



ALGORITMOS Y ESTRUCTURAS DE DATOS I

- Junio 2001 -

PARTE I

Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)

Apellidos :	DNI :
Nombre :	Especialidad :

Sección Informativa:

- El examen se compone de dos partes, siendo la presente, la primera.
- Dispone de 1 hora y 30 minutos para la realización de esta Parte 1, integrada por dos ejercicios.
- Solo se podrán realizar consultas durante los primeros 20 minutos.
- Se agotará la convocatoria transcurridos los primeros 20 minutos.

Ejercicios:

1.- (3.5 Puntos).

Deseamos hacer un programa que gestione los discos compactos (CDs) de música para una tienda. Los datos que queremos almacenar de cada uno de ellos son el título del disco, el intérprete principal, el título de cada una de canciones y su correspondiente intérprete, el año en el que se ha editado el CD y un campo de observaciones para poner alguna frase, como por ejemplo si ha sido descatalogado, si es una versión actualizada de otro anterior, si es un doble CD, etc.

Tenemos también los siguientes requisitos:

- Es preciso almacenar el número de copias de un mismo CD disponibles en la tienda.
- Las operaciones más frecuentes serán:

- las de búsqueda, sobre todo por el nombre del intérprete principal del CD, aunque también podrían ser por otros como por ejemplo el título del CD, el campo título de canción o intérprete, etc. En todos estos casos habría que mostrar todos los CDs que cumplan el hecho de tener un título coincidente con el solicitado, todos los CDs que contengan una canción del título o autor solicitado en caso de que la búsqueda sea por título o autor.
- en menor medida, las de alta de un nuevo CD cuando llegue a la tienda.

- Las operaciones menos frecuentes, pero también necesarias son la modificación de los datos de un CD.

- No queremos eliminar un CD ni siquiera cuando se agota, para que su información siga estando presente.

- Desconocemos a priori el número de CDs diferentes que hay o puede llegar a haber en la tienda. El número de canciones por CD es también desconocido.
- El ordenador permanecerá encendido durante las horas en las que la tienda se encuentre abierta apagándose durante el resto del tiempo, de modo que lógicamente la información se recupere entre sesiones.
- Suponemos que el ordenador dispone de suficiente espacio tanto en memoria como en disco para contener la información de todos los CDs de la tienda.

Se pide:

a) **Indique las estructuras de datos** que considere óptimas para el programa del problema descrito y **justifíquelas** (por qué ha preferido esas y no otras). Puede utilizar cualquiera de las estructuras de datos estudiadas durante el curso.

b) Realice las secciones de declaración de tipos y variables **completa** que considere necesitarla ese programa, en base a lo que acaba de indicar en el apartado a).

2.- (2.5 Puntos).

Los valores de la función de Ackermann son:

$$\begin{array}{ll} \text{Ack}(0,0)=1 & \text{Ack}(1,0) = \text{Ack}(0,1) \\ \text{Ack}(0,1)=2 & \text{Ack}(2,0) = \text{Ack}(1,1) \\ \text{Ack}(0,2)=3 & \text{Ack}(3,0) = \text{Ack}(2,1) \\ \text{Ack}(0,3)=4 & \text{Ack}(4,0) = \text{Ack}(3,1) \end{array}$$

$$\text{Ack}(m,n) = \text{Ack}(m-1, \text{Ack}(m,n-1)) \text{ si } m \text{ y } n \text{ son mayores que cero}$$

a) Diseñe una función recursiva para calcular el valor de la función de Ackermann.

b) Realice un seguimiento a la llamada Ack(2,2) y rellene los siguientes valores calculados al realizar dicho seguimiento:

	<u>Valor entero devuelto</u>
Ack(1,0) =	
Ack(1,1) =	
Ack(1,2) =	
Ack(1,3) =	
Ack(1,4) =	
Ack(1,5) =	
Ack(2,0) =	
Ack(2,1) =	
Ack(2,2) =	



ALGORITMOS Y ESTRUCTURAS DE DATOS I

- Junio 2001 -

PARTE II

Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)

Sección Informativa:

- Dispone de 1 hora y 30 minutos para la realización de esta parte 2, integrada por un ejercicio.
- Solo se podrán realizar consultas durante los primeros 20 minutos.

3.- (4 Puntos).

Se desea construir, a partir de un archivo de acceso directo (con organización secuencial) en el que los elementos no se encuentran ordenados, una lista doblemente enlazada donde los elementos se encuentren ordenados de modo ascendente.

Cuenta con la siguiente sección de declaración:

tipo

```
cadena=tabla [1..12] de caracter;  
fic=fichero de entero; (* fichero de acceso directo *)  
punt=puntero a elemento;  
elemento = registro  
                numero: entero;  
                anterior,siguiente: punt;  
fregistro;
```

ftipo

Para ello, deberá implementar el siguiente subprograma:

```
accion traslado (nomb_fi:cadena; long:entero; var ini,fin:punt);
```

(* nomb_fi es el nombre del fichero, long es el número de elementos del fichero nomb fi, y ini y fin son los punteros al inicio y fin de la lista doblemente enlazada *)

var

f: fic;

fvar

inicio

...

...

fin



ALGORITMOS Y ESTRUCTURAS DE DATOS I SOLUCIÓN DEL EXAMEN

- Junio 2001 -

Dpto. Ing. Electrónica, Sistemas Informáticos y Automática
E.P.S. La Rábida (Universidad de Huelva)

PROBLEMA 1

a) Como mínimo, se considera necesaria una estructura con estas especificaciones, o bien otra más compleja que la mejor:

- Se emplearía una estructura de datos dinámica pues conjunto de CDs puede ser variable, y se desea hacer accesos frecuentes y eficientes por lo que es necesario disponer en memoria RAM de la estructura de datos.

- Dentro de las estructuras de datos dinámicas sería preferible la de tipo lista simplemente enlazada porque:

- No son necesarias las búsquedas recorriendo la estructura hacia atrás
- Un solo puntero de enlace entre registros consume la mínima memoria para éste cometido de enlazar la secuencia de nodos.

- Los nodos serán de tipo registro con los siguientes campos:

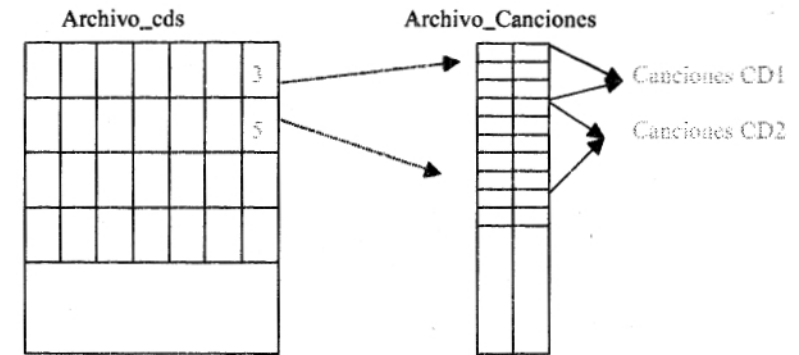
- título del CD
- intérprete principal
- lista de títulos e intérpretes de cada una de las canciones, pues no sabemos de antemano el número de canciones. Esta lista será también de tipo simplemente enlazada.
- año de edición
- observaciones
- cantidad disponible
- puntero de enlace para la lista.

- Los nodos de la lista de CDs estarían ordenados por la operación más frecuente: el intérprete principal, es decir, alfabéticamente por el nombre del intérprete.

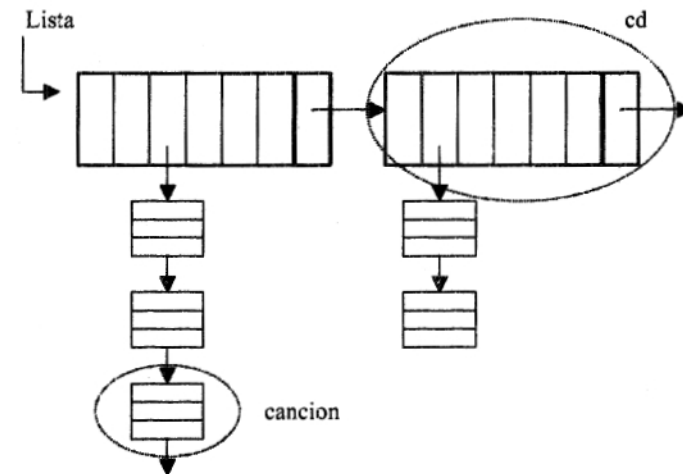
- Sería necesario manejar ficheros para cargar y volcar la lista al empezar y terminar la ejecución del programa. El volcado podría utilizar dos ficheros:

- Uno secuencial de CDs en el que sus elementos serían registros cuyos campos serían similares al registro de la lista de CDs salvo porque se eliminarían los punteros a las canciones y el campo de enlace de los nodos y se añadirían dos campos más: un índice entero que nos indique a partir de qué posición se colocan las canciones del CD y un contador del número de canciones del CD.
- Uno para las canciones, también de acceso secuencial, puesto que solo lo utilizamos como respaldo o almacenamiento.

Gráficamente en disco será :



Gráficamente en memoria será :



b)
tipo

tipo_puntero_cancion- puntero a tipo_nodo_cancion;

tipo_nodo_cancion= registro
 titulo_cancion: tabla [1..25] de caracter;
 interprete_cancion: tabla [1..20] de caracter;
 siguiente: tipo-puntero canción;
fregistro

tipo_CD = registro
 titulo: tabla [1..25] de caracter;
 interprete: tabla [1..20] de caracter;
 temas: tipo_puntero_cancion;
 editado: entero;
 cantidad: entero;
 observaciones: tabla [1..20] de caracter;
fregistro

tipo_puntero_CD = puntero a tipo_nodo_CD;

tipo_nodo_CD = registro
 datos: tipo_CD;
 siguiente: tipo_puntero_CD;
fregistro

canciones = registro
 titulo_cancion: tabla [1..25] de caracter;
 interprete_cancion: tabla [1..20] de caracter;
fregistro

CD_disco = registro
 titulo: tabla [1..25] de caracter;
 interprete: tabla [1..20] de caracter;
 editado: entero;
 cantidad_CDs: entero;
 observaciones: tabla [1..20] de caracter;
 num_canciones: entero;
fregistro

ftipo

var

lista:tipo_puntero_CD;
ed:tipo_CD;
canción:tiponodo_canción;
archivo_cds:CD_disco;
archivo_canciones:fichero de canciones;

fvar

PROBLEMA 2

a)

función Ack(m,n : entero) : entero;
inicio

si m = 0 → retorna (n+1) (* condición de fin *)

□ m > 0 →

si n = 0 → retorna (Ack(m-1,1))

 □ n > 0 → retorna (Ack (m-1, Ack (m,n-1)));

ffunción

b) Análisis recursivo para Ack (2,2)

Ack(2,2)
 Ack(1,Ack(2,1))
 Ack(1,Ack(2,0))
 Ack(1,1)
 Ack(0,Ack(1,0))
 Ack(0,1) 2
 Ack(0,2) 3
 Ack(1,3) 3
 Ack(0,Ack(1,2))
 Ack(0,Ack(1,1))
 Ack(0,3) 4
 Ack(0,4) 5
 Ack(1,5) 5
 Ack(0,Ack(1,4))
 Ack(0,Ack(1,3))
 Ack(0,5) 6
 Ack(0,6) 7

Luego Ack(2,2) devuelve el valor 7

	<u>Valor entero devuelto</u>
Ack (1,0) =	2
Ack (1,1) =	3
Ack (1,2) =	4
Ack (1,3) =	5
Ack (1,4) =	6
Ack (1,5) =	7
Ack (2,0) =	3
Ack (2,1) =	5
Ack (2,2) =	7

PROBLEMA 3

Existen varias formas de solucionar el problema. En éste caso se ha preferido la siguiente:

- Leer todos los elementos del fichero secuencialmente desde el primero hasta el último, e insertarlos al mismo tiempo en la lista doblemente enlazada, haciéndolo siempre por el final de dicha lista.
- Ordenarla lista.

Las ventajas principales de hacerlo así son:

- Eficiencia: Puesto que la ordenación se produce en memoria principal (sobre la lista) y no sobre fichero, lo cual sería más lento.
- Sencillez de implementación: La inserción en la lista se hace siempre por el final.

funcion Vacía_L(ini,fin:punt):booleano;
{ Devuelve verdadero en caso de estar vacía la lista doblemente enlazada }
inicio

Vacía L:= (ini=nulo) y (fin=nulo)

ffuncion

accion Insertar_L(dat:entero; var ini,fin:punt);
{ Inserta siempre por el final }

var
pos:punt;
fvar
inicio
reservar (pos);
pos^.numero:=dat;
si Vacía L(ini,fin) →
inicio
ini:=pos;
fin:=pos;
pos^.siguiente:=nulo;
pos^.anterior:=nulo;
fin

□ no Vacía L(ini,fin) →
inicio
fin^.siguiente:=pos;
pos^.anterior:=fin;
pos^.siguiente:=nulo;
fin:=pos;
fin

faccion

accion Intercambiar (var pos1,pos2:punt);
{ Intercambio de elementos dentro de la lista: realmente solo es necesario intercambiar su campo de información }

var
temp:entero;
fvar
inicio
temp:=pos1^.numero;
pos1^.numero:=pos2^.numero;
pos2^.numero:=temp;

faccion

accion Ordenar (var ini,fin:punt);
{ Se trata de ordenar la lista de alguna forma, por ejemplo aquí se implementa el algoritmo Burbuja por sencillez }

var

pos,ele:punt;

fvar

inicio

pos:=ini;

mientras (pos<>fin) hacer

inicio

ele:=fin;

mientras (ele<>pos) hacer

inicio

si (ele^.numero<ele^.anterior^.numero) →

Intercambiar (ele,ele^.anterior)

□ no (ele^.numero<ele^.anterior^.numero) →

nada;

ele:=ele^.anterior;

fin

pos:=pos^.siguiente;

fin;

faccion

accion Traslado (nomb_fi:cadena; long:entero; var i, fin:punt);
{ Abre el fichero, lo carga elemento a elemento en la lista insertando siempre por el final, y cuando están todos los elementos en la lista la ordena }

var

f:fic;

dato,i:entero;

fvar

inicio

abrir(f,l,nomb_fi);

para i:=1 hasta long hacer

inicio

leer(f, i, dato); {Recuerde que el fichero es de acceso directo}

Insertar_L(dato,ini,fin);

fin;

cerrar(f);

Ordenar(ini,fin);

faccion