UNIVERSIDAD DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación

LABORATORIO DE PROGRAMACIÓN

Examen Ordinario de Junio

1º Sistemas Curso 2009/2010

24/06/2010 a las 16:30

APELLIDOS_____ NOMBRE_____

DNI _____ ORDENADOR _____ GRUPO (A/B/C)_____

La empresa estatal de *Formación para el Empleo* desea desarrollar una aplicación para gestionar las solicitudes de los cursos que imparten. De cada curso se desea almacenar el código del curso (entero), el colectivo al que va dirigido ese cuso (un enumerado con valores `Desempleados`, `Jovenes`, `Mayores`, `Todos`) y una cola donde se almacenan los solicitantes de dicho curso. Los solicitantes se identifican por su NIF (cadena de caracteres). Como se muestra en la figura adjunta, para almacenar toda la información usaremos una lista enlazada, donde cada nodo de la lista enlazada guarda la información de un curso diferente. Se pide implementar un programa llamado `formacion.cpp` que presente el siguiente menú:

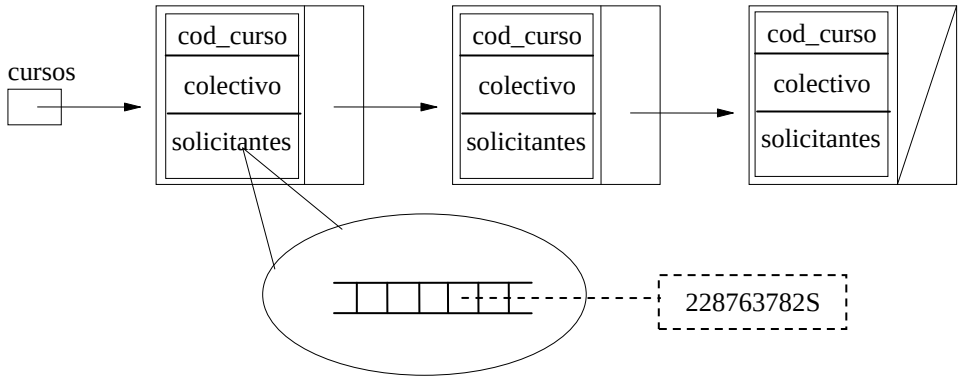

Figura. Esquema de la Estructura

```
Nombre: (Apellidos, Nombre)          Curso: 1º
Especialidad: Sistemas               Grupo: A/B/C
Puesto: número de ordenador          Fecha: 24/06/2010

Formacion para el Empleo
========================
A. Insertar Curso.
B. Insertar Solicitante.
C. Cargar Cursos desde Fichero de Texto.
D. Mostrar Todos los Cursos.
E. Salvar Todos los Cursos a Fichero de Texto.
F. Eliminar Curso.
G. Anular Inscripción a Cursos.
H. Transferir un Curso.
X. Salir del Programa

Introduzca Opción:
```

Ejemplo Fichero: cursos.txt

```
4
Desempleados#3
40111111A
40222222B
40333333C
2
Mayores#1
20111111A
6
Desempleados#0
5
Todos#2
50111111A
50222222B
```

De las opciones del menú anterior, las OPCIONES BÁSICAS que deben realizar TODOS LOS ALUMNOS son:

A. Insertar Curso. Se pedirá desde teclado el código del curso (que deberá leerse como un entero) y el colectivo al que va dirigido el curso y se insertará en la estructura de datos *ordenado por el código del curso*. Si el colectivo introducido no es correcto se pedirá al usuario que vuelva a introducirlo hasta que se introduzca uno correcto. Si ya existiera en la estructura un curso con el mismo código que el introducido o no hubiera memoria se informaría del error. Al dar de alta un nuevo curso la cola de solicitantes estará vacía.

B. Insertar Solicitante. Se pedirá desde teclado el código de un curso (que deberá haberse insertado anteriormente), y si existe se mostrarán los datos del curso por teclado. Si no existiera en la estructura de datos un curso con el código introducido se informará del error. Si el curso existe, se pedirán los datos del solicitante (su NIF como cadena de caracteres) y se insertará en la cola de solicitantes del curso correspondiente. Si no hubiera memoria en la cola para insertar el nuevo solicitante también se informaría del error. No es necesario controlar si el solicitante ya existe en el curso.

C. Cargar Cursos desde Fichero de Texto. Se pedirá al usuario el nombre de un fichero de texto que tendrá el formato que se ha mostrado anteriormente (junto al menú) y se cargará toda la información en memoria. Se deberá informar de cualquier error relativo al manejo de ficheros o de memoria. Se descartarán los datos que hubiera previamente en memoria. Se supone que el formato del fichero es correcto.

Formato del Fichero:

```
<COD_CURSO><ENTER>
<COLECTIVO>#<NUMERO SOLICITANTES><ENTER>
<NIF>
...
<NIF>
```

D. Mostrar Todos los Cursos. Se mostrará por pantalla un listado de todos los cursos, utilizando el formato que el alumno considere más apropiado.

X. Salir del Programa. Se pedirá confirmación de salida.

Los ALUMNOS QUE NO TENGAN SUPERADO EL TRABAJO EN CLASE (tienen una nota acumulada < 2), además tendrán que implementar las siguientes opciones del menú principal:

E. Salvar Todos los Cursos a Fichero de Texto. Se pedirá el nombre de un fichero de texto y se salvará toda la información con el mismo formato del apartado C. Si el fichero no existe se creará y si ya existiera se borrará su contenido. El contenido de la lista en memoria principal no se modifica al guardar los datos en el fichero.

F. Eliminar Curso. Se pedirá desde teclado el código de un curso (que deberá haberse insertado anteriormente) y, si existe, se mostrarán los datos del curso por teclado. Si no existiera en la estructura de datos un curso con el código introducido se informará del error. Si el curso existe, se pedirá confirmación al usuario y si el usuario confirma que sí se borre, se borrará de la estructura. Los solicitantes que tuviera ese curso se descartarán.

Una vez completadas las operaciones básicas pedidas, PARA SACAR MÁS NOTA habrá que implementar el resto de opciones del menú:

G. Anular Inscripción a Cursos. Se pedirá al usuario el nif de un solicitante y se borrará de la estructura todas las solicitudes que ese solicitante haya realizado, sacando su solicitud de las colas correspondientes. Se mostrará por pantalla el código de los cursos en los que el usuario se encontraba inscrito. Hay que tener en cuenta que un solicitante puede haber solicitado tantos cursos como desee y que podría pertenecer a varios colectivos simultánemanete. Si el solicitante no está inscrito a ningún curso no se hará nada.

H. Transferir un Curso. Se pedirá al usuario el código del curso y se borrará de la estructura. Antes de borrarlo, todos los solicitantes de ese curso se encolarán en el siguiente curso existente que esté dirigido al mismo colectivo que el curso borrado. Si no existe en la estructura ningún otro curso del mismo colectivo el curso no podrá darse de baja y se informará del error. Si el curso al que se transfieren los datos se llena (por no haber memoria insuficiente) se mostrarán por pantalla los datos de los solicitantes que queden por transmitir, informando de que esos participantes no ha sido posible transferirlos y serán descartados.

Módulos a Implementar:(<u>LA DEFINICIÓN DE LOS MÓDULOS NO SE PODRÁ MODIFICAR</u>)

- `MError`: Manejo de errores.
  - Define el tipo `TError` como un enumerado con los posibles errores generados.
  - `MostrarError`: Muestra el error por pantalla.
- `MCurso`: Manejo del registro `TCurso` y del enumerado `TColectivo`.
  - Define las constantes `SP` (espacio), `SEP` (#); y `ENTER`.
  - Define el tipo `TCurso como` un registro con los campos `codigo` (entero), `colectivo` (enumerado) y `solicitantes` (cola).
  - Define el tipo `TColectivo` (enumerado de valores `Desempleados, Jovenes, Mayores, Todos`)
  - `LeerCurso(curso)`: Lee un curso (código y colectivo) desde teclado. La lista de solicitantes estará vacía.
  - `EscribirCurso(curso)`: Escribe un curso por pantalla (código, colectivo e información de los solicitantes).
  - `InsertarSolicitanteCurso(curso,solicitante,error)`: Inserta los datos de un solicitante (su NIF) en la cola de solicitantes del curso que se pasa como parámetro. Informará del error si el curso no existe o no hay memoria para insertar al solicitante.
  - `CargarCursoFicheroTexto(curso,fich,error)`: Lee un curso (código, colectivo y solicitantes) desde un fichero de texto cuyo descriptor se pasa como parámetro. Informará del error en caso de que haya algún problema con los ficheros o no haya memoria.
  - `DestruirCurso(curso)`: Libera los recursos de memoria asignados previamente a un curso.
  - `SalvarCursoFicheroTexto(curso,fich,error)`: Escribe un curso (código, colectivo y solicitantes) en un fichero de texto cuyo descriptor se pasa como parámetro. Informará del error en caso de que haya algún problema con el fichero. **(Sólo para quien tenga que hacer las opciones E-F)**
  - `AnularInscripcionCurso(curso, solicitante, error)`: Si el solicitante se encuentra inscrito en el curso anula su inscripción sacándolo de la cola de solicitantes. Informará del error en caso de que el solicitante no se encuentre inscrito al curso **(Sólo para quien tenga que hacer las opciones G-H)**
  - `SacarSolicitanteCurso(curso, solicitante, error)`: Dado un curso, devuelve el solicitante que se encuentre en primer lugar en la cola de solicitantes, sacándolo de la misma. Dará un error si no hay más

solicitantes en el curso y por tanto no se ha podido extraer ninguno **(Sólo para quien tenga que hacer las opciones G-H)**

- `TieneSolicitantes(curso)`: Devuelve un valor booleano indicando si el curso que se pasa como parámetro tiene o no algún solicitante **(Sólo para quien tenga que hacer las opciones G-H)**

- `MListaCursos`: Manejo de una lista de cursos.
  - Define los tipos `TListaCursos` y `TNodo` para representar una lista enlazada de cursos.
  - `CrearLista(lista)`: Crea una lista de cursos vacía.
  - `ListaVacia(lista)`: Nos dice si una lista de cursos está vacía.
  - `ListaLlena(lista)`: Nos dice si una lista de cursos está llena (no hay más memoria).
  - `InsertarCursoLista(lista,curso,error)`: Inserta un curso en la lista ordenado por el código del curso. No podrá haber cursos repetidos y deberá informarse de cualquier error producido.
  - `InsertarSolicitanteLista(lista,codigo,solicitante,error)`: Inserta un nuevo solicitante en un curso existente previamente en la lista y cuyo código se pasa como parámetro. En caso de que el curso no exista se informará del error.
  - `BuscarCursoLista(lista,código,curso,error)`: Busca en la lista de cursos el curso cuyo código se pasa como parámetro, devolviendo en el parámetro curso toda la información sobre el mismo. Si el curso no existe se informará del error.
  - `MostrarLista(lista)`: Se muestra toda la información almacenada en la lista de cursos.
  - `DestruirLista(lista)`: Destruye una lista de cursos.
  - `CargarListaFicheroTexto(lista,nombre,error)`: Carga los datos del fichero de texto cuyo nombre se pasa como parámetro en la lista de cursos. Informa de cualquier error que se produzca con el fichero.
  - `SalvarListaFicheroTexto(lista,nombre,error)`: Guarda los datos de la lista enlazada en el fichero de texto cuyo nombre se pasa como parámetro. Informa de cualquier error que se produzca con el fichero **(Sólo para quien tenga que hacer las opciones E-F)**.
  - `BorrarCursoLista(lista,código,error`: Borra de la lista un curso cuyo código se pasa como parámetro. En caso de que el curso no exista se informará del error **(Sólo para quien tenga que hacer las opciones E-F)**.
  - `AnularInscripcion(lista,solicitante)`: Anula la inscripción de un solicitante a todos los cursos en los que esté inscrito, sacándolo de las colas correspondientes. Se mostrará por pantalla el código de todos los cursos en los que el solicitante estaba inscrito. Si no está inscrito en ningún curso no se hace nada **(Sólo para quien tenga que hacer las opciones G-H).**
  - `TransferirCursoLista(lista, codigo, error)`: Transfiere todos los solicitantes del curso cuyo código se indica como parámetro a otro curso de la lista que esté dirigido al mismo colectivo, borrando después el curso original. Si no existe el curso original se informará del error. Si no existe ningún otro curso con el mismo colectivo al que hacer la transferencia se informará del error y no se borrará el curso original. Si la cola de solicitantes del curso destino se llena se descartarán los solicitantes que no se hayan podido transferir, mostrando su NIF por pantalla. En este último caso el curso original sí se borrará **(Sólo para quien tenga que hacer las opciones G-H).**
- `MCola`: Manejo de una cola de solicitantes (implementada de forma eficiente), identificados por su NIF (cadena de caracteres).
  - Define el tipo `TCola` para implementar una cola de forma eficiente.
  - `CrearCola(cola)`: Crea una cola.
  - `DestruirCola(cola)`: Destruye una cola.
  - `MeterCola(cola,valor,error)`: Inserta un solicitante en la cola. Informa del error si no hay memoria.
  - `SacarCola(cola,valor,error)`: Saca un solicitante de la cola. Informa del error si la cola estaba vacía.
  - `ColaVacia(cola)`: Nos dice si la cola está vacía.
  - `ColaLlena(cola)`: Nos dice si la cola está llena (no hay más memoria).
  - `ImprimirCola(cola)`: Muestra por pantalla los datos almacenados en la cola.
  - `CargarColaFicheroTexto(cola,fich,error)`: Carga desde un fichero de texto, cuyo descriptor se pasa como parámetro, los datos de la cola. Informa del error si hay algún problema con el fichero.
  - `SalvarColaFicheroTexto(cola,fich,error)`: Guarda en un fichero de texto, cuyo descriptor se pasa como parámetro, los datos de la cola. Informa del error si hay algún problema con el fichero **(Sólo para quien tenga que hacer las opciones E-F).**

NOTAS

1. **Es obligatorio trabajar en una carpeta con nombre LPSIS10.**
2. Todo **PROGRAMA QUE NO COMPILE** o tenga **efectos laterales** se considerará **SUSPENSO**.
3. **Añadir procedimientos o funciones a la definición de un módulo y el uso de detalles de implementación de la lista o de la cola fuera del módulo de implementación** de las mismas (`MlistaCursos.cpp, MCola.cpp`) será **CAUSA de SUSPENSO**, es decir, en el programa principal o en módulos distintos a los mencionados anteriormente no se podrá hacer cosas tipo: `l= NULL`, `ptr=ptr->sig`, etc.
4. 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.