Curso 2007/2008 Estructura de Datos y de la Información
I. Informática, I. T. Informática de Gestión y de Sistemas

PRÁCTICA 1

1 El problema

Se trata de automatizar a través de un portal web la gestión de reservas de butacas en auditorios, palacios de congresos y teatros, dependientes del Ayuntamiento de la ciudad. Para ello necesitamos diseñar una nueva estructura de datos que permita almacenar toda la información necesaria. Por un lado, para poder efectuar cualquier reserva, se necesita mantener un mapa actualizado del local donde cada butaca se identifique por un número dentro de la fila, y un estado—libre, ocupada o reservada. Por otro lado, para gestionar las reservas y los pagos es necesario almacenar el nombre y apellidos de los espectadores que han hecho la reserva. A cada espectador sólo se le permite realizar una reserva. Para simplificar el problema, en esta práctica trabajaremos sólo con UN TEATRO y supondremos que la platea se numera correlativamente, de esta forma tan sólo es necesario modelizar UNA SOLA FILA.

2 TAD Lista

Para mantener un mapa de la ocupación de las butacas de un local se utilizará un TAD Lista, en el que se registran las butacas, numeradas correlativamente de 1 a n, su estado—libre, o reservada—y en su caso, el espectador que hizo la reserva o el pago. Se realizarán dos implementaciones del TAD Lista: una estática con arrays (unit ListaEstatica.pas) y otra dinámica, simplemente enlazada, con punteros (unit ListaDinamica.pas).

2.1 Tipos de datos incluidos en el TAD Lista

- **tLista** Representa a una lista ordenada por número de butaca.
- **tNumeroButaca** Número de la butaca.
- **tEstadoButaca** Estado de la butaca—libre o reservada (tipo enumerado).
- **tNombreEspectador** Apellidos y nombre del espectador.
- **tInfo** Dato de un elemento de la lista, compuesto por los campos *NumeroButaca*, *EstadoButaca* y *NombreEspectador* que contienen el número de la butaca, su estado y caso de reserva el espectador que la efectúa, respectivamente.
- **tPos** Posición de un elemento de la lista.
- **NULO** Constante utilizada para representar posiciones nulas.

2.2 Operaciones incluidas en el TAD LISTA

Una precondición común para todas las operaciones (salvo ListaVacia) es que la lista debe estar previamente inicializada.

- `ListaVacia (tLista)` → `tLista`
  Crea una lista vacía.
- `esListaVacia (tLista)` → `Boolean`
  Determina si la lista está vacía
- `Primero (tLista)` → `tPos`
  Devuelve la posición del primer elemento de la lista (o NULO si la lista está vacía).
- `Ultimo (tLista)` → `tPos`
  Devuelve la posición del último elemento de la lista (o NULO si la lista está vacía).
- `Siguiente (tLista, tPos)` → `tPos`
  Devuelve la posición en la lista del siguiente elemento a la posición indicada (o NULO si la posición no tiene siguiente).
  PreCD: La posición tiene que ser válida.
- `Anterior (tLista, tPos)` → `tPos`
  Devuelve la posición en la lista del anterior elemento a la posición indicada (o NULO si la posición no tiene anterior).
  PreCD: La posición tiene que ser válida.
- `Insertar (tLista, tNumeroButaca)` → `tLista, Boolean`
  Inserta una nueva butaca con número tNumeroButaca en la lista. Devuelve un valor falso si no hay memoria suficiente para realizar la operación.
  PreCD: No existe una butaca con ese número
  PosCD: Se asigna a la butaca el estado libre y el nombre del espectador se inicializa a “”
- `Borrar (tLista, tPos)` → `tLista`
  Borra de la lista la butaca que está en la posición indicada.
  PreCD: La posición tiene que ser válida.
- `ObtenerDato (tLista, tPos)`→ `tNumeroButaca, tEstadoButaca, tNombreEspectador`
  Devuelve el dato situado en la posición indicada de la lista.
  PreCD: La posición tiene que ser válida.
- `ActualizaDato(tLista,tPos,tNombreEspectador,tEstadoButaca)`→ `tLista`
  Actualiza el dato situado en la posición indicada con nombre tNombreEspectador y estado tEstadoButaca.
  PreCD: La posición tiene que ser válida.
- `Buscar (tLista, tNombreEspectador)` → `tPos`
  Devuelve la posición del elemento con nombre tNombreEspectador (o NULO si el elemento no existe).

3 Descripción de la tarea

El alumno diseñará un único programa principal (`principal.pas`) que lea, a través de un fichero, las distintas operaciones de los usuarios—reserva, cancelación o pago—las procesará y visualizará por pantalla los resultados. El objetivo de la práctica es practicar la independencia de la implementación en los tipos abstractos de datos (TADs).

El programa funcionará en modo batch (sin interactividad con el usuario) y de manera transparente con cada una de las dos implementaciones del TAD Lista (unit ListaEstatica y unit ListaDinamica), ejecutando las siguientes tareas:

1. Lectura de un fichero de texto `platea.txt` con el número de butacas del teatro (máximo 100), y creación de la lista correspondiente.

2. Lectura en un fichero de texto `operaciones.txt` de la siguiente operación a procesar. El fichero contiene las distintas operaciones permitidas—reserva, cancelación o pago—según el orden en que los espectadores las van efectuando. En la operación consta el código—R (reserva), C (cancelación), P (pago)—apellidos y nombre de la persona y el número de butacas que solicita si se trata de una reserva[1].

3. Atender cada operación tal y como sigue:

   a. Leer una operación del fichero. Se imprimirá un mensaje del tipo:

   ```
   **********************************************
   Nueva operación XXXXXXX: Nombre – NumeroButacas
   **********************************************
   ```

   donde XXXXXXX es el tipo de operación—reserva, cancelación o pago.

   b. Si la operación es `Reserva` buscar en la lista el primer grupo de butacas libres **correlativas** con un número suficiente para atender la solicitud. Si no existe imprimir un mensaje del tipo:

   ```
   Operación NO atendida. No hay butacas suficientes
   ```

   Si existe, actualizar el estado de las butacas a *reservada* así como el nombre de quien realiza la reserva e imprimir un mensaje del tipo:

   ```
   Reserva de XXXXXXXXXXX
        Butaca XX
        Butaca XX
   ```

   donde XXXXXXX es el nombre del espectador, y XX es el número de la butaca.

   c. Si la operación es `Pago` buscar el nombre de la reserva en la lista de butacas, recuperar los datos de la reserva (nombre y posición de las butacas), borrar las butacas de la lista e imprimir un mensaje del tipo:

   ```
   Pago de XXXXXXXXXXX
        Butaca XX
        Butaca XX
   ```

   donde XXXXXXX es el nombre del espectador, y XX es el número de la butaca.

   d. Si la operación es `Cancelacion` buscar el nombre de la reserva en la lista de butacas, recuperar los datos de la reserva (nombre y posición de las butacas), actualizar el estado de las butacas a *libre* e imprimir un mensaje del tipo:

   ```
   Cancelacion de XXXXXXXXXXX
        Butaca XX
        Butaca XX
   ```

[1] Las operaciones de cancelación o pago de una reserva implican que ésta ha sido efectuada previamente.

donde XXXXXXX es el nombre del espectador, y XX es el número de la butaca.

4. Visualizar en pantalla la platea (en nuestro caso una única fila). El formato será el siguiente:

```
Butaca XX        Estado
…
Butaca XX        Estado
```

   donde XX es el número de la butaca.

5. Volver al paso 2.

4 Lectura de los ficheros

Para facilitar el desarrollo de la práctica se proporciona un fichero "`lectura.pas`". Este fichero contiene un ejemplo de lectura y manejo de los distintos ficheros mencionados. También se proporciona un ejemplo de prueba de los ficheros `platea.txt` y `operaciones.txt`.

5 Normas de realización

- Estructura que deberá de tener cada programa/unit desarrollado
  - Encabezamiento del programa/unit. Constará la siguiente información entre comentarios

```
TÍTULO: Prácticas de EDI
SUBTÍTULO: Practica 1
AUTOR 1: _____ LOGIN 1: _____
AUTOR 2: _____ LOGIN 2: _____
GRUPO: E.I / E.T.I.X. / E.T.I.S
FECHA: __/__/____
```

  - Cuerpo del programa/unit
    - El código irá comentado. Los comentarios han de ser concisos pero explicativos.
    - Después de la cabecera de cada procedimiento o función se incluirá lo siguiente: objetivo, entradas, salidas, precondiciones (condiciones que han de cumplir las entradas para el correcto funcionamiento de la subrutina) y poscondiciones (otras consecuencias de la ejecución de la subrutina que no quedan reflejadas en la descripción del objetivo o de las salidas).
- Criterios de valoración de la práctica:
  - *Eficacia*: que se cumplan las especificaciones.
  - *Control de Errores*: Control intensivo de todos los errores de ejecución que sea posible detectar.
  - *Claridad*: que el programa se pueda entender con facilidad, que contenga comentarios oportunos, indentación adecuada, nombres de variables significativos, etc.
  - *Modular*: que esté construido con módulos intercambiables y reutilizables.

6 Normas de entrega

- Las prácticas son OBLIGATORIAS y serán realizadas en grupos de DOS PERSONAS.
- La entrega de TODAS las prácticas en las FECHAS indicadas es requisito IMPRESCINDIBLE para aprobar la asignatura en la convocatoria ordinaria de JUNIO. Para las convocatorias de SEPTIEMBRE y DICIEMBRE las prácticas serán las mismas pero se fijarán otras fechas de entrega.
- Las prácticas entregadas tendrán que ser compilables en FreePascal en los ordenadores del laboratorio de docencia asignado.
- **Fecha límite de entrega: 30 de Abril de 2008.**
- Forma de entrega: Las prácticas quedarán depositadas en la red según el procedimiento del CECAFI. No se admitirán discos ni papel. El proceso para depositarlas será el siguiente:
  1. Conectarse a las máquinas **xurxo** o **limia**.
  2. Situarse en el directorio /PRACTICAS/XXXX/EDI/P1 donde XXXX es EI, ETIX o ETIS
  3. Situarse en el directorio que coincida con el login del usuario y copiar allí la práctica (sólo los ficheros fuentes .pas). IMPORTANTE: El nombre del fichero fuente que contenga el programa principal deberá ser `principal.pas`.
  4. Pasada la fecha de entrega no se permitirá el acceso a estos directorios.