UTN – FRBA – Algoritmos y Estructura de Datos – Examen Final – 14/12/2016

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.

En la aplicación instalada en el celular de un usuario se tiene:

- La información de sus conversaciones que se almacenan en una lista enlazada. Cada elemento tiene un idConversacion que identifica unívocamente la conversación, el nombre del grupo si es una conversación grupal o "" si es una conversación de mensajes directos con otro usuario; la configuración de notificaciones con dos valores booleanos: 1. activadas o desactivadas, 2. mostrar vista previa o no.

Info				Sgte
IdConversacion	Nombre	Notificar	VistaPrevia	

- La información de los contactos del usuario que se almacena en un arreglo. Cada elemento tiene un idUsuario, el nombre del contacto y un indicador de bloqueado o no.

Cuando llega un evento remoto al celular, este se procesa invocando un subprograma específico para cada evento. *Su tarea es resolver la notificación al usuario de un mensaje nuevo.*

Cuando llega un mensaje nuevo debe mostrarse una notificación al usuario dependiendo de la configuración de la conversación y del usuario emisor. *El mensaje recibido debe notificarse si la conversación tiene activadas las notificaciones y el usuario emisor no está bloqueado en la lista de contactos*. En caso de mostrarse la notificación debe *EmitirNotificacion(Titulo, Descripcion)*

	Conversación Directa	Conversación Grupal
Con Vista Previa	T→<Nombre del contacto emisor> D→<Vista Previa del mensaje>	T→<Nombre del grupo> D→<Nombre del emisor>: <Vista Previa del mensaje>
Sin Vista Previa	T→<Nombre del emisor> D→"Nuevo Mensaje."	T→<Nombre del grupo> D→"Nuevo Mensaje de:" <Nombre del emisor>

Aclaraciones:

- Si el usuario emisor no está en la lista de contactos no está bloqueado. En este caso en vez de mostrar el nombre del contacto debe mostrar el número telefónico utilizando, sin desarrollar, la función: s*tring NumeroTelefonicoUsuario(int idUsuario).*
- Para obtener la vista previa de un mensaje invoque sin desarrollar la función: *string VistaPrevia(Mensaje mensaje).*
- Para concatenar dos cadenas utilice el operador +.
- El tipo Mensaje es un tipo abstracto que no es necesario declarar.
- La conversación existe en la lista.

Se pide

1. Defina y declare todas estructuras de datos para las conversaciones y los usuarios (son 200 como máximo).
2. Declare el prototipo de la función NotificarMensajeRecibido, que recibe la lista de conversaciones, el arreglo de contactos, el tamaño del arreglo, el idConversacion, el idUsuarioEmisor y el mensaje (de tipo Mensaje).

3. Desarrolle la función **NotificarMensajeRecibido**, que prepara la notificación del mensaje recibido al usuario si corresponde. Para lanzar una notificación invocar sin desarrollar *void EmitirNotificación(string titulo, string descripcion).* Además dispone de las funciones: *string NumeroTelefonicoDelUsuario(IdEmisor)* *string VistaPreviaMensaje(Mensaje)*	La función debe: ✓ Buscar la conversación ✓ Buscar el contacto ✓ Hacer las notificaciones según corresponda o Usuario existe y no esta bloquedo o Usuario existe pero está bloqueado o Usuario no existe o Conversación directa o grupal o Con o sin vista previa Dispone para usar sin desarrollar las funciones: NumeroTelefonicoDelUsuario(IdEmisor) VistaPreviaMensaje(Mensaje) EmitirNotificacion(Titulo, Descripcion)

Solución propuesta:

1.

```
struct Conversacion {
    int idConversacion;
    string nombre;
    bool notificar;
    bool vistaPrevia;
}
struct Contacto {
    int idUsuario;
    string nombre;
    bool bloqueado;
}
struct Nodo {
    Conversacion info;
    Nodo* sgte;
}
```

2.

```
void NotificarMensajeRecibido(Nodo* Conversaciones, Contacto contactos[], int cantContactos, int idConversacion, int idUsuarioEmisor, Mensaje mensaje);
```

3.

```
// funciones auxiliares
int BuscarContacto(Contacto contactos[], int n, int idUsuario) {
    int pos = -1;
    int i = 0;
    while (i < n && pos != -1) {
        if (contactos[i].idUsuario == idUsuario) {
            pos = i;
        }
        i++;
    }
    return pos;
}

// asume que la conversación existe
Nodo* BuscarConversacion(Nodo* conversaciones, int idConversacion) {
    while (conversaciones != null && conversaciones->info.idConversacion != iConversacion) {
        conversaciones = conversaciones->sgte;
    }
    return conversaciones;
}
```

```
void NotificarMensajeRecibido(Nodo* conversaciones, Contacto contactos[], int cantContactos, int idConversacion, int
idUsuarioEmisor, Mensaje mensaje) {
    // Buscamos la conversación:
    Nodo* conversacion = BuscarConversacion(conversaciones, idConversacion);

 // Buscamos el contacto, -1 es no encontrado:
    int pos = BuscarContacto(contactos, cantContactos, idUsuarioEmisor);
    bool bloqueado;
    string nombreContacto;
   // si encuentra al usuario verifica si está bloqueado y toma el nombre
    if (pos != -1) { // Encontrado
        bloqueado = contactos[pos].bloqueado;
        nombreContacto = contactos[pos].nombre;
    }
    else {//en caso de no encontrarlo toma el número de teléfono como nombre usa función provista
        bloqueado = false;
        nombreContacto = NumeroTelefonicoUsuario(idUsuarioEmisor);
    }// fin del control del usuario para determinar si está o no esta

    //solo notifica a usuarios no bloqueados, por lo que verifica si no esta bloqueado y quiere ser notificado
    if (!bloqueado && conversacion->info.notificar) { // Notificar
        string titulo, descripcion;
        if (conversacion->info.nombre == "") { // Si es conversación Directa
            titulo = nombreContacto; //para emitir en la notificación solo el nombre del emisosr
            if (conversacion->info.vistaPrevia) {//Si desea vista previa
                descripcion = VistaPrevia(mensaje);//la descripción para emitir notificación
            }
            else {
                descripcion = "Nuevo Mensaje.";
            }
        }// fin de conversación directa
        else { // Conversacion Grupal
            titulo = conversacion->info.nombre; // debe mostrar nombre del grupo
            if (conversacion->info.vistaPrevia) {
                descripcion = nombreContacto + ": " + VistaPrevia(mensaje);
            }
            else {
                descripcion = "Nuevo Mensaje de " + nombreContacto;
            }
        }// fin conversación grupal
        EmitirNotificacion(titulo, descripcion);//si corresponde emite
    }   // fin condicional de usuario que se debe notificar
    return; // fin de la funcion
}
```