Estructuras de Datos y Algoritmos
Grado en Ingeniería Informática
Universidad de Zaragoza – curso 2024-2025

Sesión de problemas 2
Más especificación e implementación de TAD lineales

Objetivos

- Trabajar con la especificación no formal.
- Implementar (en pseudocódigo) TAD lineales.

1. Especificación del TAD `guardería`

Se trata de **especificar un TAD `guardería`**, que pueda utilizarse para desarrollar aplicaciones en contextos en los que un colectivo de adultos conocidos y registrados en un sistema guardería, puedan servir como adultos de contacto responsables (guardianes) de uno o varios niños pertenecientes a un colectivo de niños registrados en el sistema `guardería`. Un ejemplo de posible uso sería una aplicación para la gestión y monitorización de los adultos y niños que entran y pueden despistarse en el interior de un recinto como un centro comercial, un parque de atracciones, o en situaciones asimilables, y que por ejemplo permita resolver fácilmente situaciones de niños extraviados, prevención de secuestros, etc., si se les ha colocado una pulsera identificativa y registrado en la aplicación al llegar al lugar.

Para cada niño, el sistema deberá mantener información de: su código de identificación, su nombre y apellidos, edad, sexo, domicilio, y siempre la información de como mínimo un adulto y como máximo seis adultos distintos, previamente registrados en el sistema, y que servirán como guardianes o adultos de contacto válidos para ese niño. Para cada adulto, el sistema deberá mantener información de: su código de identificación, su nombre y apellidos, edad, sexo, teléfono de contacto, su dirección, y la información de para cuál o cuáles de los niños está actualmente registrado como guardián. **Todo niño registrado en el sistema deberá tener en todo momento al menos un adulto guardián**

que esté registrado en el sistema. Ningún adulto podrá figurar como guardián de un niño que no esté registrado en el sistema, ni viceversa. Se podrá eliminar de la guardería la información de adultos o niños, pero únicamente si al hacerlo ningún niño queda sin al menos un adulto guardián registrado en el sistema. Para garantizar la seguridad, tanto para eliminar un niño del sistema como para añadir otro adulto como guardián a un niño, se exigirá dar la identificación del niño y la identificación de uno de los adultos guardianes ya registrados para el niño.

Las operaciones imprescindibles para el TAD `guardería` son:

- `crear` una guardería vacía, sin niños ni adultos;
- `añadirAdulto`, que registre, si es posible, los datos de un adulto en el sistema, sin niños a su cargo;
- `añadirNiño`, que registre, si es posible, los datos de un niño en el sistema;
- `eliminarAdulto`, que elimine, si es posible, la información de un determinado adulto del sistema;
- `eliminarNiño`, que elimine, si es posible, la información de un determinado niño del sistema;
- `añadirGuardianANiño`, que registre en el sistema, si es posible, la información de que un determinado niño tendrá como guardián a un adulto dado;
- `quitarGuardianANiño`, que registre en el sistema, si es posible, la información de que un determinado adulto deja de ser guardián de un niño dado;
- `obtenerGuardianes`, que obtenga, si es posible, una cadena de caracteres con el listado de las identificaciones de los adultos guardianes registrados para un determinado niño;
- `obtenerNiñosCustodiados`, que obtenga, si es posible, una cadena de caracteres con el listado de las identificaciones de los niños para los que un determinado adulto está registrado como guardián.

Se debe escribir la especificación del TAD `guardería` con al menos las operaciones descritas en el enunciado.

2. Implementación eficiente (`quitaRepetidos`)

Sean l_1 y l_2 dos listas enlazadas con punteros que almacenan secuencias de números enteros distintos, y ordenadas de menor a mayor. Se trata de crear una nueva lista que, dadas l_1 y l_2, sea la resultante de borrar de l_2 los elementos que aparecen en l_1, de la forma más eficiente posible. Para ello, **diseña el procedimiento `quitaRepetidos` e indica su coste asintótico temporal en función de las longitudes de ambas secuencias**.

```
tipos  lista = ↑nodoLista;
   nodoLista = registro
                    dato: entero;
                    sig: lista
               freg

procedimiento quitaRepetidos(ent lista1, lista2: lista; sal res: lista)
{Pre: lista1 es una lista de enteros distintos ordenada de forma creciente.
      lista2 es una lista de enteros distintos ordenada de forma creciente.
 Post: devuelve en res la lista igual a la resultante de eliminar de lista2
      los elementos que también están en lista1.}
```

3. Implementación eficiente (`borraRepeticiones`)

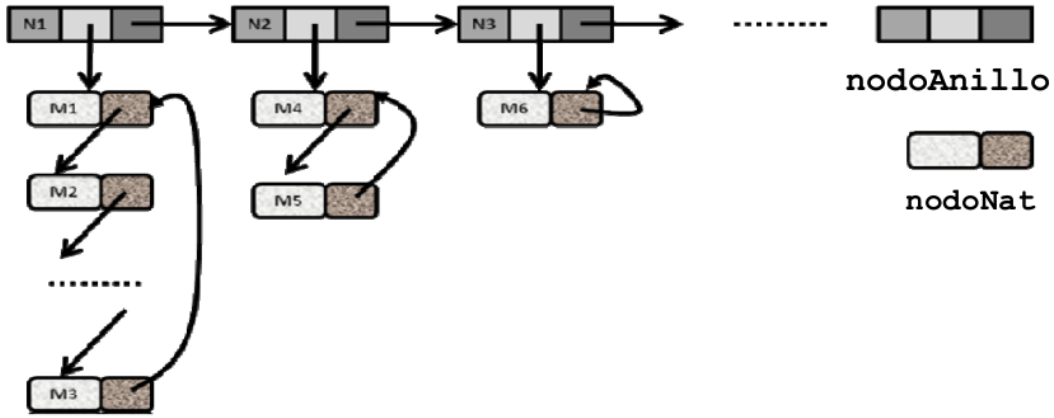
Implementa en pseudocódigo un procedimiento `borraRepeticiones` que, dada una lista enlazada mediante punteros de datos de tipo entero, ordenada por orden no decreciente, **devuelva la lista resultante de eliminar las repeticiones de números** (dejando una única aparición de cada entero distinto de la lista original).

```
tipos  lista = ↑nodoLista;
   nodoLista = registro
                    dato: entero;
                    sig: lista
               freg

procedimiento borraRepeticiones(e/s l: lista)
{Dada una lista enlazada mediante punteros l de datos de tipo entero y
    ordenada por orden creciente (<=), elimina de l las repeticiones de
    enteros, dejando únicamente la primera aparición de cada entero
    distinto y la lista ordenada de menor a mayor.}
```


4. TAD `listaAnillos`

Se quiere definir la estructura de datos lista de anillos. Cada anillo tiene un identificador representado por un número natural. Un anillo es una lista circular de números naturales. Una posible representación gráfica de esta estructura es la siguiente:



La lista lineal formada por nodos de tipo `nodoAnillo` ha de estar ordenada de mayor a menor identificador. No hace falta que los anillos, es decir, las listas circulares formadas por nodos de tipo `nodoNat`, estén ordenadas. La estructura lista de anillos ha de incluir una operación `borrar` que, dados dos números naturales N y M, elimine un `nodoNat` con el valor M de la lista circular apuntada por el `nodoAnillo` con identificador igual a N. Además, si este anillo quedara vacío, también tendría que eliminar el `nodoAnillo` con identificador N.

Se pide:

a) **Definir, en pseudocódigo, los tipos de datos necesarios para almacenar la estructura descrita**.

b) **Implementar, en pseudocódigo, la operación** `borrar`.

