UTN – FRBA – Algoritmos y Estructura de Datos – Examen Final – 07/12/2016

Apellido y nombre:_____ Legajo: _____ Cursó con Prof: _____

- Si luego de la lectura del examen, durante la resolución tiene alguna duda, escriba hipótesis de trabajo, las cuales también serán evaluadas.
- Los puntos que solicitan codificación puede ser respondidos en C, ó C++, pero debe indicar el lenguaje utilizado.
- En C y C++ prototipo refiere a la declaración de la función, es decir tipo de dato retornado, nombre de la función, y tipos de los parámetros.

WhatsApp

Temas evaluados: Abstracción, estructuras de datos enlazadas.

Definición del problema

Usted es parte de un equipo que desarrolla aplicaciones para celulares WhatsApp. La aplicación actualiza las conversaciones según los eventos remotos que se producen: llegadas de mensajes nuevos, recepción y lectura de mensajes enviados, conversaciones nuevas. Usted es el encargado de procesar los mensajes nuevos.

Dinámica del mantenimiento de las conversaciones.

La información disponible se encuentra en las siguientes estructuras de datos:

- ***LasConversaciones,*** una lista donde cada elemento tiene un *idConversacion* (entero) que identifica unívocamente la conversación, el *IdDelEmisor*, el *IdDelReceptor,* ambos enteros. La *Fecha* con formato AAMMDDHHMM, que se actualiza con cada mensaje recibido o enviado y punteros a sublistas de *LosMensajesRecibidos* y *LosMensajesEnviados*;
- ***LosUsuarios,*** una lista de usuarios participantes, con un *IdUsuario* que lo identifica unívocamente, *Recibido*, un dato de tipo booleano que debe pasar a true cuando se notifica a un usuario que recibió un mensaje, *Entregado*, cuando se notifica a un ususario que su mensaje fue enviado y *OtrosDatos* en una cadena de caracteres;
- ***LosMensajesEnviados***, ***LosMensajesRecibidos,*** sublistas de cada conversación con el intercambio de mensajes entre usuarios, ordenadas por fecha decreciente (AAMMDDHHMM). Cada elemento de las listas secundarias contienen la *Fecha*, con el formato decripto, el *IdMensaje* (entero que identifica al mensaje en la conversación), *Leido* un tipo de dato booleano con valor true, si el mensaje fue leído y false en caso contrario y el Mensaje, una cadena de caracteres.

Dinámica de los mensajes

Cuando llega un evento remoto, este se procesa invocando un subprograma específico para cada de evento. Su tarea es resolver el procesamiento de un evento de mensaje nuevo.

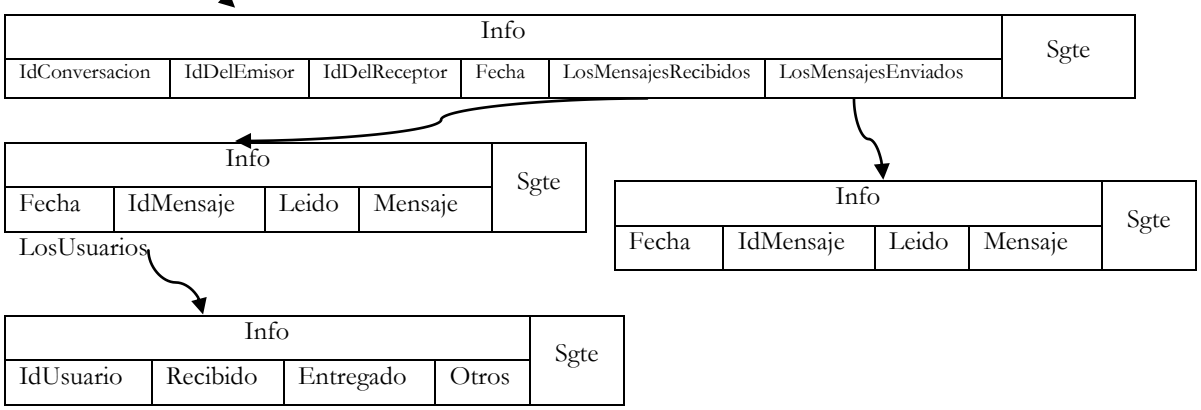
Cuando llega un mensaje nuevo este debe agregarse a *LosMensajesRecibidos* de la conversación a la que pertenece. La sublista debe mantener el orden por fecha y la conversación debe pasar al principio de la lista *LasConversaciones*. Se debe además notificar al usuario que lo envió que el mismo fue recibido. Se recibe *IdConversacion*, char Mensaje[N]. Se asume que las conversaciones existen

Se dispone de las siguientes funciones que puede utilizar sin desarrollar para el tipo de dato que decida

1. Nodo* insertarOrdenado (Nodo* & l, tipoDato x) Inserta siempre el valor x en la estructura apuntada por l.
2. Nodo* eliminarNodo (Nodo* &l, tipoDato x) que saca el nodo que contiene el valor x en el campo de la información y lo retorna.
3. Nodo* buscarEnLista(Nodo* l, tipoDato x) busca el nodo con x en el campo info
4. int fechaAEntero() toma la fecha del sistema y retorna un entero con formato AAMMDDHHMM.
5. int numeroMensaje() genera un numero único para cada mensaje

Se pide

1. Defina y declare todas estructuras de datos necesarias para la resolución del problema
2. Declare el prototipo de la función LosMensajesRecibidos, para cumplir con lo que se requiere cuando llega un mensaje nuevo.
3. Codifique o diagrame la función anterior. Recuerde: 1) Una conversación puede constar de uno o varios mensajes, los cuales deben estar en la conversación en el orden en el que fueros realizados. 2) al recibir un mensaje la conversación pasa al primer lugar.



1. Defina y declare todas estructuras de datos necesarias para la resolución del problema

```
struct InfoLosMensajes {
        int Fecha;
        int  IdMensaje;
        bool Leido;
        char Mensaje[N]
        };
struct NodoLosMensajes {
        InfoLosMensajes Info;
        NodoLosMensajes* Sgte
        ;
struct InfoConversacion{
        int IdConversacion;
        int IdDelEmisor;
        int IdDelReceptor;
        int Fecha;
        NodoLosMensajes* LosMensajesRecibidos;
        NodoLosMensajes* LosMensajesEnviados
        };
struct NodoConversacion {
         InfoConversacion Info;
        NodoConversacion* Sgte
        };
struct InfoLosUsuarios {
        int IdUsuario;
        bool  Recibido;
        bool Entregado;
        char Otros[N]
        };
struct NodoLosUsuarios {
        InfoLosUsuarios Info;
        NodoLosUsuarios* Sgte
        ;
```

2. Declare el prototipo de la función LosMensajesRecibidos, para cumplir con lo que se requiere cuando llega un mensaje nuevo.

```
void LosMensajesRecibidos(NodoConversacion* & , NodoLosUsuarios* , int , char [ ])
```

// la función retorna void y recibe el numero de conversación y el mensaje, pero como debe modificar las listas de conversación y usuarios recibe un puntero a las mismas pasado por referencia, aunque en el caso de la lista de usuarios podría ser pasado por valor ya que el puntero permanecerá sin modificaciones, no asi en el caso de la conversación ya que el nodo que representa a la misma deberá colocarse en la primera posición de la estructura.

3. Codifique o diagrame la función anterior. Recuerde: 1) Una conversación puede constar de uno o varios mensajes, los cuales deben estar en la conversación en el orden en el que fueros realizados. 2) al recibir un mensaje la conversación pasa al primer lugar.

4.
```
void LosMensajesRecibidos(NodoConversacion* & lc, NodoLosUsuarios* lu, int IdConversacion, char Mensaje[ ])
{
InfoLosMensajes Info; // para actualizar la sublista
NodoLaConversacion* p = eliminarNodo(lc, IdConversacion); //busca el nodo de la conversación
NodoLosUsuarios* q = buscarEnLista(lu, p->Info.Emisor) //busca al usuario que envió el mensaje
q->Info.Recibido = true; // notifica al emisor que el mensaje fue recibido
p->sgte = lc; //el sgte del nodo de la conversación apunta al principio de la lista
lc = p; //hace que el nodo de la conversación quede al principio
p->Info.Fecha = fechaAEntero(); //actualiza la fecha con la función provista
Info.Fecha = p->Info.Fecha; // genera la información de la sublista
Info.IdMensaje = numeroMensaje();
Info.Mensaje = Mensaje;
insertarOrdenado(p->Info.LosMensajesRecibidos, Info) // Inserta el nodo de la sublista
return;
}
```