IBBDD 2014 – Primera Fecha de Primer Parcial – 17/6/2014

En la primer hoja del examen escribir claramente legajo, apellido y nombre, turno (mañana: 1 - tarde: 2), temas que rinde (por su número) y cantidad de hojas que entrega. Numerar cada hoja.

1. Archivos Secuenciales

Defina constantes y tipos de datos y la implementación de una primitiva (en una Unit) para agregar un registro de longitud variable al final de un archivo con bloques de 2048 bytes, abierto sólo para escritura y con control de espacio libre en un archivo de bytes libres por bloque.

En cualquier programa que use la unidad, los registros se manejan con variables de tipo tFactura:

tLinea = **Record** *{registro de venta de un producto}*

codProd: **LongWord**; *{código de producto}*

cant: **Word**; *{cantidad vendida}*

precioU: **Real** *{precio unitario}*

end;

tFactura = **Record** *{registro de factura de venta}*

nroFac, fecha: **LongWord**; *{número y fecha de factura}*

cantLineas: 1..10; *{cantidad de líneas}*

líneas: **Array**[1..10] **of** tLinea *{detalle de productos vendidos}*

end;

Bosquejo de Solución

Unit Facturas**;**

Interface

Const

LongBloque = 2048;

Type

tNroBloque = **Word**;

tBloque = **Array**[1..LongBloque] of **Byte**;

abFfacturas = **File**; *{tipo de archivo de bloques de registros de facturas}*

tEstado = (C, E, LE); *{estado del archivo: (C(errado), abierto para E(scrituras) secuenciales sin búsqueda de espacio libre (appends), o para L(ectura)E(scritura) con posicionamiento previo en bloques}*

tLinea = …

tFactura **=** …

ctlFacturas = **Record** *{tipo de registro de control para archivo de facturas (handle); se pasa como parámetro por referencia a cualquier subprograma que acceda al archivo}*

estado: tEstado;

arch: abFacturas; *{archivo de bloques de facturas}*

b: tBloque *{bloque buffer para leer o escribir en el archivo de facturas}*

ib: **Word**; *{índice de posicionamiento en b}*

libres: **File of Word**; *{archivo de bytes libres en cada bloque del archivo de datos}*

libre: **Word**; *{variable buffer para leer libres de pb}*

fe: **Array**[1..126] **of Byte**; *{registro buffer para registros empaquetados de facturas}*

```
            lfe: Byte; {longitud de registro empaquetado de factura}
            f: tFactura; {registro buffer para registros desempaquetados de facturas}
        end;
    …
    Procedure Agregar(var a: ctlFacturas);
    …
Implementation
    Procedure Agregar(var a: ctlFacturas);
    {Recibe a ctlFacturas.arch con estado E (abierto para escritura), con el último bloque cargado en
    ctlFacturas.b, sus bytes libres en ctlFacturas.libre y ctlFacturas.ib indicando el primer byte libre del bloque}
    begin
    …
    end;
```

Observaciones

Las variables para el registro empaquetado de facturas y su longitud podrían ser locales al procedimiento Agregar en lugar de ser campos de ctlFacturas; asimismo la variable para el registro en memoria de factura f podría ser un parámetro, preferentemente por referencia (var), del procedimiento en lugar de estar como un campo de ctlFacturas.

Es importante que hayan definido tEstado para asumir que el archivo está abierto sólo para escritura (si no lo mencionan como precondición o como comentario no importa, pero debe estar definido el tipo para especificar el estado del archivo de bloques. El registro ctlFacturas siempre contendrá la información para agregar un registro al último bloque, sin necesidad de tener que leerlo ni inicializar los campos de control: sólo se debe agregar el registro en el bloque sin grabarlo después, y si no hubiera lugar, grabar el bloque actual e inicializar un bloque nuevo en ctlFacturas, sin grabarlo luego de agregar la factura.

2. Árboles en Archivos

Dado el árbol B+ que se detalla más abajo, con capacidad máxima de 2 registros por nodo y mínima 1, muestre los estados sucesivos **completos** al realizar la siguiente secuencia de operaciones:

+500, -254, +800, -432.

Para cada operación indique la secuencia de nodos que debió leer o escribir. Considere que al partirse un nodo se agrega uno nuevo a la derecha, y si es hoja, el partido queda más cargado.

2: 0 (432) 3 (724) 1

0: (254) 3 | 3: (432)(611) 1 | 1: (724)(905) -1

+500

6: 2 (611) 5			
2: 0 (432) 3		5: 4 (724) 1	
0: (254) 3	3: (432)(500) 4	4: (611) 1	1: (724)(905) -1

Costo: L2L3 <u>E3E4</u> <u>E2E5</u> E6

-254

6: 2 (611) 5			
2: 0 (500) 3		5: 4 (724) 1	
0: (432) 3	3: (500) 4	4: (611) 1	1: (724)(905) -1

Costo: L6L2L0L3 <u>E0E3</u> E2

+800

6: 2 (611) 5				
2: 0 (500) 3		5: 4 (724) 1 (905) 7		
0: (432) 3	3: (500) 4	4: (611) 1	1: (724)(800) 7	7: (905) -1

Costo: L6L5L1 <u>E1E7</u> E5

-432

6: 2 (724) 5			
2: 0 (611) 4		5: 1 (905) 7	
0: (500) 4	4: (611) 1	1: (724)(800) 7	7: (905) -1

Libre: 3

Costo: L6L2L0L3E0L5 <u>E2E5</u> E6

Observación

Los costos pueden variar y ser correctos sólo si se invierte el orden de escritura de nodos hermanos (los subrayados).

3. Archivos Directos

Dado el estado de un archivo directo con cubetas para a lo sumo 4 registros y con dispersión extensible que se detalla:

a) Muestre los estados sucesivos **completos** al realizar la siguiente secuencia de operaciones: +202, +669, -24.

Para cada operación indique la secuencia de cubetas que debió leer o escribir.

b) Sobre el estado final resultante de las operaciones previas, indique para cada cubeta, si al vaciarse sin que las restantes se modifiquen, se podría liberar o no. Justifique.

a)

Tabla:

2 0 1 0

Cubetas Libres: -

Claves de registros en cubetas:

0: 1 | 835 831 347 401

1: 2 | 250 526 390 366

2: 2 | 24

+202:

Tabla:

2 0 3 0 2 0 1 0

Cubetas Libres: -

0: 1 | 835 831 347 401

1: **3** | 526 390 366

2: 2 | 24

3: **3** | 250 **202**

Costo: L1 E1E3

+669:

Tabla:

2 4 3 0 2 4 1 0

Cubetas Libres: -

0: **2** | 835 831 347

1: 3 | 526 390 366

2: 2 | 24

3: 3 | 250 202

4: **2** | 401 **669**

Costo: L0 E0E4

-24:

Tabla:

2 4 3 0 2 4 1 0

Cubetas Libres: -

0: 2 | 835 831 347

1: 3 | 526 390 366

2: 2 |

3: 3 | 250 202

4: 2 | 401 669

Costo: L2E2 - No se puede liberar la cubeta 2 porque a partir de la posición 24 mod 8 = 0 de la tabla, inspeccionando los registros anterior y posterior de la tabla a distancia circular 2^(**2**-1)=2 (**2** son los bits de dispersión de la cubeta) se encuentran **distintos** números de cubeta (1 y 3).

b)

Cubeta 0: 2^(2-1)=2; desde el registro 3 de la tabla, inspeccionando el 3-2=1 y 3+2=5 se encuentra un mismo número de cubeta (4); entonces se puede liberar.

Cubeta 1: 2^(3-1)=4; desde el registro 6 de la tabla, inspeccionando el 6-4=2 y (6+4) mod 8=2 se encuentra un mismo número de cubeta (3); entonces se puede liberar.

Cubeta 2: por la última operación, ya se verificó que no se puede liberar.

Cubeta 3: ídem cubeta 1.

Cubeta 4: ídem cubeta 0.