UNIVERSIDAD DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación

LABORATORIO DE PROGRAMACIÓN

E.T.S.I. Informática- 1º SISTEMAS. 26-JUNIO-2008

APELLIDOS_____NOMBRE_____

DNI _____ ORDENADOR_____GRUPO (A/B/C)_____

Debido al gran número de eventos que tenemos a lo largo del mes de Julio necesitamos un programa que gestione nuestras citas. Para ello implementaremos una estructura que será una array de 31 posiciones (una por cada día del mes) y cuyo tipo base será una lista enlazada de citas que estará ordenada de forma creciente por hora (hora y minutos) (ver **Figura 1**). La estructura se definirá y gestionará mediante los módulos `MAgenda`, `MLista`, `MCita` y `MCadena` que se especifican más adelante. El programa principal (`julio.cpp`) deberá presentar el siguiente menú:

```
 Nombre: (Apellidos, Nombre)    Curso: 1º
 Especialidad: Sistemas         Grupo: A/B/C
 Puesto: número de ordenador    Fecha: 26/06/2008

     Agenda del Mes de Julio
     =======================
 A. Insertar Cita.
 B. Cargar Citas desde Fichero de Texto.
 C. Mostrar Todas las Citas.
 D. Eliminar Cita Hora.
 E. Salvar Todas las Citas a Fichero de Texto.
 F. Buscar Cita.
 G. Eliminar Tipos de Cita.
 H. Modificar Cita.
 I. Generar Carta de Disculpa.
 X. Salir del Programa.

Introduzca Opción: _
```

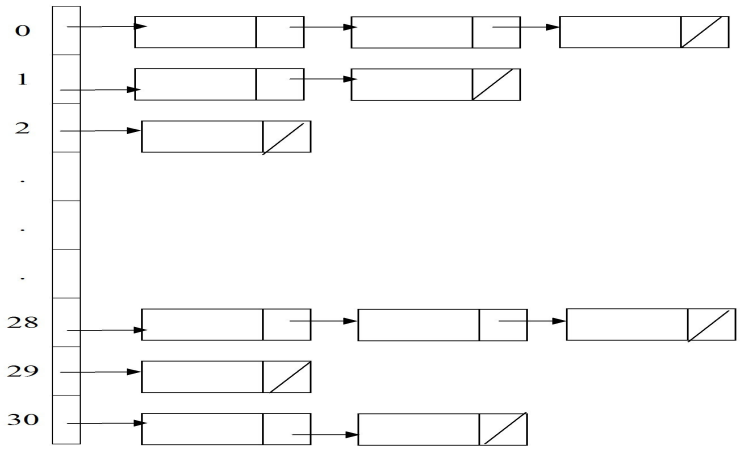

Figura 1. Esquema de la Estructura `TAgenda`

Descripción de Opciones:

A. Insertar Cita. Se leerá desde teclado el día y toda la información de una cita (hora, descripción y tipo de cita) y se insertará en la agenda. El tipo de cita se deberá pedir al usuario hasta que sea correcto. Si el día es incorrecto (no está entre 1 y 31) o la agenda está llena o ya existiera una cita en ese día y hora se informará del correspondiente error. Ejemplo:

```
Dia          : 27
Descripcion  : Ir a la playa
Hora(0-23)   : 11
Minutos(0-59): 30
Tipo de Cita (Personal, Trabajo, Ocio): Personal
```

B. Cargar Citas desde Fichero de Texto. Se pedirá al usuario el nombre de un fichero de texto en el que estén almacenados las citas y se insertarán en la agenda que se pasa como parámetro. El contenido de la agenda NO deberá ser borrado antes de cargar los nuevos datos y se ignorará la inserción de una cita en caso de que ya existiera otra cita en ese día y hora. Si el fichero no existe o no puede ser abierto se informará del error producido. El formato del fichero queda a elección del alumno.

C. Mostrar Todas las Citas. Se mostrará por pantalla un listado de todas las citas de cada día.

D. Eliminar Cita. Se pedirá por teclado el día y la hora de la cita y se mostrará (si existe) la información de dicha cita. Posteriormente se pedirá confirmación de borrado y si es afirmativa se borrará. Si el día es incorrecto (no está entre 1 y 31) o no existe una cita en ese día y hora se informará del error. Ejemplo:

```
Dia          : 27
Hora(0-23)   : 11
Minutos(0-59): 30
Se procede a borrar la siguiente cita ...
Descripcion  : Ir a la playa
Hora         : 11:30
Tipo de Cita : Personal

¿Esta seguro(S/N)?
```

E. Salvar Todas las Citas a Fichero de Texto. Se pedirá al usuario el nombre de un fichero de texto que será creado, o si existiera se borrará su contenido, y se salvarán todos los datos de la agenda en el fichero. El contenido de la agenda no será modificado y se informará de cualquier error producido. El formato del fichero será el mismo que se haya elegido para la lectura.

F. Buscar Cita. Se pedirá por teclado el día y la hora de la cita y se mostrará (si existe) la información de dicha cita. Si el día es incorrecto (no está entre 1 y 31) o no existe una cita en ese día y hora se informará del error. Ejemplo:

```
Dia          : 27
Hora(0-23)   : 11
Minutos(0-59): 30

CITA
====
Descripcion  : Ir a la playa
Hora         : 11:30
Tipo de Cita : Personal
```

G. Eliminar Tipos de Cita. Se pedirá por teclado el día y el tipo de la cita y se borrarán de la agenda todas las citas de ese tipo que hubiera en ese día. Si el día es incorrecto (no está entre 1 y 31) se informará del error. Ejemplo:

```
Dia          : 27
Tipo de Cita : Personal
```

H. Modificar Cita. Se leerá desde teclado el día y toda la información de una cita (hora, descripción y tipo de cita). Si el día es incorrecto (no está entre 1 y 31) se informará del correspondiente error. Si hay una cita en ese día y hora se modificará su descripción y tipo de cita por la leída. Si no existiera una cita en ese día y hora se insertará como si fuera nueva. Ejemplo:

```
Dia          : 27
```

```
Descripcion  : Ir a la playa con los amigos
Hora(0-23)   : 11
Minutos(0-59): 30
Tipo de Cita (Personal, Trabajo, Ocio): Personal
```

I. Generar Carta de Disculpa. Se solicitará al usuario el nombre de un fichero de texto donde hay un modelo de carta y el nombre de un fichero de texto donde se generará la nueva carta. Además, hay que solicitar al usuario el día y la hora de la cita para la que quiere generar la carta. Si el día es incorrecto o la cita no existe se informará del error. Si no se puede abrir algún fichero también se notificará el error. El proceso de generación de la carta será ir leyendo el modelo de carta y cuando se encuentre una palabra clave sustituir esa palabra por el valor de ese campo en la cita.

Palabras clave:

Clave	Valor	Clave	Valor
DIA	día de la cita	$HORA$	**hora de la cita**
$TIPO$	**tipo de cita**	**$DESCRIPCION$**	**descripción de la cita**

Ejemplo:

```
Nombre Fichero con la Carta Modelo: modelo_carta.txt
Nombre Fichero de la Carta Generada: carta.txt
Dia          : 14
Hora(0-23)   : 15
Minutos(0-59): 15
```

carta_modelo.txt

```
Estimado Señor,
lamento no poder acudir a la reunión del día $DIA$ ya que tengo una
cita de $TIPO$ a las $HORA$ consistente en "$DESCRIPCION$".
Muchas Gracias, atentamente José Luis
```

carta.txt

```
Estimado Señor,
lamento no poder acudir a la reunión del día 14 ya que tengo una cita
de Ocio a las 15:15 consistente en "Ver etapa del Tour de Francia".
Muchas Gracias, atentamente José Luis
```

X. Salir del Programa. En esta opción se pedirá confirmación de salida. Si es afirmativa, se terminará el programa, liberando todos los recursos que hayan sido reservados y si es negativa, se volverá al menú principal.

Módulos a Utilizar:

- `MCadena` (`MCadena.h, MCadena.cpp`): Módulo de manejo de cadenas. Utiliza el módulo `MCadena` que se proporciona, en el que se define:
  - Las constantes necesarias y el tipo `TCadena` como un array de 80 caracteres.
  - El enumerado `TCompara` con los valores: `Menor, Igual, Mayor`.
  - La función `CopiaCadena` y `ComparaCadena`.

Módulos a Implementar:

- `Programa Principal` (`julio.cpp`)
- `MCita` (`MCita.h, MCita.cpp`): Módulo de manejo de Citas
  - Define las constantes `necesarias`.
  - Define los tipos `TNatural` (entero sin signo), `TTipoCita` (enumerado), `THora` (registro hora y minutos), y `TCita` (registro con la descripción, hora y tipo de una cita).
  - Define la función `LeerHora`: lee una hora correcta desde teclado.
  - Define la función `SUC_TipoCita`: retorna el sucesor del `TTipoCita` que se le pasa.
  - Define la función `TipoCita_a_Cadena`: convierte un `TTipoCita` a cadena.
  - Define la función `Cadena_a_TipoCita`: convierte una cadena a `TTipoCita`.
  - Define la Función `LeerTipoCita`: retorna un `TTipoCita` leído desde teclado.
  - Define la función `LeerCita`: lee un registro `TCita` desde teclado.

- Define la función `EscribirCita`: escribe por pantalla un registro `TCita`.
- Define la función `LeerCitaFicheroTXT`: lee un registro `TCita` desde un fichero de texto cuyo manejador se le pasa como parámetro.
- Define la función `EscribirCitaFicheroTXT:` escribe un registro `TCita` en un fichero de texto cuyo manejador se le pasa como parámetro.
- Define la función `HoraCita` que devuelve la hora de una cita.
- Define la función `TipoCita` que devuelve el tipo de cita de una cita.
- Define la función `ComparaHoras:` función que dadas 2 horas nos indica si la primera es `Menor`, `Igual` o `Mayor` que la de la segunda devolviendo un `TCompara` (módulo `MCadena`).

- `MLista` (`MLista.h, MLista.cpp`): Módulo de lista ordenada por horas de Citas.
  - Define los tipos `TLista, TNodo y TErrorLista`.
  - Define la función `CrearLista`: Crea una lista vacía.
  - Define la función `ListaVacia`: Nos dice si una lista está vacía.
  - Define la función `ListaLlena`: Nos dice si una lista está llena.
  - Define la función `InsertarLista`: Inserta una cita en la lista ordenada por hora.
  - Define la función `MostrarLista`: Muestra por pantalla las citas de la lista.
  - Define la función `EliminarLista`: Elimina de la lista una cita dada su hora.
  - Define la función `SacarPrimeroLista`: Extrae de la lista la primera cita almacenada.
  - Define la Función `DestruirLista`: Destruye una lista liberando toda la memoria.
- `MAgenda (MAgenda.h, MAgenda.cpp)`: Módulo de manejo de la `Agenda`.
  - Define las constante `necesarias y` los tipos `TErrorAgenda y TAgenda`.
  - Define la Función `CrearAgenda`: Crea un `agenda` vacía.
  - Define la Función `AgendaVacia`: Nos dice si una `agenda` está vacía
  - Define la Función `AgendaLlena`: Nos dice si una agenda está llena.
  - Define la Función `InsertarAgenda`: Inserta una cita en la agenda.
  - Define la Función `MostrarAgenda`: Muestra todos las citas de la agenda.
  - Define la Función `EliminarAgendaHora`: Elimina (si existe) una cita de la agenda dados su dia y hora.
  - Define la Función `EliminarAgendaTipoCita`: Elimina (si existe) todas las citas que sean de un tipo de cita que se pasa como parámetro y en el día indicado.
  - Define la Función `BuscarAgenda`: Retorna (si existe) una cita dados su día y hora..
  - Define la Función `DestruirAgenda`: Destruye una agenda liberando toda la memoria.
  - Define la Función `CargarAgendaFichero`: Inserta todos las citas que haya en un fichero cuyo nombre se pasa como parámetro y con el formato indicado anteriormente.
  - Define la Función `SalvarAgendaFichero`: Almacena todos las citas en un fichero cuyo nombre se pasa como parámetro en el formato indicado anteriormente.

NOTAS

1. Es obligatorio trabajar en el directorio **`C:\LPSJUN08`**.
2. **Mínimo Obligatorio para Aprobar para quien tenga superado el trabajo en clase** (3 puntos): correcta la definición de tipos, la modularización y funcionar **CORRECTAMENTE** las Opciones A, B, C, D y X del menú.
3. **Mínimo Adicional Obligatorio para Aprobar para quien NO tenga superado el trabajo en clase** (2 puntos): funcionar **CORRECTAMENTE** las Opciones E y F del menú.
4. **El uso de detalles de implementación de la lista fuera del módulo de implementación** de la misma (`MLista.cpp`) será **CAUSA de SUSPENSO**, es decir, no se podrá hacer cosas tipo: `l= NULL`, `ptr=ptr->sig`, etc. Ni en el programa principal ni en el módulo de agenda. Tampoco podrá hacerse uso de los detalles de implementación de la agenda en el programa principal, es decir, no se podrá hacer nada del tipo `agenda[i]`, etc.
5. **Añadir procedimientos o funciones a la definición de un módulo** será **CAUSA de SUSPENSO**
6. Todo **PROGRAMA QUE NO COMPILE** o tenga **efectos laterales** se considerará **SUSPENSO**.
7. No se distinguirá entre mayúsculas y minúsculas.
8. Se recomienda y valora el tratamiento de errores y la buena descomposición del programa principal en procedimientos y funciones, así como el uso de procedimientos y funciones auxiliares dentro de la implementación de los módulos cuando estas sean necesarias.