

4. Tratamiento de Listas

Ejercicio 4.1

Dada la siguiente declaración y el siguiente esquema:

```
P, Q, R: puntero a NODO
NODO: registro
DATO: entero
PROX: puntero a nodo
Fin registro
```



P	Q	R	Comentario

¿Qué hacen las siguientes órdenes?

Considerar que se ejecutan secuencialmente.

1. $P := P \uparrow . PROX$
2. $Q := P$
3. $R := P \uparrow . PROX$
4. $P \uparrow . DATO := Q \uparrow . DATO$
5. $P \uparrow . DATO := (Q \uparrow . PROX) \uparrow . DATO$
6. $R \uparrow . PROX := P$

Ejercicio 4.2

Dada una lista simplemente encadenada de números enteros, diseñar un algoritmo que a partir de ella genere otra lista conteniendo los nodos cuyos datos sean múltiplos de 3, dichos elementos deberán ser eliminados de la lista original. Se asume que la lista está cargada y que el algoritmo recibe como parámetro de entrada la dirección del primer elemento.

Ejercicio 4.3

Dada una lista doblemente encadenada que contiene datos de todas las provincias de la República Argentina: nombre, capital, cantidad total de habitantes y cantidad de analfabetos, y está ordenada alfabéticamente por nombre de provincia, se desea generar otra lista simplemente encadenada pero ordenada en orden decreciente por número de habitantes analfabetos.

Ejercicio 4.4

Dada una lista circular con los datos de los socios de un club: número, nombre y condición ('A'= alta, 'B'= baja); hacer un algoritmo que cree otra lista simplemente encadenada con los socios múltiplos de 100, que no estén dados de baja y eliminar de la lista original los socios dados de baja (cond = 'B').

Ejercicio 4.5

Una empresa FARMACÉUTICA, posee una lista simplemente enlazada de pedidos realizados por sus clientes en el último mes. Se desea generar dos grupos con la siguiente información:

- Los pedidos de los clientes deudores.
- Los pedidos de los clientes regulares.

Para ello se cuenta con un archivo "CLIENTES" indexado por Nro_cliente.

Deben tenerse en cuenta las siguientes consideraciones:

- La lista de entrada está ordenada por Nro_cliente y pueden existir varios pedidos de un mismo cliente.
- En el caso de recibir un pedido de un cliente que se encuentre dado de baja, debe ser dado de alta automáticamente.

- El total (Monto \$) de Dinero_deuda debe ser actualizado con la suma de los pedidos en Cuenta Corriente. Modificar la condición a Deudor si esa suma es mayor a cero.

Formato lista de entrada

Nro_cliente	Nro pedido	Precio pedido	TipoPedido (C:Contado, R: Cuer
◀			▶

Formato archivo "Clientes"

Nro_cliente	DNI	Domicilio	Dinero_deuda	Deudor (si/no)	Baja
-------------	-----	-----------	--------------	----------------	------

Formato lista salida

Nro_cliente	Total pedido
-------------	--------------

Ejercicio 4.6

Usando las operaciones de Lista, escribir un procedimiento AGRUPA (PRIM,ULT: puntero a nodo; A:entero) que, dada una lista doblemente encadenada de enteros L agrupe (sume) elementos de tal manera que en L queden sólo elementos mayores o iguales que A. El algoritmo recorre la lista y cuando encuentra un elemento menor empieza a agrupar (sumar) el elemento con los siguientes hasta llegar al valor A o hasta que se acabe la lista (el elemento menor se debe eliminar de la lista.)

Por ejemplo si:

$$L = \{3,4,2,4,1,4,4,3,2,2,4,1,4,1,4,4,1,4,4,2\},$$

entonces AGRUPA(PRIM,ULT,10) da

$$L = \{13,12,13,10,10\}.$$

En la lista final NO deben quedar elementos menores que A, salvo, eventualmente, el último.

Ejercicio 4.7

La empresa "Remises Yapú S.A." necesita solucionar el problema de asignación de vehículos para sus clientes. La empresa posee 105 autos, y maneja dos colas (estructuras FIFO) una llamada "No

"Asignados" y otra llamada "No Asignados". En la primera se encuentran todos los autos que no han sido asignados a algún cliente en lo que va del día y en la segunda los autos que ya han sido asignados. La idea de la cola "Asignados" es manejar en forma equitativa la asignación de turnos, es decir que cada vehículo tenga igual oportunidad de obtener un cliente. Los vehículos se asignan de la siguiente forma:

- Si existen autos "No Asignados", se toma uno de ellos y se lo mueve a la cola de "Asignados".
- Si no existen autos "No Asignados", se toma uno de la cola de "Asignados", y se lo coloca al final de la cola.

Siempre que se asigna un vehículo se debe incrementar en uno un campo "cantidad_de_clientes", a fin de saber cuantos clientes tuvo. Por final de proceso se requiere saber la cantidad total de clientes de ese día. Se pide: Confeccionar una subacción que realice la asignación de vehículos y el mantenimiento de las colas, usando listas.

Ejercicio 4.8

Se dispone de una lista doblemente enlazada con un conjunto de números naturales. Este conjunto está dividido en 10 grupos. En cada grupo el primer elemento indica la cantidad de números subsiguientes a él que integran dicho grupo. Se desea generar una nueva lista circular, simplemente encadenada, en la cual cada nodo contendrá el valor máximo de cada grupo. Dicha lista debe quedar ordenada en forma ascendente y, al final se debe informar cuál es el promedio de todos los máximos.

Ejercicio 4.9

El ciclo de desarrollo de software tradicional, incremental e iterativo cuenta con las siguientes etapas: Captura de Requisitos, Análisis, Diseño, Desarrollo, Implementación y Prueba. De la última etapa se obtiene un informe que contiene las modificaciones que deben realizarse al software para que cumpla con las funcionalidades para las cuales fue creado inicialmente.

Suponga que cuenta con una lista de proyectos de software de los cuales se tiene la siguiente información:

Proy AN(20)	Resp AN(30)	Fch_Ini Fecha	C_Error N(3)
-------------	-------------	---------------	--------------

El dato Cant_Errores indica la cantidad de errores que deben solucionarse en cada proyecto.

La información respecto a cada Error se almacena en una lista de acuerdo al orden de los proyectos y teniendo en cuenta la cantidad que le corresponden a éste. Por ejemplo si en la Lista 1 el Proyecto A tiene 3 errores, los 3 primeros nodos de la Lista 2 corresponden al Proyecto A. La información que se almacena por Error es la siguiente:

Desc_E AN(30)	Est AN(1)
---------------	-----------

Est – M: En Modificación | O: En Observación | R: Resuelto

Durante el proceso de Control de Proyectos se analiza cada error de cada proyecto, y se le consulta al Responsable el estado del Error (en caso que no haya sido Resuelto) para actualizarlo. El Responsable debe tener la opción de recorrer todos los proyectos, en el orden almacenados, las veces que considere necesario, luego de analizar el último en la lista.

Suponiendo que se necesita simular el control de los proyectos, creando dos listas para almacenar por un lado Proyectos y por otro Errores. Y que además se debe informar al final del proceso qué cantidad de proyectos pasaron el control con todos sus errores resueltos. Se le solicita a ud:

1. Dado el Algoritmo resuelto pero incompleto, analice la propuesta presentada e indique las sentencias incorrectas, encerrándolas con un círculo, e indique las modificaciones necesarias para que funcione, complete la líneas de puntos y agregue las acciones que falten.
2. Desarrolle la acción Control_Proyectos y complete el ambiente si fuera necesario

```

Accion Ej_1 (PRIM: Puntero a Nodo, PRIM2: Puntero a Nodo2) Es
    Ambiente
        NODO = Registro
        Proy: N(30)
        Resp: AN(30)
        Fecha: Fecha
        C_error: N(3)
        Prox: Puntero a Nodo
        FinRegistro
        NOD01 = Registro
        Desc_E: AN(30)
        Est: AN(1)
        Prox: Puntero
        FinRegistro
        P, K: Puntero a Nodo
        Q : Puntero a Nodo1
        Cont, cant_proy: N(3)
        Opc: caracter

        Subaccion Cargar_Proyectos es
            Escribir("Ingrese 'S' para empezar o cualquier otro caracter para salir")
            Leer(opc)
            Mientras opc = 'S' hacer
                .....

```

```

Leer(*P.Proy, *P.Resp, *P.Fch_Ini, *P.C)

Si ..... Entonces
    P := *P.prox;
    PRIM:=P;
Sino
    K := PRIM
    Mientras (*K.prox <> PRIM) hacer
        K := *K.prox
    FinMientras
    *K.prox := P
    *P.prox := PRIM
Fsi;
Cont:= 0
Cargar_Errores
Escribir("Ingrese 'S' para seguir cargando proyectos o cualquier
otro caracter para salir")
Leer(opc)
Fin Mientras
Fin Subaccion

Subacción Cargar_Errores es
Mientras Cont <= *P.C_error hacer
    NUEVO(Q)
    Escribir(*Q.Desc_E, *P.Est)
    Si PRIM2 = NIL Entonces
        *Q.prox := Nil
        ..... := Q
    Sino
        *Q.prox := PRIM2
        PRIM2 := Q
    FinSi
    .....
Fin Mientras
FinSubaccion

Algoritmo
    Cargar_Proyectos;
    Control_Proyectos;
    Escribir('La cantidad de proyectos con errores resueltos es de', *P.C_error)
Fin Accion

```