



Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Junio de 2000 (2º parcial)	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

Soluciones al examen

Preguntas teórico-prácticas (1,5 puntos cada pregunta)

1. Ordenación de archivos secuenciales. ¿Por qué la ordenación de archivos secuenciales debe utilizar técnicas distintas que la ordenación de arrays? Enumere y explique brevemente las distintas técnicas de ordenación de archivos secuenciales que conozca.

Práctica

Dado un archivo secuencial con las siguientes claves,

8	15	13	21	43	24	16	38	43	33	36	2	90
---	----	----	----	----	----	----	----	----	----	----	---	----

Realice un seguimiento que ordene dicho archivo por la clave utilizando el método de ordenación por mezcla natural indicando como quedarían dispuestas las claves en cada una de las pasadas.

Primera pasada

F1	8	15	24	33	36								
F2	13	21	43	16	38	43	2	90					
F3	8	13	15	21	24	33	36	43	16	38	43	2	90

Segunda pasada

F1	8	13	15	21	24	33	36	43	2	90			
F2	16	38	43										
F3	8	13	15	16	21	24	33	36	38	43	43	2	90

Tercera pasada

F1	8	13	15	16	21	24	33	36	38	43	43		
F2	2	90											
F3	2	8	13	15	16	21	24	33	36	38	43	43	90

2. Programación Estructurada de Jackson: Describa la simbología que utiliza la metodología de Jackson para representar las distintas estructuras de control. Enumere y explique brevemente los pasos que hay que seguir para resolver un problema utilizando la metodología de Jackson.



Práctica

Una compañía aérea mantiene sus vuelos en un archivo secuencial ordenado por el número de vuelo. Los campos del archivo son:

Campo	Descripción
NumVuelo	Campo clave del archivo. El archivo está ordenado por este campo
Origen	Aeropuerto de origen
Destino	Aeropuerto de destino
Salida	Fecha y hora de la salida del vuelo
Num-plazas	Número total de plazas del vuelo

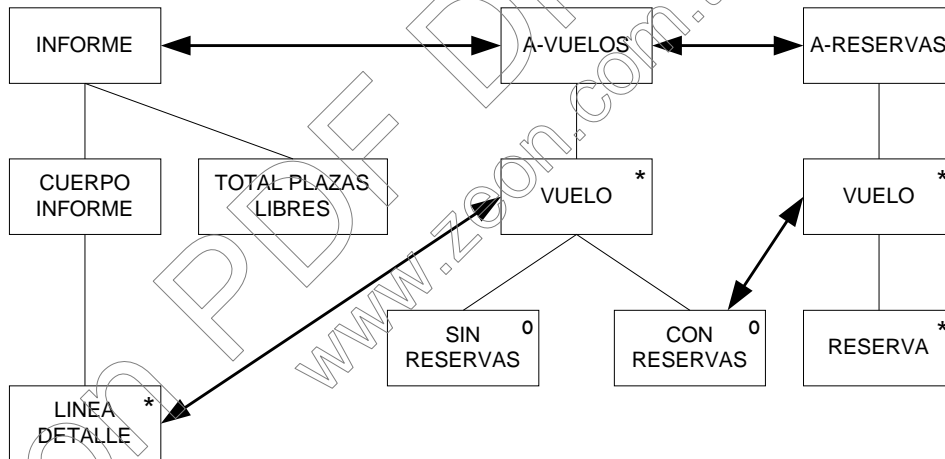
Las reservas efectuadas están almacenadas en otro archivo secuencial también ordenado por número de vuelo. En este archivo, cada reserva supondrá una plaza en el vuelo.

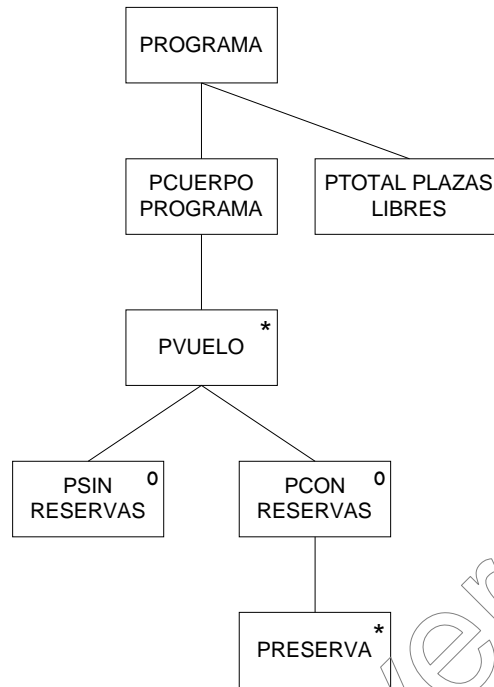
Campo	Descripción
NumVuelo	Campo clave del archivo. El archivo está ordenado por este campo
NumReserva	
DatosReserva	Datos adicionales de la reserva

- Todos los vuelos del archivo de reservas existen en el de vuelos
- En un vuelo puede haber más de una reserva.
- No todos los vuelos tienen reservas

Con estos dos archivos se desea hacer un listado en el que aparezca el número de vuelo y las plazas que quedan libres en dicho vuelo. Además, al final del listado se aparecerá el total de plazas libres.

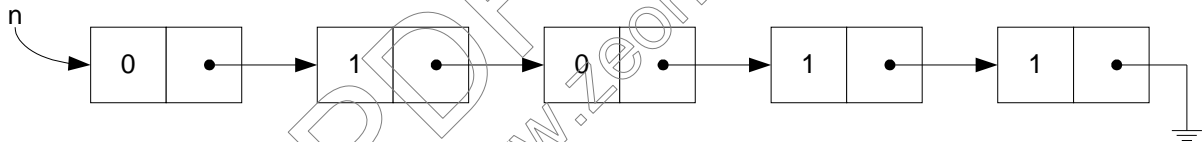
Diseñe, utilizando la metodología de Jackson, la estructura del programa necesario para resolver el problema (sólo será necesario aplicar los tres primeros pasos de la metodología).





Pruebas prácticas

- Se desea implementar números binarios utilizando listas enlazadas. Cada nodo de la lista almacenará el dígito binario (un 0 o un 1). El dígito menos significativo ocupará la primera posición de la lista. Por ejemplo, si el número n fuera 11010, se almacenaría como:



- Diseñe las estructuras de datos necesarias para realizar el problema.
- Escriba un procedimiento que permita leer un número binario desde teclado y almacenarlo en una lista enlazada de la forma indicada.
- Escriba un procedimiento que permita sacar el número por pantalla.
- Escriba un procedimiento que reciba dos números binarios almacenados en una lista enlazada y devuelva otra lista enlazada con la suma de ambos.

Puntuación: 3,5 puntos

Apartado A

tipos

```

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

Otros procedimientos utilizados en la solución

```

procedimiento ListaNueva(E lista : l)
inicio
  l ← nulo
  
```



fin_procedimiento

```
lógico : función EsListaVacia( l : lista)
inicio
    devolver(l = nulo)
fin_función
```

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

```
procedimiento LPrimero(E lista : l ; E/S TipoElemento : e)
inicio
    si EsListaVacia(l) entonces
        // Error, lista vacía
    si_no
        e ← l↑.info
    fin_si
fin_procedimiento
```

```
procedimiento LSiguiente(E lista : l ; E/S lista : lista)
inicio
    si EsListaVacia(l) entonces
        // Error, lista vacía
    si_no
        aux ← l↑.sig
    fin_si
fin_procedimiento
```

Apartado B

```
procedimiento LeerBinario(E/S lista : lista)
var
    entero : digito
    lista : aux :
inicio
    n ← nulo
    leer(digito)
    //Se leen dígitos hasta que alguno sea menor que 0 o mayor que 1
    mientras (digito >= 0) y (digito <=1) hacer
        LInsertar(n,digito)
        ReadLn(digito)
    fin_mientras
fin_procedimiento
```

Apartado C

```
procedimiento EscribirBinario(E lista : n)
var
    TipoElemento : digito
inicio
    //Se utiliza recursividad para escribir la lista desde
    //el dígito más significativo al menos significativo
    si no EsListaVacia(n) entonces
        EscribirBinario(n↑.sig)
        LPrimero(n,digito)
```



```
    escribir(digito)
  fin_si
fin_procedimiento
```

Apartado C

```
procedimiento SumarBinario(E lista : n1,n2 ; E/S lista : s)
var
  entero : acarreo,digito,d1,d2
  lista : aux
  lista : ultimo //Guarda la dirección del último nodo insertado
inicio
  ListaNueva(s)
  acarreo ← 0
  mientras no EsListaVacía(n1) o no EsListaVacía(n2) hacer
    //Si la lista n1 no está vacía se lee el siguiente dígito (d1)
    //y se pasa al siguiente. Si está vacía d1 es 0
    si no EsListaVacía(n1) entonces
      LPrimero(n1,d1)
      LSiguiente(n1,n1)
    si_no
      d1 ← 0
    fin_si
    //Si la lista n2 no está vacía se lee el siguiente dígito (d2)
    //y se pasa al siguiente. Si está vacía d2 es 0
    si no EsListaVacía(n2) entonces
      LPrimero(n2,d2)
      LSiguiente(n2,n2)
    si_no
      d2 ← 0
    fin_si
    //El dígito del resultado es la suma de los dos más el acarreo
    digito ← d1 + d2 + acarreo
    //Si el resultado es menor o igual que uno no hay acarreo
    si digito <= 1 entonces
      acarreo ← 0
    si_no
      //Si el resultado es mayor que uno se vuelve a calcular
      //el dígito y el acarreo
      acarreo ← digito div 2
      digito ← digito mod 2
    fin_si
    //Si es el primer nodo de la lista, se inserta al principio
    //si no, se inserta después del último nodo
    //La variable ultimo siempre apuntará al último nodo insertado
    si s = nulo entonces
      LInsertar(s,digito)
      ultimo ← s
    si_no
      LInsertar(ultimo↑.sig,digito)
      ultimo ← ultimo↑.sig
    fin_si
  fin_mientras
  //Si el último acarreo es 1 se añade a la lista
  si acarreo > 0 entonces
    LInsertar(ultimo↑.sig,acarreo)
  fin_si
fin_procedimiento
```



2. Una empresa dispone de un archivo de organización directa con información sobre los productos que distribuye. El archivo tiene una capacidad de 1000 registros y presenta la siguiente estructura:

Campo	Tipo de dato	Observaciones
CodProducto	Alfanumérico de 8 posiciones	Clave de acceso al registro
Nombre	Alfanumérico de 50 posiciones	
Precio	Númérico entero	
Ocupado	Lógico	Indica si el registro continúa información. Toma el valor verdad si tiene información o falso si no la tiene.

Se desea crear un índice utilizando un árbol binario de búsqueda en el que la clave sea el campo CodProducto. Cada elemento del índice contendrá el código del producto y el número de registro en el que está almacenado.

- Diseñe las estructuras de datos necesarias para realizar el problema.
- Escriba los procedimientos necesarios para crear el índice a partir de los datos del archivo.
- Escriba un procedimiento que realice un listado del archivo ordenado por código de producto.

Puntuación: 3,5 puntos

Apartado A

constantes

MaxReg = 1000

tipos

registro

cadena : clave

entero : nreg

fin_registro = TipoElemento

puntero_a nodo = arbol

registro

TipoElemento : info

Arbol : hi, hd

fin_registro : nodo

registro

cadena : CodProducto, nombre

entero : precio

lógico : ocupado

fin_registro = producto

archivo_d de producto = productos

Apartado B

procedimiento CrearIndice(**E/S** productos : A; **E/S** arbol : a)

var

producto : R

TipoElemento : e

entero : i

inicio

//Se supone que el archivo A ya está abierto

a ← nulo

desde i ← 1 **hasta** MaxReg **hacer**

leer(A,R,i)

si R.ocupado **entonces**

e.clave ← R.CodProducto

e.nreg ← i

InsertarEnArbol(a,e)

fin_si

fin_desde

fin_procedimiento

procedimiento InsertarArbol(**E/S** arbol : a ; **E** TipoElemento : e)



```
inicio
  si a = nulo entonces
    nuevo(a)
    a↑.info ← e
    a↑.hi ← nulo
    a↑.hd ← nulo
  si_no
    si a↑.info.CodProducto < e.CodProducto entonces
      InsertarArbol(a↑.hd,e)
    si_no
      InsertarArbol(a↑.hi,e)
    fin_si
  fin_si
fin_procedimiento
```

Apartado C

```
procedimiento ListarArchivo(E arbol : a ; E/S productos : A)
var
  producto : R
inicio
  //Recorre el arbol en inorden
  //Por cada nodo accedemos al número de registro indicado
  //en el campo campo nreg de la información del nodo
  si a <> nulo entonces
    ListarArchivo(a↑.hi ,A)
    leer(A,R, a↑.info.nreg)
    escribir(R.CodProducto,R.Nombre,R.Precio)
    ListarArchivo(a↑.hi ,A)
  fin_si
fin_procedimiento
```