Bely - UTNiamos

Apellido, Nombres:		Legajo:	Hoja: /
e-mail	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructuras de Datos 28 de febrero de 2009

La Empresa "SINMONEDAS", contratada por la Secretaría de Transportes de Buenos Aires para implementar el uso de tarjetas en el transporte automotor colectivo, necesita procesar la información.
Para ello, cada vehículo al finalizar su "recorrido" transmite la actividad generada por la correspondiente máquina. Los archivos a utilizar en el proceso son:

- **M) MAQUINA** ***ordenado cronológicamente,*** con los datos del recorrido de l(un) colectivo, los campos Línea e Interno poseen los mismos valores para todos los registros.
  - Línea (3 dígitos)
  - Interno (4 dígitos)
  - Tipo de tarjeta (char)
  - Número de tarjeta (9 dígitos)
  - Valor del boleto (single, máximo 8 valores posibles)
  - Fecha AAAAMMDD (longint)
  - Hora HHMMSS (longint)

R) RESUMEN con un registro por proceso de consolidación efectuado:

- Línea (3 dígitos)
- Interno (4 dígitos)
- Fecha AAAAMMDD (sacar del sistema GETDATE)
- HoraHHMM (sacar del sistema GETTIME)
- Monto total recaudado (real)
- Boletos vendidos por Valor (se repite 8 veces); completar con cero los no utilizados.
  - Valor del boleto (single)
  - Cantidad (word)

Se pide desarrollar la metodología necesaria para realizar un programa que:

1. Generar **un solo registro y agregar** al final del archivo **RESUMEN,** con los valores finales del proceso.
2. Listar la cantidad de boletos vendidos ***ordenados por tipo de tarjeta y valor del boleto,*** *ambos ascendentes.*

Listado de Cantidades de Boletos Vendidos por el Interno: 9999 Línea 999
Tipo de Tarjeta: X

Valor del Boleto	Cantidad de Boletos
99.99	99999
99.99	99999

Recursos, Restricciones y Observaciones:

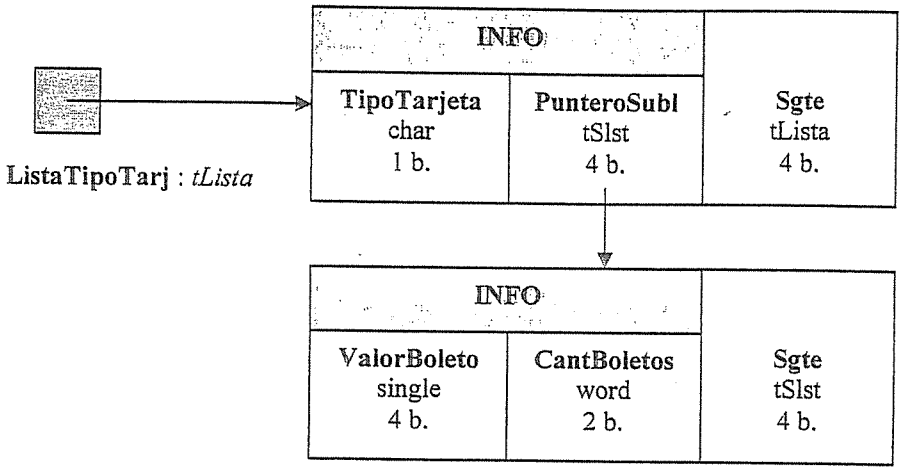
- **Memoria para arrays:** 0 bytes para arreglos auxiliares.
- **Memoria para estructuras dinámicas:** Nodos de a lo sumo 10 bytes.
- **Accesos al archivo MAQUINA:** secuencial una vez. **RESUMEN:** un solo acceso al archivo
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. **Desarrollar todos los módulos invocados.**
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pag. x de y.

ALGORITRESFIN

Apellido, Nombres:		Legajo:	Hoja:
e-mail	Docente que Firmó Libreta:		Nota:

Estructuras de datos

Lista de tipo de tarjeta y sublista por valor del boleto (punto2), ordenada por tipo de tarjeta y valor del boleto ascendente



Tamaño del nodo: 9 bytes. Un nodo por cada tipo de tarjeta diferente generado a partir de recorrer secuencialmente **MAQUINA.DAT**

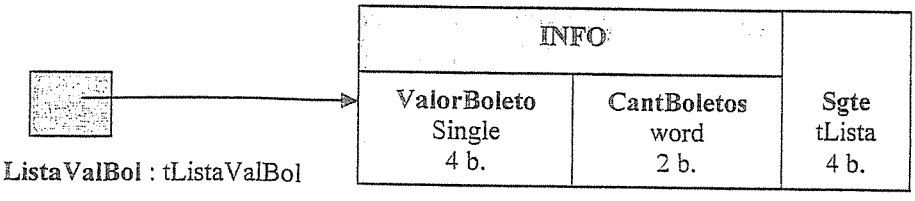
Tamaño del nodo: 10 bytes. Un nodo por cada valor de boleto diferente de pendiente de cada tipo de tarjeta generado a partir de recorrer secuencialmente **MAQUINA.DAT**

Estrategia

- Asignar y abrir archivos.
- Inicializar lista y registro de Resumen: fecha, hora, monto en cero, y array de registro con ceros.
- Recorrer secuencialmente archivo de Máquina:
  - Leer registro.
  - Recorrer array de registro de resumen buscando el valor del boleto.
    - Si no se encuentra, agregar el valor del boleto en el array o En ambos casos contar un boleto más en la posición del array o Armar registros e Insertar en lista y sublista de tipo de tarjetas, y contar un boleto más.
  - Acumular valor del boleto en campo monto del registro de Resumen.
- Completar datos en registro de Resumen (línea, interno).
- Posicionarse al final del archivo con seek de filé sise del archivo Resumen.
- Grabar registro en archivo resumen (punto 1).
- Listar contenido de lista y sublista (punto 2).
- Liberar lista.
- Cerrar archivos.

Otras opciones para el punto 1 :
*1) pueden usar una lista para acumular cantidad de boletos por valor del boleto, y al final asignar los valores finales al array del registro

Lista por valor de boleto



T.N.: 10 bytes.

2) al final del proceso a partir de las sublistas podrían ir acumulando en el array las cantidades de boletos por valor del boleto.

Apellido, Nombres:		Legajo:	Hoja: /
e-mail:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructura de Datos 7 de marzo de 2009

La empresa que administra los peajes en la Ciudad Autónoma de Bs. As. desea obtener las franjas horarias con mayor circulación de vehículos del último año. Para ello cuenta con la siguiente información:

Un archivo binario, **CIRCULACION.Dat,** ***ordenado por fecha y hora,*** de un año, con el siguiente diseño:

Fecha (mmdd, word)	Hora (hhmmss, longint)	Importe abonado (single)
Tipo de vehículo (1..5, byte)	Tipo de paga('P': pase, 'E': efectivo, char)	

Se pide desarrollar la metodología necesaria para **emitir** el siguiente **listado anual,** con *orden descendente por mayor cantidad de vehículos que circularon en las franjas horarias,* y dentro *de las franjas horarias ascendente por tipo de vehículo y fecha:*

Listado de franjas horarias ordenadas por mayor circulación de vehículos en el último año
Franja horaria de 99.00 a 99.59
Tipo de vehículo: Motocicletas **Cantidad total de Vehículos: 999999999**

Fecha	Cantidad de Vehículos
dd-mm	999999999
dd-mm	999999999

.....

Tipo de vehículo: Vehículos de hasta 2 ejes **Cantidad total de Vehículos: 999999999**

Fecha	Cantidad de Vehículos
dd-mm	999999999
dd-mm	999999999

Cantidad Total vehículos en la franja horaria: 99999999

.....

Recaudación Anual en Pases: 9999999.99
Recaudación Anual en Efectivo: 9999999.99

Las franjas horarias son de una hora reloj, por ejemplo: de 0.00 a 0.59, de 1.00 a 1.59..... de 12.00 a 12.59,... y de 23.00 a 23.59.

Los tipos de vehículos están clasificados de la siguiente manera: 1 : Motocicletas, 2: Vehículos de hasta 2 ejes, 3: Vehículos de más de 2 ejes y hasta 4 ejes, 4: Vehículos de más de 4 ejes y hasta 6 ejes, y 5: Vehículos de más de 6 ejes.

Recursos, Restricciones y Observaciones:

- **Memoria para arrays:** 600 bytes.
- **Memoria para estructuras dinámicas:** Nodo de 10 bytes
- **Memoria en disco:** 0 bytes
- **Accesos a disco: 1 solo** acceso a cada registro del archivo.
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** solo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos, afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. **Desarrollar todos los módulos invocados.**
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pag. x de y.

Apellido, Nombres:		Legajo:	Hoja: /
e-mail:	Docente que Firmó Libreta:		Nota:

Estructuras de datos

El final se puede desarrollar con diferentes tipos de estructuras debido a que no tiene restricciones en cantidad de nodos:

Opciones:

1. Matriz de franjas horarias y tipo de vehículo de listas con un nodo por día con fecha y cantidad acumulada de vehículos, o con un nodo por cada registro con fecha, cada nodo es un vehículo, pero luego tendrán que contabilizar los vehículos.
2. Array de franjas horarias de listas con un nodo por tipo de vehículo y día como una sola clave acumulando cantidad de vehículos, o donde cada nodo es un vehículo, pero luego tendrán que contabilizar los vehículos.
3. Array de listas de 120 elementos donde cada 5 elementos es una franja horaria, deberán hacer cuentas para ubicarse en el array según franja horaria y tipo de vehículo. Cada nodo contendrá fecha y cantidad acumulada de vehículos, o un nodo por cada registro con fecha, cada nodo es un vehículo, pero luego tendrán que contabilizar los vehículos.
4. Una sola lista con un nodo por cada vehículo con hora, tipo y fecha, o franja horaria, tipo y fecha donde cada nodo es un vehículo, pero luego deberán sumar y realizar corte de control en la lista.
5. Lista de franja horaria y tipo de vehículo con sub-lista acumulando por fecha, o con un nodo por cada vehículo.

EN NINGÚN CASO PUEDEN GUARDAR POSICIÓN AL ARCHIVO EN NINGUNA ESTRUCTURA

EN TODOS LOS CASOS DEBERÁN ORDENAR LAS FRANJAS HORARIAS POR MAYOR CIRCULACIÓN PARA PODER LISTAR, para ello pueden usar un array de 24 elementos o una lista asociada a la otra estructura.

Punteros a listas ordenadas por fecha tListaFec 4 b.	Tipos de Vehículos				
Franjas Horarias	1	2	3	4	5
0	/	/	/	/	
1	/	/	/	/	/
2					
.					
.					
23	/	/	/	/	/

mFHoraTVeh[i, j]
M.E.: 24 x 5 x 4b. = 480 bytes.

INFO		
Fecha mmdd 2b.	CantVehxDia longint 4b.	Sgte tListaFec 4b.

T.N.: 10 bytes.

Se genera un nodo por cada registro leído de CIRCULACION.DAT sin repetir a igual fecha.

Franjas Horarias	FranjaHora byte 1b.	TotVehxFHora longint 4b.
0	0	0
1	1	0
.	.	.
23	23	0

vrtVehxHora[i]
M.E. total: 480b. + 120b.= 600 bytes.

Apellido, Nombres:		Legajo:	Hoja: /
e-mail:	Docente que Firmó Libreta:		Nota:

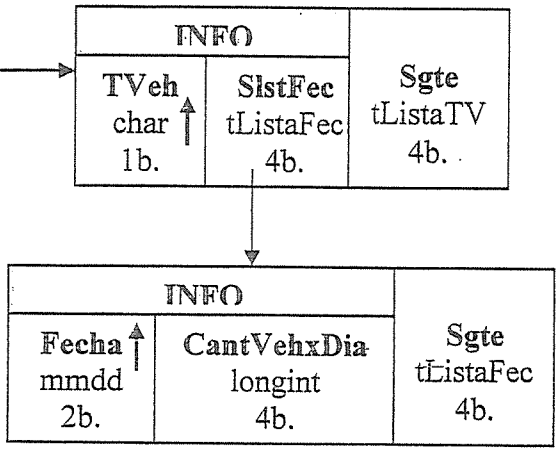
ALTERNATIVA:

T.N.: 9 bytes.
Se genera 1 nodo por cada tipo de vehículo sin repetir, para cada franja horaria., colgado en la franja horaria correspondiente al reg. leído; máximo 5 nodos.

Franjas Horarias

	FranjaHora byte 1b.	TotVehxFHora longint 4b.	ListaTVeh tListaTV 4b.
0	0	0	
1	1	0	/
.			
.			
23	23	0	/

vrFHora : tvrFHora
M.E.: 24 x (1b.+ 4b.+ 4b.) = **216 bytes.**
Luego de procesar los registros de Circulacion, se ordena por el campo **TotVehxFHora** decreciente.



Observación: Los nodos de la lista principal podrían **pre-armarse**, en ese caso el campo **TVeh** no sería necesario, ya que la posición ordinal indicaría el tipo de vehículo.

T.N. 10 bytes.
Se genera 1 sub-nodo por cada reg. leído colgado de la franja horaria y del tipo de vehículo correspondiente, sin repetir a = fecha.

Estrategia

- Abrir archivo.
- Inicializar matriz, array de totales, totales para recaudación.
- Recorrer archivo secuencialmente y por cada registro.
  - Leer registro.
  - Obtener franja horaria un valor de 1 a 24 o de 0 a 23.
  - insertar o buscar nodo con la misma fecha para contar un vehículo mas.
  - Sumar 1 a contador de vehículo en array de totales según franja horaria.
  - Acumular el importe abonado según tipo de pago en totales para recaudación.
- Ordenar array de totales.
- Emitir listado.
  - Recorrer array de totales ordenado y por cada elemento.
    - Ubicarse en la fila de la matriz.
    - Listar la franja horaria y los nodos según diseño.
  - mostrar totales generales
- cerrar archivo.

Apellido, Nombres:			Legajo:	Hoja: /
Aula:	e-mail:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructuras de Datos – 7 de octubre de 2009

Se desea publicar una nueva edición de la GUIAFILCAR de calles de la Capital Federal, para ello, se cuenta con los siguientes archivos:

a) CALLES.DAT, *ordenado alfabéticamente por nombre de la calle y altura desde*, con uno o varios registros con la misma calle pero con diferentes intervalos de alturas, con el siguiente diseño:
   - a.1) Nombre de la calle (25 caracteres)
   - a.2) Código de calle (longint)
   - a.3) Altura desde (Word)
   - a.4) Altura hasta (Word)
   - a.5) Ubicación en el mapa
     - a.5.1) Página (byte)
     - a.5.2) Fila (char)
     - a.5.3) Columna (byte)

b) CODIGOPOSTAL.DAT, ***sin orden***, con uno o varios registros con la misma calle, pero con diferentes intervalos de alturas y código postales, con el siguiente diseño:
   - b.1) Código Postal (Word)
   - b.2) Código de calle (longint)
   - b.3) Altura desde (Word)
   - b.4) Altura hasta (Word)

Se pide desarrollar la metodología necesaria para realizar un programa que emita la guía con el mismo orden del archivo de calles, según el siguiente diseño:


CAPITAL FEDERAL

Indice Alfabético de Calles - Códigos Postales - Alturas

AZURDUY Juana
1501/3400 cp 1429 — 3401/3700 cp 1430
1501 - 2500 ... 49 - B - 2
2501 - 2950 ... 57 - I - 5
2951 - 3700 ... 56 - G - 6

BACACAY
3401/4050 cp 1405 — 1601/3300 cp 1406
3501/5700 cp 1407 — 5701/5900 cp 1408
601 - 1000 ... 46 - E - 1
1001 - 1050 ... 46 - D - 6
1601 - 2150 ... 45 - A - 5
2151 - 5200 ... 58 - F - 4
5201 - 5900 ... 60 - I - 3

BACH Juan Sebastián ... cp 1430
1301 - 4500 ... 57 - D - 2

BACILE César H. ... cp 1407
3801 - 3900 ... 53 - E - 2

BARRAGAN ... cp 1406
1 - 1200 ... 60 - D - 5

BARRIENTOS ... cp 1115
1501 - 1600 ... 40 - E - 2

BARROS PAZOS José
1601/3700 cp 1437 — 3701/7000 cp 1439
1601 - 2800 ... 44 - F - 4
2801 - 3600 ... 44 - C - 1
3601 - 4100 ... 43 - B - 4
4101 - 4300 ... 51 - I - 5
5401 - 6600 ... 51 - F - 2
6601 - 7000 ... 50 - D - 6

BARZANA
1001/1400 cp 1427 — 1401/2200 cp 1431
1001 - 2200 ... 55 - F - 1

BASAVILBASO D. de ... cp 1006
201 - 1400 ... 34 - B - 5

BASAVILBASO Leopoldo ... cp 1428
701 - 900 ... 49 - F - 2

BASUALDO ... 1/400 cp 1408
401/2390 cp 1440 — 2301/3300 cp 1439

1 - 300 ... 53 - D - 2

BENIELLI Carlos Javier ... cp 1440
7051 - 7100 ... 59 - E - 1

BERG Carlos ... cp 1437
2701 - 3550 ... 44 - F - 2
3551 - 3650 ... 43 - G - 6

BERGANTIN
CONGRESO NACIONAL ... cp 1439
6051 - 6100 ... 50 - E - 6

BERGANTIN ECHAGÜE ... cp 1439
5901 - 6000 ... 51 - F - 1

BERGANTIN
GENERAL BALCARCE ... cp 1439
5901 - 6100 ... 51 - F - 1

BERGANTIN
GENERAL BELGRANO ... cp 1439
5901 - 6100 ... 51 - F - 1

BERGANTIN
SOBERA GRAL. ESPORA ... cp 1439
s/n ... 51 - B - 2

BERGANTIN NANCY ... cp 1439
5151 - 5250 ... 50 - G - 6

Recursos. Restricciones y Observaciones:

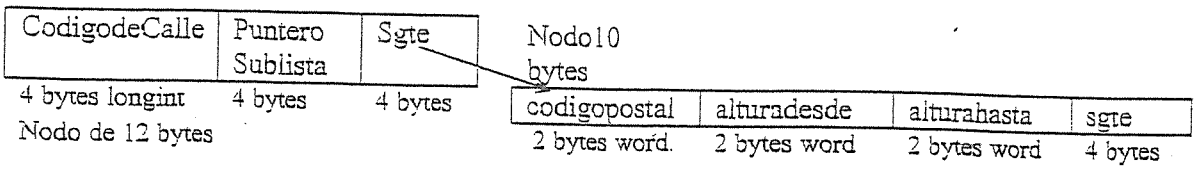
- **Memoria para arrays:** 0 bytes para arreglos auxiliares - **Espacio en disco:** 0 bytes
- **Memoria para estructuras dinámicas:** nodos de a lo sumo 12 bytes para almacenar la información del archivo CODIGOPOSTAL
- **Accesos al archivo:** un solo recorrido secuencial a cada archivo
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. Desarrollar todos los módulos invocados.
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

ALGORITRESFIN

Apellido: Nombres:		Legajo:	Hoja: /
Aula:	e-mail:	Docente que Firmó Libreta:	Nota:

Estructuras

Lista de calles con sublista de códigos postales



Estrategia

- Asignar y abrir archivos
- Inicilizar lista
- Recorrer archivo de código postales y cargar lista y sublista
- Informar título
- Recorrer secuencialmente archivo de Calles con corte de control por nombre de calle
  - Leer registro con lectura anticipada
  - Informar nombre de la calle
  - Ubicar código de calle en lista
  - Recorrer sublista y por cada nodo (podrían ir liberando sublista)
    - informar la alturaD-H/código postal
  - Mientras sea la misma calle y no fin de archivo
    - Informar del registro de calles alturaD-H, página, fila, columna
    - Leer siguiente registro
- Liberar lista
- Cerrar archivos

Apellido, Nombres:		Legajo:	Hoja: /
e-mail:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructuras de Datos 22 de agosto de 2009

La Empresa de "GASNATURAL" incrementó el valor del metro cúbico de gas natural para la demanda interna. Para la aplicación de estas medidas, se crearon nuevas categorías. Estas se definen en cada período de facturación, tomando el último año (6 últimos consumos bimestrales). Al tomar el total de consumos del período, es posible que un cliente pueda cambiar de categoría si varía el consumo entre un período anual y otro.

Para ello, se cuenta con los siguientes archivos:

C) CATEGORIAS *ordenado por m^3 desde*, con un registro por cada categoría y el intervalo de consumo correspondiente a esa categoría, con el siguiente diseño:

- Categoría (3 caracteres, 8 categorías)
- M^3 anual desde (4 dígitos)
- M^3 anual hasta (4 dígitos

T) CLIENTES *ordenado por código de cliente*, con un registro por cliente y el siguiente diseño:

- Código de cliente (8 dígitos)
- Datos del titular (15 caracteres)
- Categoría anterior(3 caracteres)
- Lectura anterior del medidor (4 dígitos)
- Historial de consumos, últimos 6 bimestres con los siguientes datos.
  - Mes y Año del bimestre (mmaaaa)
  - M^3 consumidos del bimestre (4 dígitos)

M) MEDICIONES *sin orden*, cada registro contiene la lectura actual de cada medidor con el siguiente diseño:

- Código de cliente (8 dígitos)
- Lectura actual del medidor (4 dígitos)
- Fecha de la medición (ddmmaaaa)

Se pide desarrollar la metodología necesaria para realizar un programa que calcule el consumo del bimestre actual, a partir de los consumos de los bimestres anteriores y el actual se recategorice a los clientes, y realice los siguientes procesos:

1. Aplique pero no desarrolle la siguiente función que retorna el nuevo consumo anual
   CálculoConsumoAnual(HistorialConsumo:thistorial; ConsumoBimestre:word):word;

2. Emitir el siguiente listado *agrupado por categoría*, de aquellos clientes que cambiaron de categoría.

   Listado de clientes con cambios de categoría agrupado por categoría
   Categoría: XXX

Código de cliente	Categoría anterior
999999999	XXX
999999999	XXX

3. Actualice los siguientes campos en el archivo de clientes: la categoría (si fue modificada), lectura anterior del medidor con lectura actual; y el historial de consumos de los últimos 6 bimestres, eliminado el primer bimestre y agregando al final el nuevo bimestre.

Recursos, Restricciones y Observaciones:

- **Memoria para arrays:** 96 bytes para arreglos auxiliares.
- **Memoria para estructuras dinámicas:** Nodos de 12 bytes, uno por cada cliente que cambie de categoría.
- **Accesos al archivo** CATEGORIA y MEDICIONES: secuencial una vez.
  CLIENTES: búsqueda binaria+ 1 acceso para actualizar a los clientes.
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. Desarrollar todos los módulos invocados.
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

Apellido, Nombres:		Legajo:	Hoja: /
e-mail:	Docente que Firmó Libreta:		Nota:

Estructuras

Array de categorías y puntero a pila con cliente que cambia la categoría

	Categoria	M^3 anual desde	M^3 anual hasta	Pila
1				
2				
4				
8	4 bytes longint	2 bytes Word	2 bytes Word	4 bytes

Nodo12 bytes

codigocliente	Categanterior	sgte
4 bytes longint	4 bytes string	4 bytes

Memoria para array (4+2+2+4) * 8 = 12 * 8= 96 bytes

Estrategia

- Asignar y abrir archivos
- Recorrer archivo de categorías y cargar array e inicializar puntero a las pilas
- Recorrer secuencialmente archivo de Mediciones:
  - Leer registro
  - Busqueda binaria en archivo de clientes
  - Calcular consumo actual como Lectura actual – lectura anterior
  - Invocar a la función calculoconsumoanual con el historial y el nuevo consumo
  - Recorrer array de registro buscando la nueva categoría de acuerdo al consumo anual según el intervalo desde-hasta
  - Si la nueva categoría es diferente a la categoría anterior, armar registro nodo y meter nodo en la pila según categoría y modificar campo categoría del registro de cliente
  - Modificar registro de clientes con lectura actual, realizar corrimiento en el array de historial de consumos, perdiendo el primer valor y en el sexto elemento poner el consumo actual. Posicionarse y Grabar registro cliente
- Recorrer array de registro de categorías y emitir listado de acuerdo punto 1
- Cerrar archivos

Apellido, Nombres:			Legajo:	Hoja: /
e-mail:	Aula:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructura de Datos 24 de julio de 2010

La Agrupación de cines argentinos desea obtener información de la concurrencia a los cines de las 10 películas con mayor cantidad de espectadores. Para ello cuenta con dos archivos binarios:

P) **PELICULAS.DAT**, ***con un registro por cada película*** que está en cartelera, según el siguiente diseño, ***ordenado creciente por código de película***:

Código de Película (6 dígitos, longint)	Título de la Película (25 caracteres)
Semanas en cartel (byte)	Acumulado de espectadores (8 dígitos)

S) **SEMANAL.DAT,** que contiene información sobre la concurrencia de espectadores en cada cine durante los días jueves a domingo de la última semana, ***con un registro por cada cine***, ***sin orden*** y con el siguiente diseño:

Código de Película (6 dígitos, longint)	Cantidad de espectadores (8 dígitos)

Se pide desarrollar la metodología necesaria para realizar un algoritmo que:

1) Emita el siguiente listado con las 10 películas con mayor concurrencia, ordenado en forma decreciente por cantidad de espectadores en la última semana.

Concurrencia a los cines

		Espectadores	
Película	Salas	Jueves a domingo	Acumulado
1 TOY STORY 3 Semanas en cartel: 2	108	320.678	1.094.667
2 KARATE KID Semanas en cartel: 1	82	49.192	49.192
3 EL PRINCIPE DE PERSIA Semanas en cartel: 5	107	27.115	701.638
4 BRIGADA A: LOS MAGNIFICOS Semanas en cartel: 1	63	25.229	25.229
5 LA CARRETERA Semanas en cartel: 1	38	19.578	19.578
6 CARANCHO Semanas en cartel: 8	68	16.928	579.697
7 SEX AND THE CITY 2 Semanas en cartel: 4	67	14.784	255.423
8 ROBIN HOOD Semanas en cartel: 7	40	8828	704.472
9 NEW YORK, I LOVE YOU Semanas en cartel: 2	15	7.367	24.169
10 CARTAS A JULIETA Semanas en cartel: 3	19	6424	46.695
Total de espectadores de jueves a domingo		[illegible]	

2) Actualice los registros del archivo **PELICULAS.DAT**, con la nueva información que se encuentra en el archivo **SEMANAL.dat**

🗁 **Recursos, Restricciones y Observaciones**:

- **Memoria para arrays: 360** bytes.
- **Memoria para estructuras dinámicas: nodos** de 13 bytes * **cada película**.
- **Accesos a archivos:** un solo recorrido secuencial para cada archivo, y otro acceso por cada registro del archivo de **PELICULAS.DAT. No utilizar búsqueda binaria.**
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. **Desarrollar todos los módulos invocados**.
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

o, Nombres:			Legajo:	Hoja: /
il:	Aula:	Docente que Firmó Libreta:		Nota:

Lista películas con espectadores de la semana

info			
Código de película	Cantidad de cines	Espectadores de la semana	

4 + 1+4+4= 13 bytes

Vector 10 mejores

	Título	Semanas en cartel	Cines	Espectadores	Acumulado
1	26 bytes	1 byte	1 byte	4 bytes	4 bytes
2					
...					
10					

10 *(26+1+1+4+4)=10 * 36 =360 bytes

Estrategia

- Abrir archivos
- Inicializar lista y vector
- Recorrer archivo semanal y insertar en lista acumulando espectadores y contando cines, ordenado por código de película
- Recorrer simultáneamente o con apareo el archivo de películas y la lista, y por cada película
  - Actualizar acumulado de espectadores y incrementar en 1 la cantidad de semanas en el registro del archivo de películas y grabar registro(punto2)
  - Insertar en vector la películas si esta dentro de las 10 mejores(punto 1)
- Mostrar vector según diseño y orden punto1
- Liberar lista
- cerrar archivos

Apellido, Nombres:			Legajo:	Hoja: /
e-mail:	Aula:	Docente que firmó Libreta:		Nota:

Final Algoritmos y Estructuras de Datos 19 de Febrero de 2011

Se requiere evaluar a un conjunto de estudiantes mediante 5 exámenes parciales de manera que cada examen consta de 20 ítems a responder.
Cada ítem vale un punto por lo que cada examen puede tener un puntaje máximo de 20 puntos y el máximo puntaje a lograr por un alumno es de 100 puntos.

Se cuenta con la siguiente información:
a) Un archivo binario ***'Resultados.dat'***, con 5 registros, uno por cada examen con el siguiente diseño:

a.1 código de examen (4 caracteres)
a.2 20 (veinte) valores de tipo byte con la respuesta correcta (1..4) de cada ítem del examen.

b) Un segundo archivo ***'Examenes.dat'***, con los exámenes rendidos por los alumnos, con un registro por cada ítem respondido, ***ordenado por código de examen y DNI del alumno***, con el siguiente diseño:

b.1 código de examen (4 caracteres)
b.2 DNI del alumno (longint)
b.3 número de ítem (1..20, byte)
b.4 opción elegida (1..4, byte)

Se pide: Diseñar las estructuras, la estrategia y el algoritmo que emita el siguiente listado, ***ordenado en forma creciente por DNI.***

Listado de calificaciones

N° orden	DNI	Notas					
		1° examen	2° examen	3° examen	4° examen	5° examen	Nota final
1	1111	19/20	20/20	18/20	20/20	20/20	97/100
2	1150	19/20	18/20	19/20	20/20	20/20	96/100
3	2222	20/20	17/20	18/20	20/20	19/20	94/100
4.....							
.....							
.....							

Recursos y Restricciones:

- **Memoria para arrays**: 125 bytes.
- **Memoria para estructuras dinámicas**: nodos de hasta 14 bytes.
- **Memoria en disco**: No hay lugar disponible para crear archivos.
- **Accesos a disco**: Un acceso a cada registro de los archivos.

Observaciones

- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. **Desarrollar todos los módulos invocados**.
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

Alumno.DAT

CodEx	RtaCor [1..20]
Entero [4]	1 Byte x 20
5 bytes	20 bytes

Examen.DAT Ordenado por código de examen y DNI del alumno

CodEx	DNI	NroItem	Opción
Entero [4]	LongInt	Byte	Byte
5 bytes	4 bytes	1 byte	1 byte

Memoria =

VecRes

	CodEx	RtaCor [1..20]
1		
↓		
20		

[Manejo] 1 byte x 20

(1 byte + 20 bytes) * 5 = 125 bytes

Lista

nro		
DNI	SubL	Sig
LongInt	^	^

= 4 bytes + 4 bytes + 4 bytes = 12 bytes

SubL =

nro		
CodEx	NroItem	Opción
[Manejo]	Byte	Byte
5 bytes	1 byte	1 byte

Sig: 4 bytes

= 11 bytes

©

Abrir Archivos

ArchRes1 ; ArchExam

Procesar Resultados

ArchRes1 ; VecRes

Procesar Examenes

Lista ; ArchExam

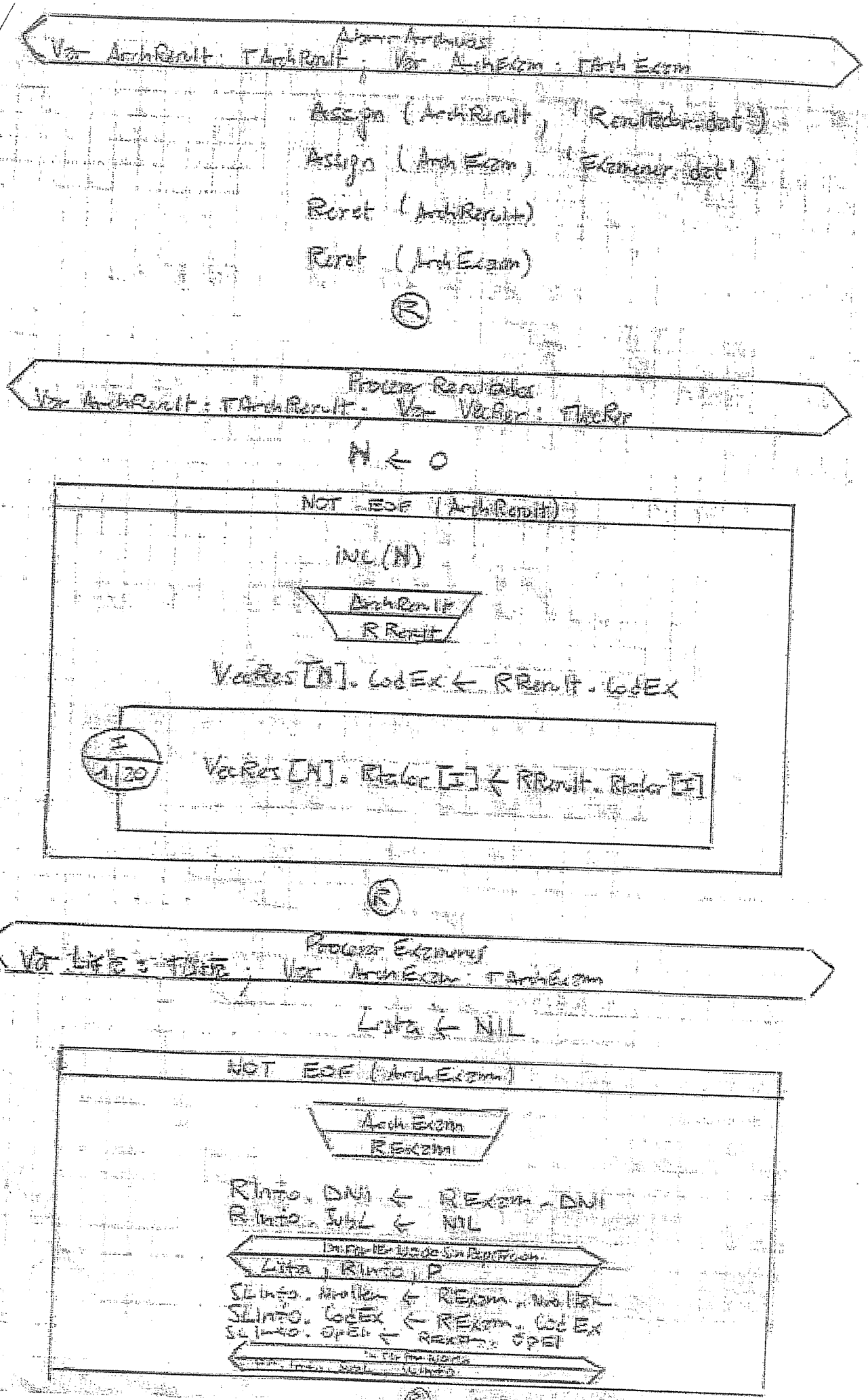
Mostrar Listado

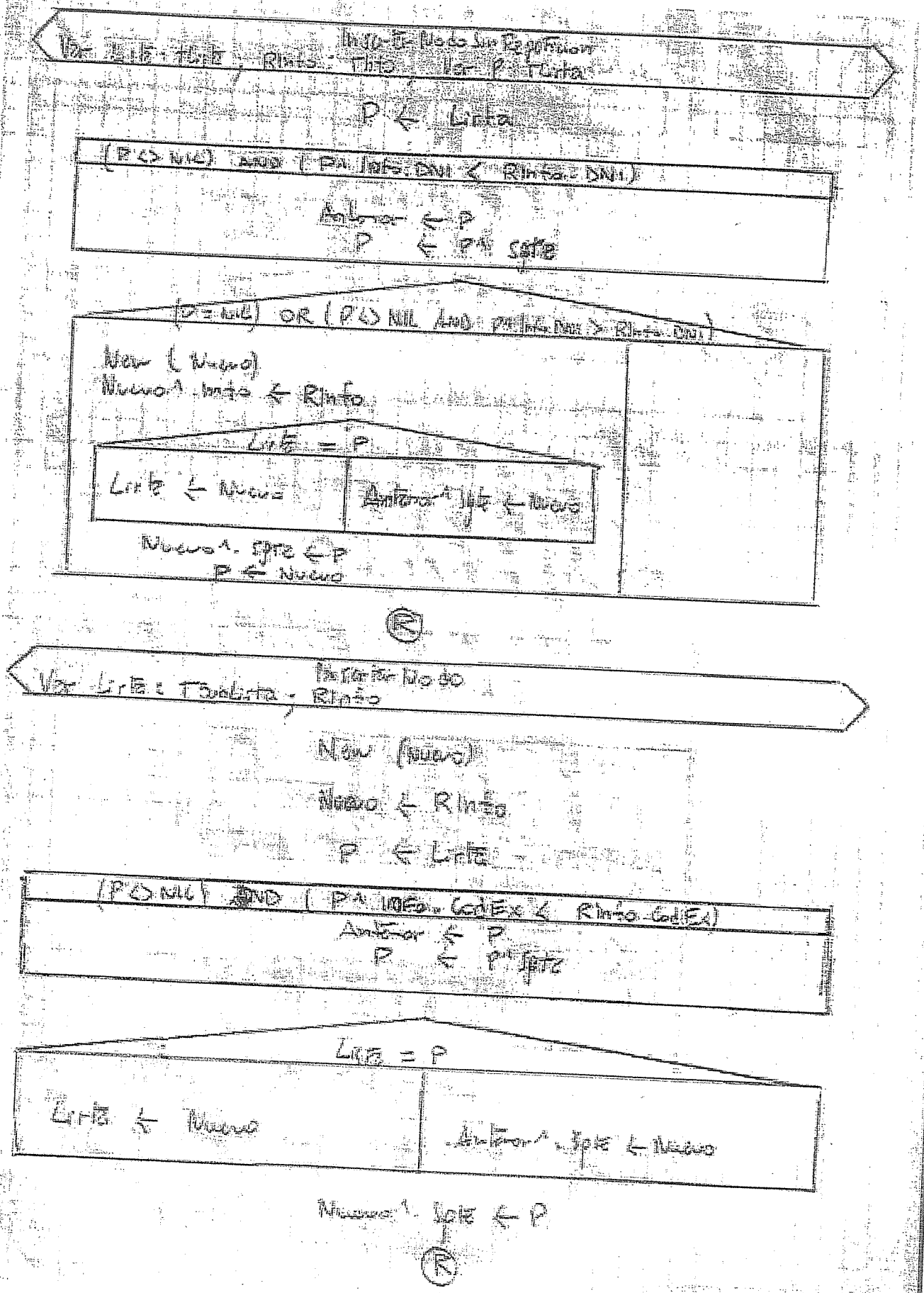
Lista ; VecRes

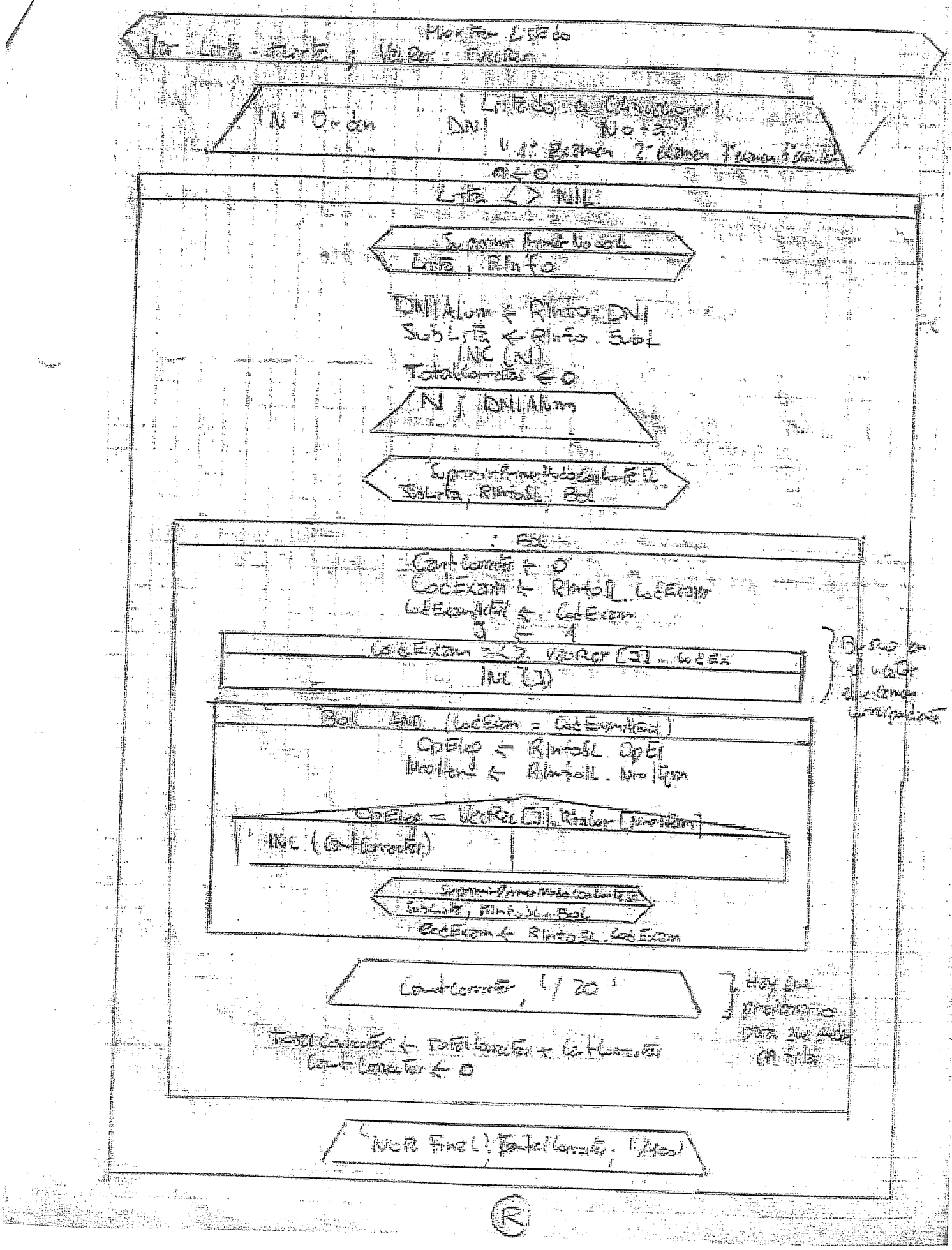
Cerrar Archivos

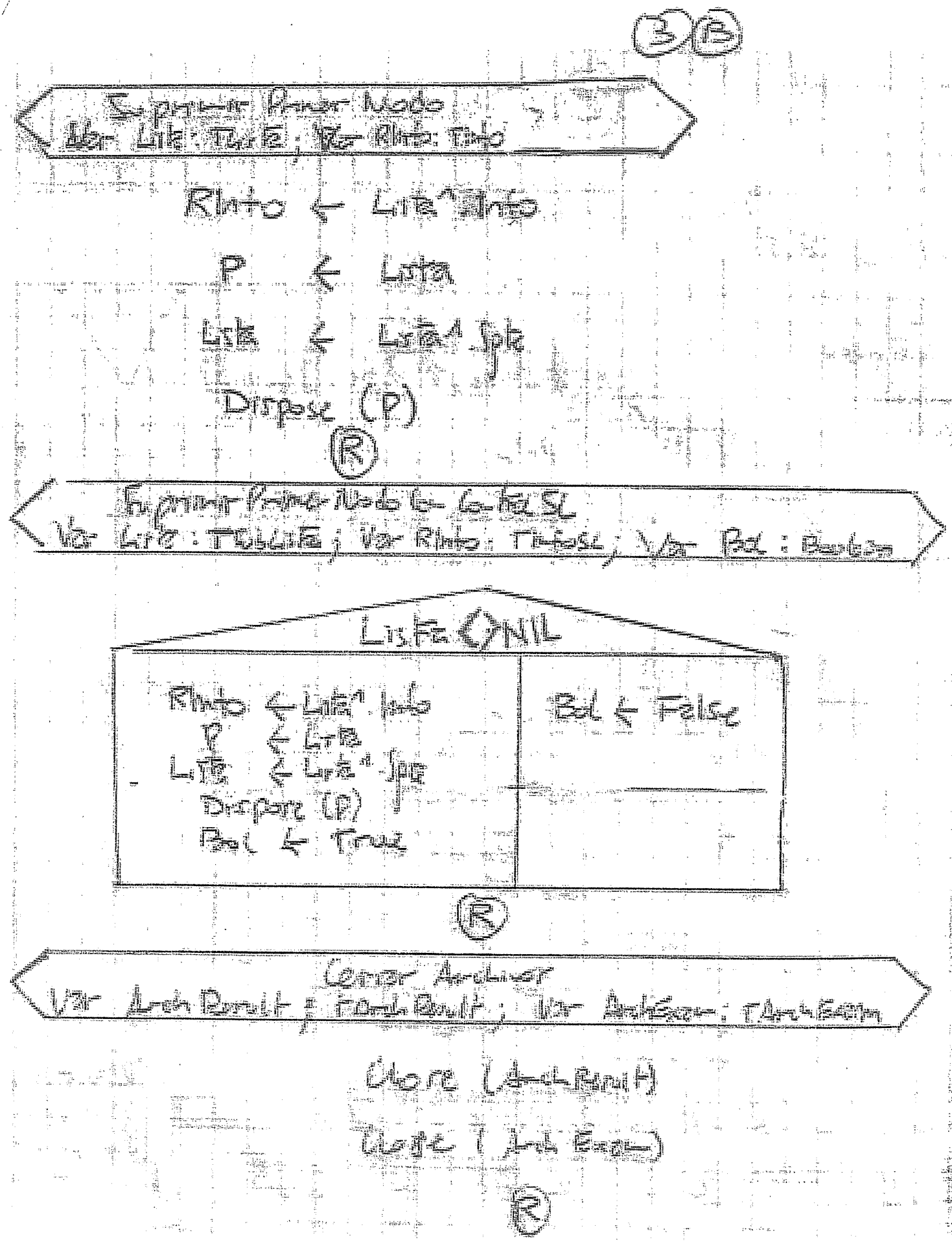
ArchRes1 ; ArchExam

Ⓕ


Abrir Archivos
Var ArchResult : TArchResult ; Var ArchExam : TArchExam
Assign (ArchResult , 'Resultados.dat')
Assign (ArchExam , 'Examenes.dat')
Reset (ArchResult)
Reset (ArchExam)
R
Procesar Resultados
Var ArchResult : TArchResult ; Var VecRes : TVecRes
N ← 0
NOT EOF (ArchResult)
INC (N)
ArchResult
RResult
VecRes [N]. CodEx ← RResult . CodEx
I
1 | 20
VecRes [N]. ResCor [I] ← RResult . Resp [I]
R
Procesar Examenes
Var Lista : TLista ; Var ArchExam : TArchExam
Lista ← NIL
NOT EOF (ArchExam)
ArchExam
RExam
RNuevo . DNI ← RExam . DNI
RNuevo . Sig ← NIL
Lista , RNuevo , P
SLNuevo . NroIns ← RExam . NroIns
SLNuevo . CodEx ← RExam . CodEx
SLNuevo . Opci ← RExam . Opci
R


Insertar Nodo sin Repetirse
Var Lista: TipoLista; Rinfo: TipoInfo
P ← Lista
(P <> NIL) AND (P^.Info.DNI < Rinfo.DNI)
Anterior ← P
P ← P^.Sgte
(P = NIL) OR (P <> NIL AND P^.Info.DNI > Rinfo.DNI)
New (Nuevo)
Nuevo^.Info ← Rinfo
Lista = P
Lista ← Nuevo
Anterior^.Sgte ← Nuevo
Nuevo^.Sgte ← P
P ← Nuevo
R
Insertar Nodo
Var Lista: TipoLista; Rinfo
New (Nuevo)
Nuevo ← Rinfo
P ← Lista
(P <> NIL) AND (P^.Info.Codigo < Rinfo.Codigo)
Anterior ← P
P ← P^.Sgte
Lista = P
Lista ← Nuevo
Anterior^.Sgte ← Nuevo
Nuevo^.Sgte ← P
R




Function Primer Nodo
Var Lis: TipoLis; Var Nro: Tipo
Nro ← Lis↑.Info
P ← Lis
Lis ← Lis↑.Sig
Dispose (P)
Function PrimerNodoLista
Var Lis: TipoLis; Var Nro: Tipo; Var Bol: Boolean
Lis.Fin ONIL
Nro ← Lis↑.Info
P ← Lis
Lis ← Lis↑.Sig
Dispose (P)
Bol ← True
Bol ← False
Crear Arbol
Var ArbolDer: TipoArbol; Var ArbolIzq: TipoArbol
Dispose (ArbolDer)
Dispose (ArbolIzq)

Apellido, Nombres:			Legajo:	Hojas:
e-mail:	Aula:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructura de Datos 26/02/2011 U.T.N. F.R.B.A.

Una empresa lleva el registro de entrada y salida de sus empleados, por tres diferentes puertas principales, y almacena los movimientos de un mes en un archivo binario, con el siguiente diseño: **EntradaSalida.Dat**, con un registro por cada movimiento, ***ordenado por número de puerta, fecha y hora***:

1. **ID de puerta** (1,2,3)
2. **Fecha** (aaaammdd)
3. **Hora** (hhmm)
4. **Evento** ('E':entrada, 'S':salida)
5. **Legajo empleado** (1..1000)

Los empleados pueden entrar o salir por cualquiera de las tres puertas principales, y varias veces al día. Toda entrada registrada en un día tiene registrada la salida, no hay inconsistencias.
También posee otro archivo **Empleados.Dat,** con un registro por cada empleado, máximo 1000 empleados, ***ordenado alfabéticamente por apellido y nombre*** y con el siguiente diseño:

1. **Legajo** (1..1000)
2. **Apellido y nombre** (25 caract)
3. **Departamento** (15 caract)

Se pide:

1. Obtener la fecha del sistema.
2. Desarrollar la metodología necesaria para realizar un algoritmo que emita el siguiente reporte, ***ordenado alfabéticamente por apellido y nombre*** **del empleado, fecha y hora**:

Reporte del e-lock
Fecha del reporte: dd/mm/aaaa
Total General entradas/salidas Puerta 1 9999
Total General entradas/salidas Puerta 2 9999
Total General entradas/salidas Puerta 3 9999
Apellido y Nombre: *Perez, Juan* **Departamento:** *Sistemas*

Fecha	Hora	Evento	ID puerta
08/07/2008	14:49	Entrada	1
08/07/2008	18:15	Salida	2
10/07/2008	8:30	Entrada	2
10/07/2008	15:10	Salida	3
14/07/2008	8:26	Entrada	3
14/07/2008	15:09	Salida	1
15/07/2008	8:24	Entrada	1
15/07/2008	10:32	Salida	2
15/07/2008	11:40	Entrada	2
15/07/2008	13:13	Salida	3

Cantidad dias trabajados de *Juan Perez*: 99

Recursos, Restricciones y Observaciones:

- **Memoria para arrays: 4000 bytes + 6 bytes.**
- **Memoria para estructuras dinámicas: nodos de 12 bytes.**
- **Accesos a archivos:** un solo recorrido secuencial para cada archivo
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos.
- **Desarrollar todos los módulos invocados.**
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

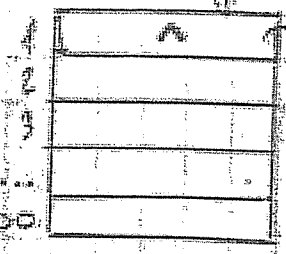
lida.dat Reg x Hor Ordenado por NroDePuerta, Fecha y Hora

a	Fecha	Hora	Evento	Leg
	Longint	Word	Char	NuSerie
E	4 bytes	2 bytes	1 byte	2 bytes

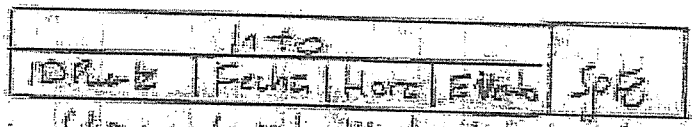
pleador.dat Hasta 1000 Empleador, Ordenado por Apellido y Nom

	AyN	Dep
Fd	String[25]	Byte[1..5]

cion = VecLeg

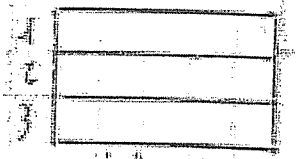


→ Lista



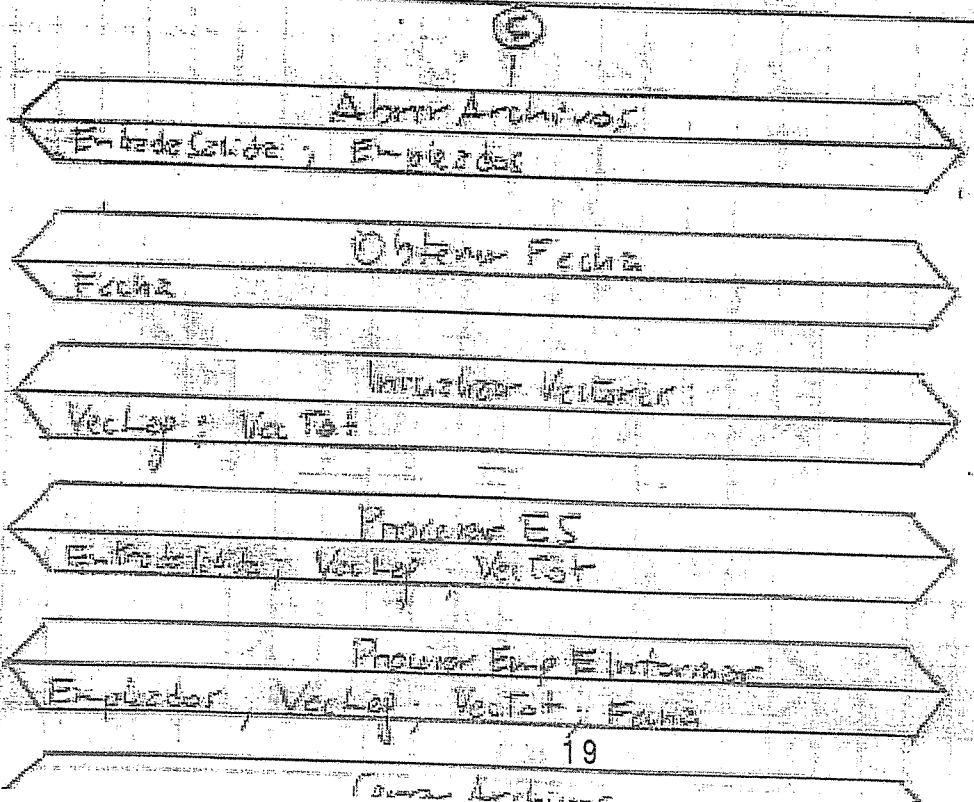
4 bytes * 1000 = 4000 bytes

VecTot



Word

2 bytes * 3 = 6 bytes



Abrir Archivos

Var ArchES : FArchES ; Var ArchEmp : FArchEmp

Assign (ArchES, 'Entrada Salida.dat')
Assign (ArchEmp, 'Empleados.dat')
Reset (ArchES)
Reset (ArchEmp)

Ⓡ

ObtenerFecha

Var Fecha : String[10]

GetDate (A, M, D, DS)

Fecha ← 'D'/'M'/'A...

Ⓡ

InicializarVectores

Var VecLog : TVecLog ; Var VecTot : TVecTot

I
1 .. 100

VecLog [I] ← NIL

J
1 .. 3

VecTot [I] ← 0

Ⓡ

ProcesarEntradasSalidas

Var ArchES : FArchES ; Var VecLog : TVecLog ; Var VecTot : TVecTot

NOT EOF (ArchES)

ArchES
RegES

LogEmp ← RegES.Log
RInfo.IDPuerta ← RegES.IDPuerta
RInfo.Fecha ← RegES.Fecha
RInfo.Hora ← RegES.Hora
RInfo.Evento ← RegES.Evento

INC (VecTot [RegES.IDPuerta]) } Suma 1 movimiento

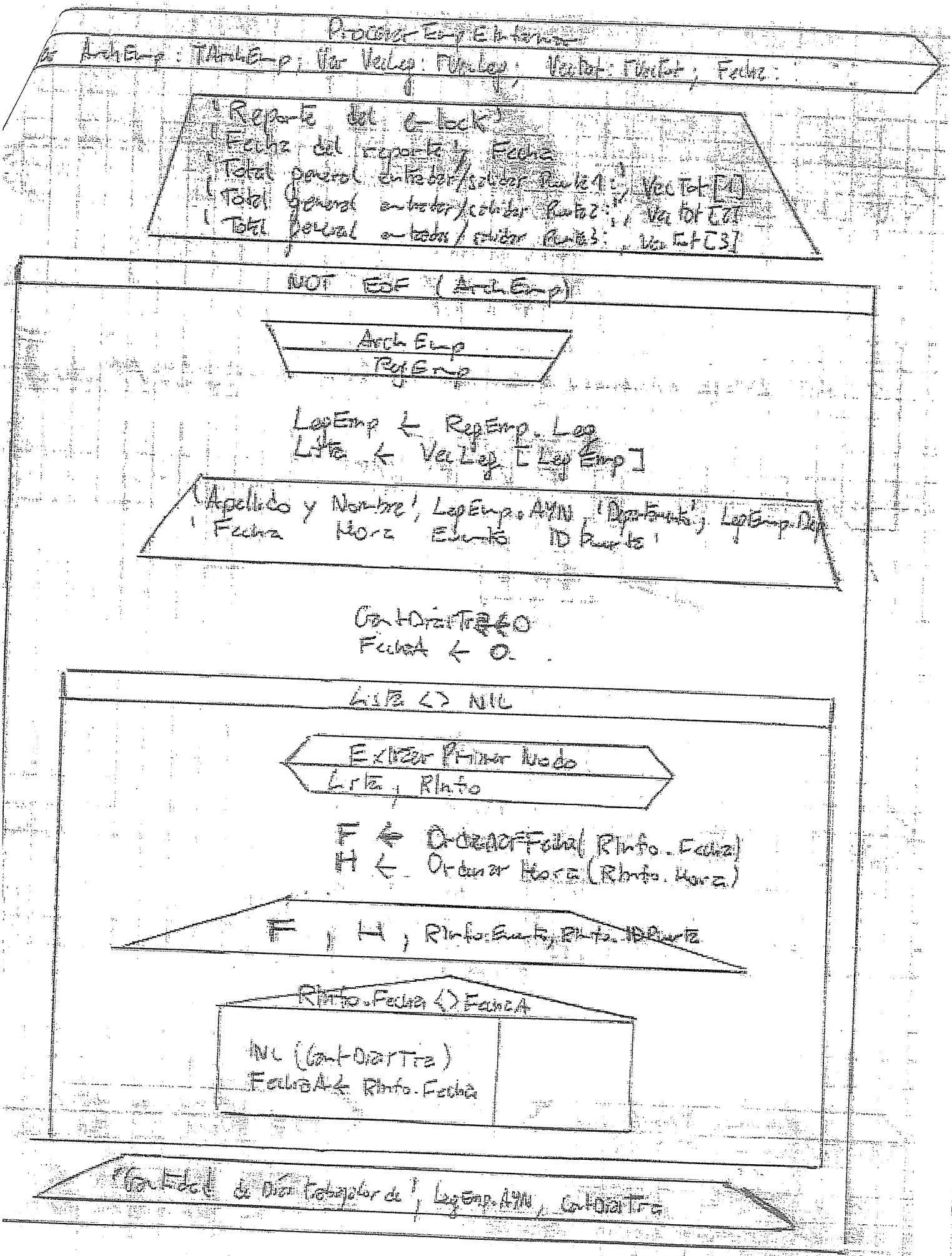
InformarModo

VecLog [LogEmp], RInfo

I
1 .. 3

VecTot[I] ← VecTot[I]/2

Ⓡ


Procesar Emp Enfermeras
ArchEmp : TArchEmp; VecLeg : TVecLeg; VecTot : TVecTot; Fecha :
'Reporte del Check'
'Fecha del reporte: ', Fecha
'Total general entrados/salidos Ruta1: ', VecTot[1]
'Total general entrados/salidos Ruta2: ', VecTot[2]
'Total general entrados/salidos Ruta3: ', VecTot[3]
NOT EOF (ArchEmp)
ArchEmp
RegEmp
LegEmp ← RegEmp.Leg
Lista ← VecLeg[LegEmp]
'Apellido y Nombre', LegEmp.AYN, 'Departamento', LegEmp.Dep
'Fecha Hora Estado ID Ruta'
ContOrdTrz ← 0
Fecha ← 0
Lista <> NIL
Extraer Primer Nodo
Lista, RInfo
F ← OrdenarFecha(RInfo.Fecha)
H ← Ordenar Hora(RInfo.Hora)
F, H, RInfo.Estado, RInfo.IDRuta
RInfo.Fecha <> FechaA
NL (ContOrdTrz)
FechaA ← RInfo.Fecha
'Total de Ordenes trabajadas de', LegEmp.AYN, ContOrdTrz

Cerrar Archivos

Var ArchEs : FArchEs ; Var ArchE-p : FArchE-p

Close (ArchEs)
Close (ArchE-p)

(R)

Intercalar Nodo

Var Lista : TLista ; Nuevo : TNodo

New (Nuevo)
Nuevo^.Info ← RInfo
P ← Lista

(P <> Lista) AND (P^.Info.Fecha < RInfo.Fecha OR (P^.Info.Hora = RInfo.Fecha AND P^.Info.Hora < RInfo.Hora))

Anterior ← P
P ← P^.Sgte

Lista = P

Lista ← Nuevo	Anterior^.Sgte ← Nuevo

Nuevo^.Sgte ← P

(R)

Extraer Primer Nodo

Var Lista : TLista ; Var RInfo : TInfo

RInfo ← Lista^.Info
P ← Lista
Lista ← Lista^.Sgte
Dispose (P)

(R)

Ordenar Fecha

(F: Comp) : STRING[6]

A ← F DIV 10000
M ← (F MOD 10000) DIV 100
D ← F MOD 100

OrdenarFecha ← D + '/' + M + '/' + A

(R)

Ordenar Hora

(Hora: Word) : STRING[5]

H ← Hora DIV 100
M ← Hora MOD 100

OrdenarHora ← H + ':' + M

(R)

(R) ALGORITRESFIN

Apellido, Nombres:			Legajo:	Hojas:
e-mail:	Aula:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructura de Datos 3 de diciembre de 2011 U.T.N. F.R.B.A.

Una Empresa comercial requiere de un proceso que asigne a los camiones las cantidades de bultos de los pedidos a distribuir en las zonas de repartos -un camión por zona-, contando con los siguientes archivos binarios de datos:

PEDIDOS.DAT sin orden, con el siguiente diseño:

a.1) Número de pedido (6 díg.) a.2) Cantidad de bultos (byte) a.3) Número de Cliente (6 díg)
a.4) Código de zona de reparto (1..50)

CLIENTES.DAT ordenado por Número de Cliente, con el siguiente diseño:

b.1) Número de Cliente. b.2) Razón social (30 caracteres)
b.3) Dirección de entrega (30 caracteres)

CAMIONES.DAT sin orden, con el siguiente diseño:

c.1) Patente (6 caracteres) c.2) Capacidad en cantidad de bultos (4 díg.)

Se pide desarrollar la metodología necesaria para obtener un algoritmo que:

1. Aplique en algún momento del proceso, pero NO desarrolle, la siguiente función cuyo prototipo es:

CamionCSM (var Lista : tipoLista; CantBultosZona : word) : tipoLista;

que recibe la lista de camiones disponibles, la cantidad de bultos de una zona, y retorna el puntero al nodo del camión cuya capacidad de carga satisfaga la cantidad de bultos requeridos para la zona pedida, y que no haya sido asignado a otra zona.

2. Listado de repartos, ordenado por Zona, con el siguiente diseño:

LISTADO DE DISTRIBUCIÓN DE PEDIDOS POR ZONA
CÓD-ZONA: 99 NRO.PATENTE: XXX999 CANT-BULTOS-ZONA: 9999

NRO-ORDEN	DIRECCIÓN	RAZÓN SOCIAL	NRO.PEDIDO	CANT.BULTOS
999	XXXXXXXX...XXXXXXX	XXXXX...XXXXXXX	999999	999

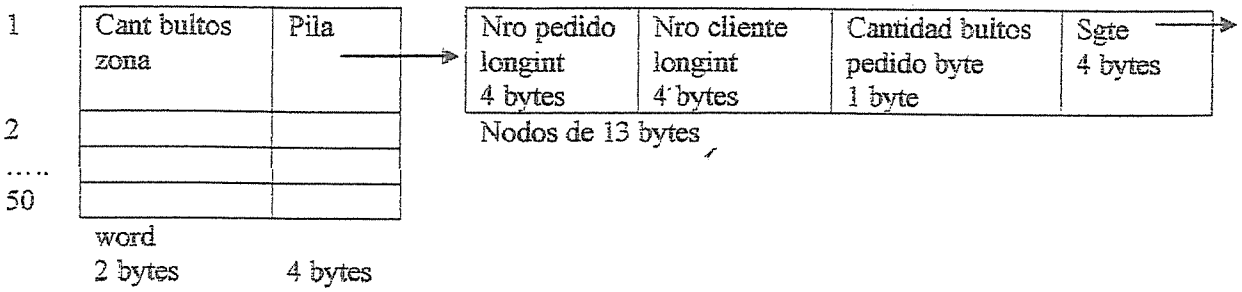
Recursos. Restricciones y Observaciones:

- Memoria para arrays: 300 bytes.
- Memoria para estructuras dinámicas: nodos de 13 bytes
- Lugar en disco: 0 bytes
- Accesos a archivos: un solo recorrido secuencial en los archivos PEDIDOS.DAT y CAMIONES.DAT. Aplique pero no desarrolle búsqueda binaria en CLIENTES.DAT.
- Utilizar procedimientos y funciones para desarrollar el algoritmo.
- Bloque principal sólo invocaciones a módulos.
- Optimización: dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. Desarrollar todos los módulos invocados.
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

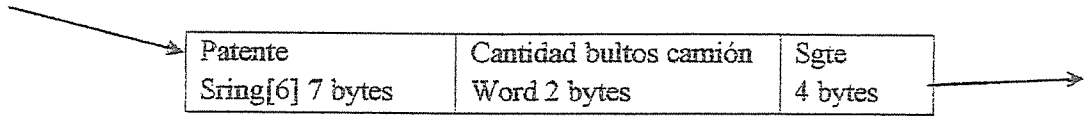
Apellido, Nombres:			Legajo:	Hojas:
e-mail:	Aula:	Docente que Firmó Libreta:		Nota:

Estructuras

array de zonas 50 * (2+4)= 300



Lista de camiones nodos de 13 bytes



Estrategia

- abrir archivos
- inicializar vector y lista
- recorrer archivo de camiones, y por cada registro insertar nodo en la lista de camiones
- recorrer archivo de pedidos, y por cada registros ubicar zona en array, armar nodo y meter en la pila correspondiente(podrían usar listas)
- recorrer array y por cada zona
  - invocar a la función de *CamionCSM* con puntero a la lista y la cantidad de bultos de la zona
  - poner cantidad de bultos del camión en cero en el nodo de la lista
  - emitir listado de esa zona sacando los nodos de la pila, y accediendo por cada nodo con búsqueda binaria al archivo de clientes para obtener los datos
- cerrar archivos
- fin

Apellido, Nombres:			Legajo:	Hojas:
e-mail:	Aula:	Docente que Firmó Libreta:		Nota:

Final Algoritmos y Estructura de Datos 17 de diciembre de 2011 U.T.N. F.R.B.A.

Un supermercado requiere controlar sus ventas al final del día, y para ello cuenta con los siguientes archivos:
a) Archivo binario, VENTAS.DAT, *con orden natural y un mismo artículo se puede encontrar en varios registros.*

a.1 Id_Caja (longint)
a.2 Código de artículo (7 caracteres)
a.3 Cantidad de unidades (longint)

b) Archivo binario, **EXISTENC.DAT**, *ordenado ascendentemente por Código de artículo.*

b.1 Código de artículo (7 caracteres)
b.2 Stock actual (longint)
b.3 Fecha última venta (aaaammdd)

Se pide desarrollar la metodología necesaria para obtener un algoritmo que:

1) Obtener una sola vez la fecha del sistema

2) Actualizar el archivo **EXISTENC.DAT** (una única vez por artículo) en sus campos **b.2 Stock actual y b.3 Fecha última venta** (con la fecha del sistema).

3) Para los artículos **NO vendidos**, emitir un listado *ordenado ascendentemente por Cantidad de días sin venta (1, 2, 3, ..., 30 o más) y Código de artículo* según se indica:

ARTÍCULOS SIN VENTA

Cantidad de días sin venta	Código de Artículo	Cantidad en stock
1	XXXXXXX	999999
2	XXXXXXX	999999
2	XXXXXXX	999999
...		
30	XXXXXXX	999999

Tener en cuenta que para cada cantidad de días sin venta puede haber uno o varios artículos diferentes en condiciones similares.
Aplique pero ***no desarrolle*** la **función:** *CalculaDias (FechaUltimaVenta, FechaDelSistema: longint): byte;*
Tal que, dadas dos fechas en formato **aaaammdd**, retorna la cantidad de días transcurridos entre esas dos fechas.

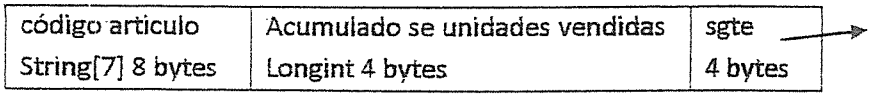
🗁 Recursos, Restricciones y Observaciones:

- **Memoria para arrays: 120** bytes.
- **Memoria para estructuras dinámicas:** nodos de 16 bytes uno por cada artículo
- **Lugar en disco:** 0 bytes
- **Accesos a archivos:** un recorrido secuencial a cada archivo + un acceso directo al archivo **EXISTENC.DAT.**
- **Utilizar procedimientos y funciones** para desarrollar el algoritmo.
- **Bloque principal** sólo invocaciones a módulos.
- **Optimización:** dado que el uso de ciclos afecta el tiempo de ejecución de un proceso, se evaluará la eficiencia en el uso de los mismos. **Desarrollar todos los módulos invocados.**
- Utilizar nombres significativos para los identificadores, dibujos para las estructuras a utilizar, rotulando cada elemento, tamaño, breve leyenda de cómo se generan y estado inicial, respetar esos nombres para utilizarlos en el algoritmo. Letra clara, trazo fuerte y tamaño apreciable para que pueda leerlo un tercero. Escribir una carilla por hoja rotulando c/u. de ellas con su Apellido, Nombre y Nro. Pág. x de y.

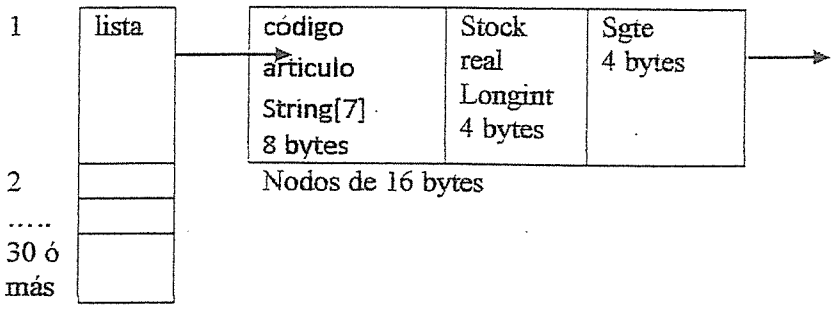
Apellido, Nombres:			Legajo:	Hojas:
e-mail:	Aula:	Docente que Firmó Libreta:		Nota:

Estructuras

lista de arcticulos vendidos nodos de 16 bytes uno por cada articulo



array de días 30 * 4=120



Estrategia

- abrir archivos
- inicializar vector y lista
- obtener fecha del sistema
- recorrer archivo de ventas y por cada registro
  - insertar en lista ordenada por artículo sin repetición acumulando unidades vendidas
- realizar apareo entre archivo de existencia y con la la lista de artículos vendidos
  - si el articulo esta en la lista y en el archivo
    - actualizar stock
    - actualizar fecha de ultima venta en el registro con la fecha del sistema
    - posicionarse en el registro del archivo y grabar
  - si el articulo del archivo no esta en la lista
    - invocar a la función calculadias con la fecha de ultima venta y la del sistema
    - según la cantidad de días transcurridos, armar nodo e insertar en la lista según posición en el array
- recorrer array y emitir listado
- cerrar archivos
- fin