

Práctica 4b - Listas

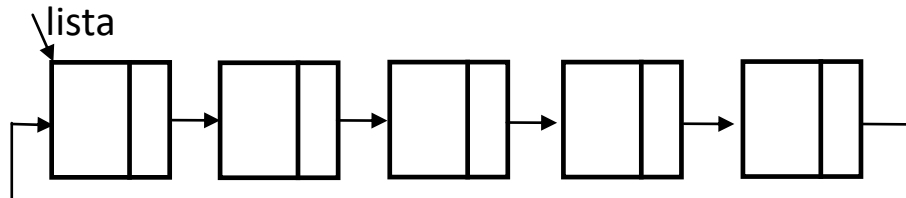
Estructuras de Datos

En esta práctica se profundizará más sobre el aprendizaje de listas, implementando algunas de sus posibles versiones, para después hacer uso de las mismas. La práctica consta de dos partes:

Primera parte

El comportamiento de un TAD Lista viene determinado por su especificación formal. Siguiendo la especificación formal vista en clase se pide implementar lo siguiente:

- Implementación de la unidad `uListaEst`, como una lista estática que simule memoria dinámica utilizando un array como almacén para guardar los nodos.
- Implementación de la unidad `uListCircular` como una lista circular con enlace simple y puntero al último elemento. Una posible representación gráfica se puede ver a continuación:



En ambos casos los elementos de las listas serán de tipo `TElem`, cuyo TAD estará definido en una unidad diferente (`uElem`).

Segunda parte

Se quiere comprobar si una subcadena está contenida dentro de una cadena. Dado que el tipo para representar cadenas en Pascal es el tipo `String`, que tiene un tamaño máximo y se quiere poder utilizar cadenas de cualquier tamaño sin tener ningún tipo de limitación, se opta por representar las cadenas mediante listas, donde cada elemento de la lista será un carácter (el `TElem` será de tipo `Char` en este caso). Así, la posible secuencia de acciones que debe tener el programa que se pide es la siguiente:

1. Crear la cadena (crear una lista con el contenido de la cadena).
2. Crear la subcadena (crear una lista con el contenido de la subcadena).
3. Comprobar si la subcadena está contenida en la cadena.
4. Eliminar ambas cadenas (eliminar las dos listas).

Se deberá extender el TAD `TListaEst` y `TListaCircular` para añadir la operación de poder recuperar el elemento que se encuentre en una posición determinada de la lista. A continuación, se presenta la especificación algebraica para dicha operación:

```

...
OPERACIONES

    (* operación observadora no selectora *)
    PARCIAL GetElemPos: TipoLista x Natural → TipoElemento

VARIABLES
    lista: TipoLista;
    p: Natural;

ECUACIONES DE DEFINITUD
    SI ((p > 0) Y (p <= Longitud(lista)) →
        DEF (GetElemPos (lista, p))

ECUACIONES
    GetElemPos(Construir(e, lista), p) = SI p=1 →
        e
        | GetElemPos (lista, p-1)

...

```

Un ejemplo de salida de diferentes ejecuciones se presenta a continuación:

La cadena [n,o,c] SI está contenida en la cadena [r,i,n,o,c,e,r,o,n,t,e]

La cadena [n,o,c] NO está contenida en la cadena [c,a,s,a]

La cadena [p,e,p,e] SI está contenida en la cadena [h,o,l,a ,p,e,p,e]

Nota 1: Se considera necesario un nivel elevado de encapsulación y abstracción.

Nota 2: Se hará uso de las normas de estilo dictadas en clase (cabecera del fichero, interfaz de la unidad con precondiciones, postcondiciones, excepciones, implementaciones con el análisis de complejidad de cada operación, nombres coherentes de variables y operaciones,...)

Plantilla de cabecera del fichero:

```

{ *****
*
*      Módulo:
*      Tipo:  Programa()      Interfaz-Implementación TAD ()      Otros()
*      Autor/es:
*      Fecha de actualización:
*      Descripción:
*
*****
*****}

```