweets
struct FechaHora {
    unsigned anyo, mes, dia, hora, minuto, segundo;
};
struct Tweet {
    std::string tweet;
    FechaHora fecha_hora;
};
typedef std::array <Tweet, MAX_TWEETS> ListaTweets;
struct Tweets {
    unsigned num_tweets;
    ListaTweets listado;
};
//------------------------------------------------------------------------
// Resulstado de las operaciones:
//  - OK: la operación se ha realizado con éxito
//  - LISTA_LLENA: la operación no ha sido posible: lista llena
//  - YA_EXISTE: la operación no ha sido posible: el elemento ya existe
//  - NO_EXISTE: el elemento no existe
//  - FIC_ERROR: error en el fichero
typedef unsigned Resultado;
const Resultado OK = 0,
                LISTA_LLENA = 1,
                YA_EXISTE = 2,
                NO_EXISTE = 3,
                FIC_ERROR = 4;

/* O, alternativamente,
enum Resultado {
  OK,
  LISTA_LLENA,
  YA_EXISTE,
  NO_EXISTE,
  FIC_ERROR
};
*/

class UsuarioTwitter {
public:
    // Constructor por defecto
    // Inicializar todos los datos vacíos.
    UsuarioTwitter();

    // Constructor extendido.
    // Inicializa el idenfificador de usuario con el 'id' que se pasa
    // como parámetro. Las listas de usuarios y tweets están
    // vacías.
    UsuarioTwitter(const std::string &id);

    // Constructor de copia
    UsuarioTwitter(const UsuarioTwitter &otro_usuario);
```

```
// Operador de asignación (¡¡¡OPCIONAL!!!)
UsuarioTwitter & operator=(const UsuarioTwitter &otro_usuario);

// Destructor de la clase
~UsuarioTwitter();

//-----------------------------------------------------------------
// MÉTODOS DE CONSULTA

// Devuelve el idenficador del usuario
std::string obtener_id() const;

// Devuelve la lista de seguidores
void obtener_seguidores(Usuarios &lista_seg) const;

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

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

// Indica si un determinado usuario es seguidor de este usuario
bool me_sigue(const std::string &otro_usuario) const;

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

// Devuelve el número de seguidores
unsigned num_seguidores() const;

// Devuelve el número de usuarios a los que se sigue
unsigned num_siguiendo() const;

// Devuelve el número de tweets del usuario
unsigned num_tweets() const;

// Imprime por pantalla la lista de seguidores
// Si num_imprime == 0, imprime todos los seguidores. Si no,
// se imprime el número de seguidores que se indica.
// PRECONDICIÓN: num_imprime <= num_seguidores
void imprimir_seguidores(unsigned num_imprime) const;

// Imprime por pantalla la lista de usuarios a los que sigue
// Si num_imprime == 0, imprime todos los usuarios a los que sigue.
// Si no, imprime el número de usuarios que se indica.
// PRECONDICIÓN: num_imprime <= num_siguiendo
void imprimir_siguiendo(unsigned num_imprime) const;

// Imprime por pantalla la lista de tweets
// Si num_imprime == 0, imprime todos los tweets del usuario.
// Si no, imprime el número de tweets que se indica.
// PRECONDICIÓN: num_imprime <= num_tweets
void imprimir_tweets(unsigned num_imprime) const;

// Guarda en fichero la lista de seguidores
void guardar_seguidores(const std::string &nom_fic, Resultado &res) const;

// Guarda en fichero la lista de usuarios a los que sigue
```

```
void guardar_seguiendo(const std::string &nom_fic, Resultado &res) const;

// Guarda en fichero los tweets del usuario
void guardar_tweets(const std::string &nom_fic, Resultado &Res) const;

// Guarda en fichero las listas de usuarios  y tweets
void guardar_todo(const std::string &nom_fic_seguidores,
                  const std::string &nom_fic_siguiendo,
                  const std::string &nom_fic_tweets,
                  Resultado &res_seguidores,
                  Resultado &res_siguiendo,
                  Resultado &res_tweets) const;

//-----------------------------------------------------------------
// MÉTODOS DE ACTUALIZACIÓN

// Modifica el identificador del usuario
void establecer_id(const std::string &nuevo_id);

// Inserta un seguidor en la lista de seguidores.
// Si el nuevo seguidor no existe, se inserta de manera ordenada
// (orden lexicográfico creciente) y se devuelve 'OK' a través de
// 'res'. Si ya existe el seguidor, no se inserta y se devuelve
// 'YA_EXISTE' a través de 'res'. Si no, si la lista de seguidores está
// llena, se devuelve 'LISTA_LLENA' a través de 'res'.
void nuevo_seguidor(const std::string &nuevo, Resultado &res);

// Inserta un usuario en la lista de usuarios a los que sigue.
// Si el nuevo usuario no existe, se inserta de manera ordenada
// (orden lexicográfico creciente) y se devuelve 'OK' a través de
// 'res'. Si ya existe el usuario, no se inserta y se devuelve
// 'YA_EXISTE' a través de 'res'. Si no, si la lista de seguidores está
// llena, se devuelve 'LISTA_LLENA' a través de 'res'.
void nuevo_siguiendo(const std::string &nuevo, Resultado &res);

// 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 del tweet es 140 caracteres, por
// lo que si el texto del tweet tiene más de 140 caracteres, los
// caracteres sobrantes por el final se eliminarán.
void nuevo_tweet(const Tweet &nuevo, Resultado &res);

// Elimina a un usuario de la lista de seguidores
// Si el usuario existe, se elimina y se devuelve 'OK' a través de
// 'res'. Si no existe el usuario, se devuelve 'NO_EXISTE' a través
// de 'res'.
void eliminar_seguidor(const std::string &usuario, Resultado &res);

// Elimina a un usuario de la lista de usuarios a los que sigue
// Si el usuario existe, se elimina y se devuelve 'OK' a través de
// 'res'. Si no existe el usuario, se devuelve 'NO_EXISTE' a través
// de 'res'.
void eliminar_siguiendo(const std::string &usuario, Resultado &res);

// Carga desde fichero la lista de seguidores,
// eliminando los seguidores actuales. Si el fichero se ha leído
// correctamente y los usuarios caben en la lista, se devuelve
// 'OK' a través de 'res'. En caso contrario, se devuelve
// 'FIC_ERROR' o 'LISTA_LLENA', respectivamente (aunque se insertan
```

```
    // solo los usuarios que caben en la lista).
    void cargar_seguidores(const std::string &nom_fic, Resultado &res);

    // Carga desde fichero la lista de usuarios a los que se sigue,
    // eliminando los usuarios seguidos actuales. Si el fichero se ha leído
    // correctamente y los usuarios caben en la lista, se devuelve
    // 'OK' a través de 'res'. En caso contrario, se devuelve
    // 'FIC_ERROR' o 'LISTA_LLENA', respectivamente (aunque se insertan
    // solo los usuarios que caben en la lista).
    void cargar_seguiendo(const std::string &nom_fic, Resultado &res);

    // Carga desde fichero la lista de tweets del usuario,
    // eliminando los tweets actuales. Si el fichero se ha leído
    // correctamente y los tweets caben en la lista, se devuelve
    // 'OK' a través de 'res'. En caso contrario, se devuelve
    // 'FIC_ERROR' o 'LISTA_LLENA', respectivamente (aunque se insertan
    // solo los tweets que caben en la lista).
    void cargar_tweets(const std::string &nom_fic, Resultado &res);


    // Carga desde fichero las listas de usuarios y tweets. Si cada fichero se ha leído
    // correctamente y los usuarios/tweets caben en la lista correspondiente, se devuelve
    // 'OK' a través del parámetro 'res_*' correspondiente. En caso contrario, se devuelve
    // 'FIC_ERROR' o 'LISTA_LLENA', respectivamente (aunque se insertan
    // solo los usuarios/tweets que caben en la lista correspondiente).
    void cargar_todo(const std::string &nom_fic_seguidores,
                     const std::string &nom_fic_siguiendo,
                     const std::string &nom_fic_tweets,
                     Resultado & res_seguidores,
                     Resultado & res_siguiendo,
                     Resultado & res_tweets);

private:
    //-----------------------------------------------------------------
    // ATRIBUTOS: A DEFINIR POR EL ALUMNO
    //
    // Identificador del usuario
    // ... id_usuario;
    // Lista de tweets
    // ... tweets;
    // Lista de usuarios a los que se estoy siguiendo
    // ... siguiendo;
    // Lista de usuarios que me siguen
    // ... seguidores;
    //-----------------------------------------------------------------

    //-----------------------------------------------------------------
    // MÉTODOS PRIVADOS:
    //
    // Busca a un usuario en la lista ordenada de usuarios. Si lo
    // encuentra, devuelve la posición de la lista donde está. Si no,
    // devuelve la posición donde debería estar según el orden de la
    // lista.
    unsigned buscar_usuario(const Usuarios &usuarios, const std::string &user) const;

    // Inserta un usuario en la lista en la posición indicada
    // PRECONDICIÓN: el usuario cabe en la lista
    // PRECONDICIÓN: la posición es correcta
    void insertar_usuario_pos(Usuarios &usuarios, unsigned pos, const std::string &usuario);
```

```
    // Elimina un usuario de una posisición
    // PRECONDICIÓN: la posición es correcta
    void eliminar_usuario_pos(Usuarios &usuarios, unsigned pos);
};
}
#endif
```

OPCIONAL Completar la clase `UsuarioTwitter` añadiendo los tipos necesarios para la gestión de *menciones* entre los usuarios. Una mención se produce cuando un usuario incluye el nombre de otro usuario de la red en uno de sus tweets. Por ejemplo, en la lista de tweets del usuario `ada_lovelace` (que tiene solo 3 tweets) [1 pts.]

```
5 2 2014 9 10 1 Me encanta la maquina de @carlos_babbage, es mi favorita con diferencia

5 2 2014 9 11 2 Creo que puedo escribir algun algoritmo para la maquina de
@carlos_babbage, no me parece dificil @blas_pascal

5 2 2014 19 1 10 He escrito un algoritmo para sumar numeros complejos con la maquina
de @carlos_babbage, creo que me hare famosa
```

se producen menciones a dos usuarios: `carlos_babbage`, en los tweets 0, 1 y 2, y `blas_pascal`, en el tweet 1.

Un objeto de la clase `UsuarioTwitter` guardará una lista con las menciones que otros usuarios hacen de él/ella en sus tweets. Para cada mención, se guardará el nombre del usuario que la realizó y el número de tweet donde lo hizo. Así, dado el ejemplo anterior, el usuario `carlos_babbage` deberá guardar en su lista de menciones tres menciones realizadas por el usuario `ada_lovelace`: {{ {"ada_lovelace", 0}, {"ada_lovelace", 1}, {"ada_lovelace", 2} }}.

Debe definirse, en el espacio de nombres `bblProgII`, los siguientes tipos:

```
//--------------------------------------------------------------------------
// Lista de menciones de otros usuarios conmigo.
// Cada mención almacena el identificador del usuario
// que me ha mencionado y el número de tweet en el que lo ha hecho
struct Mencion{
    std::string id_usuario; // Indentificador de usuario que me menciona
    unsigned id_tweet; // Identificador del tweet que me menciona
};
typedef std::array <Mencion, MAX_MENCIONES> ListaMenciones;
struct Menciones
{
    unsigned num_menciones;
    ListaMenciones listado;
};
```

La constante `MAX_MENCIONES` se puede declarar en un `namespace` sin nombre, para mantenerla oculta:

```
namespace {
    const unsigned MAX_MENCIONES = 1000; // Máximo número menciones
}
```

Añadir a la clase `UsuarioTwitter` los siguientes métodos:

```
// Devuelve la lista de menciones que se han hecho a este usuario
void obtener_menciones(Menciones &menciones) const;

// Devuelve el número de menciones
unsigned num_menciones() 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) const;

// Se obtiene la lista de menciones que el usuario actual
// realiza con el usuario que se pasa como parámetro. Para ello
// se recorre la lista de tweets del usuario actual para localizar
```

```
// los tweets donde el usuario actual menciona al usuario que se
// pasa como parámetro. Las menciones comienzan todas con el carácter
// ampsersand ('@').
void obtener_menciones(const std::string &con_usuario,
                       Menciones &menciones) const;

// Inserta una nueva mención en la lista de
// menciones de este usuario.  Si la lista de menciones está
// llena, se devuelve 'LISTA_LLENA' a través de 'res'. Si la
// mención ya existe (coincide el id del usuario y el
// id del tweet de la mención) no se inserta, y se devuelve 'YA_EXISTE' a
// través de 'res'.
void nueva_mencion(const Mencion &mencion, Resultado &res);

// Carga desde fichero la lista de menciones,
// eliminando las menciones actuales. Si el fichero se ha leído
// correctamente y las menciones caben en la lista, se devuelve
// 'OK' a través de 'res'. En caso contrario, se devuelve
// 'FIC_ERROR' o 'LISTA_LLENA', respectivamente (aunque se insertan
// solo las menciones que caben en la lista).
void cargar_menciones(const std::string &nom_fic, Resultado &res);

// Guarda en fichero la lista de menciones
void guardar_meciones(const std::string &nom_fic, Resultado &res) const;
```

Modificar los métodos `cargar_todo` y `guardar_todo` para que incluyan también las menciones:

```
// Guarda en fichero las listas de usuarios, tweets y menciones
void guardar_todo(const std::string &nom_fic_seguidores,
                  const std::string &nom_fic_siguiendo,
                  const std::string &nom_fic_tweets,
                  const std::string &nom_fic_menciones,
                  Resultado &res_seguidores,
                  Resultado &res_siguiendo,
                  Resultado &res_tweets,
                  Resultado &res_menciones) const;

// Carga de fichero las listas de usuarios, tweets y menciones
void cargar_todo(const std::string &nom_fic_seguidores,
                  const std::string &nom_fic_siguiendo,
                  const std::string &nom_fic_tweets,
                  const std::string &nom_fic_menciones,
                  Resultado &res_seguidores,
                  Resultado &res_siguiendo,
                  Resultado &res_tweets,
                  Resultado &res_menciones);
```