UNIVERSIDAD DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación

LABORATORIO DE PROGRAMACIÓN

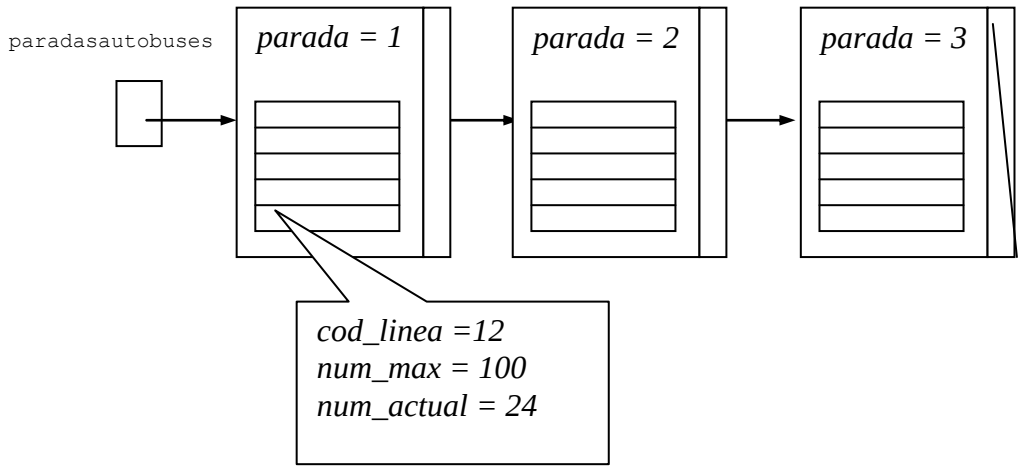
E.T.S.I. Informática- 1º SISTEMAS-GESTIÓN. 15-DICIEMBRE-2008

APELLIDOS_____NOMBRE_____

DNI _____ORDENADOR_____SISTEMAS__GESTIÓN__GRUPO (A/B/C)_____

La empresa malagueña de transporte público desea implementar un programa para gestionar el sistema de autobuses de la ciudad. Para ello, se ha diseñado el programa adjunto `autobuses.cpp`, que ofrece un menú con las opciones básicas, y que deberá completarse con las operaciones necesarias según se explica más adelante.

La estructura de datos que almacena la información (ver figura adjunta) vendrá dada por una **lista enlazada <u>ordenada</u>** por el número de parada (`TListaParadas`), donde cada nodo (`TParada`) tiene dos campos: (1) campo `parada`[1], que almacena el número de la parada, y (2) una estructura de datos (`TAutobuses`) apropiada para almacenar un máximo de 5 autobuses, que esperan en la parada. Los autobuses se almacenen según el orden de llegada a la parada. De cada autobús (`TAutobus`), la información que se quiere almacenar es la siguiente: (1) campo `cod_linea`[1], que almacena el código de la línea a la que pertenece el autobús. En una misma parada puede haber más de un autobús de la misma línea; (2) campo `num_max`[1], que almacena el número máximo de pasajeros que pueden subir a ese autobús, y (3) campo `num_actual`[1], que almacena el número de pasajeros que se encuentran en un momento dado en el autobús.



Se pide desarrollar los módulos `MListaParadas` y `MAutobuses`, necesarios para completar el módulo de programa adjunto `autobuses.cpp`, teniendo en cuenta que:

- El módulo de programa suministrado `autobuses.cpp` tendrá que modificarse apropiadamente para implementar las distintas opciones del menú.
- El módulo `MAutobuses` define el tipo `TAutobus`, el tipo `TAutobuses` y las siguientes operaciones:
  - `LeerAutobusesFich(autobuses,fich)`. Inicializa la estructura `autobuses` con la información de los autobuses leídos del fichero `fich`. Nótese que `fich` no es el nombre de un fichero, sino el manejador del fichero del que se lee. Se asume que el formato del fichero proporcionado es correcto. El registro en el fichero tendrá el siguiente formato:

```
num_autobuses<ENTER>
cod_linea<ESPACIO>num_max<ESPACIO>num_actual<ENTER>
...
```

[1] Número natural

Donde se indica, primero, el número de autobuses (como máximo podrá ser 5), y luego, una línea por cada autobús. Por ejemplo, si hay 3 autobuses el fichero sería:

```
3
12 100 24
24 123 12
12 100 0
```

- `SubirPasajeros(autobuses, cod_linea, num, numReal, ok)`. `num` pasajeros intentan subir a algún autobús de la línea `cod_linea`. Si la línea no existe, en `ok` se devuelve `false`. Si existe, en `ok` se devuelve `true` y en `numReal` se devuelve el número de pasajeros que realmente han podido subir a algun autobús, en caso de que no haya hueco suficiente para que suban todos. Es decir, que si hay más de un autobús de la línea `cod_linea`, y en el primero de ellos no entran todos los pasajeros, para los RESTANTES PASAJEROS se prueba con los siguientes autobuses hasta que todos hayan subido o hasta que no queden más autobuses de la línea deseada.
- `BajarPasajeros(autobuses, cod_linea, num, ok)`. `num` pasajeros se bajan del autobús de la línea `cod_linea`. Si la línea no existe, en `ok` se devuelve `false`. Si existe, en `ok` se devuelve `true`. Se supone que se bajan del primer autobús que haya con ese código de línea. Podría ser que `num` fuera mayor que el número real de pasajeros subidos al autobús. En dicho caso se bajaran sólo el número real de pasajeros.
- `MostrarAutobus(autobuses)`. Muestra en pantalla una línea por cada autobús.

- El módulo `MListaParadas` define el tipo `TParada`, que representa una parada, el tipo `TError`, un enumerado que permitirá informar sobre los errores que se producen en el módulo, y el tipo `TListaParadas`, que representa a una lista de paradas. Definirá además las operaciones básicas para trabajar con una lista de paradas, que serán:
  - `CrearLista(L)`. Esta operación crea en `L` una lista de paradas vacía.
  - `ListaVacia(L)`. Esta operación devuelve `true` si la lista de paradas `L` está vacía, y `false` en caso contrario.
  - `LeerParadasFichero(L,nom_fich,error)`. Esta operación crea en `L` una lista de paradas **(ordenada por el número de parada)** con la información almacenada en el fichero de texto `nom_fich`. Si la lista de entrada contenía alguna información, ésta se destruirá antes de comenzar la lectura desde el fichero. Devolverá un error en caso de que no pueda abrirse el fichero.

    El fichero estará formado por el número de parada, un espacio, y a continuación la información sobre los autobuses que hay en esa parada, según el formato descrito anteriormente en la operación `LeerParadasFich`:

    `num_parada`<ESPACIO>`información_autobuses`<ENTER>

    Por ejemplo, si hay dos paradas, una con código 1, y 3 autobuses, y otra con código 2, y 2 autobuses, su información se almacenaría de la siguiente forma:

    ```
    1 3
    12 100 24
    24 123 12
    12 100 0
    2 2
    12 100 3
    13 100 8
    ```

    Junto al enunciado se proporciona el fichero *paradas.txt* con el formato descrito.
  - `MostrarParadas(L)`. Muestra por pantalla la información de todas las paradas, incluyendo la información de los autobuses contenidos en L.

- `BorrarParada(L,num_parada,error)`. Elimina de la lista de paradas `L` aquella cuyo código es `num_parada`. Devolverá un error si la parada no existe. Los autobuses que se encuentren en esa parada desaparecen.
- `SubirBajarPasajeros(L,num_parada,cod_linea,numSuben,numRealS, numBajan, error)`. Dado un número de parada (`num_parada`), primero se comprueba si la parada existe, devolviendo un error en caso en que no exista. También se devolverá un error si no hay ningún autobús de esa línea en la parada. Si existe la parada, y hay algún autobús de la línea indicada, se bajan `numBajan` pasajeros del primer autobús de la línea `cod_línea`. A continuación, se suben `numSuben` pasajeros en algún autobus de la línea indicada, intentándolo primero en el autobús del que se han bajado anteriormente los pasajeros. Si no pueden subirse `numSuben` pasajeros a un determinado autobús, se comprobará si existe otro autobús de la misma línea en la misma parada y los pasajeros restantes intentarán subirse a dicho autobús. Se repetirá el proceso hasta que no haya ningún autobús más en la parada de la línea elegida. Es posible que no todos los pasajeros puedan subir. En la variable `numRealS` se indica el número de pasajeros que realmente han podido subir a algún autobús de la línea indicada.
- `DestruirParadas(L)`. Libera todos los recursos ocupados por la lista de paradas `L`.

LAS OPERACIONES DESCRITAS ANTERIORMENTE SON EL CONJUNTO MÍNIMO NECESARIO PARA IMPLEMENTAR LAS OPCIONES BÁSICAS DEL MENÚ PRINCIPAL:

Opciones BÁSICAS del menú principal para APROBAR:

A.- Leer paradas de un fichero de texto (cuyo nombre se lee por teclado)
B.- Mostrar paradas
C.- Borrar parada (cuyo número de parada se lee por teclado)
D.- Subir y bajar pasajeros (cuyo número de parada, código de autobús, número de pasajeros que se bajan y número de pasajeros que se suben se leen por teclado)
X.- Salir del Programa

Una vez completadas las operaciones básicas pedidas, PARA SUBIR NOTA proceda a implementar las siguientes operaciones adicionales:

- En el módulo `MListaParadas`:
  - `EscribirParadasFichero(l,nom_fich,err)`. Esta operación almacenada en el fichero de texto `nom_fich` la información almacenada en la lista `l`. Devolverá un error en caso de que no pueda abrirse el fichero. El formato del fichero será el mismo descrito en la opción `LeerParadasFichero(l,nom_fich,err)`.
  - `FueraDeServicio(l,num_parada,err)`. Esta operación pone fuera de servicio la parada indicada en `num_parada`. Si la parada no existe se devuelve un error. Si la parada existe se intentan mover los autobuses que estén en esa parada a la siguiente parada. Si en esta tampoco hay hueco para todos, se prueba con la siguiente, y así sucesivamente. El incremento de parada se hace de forma circular. Esto significa que la parada siguiente a la última de la lista enlazada será la primera parada de la lista. Si todos los autobuses pueden moverse a otras paradas la parada se deja fuera de servicio borrándola de la lista. Si no pueden moverse todos los autobuses se dará un mensaje de error y la parada no se borrará.

- En el módulo `MAutobuses`:
  - `AniadirAutobus(autobuses, autobus, ok)`. Añade un nuevo autobús a la estructura de autobuses. Si no se puede añadir porque hubiera ya 5 autobuses almacenados se devuelve `false` en `ok`. En otro caso se devuelve `true` en `ok`.
  - `ExtraerAutobus(autobuses, autobus, ok)`. Elimina de `autobuses` un autobús (elimina el primero almacenado en la estructura) si hay alguno y lo guarda en `autobus`, devolviendo `true` en `ok`. Si la estructura estuviera vacía devuelve `false` en `ok`.
  - `EscribirAutobusesFichero(autobuses,fich)`. Guarda en el fichero `fich` la información de los autobuses almacenada en la estructura `autobuses`. Nótese que `fich`

no es el nombre de un fichero, sino el manejador del fichero en el que se escribe. El formato del fichero será el mismo que el descrito anteriormente para la operación `LeerAutobusesFichero(autobuses,fich)`.

Estas operaciones son necesarias para extender el menú principal con las siguientes opciones:

Opciones ADICIONALES del menú principal para subir nota:

F.- Poner parada fuera de servicio (cuyo número de parada se lee por teclado)

G.- Guardar paradas en un fichero de texto (cuyo nombre se lee por teclado)

NOTAS

1. Es obligatorio trabajar en el directorio **`C:\LPDIC08`**.
2. Todo PROGRAMA QUE NO COMPILE correctamente se considerará SUSPENSO.
3. EL USO DE DETALLES DE IMPLEMENTACIÓN DE LA LISTA FUERA DE SU MÓDULO DE IMPLEMENTACIÓN SERÁ CAUSA DE SUSPENSO. Es decir, no se podrá hacer nada del tipo ptr=ptr->sig, l=NULL, etc., fuera de la parte de implementación del módulo MListaParadas.
4. Se recuerda que hay que liberar toda la memoria reservada antes de finalizar la ejecución del programa.