UNIVERSIDAD PONTIFICIA DE SALAMANCA EN MADRID
Facultad de Informática
Escuela Universitaria de Informática
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

Cuadernulolo de examen

ASIGNATURA	**FUNDAMENTOS DE PROGRAMACIÓN II**	CÓDIGO	**113**
CONVOCATORIA	**Parcial de mayo de 2004**	PLAN DE ESTUDIOS	**2000 / 2002**
ESPECIALIDAD	**Común**	CURSO	**1º**
TURNO	**Mañana**	CURSO ACADÉMICO	**2003/2004**
CARÁCTER	**Cuatrimestral**	PROGRAMA	**Ingeniería Informática** **Ingeniría Técnica Informática**
DURACIÓN APROXIMADA	**1 hora 45 minutos**		

APELLIDOS: ..... NOMBRE: .....

GRUPO: ..... NÚMERO DE EXPEDIENTE: .....

Solución propuesta

Pregunta teórico práctica

1. Archivos indexados. Explique la estructura de un archivo indexado y las distintas áreas de las que consta. Ilustre mediante un gráfico la conexión entre ellas. Explique sus ventajas e inconvenientes respecto a los archivos de organización secuencial y de organización directa. Explique como se realizan los accesos secuenciales y directos en un archivo indexado.

Apartados 9.12.4 y 9.13 del libro de texto y apuntes de clase.

Aplicación

Se tiene ya creado un archivo indexado. El área de datos contiene los campos `DNI`, `Nombre` y `Ciudad` y su campo clave es el `DNI`. El área de índices está ya cargada en un array en memoria y puede contener un máximo de 200 registros (no tiene porqué estar llena).

- Declare las estructuras de datos y las variables globales necesarias para mantener el archivo indexado.

```
const
   MaxReg = 200 //Número máximo de registros del archivo
tipos
   registro = Persona
      cadena : DNI,nombre,ciudad
      entero : estado
   fin_registro
   archivo_d de Persona : Personas //Área de datos del archivo indexado
   registro : índice   //Estructura del área de índices
      cadena : clave
      entero : NRR     //Número de registro relativo donde se almacena la clave
   fin_registro
   array[0..MaxReg] de índice : TablaIndices //Área de índices
```

- Codifique un procedimiento que permita listar todos los registros ordenados por el campo `DNI`.

```
procedimiento Listado(E Personas : A;E TablaIndices : t ; E entero :n)
//n es el número de elementos ocupados del índice
var
   Persona : R
   entero : i
inicio
   //Se supone que el área de datos ya está abierta
   desde i ← 1 hasta n hacer
      leer(A,t[i].NRR,R)
      escribir(R.DNI, R.Nombre,R.Ciudad)
   fin_desde
fin_procedimiento
```

Puntuación: 5 puntos

Pregunta práctica

Se tienen almacenados en una lista simplemente enlazada `L1` una serie de números enteros ordenados de forma ascendente.
Se pide:

1. Realice las declaraciones de datos necesarias para ejecutar los apartados que aparecen a continuación.
   Puntuación: 0,5 puntos

```
tipos
   entero = TipoElemento
   puntero_a nodo = lista
   registro = nodo
      TipoElemento = info
      lista = sig
   fin_registro
```

2. Codifique un procedimiento que almacene en una lista enlazada `L2` los números que ocupen las posiciones impares de la lista. Los elementos de la lista `L2` deberan almacenarse en orden ascendente.
   Puntuación: 1,5 puntos

```
procedimiento InsertarPosicionesImpares(E lista : L1 ; E/S lista : L2)
var
   entero : conta
inicio
   CrearLista(L2)
   conta ← 1
   mientras L1 <> nulo hacer
      //Si la posición es impar, se inserta ordenado en L2
      si conta mod 2 <> 0 entonces
         InsertarOrdenado(L2,L1↑.info)
      fin_si
      conta ← conta + 1
      L1 ← L1↑.sig
   fin_mientras
fin_procedimiento

procedimiento CrearLista(E/S lista : l)
inicio
   l ← nulo
fin_procedimiento

procedimiento InsertarOrdenado(E/S lista : l; E TipoElemento : e)
var
   lista : act,ant
   lógico : encontrado
inicio
   encontrado← falso
   act ← l
   mientras no encontrado y (act <> nulo) hacer
      si e = act↑.info entonces
         encontrado ← verdad
      si_no
            ant ← act
            act ← act↑.sig
      fin_si
   fin_mientras
   si act = l entonces
      LInsertar(l,e)
   si_no
      LInsertar(ant↑.sig,e)
   fin_si
fin_prodedimiento
```

```
procedimiento LInsertar(var l: lista; e : TipoElemento)
var
   lista : aux
inicio
   nuevo(aux)
   aux↑.info ← e
   aux↑.sig ← l
   l ← aux
fin_procedimiento
```

3. Codifique un procedimiento que elimine de la lista `L1` los elementos impares.
   Puntuación: 1,5 puntos

```
procedimiento EliminarElementosImpares(E/S lista : L1)
var
   lista : act,ant
inicio
   act ← L1
   ant ← nulo
   mientras act <> nulo hacer
      si act↑.info mod 2 <> 0 entonces
         //Si el elemento es impar y es el primero de la lista
         //se borra el primer elemento (al que apunta L1)
         si ant = nulo entonces
            LBorrar(L1)
            //Ahora el elemento actual es otra vez el primero
            act ← L1
         si_no
            //Si el elemento actual es impar y no es el primero de la lista
            LBorrar(ant↑.sig)
            //El elemento anterior no cambia, pero el actual será el siguiente nodo
            act ← ant↑.sig
         fin_si
      si_no
         //Si el elemento no es impar movemos los punteros anterior y actual
         ant ← act
         act ← act↑.sig
      fin_si
   fin_mientras
fin_procedimiento

procedimiento LBorrar(E/S lista : l)
var
   lista : aux
inicio
   si l = nulo entonces
      //Error, lista vacía
   si_no
      aux ← l
      l ← l↑.sig
      liberar(aux)
   fin_si
fin_procedimiento
```

4. Codifique un procedimiento preferentemente recursivo que elimine de la lista `L1` los 5 últimos elementos.
Puntuación: 1,5 puntos

```
procedimiento BorrarUltimos(E/S lista : l; E/S entero : conta)
var
   lista :aux
inicio
   si l = nulo entonces
      //Si llegamos al final de la lista inicializamos el contador de nodos
      conta ← 0
   si_no
      BorrarUltimos(l↑.sig,conta)
      //Al ser recursivo el contador contará desde
      //el último elemento hacia el primero
      conta ← conta +1
      //Si estamos antes del 5° elemento contando desde el último hacia el primero
      si conta <= 5 entonces
         //Borramos el elemento
         aux ← l
         //Como es el ultimo, apuntará a nulo
         l ← nulo
         liberar(aux)
      fin_si
   fin_si
fin_procedimiento
```