UNIVERSIDAD DE MÁLAGA
E.T.S.I. TELECOMUNICACIÓN

Memoria Dinámica

Programación Modular
(1º de Ingeniería de Telecomunicación)

1. Desarrolla los siguientes procedimientos:

   a) `ALGORITMO Crear(ES Lista l)`
   Crea una lista vacía

   b) `ALGORITMO ImprimirElementos(E Lista l);`
   Imprime por pantalla todos los elementos de una lista

   c) `ALGORITMO N NumElems(E Lista l)`
   Devuelve el número de nodos de la lista enlazada

   d) `ALGORITMO TPuntero Busca(E Lista l; E N e)`
   Devuelve el puntero al nodo que contiene a "e"si existe, y NULO en caso contrario

   e) `ALGORITMO InsPrincipio(ES Lista l, E N n)`
   Inserta un dato como primer elemento de la lista

   f) `ALGORITMO InsFinal(ES Lista l, E N n)`
   Inserta un dato como último elemento de la lista

   g) `ALGORITMO InsOrdenado(ES Lista l, E N n)`
   Inserta el dato ordenadamente, suponiendo que los demás elementos están ordenados

   h) `ALGORITMO InsPosicion(ES Lista l, E N n, E N pos)`
   Inserta el dato en la posición indicada, si pos = NumElem(l) + 1, se inserta como último de la lista

   i) `ALGORITMO TipoPuntero InsBusca(ES Lista p;E N e)`
   Añade e a la lista l, si no está en ella, y siempre devuelve un puntero al nodo que contiene a e

   j) `ALGORITMO BorrarPosicion(ES l Lista; E N n)`
   Borra el n-ésimo nodo de la lista l. Si no existe esa posición no hace nada

   k) `ALGORITMO BorrarValor(ES l Lista; E N n)`
   Borra la primera aparición de ese valor en la lista l. Si no existe no hace nada

   l) `ALGORITMO BorrarTodasCopias(ES l Lista; E N n)`
   Borra todas las apariciones de ese valor en la lista l. Si no existe no hace nada

   m) `ALGORITMO BorrarMaximo(ES l Lista; E N n)`
   Borra la primera aparición del elemento con mayor valor de la lista l

   n) `ALGORITMO BorrarTodo(ES l Lista; E N n)`
   Borra todos los elementos de la lista l

   ñ) `ALGORITMO Intercambiar(ES Lista l1; E N n, m)`
   Intercambia el contenido del n-ésimo nodo de l con el del m-ésimo. Suponemos $1 \leq n, m \leq NumElem(l)$

   o) `ALGORITMO Intercambiar2(ES Lista l1; E N n, m)`
   Intercambia el n-ésimo nodo de l con el del m-ésimo cambiando los punteros. Suponemos $1 \leq n, m \leq NumElem(l)$

   p) `ALGORITMO Burbuja(ES Lista l)`
   Ordena la lista l mediante el método de la burbula, usando los procedimientos de intercambiando

   q) `ALGORITMO Concatenar(ES Lista l1;E Lista l2)`
   Concatena una copia de l2 al final de l1

   r) `ALGORITMO Duplicar(E Lista l; S Lista nl)`
   Crea una copia de l y la almacena en nl. El contenido anterior de nl se pierde

   s) `ALGORITMO Purga(ES Lista l)`
   Elimina los elementos duplicados de la lista enlazada con punteros l

Realiza la implementación de las operaciones anteriores para las siguientes definiciones de los tipos `Lista` y `TipoPuntero`.

a) Lista Enlazada.

```
Nodo *TipoPuntero
REGISTRO Nodo
    N valor
    TipoPuntero sig
FIN
TipoPuntero Lista
```

b) Lista Enlazada Circular. La definición de los tipos coincide pero siempre se ha de mantener un enlace del último nodo al primero de la lista.

c) Lista Doblemente Enlazada.

```
Nodo *TipoPuntero
REGISTRO Nodo
    N valor
    TipoPuntero sig
    TipoPuntero ant
FIN
TipoPuntero Lista
```

d) Lista con Cabecera.

```
Nodo *TipoPuntero
REGISTRO Nodo
    N valor
    TipoPuntero sig
FIN
REGISTRO Lista
    TipoPuntero primer,ultimo
    N num_elems
FIN
```

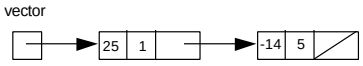
2. Ordenar recursivamente una lista ordenada mediante el método de mezcla definido en el tema de recursividad. Para ello hacer un algoritmo que, dadas dos listas ordenadas las mezcle en una nueva lista ordenada.

```
ALGORITMO Mergesort(ES Lista l)
ALGORITMO Mezcla(E Lista l1,l2;S Lista l)
```

3. Realizar algoritmos de rotación de listas enlazadas, hacia adelante, pasando los últimos elementos al principio y hacia atrás, pasando los primeros elementos pasan al final. El parámetro natural indica el número de rotaciones que hay que hacer:

```
ALGORITMO Adelante(ES lista l; E N k)
ALGORITMO Atras(ES lista l; E N k)
```

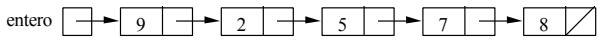
4. Hay muchas aplicaciones en las que se debe almacenar en la memoria un vector de grandes dimensiones. Si la mayoría de los elementos del vector son ceros, éste puede representarse más eficientemente utilizando una lista enlazada con punteros, en la que cada nodo es un registro con tres campos: el dato en esa posición, si es distinta de cero, el índice de esa posición y un puntero al siguiente nodo. Por ejemplo:



Esto indica que hay sólo dos elementos distintos de cero en el vector, es decir, éste es: (25, 0, 0, 0, -14, 0, 0) si consideramos los vectores con siete posiciones.

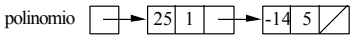
Escribe un programa que lea dos vectores por teclado, los introduzca en listas enlazadas y calcule su producto escalar. Diseña también las funciones para insertar y consultar el valor en una determinada posición del vector, teniendo en cuenta que si introducimos un elemento en una posición previamente existente se debe eliminar el valor anterior y sustituirlo por el nuevo.

5. Una forma de almacenar un número natural de valor mayor que el permitido en una computadora es introducir cada dígito en un nodo de una lista enlazada. Por ejemplo, la siguiente lista representa al número 92578:



Escribe una función que tome como parámetro un puntero a una lista enlazada y devuelva el número correspondiente en una variable de tipo unsigned int. Diseña también una función que lea por teclado una sucesión de dígitos (caracteres) y los introduzca como dígitos (naturales) en una lista enlazada, y otras dos funciones que realicen, respectivamente, la suma y el producto de números representados de esta forma.

6. Un polinomio en x de un grado arbitrario se puede representar mediante una lista enlazada con punteros, donde cada nodo contiene el coeficiente y el exponente de un término del polinomio, más un puntero al siguiente nodo. Por ejemplo el polinomio: $25x - 14x^5$ se puede representar como:



   a) Escribe una función que evalúe un polinomio P en un x = valor.
   `ALGORITMO N Evaluar (E TipoPolinomio p, E N valor)`

   b) Escribe una función que devuelva el coeficiente del término de grado `i` de un polinomio `P`.
   `ALGORITMO N Obtener (E TipoPolinomio p, E N i)`

   c) Escribe una función que sume 2 polinomios `P1` y `P2`:
   `ALGORITMO TipoPolinomio Sumar(E TipoPolinomio p1, E TipoPolinomio p2)`

   d) Escribe una función que realice la derivada de un polinomio P:
   `ALGORITMO TipoPolinomio Derivada (E TipoPolinomio p)`

7. Supongamos el tipo Conjunto definido mediante una lista enlazada según la siguiente cabecera:

```
TIPOS
    REGISTRO TipoNodo
        int dato
        TipoNodo *sig
    FIN
    TipoNodo *Conjunto
```

   Diseñar las siguientes operaciones para la intersección y unión de conjuntos:

```
ALGORITMO Conjunto interseccion (E Conjunto c1, E Conjunto c2)
ALGORITMO Conjunto union(E Conjunto c1, E Conjunto c2)
```

8. Hacer algoritmos recursivos sobre listas enlazadas para:

   a) mostrar los elementos de la lista

   b) mostrar en orden inverso los elementos de la lista

   c) devolver el mayor elemento de la lista

   d) devolver la suma de los elementos de la lista

   e) devolver la longitud de la lista

   f) invertir la lista