Muestra

FOD-2025 - Parcial - Fecha 2

EJERCICIO 1: ARCHIVOS

Enunciado

Se desea automatizar el manejo de información referente al desempeño histórico de los equipos de fútbol de la **Liga Profesional Argentina**. Para esto se cuenta con un **archivo maestro** que contiene, por cada equipo, la siguiente información: código de equipo, nombre del equipo, cantidad de partidos jugados, cantidad de partidos ganados, cantidad de partidos empatados, cantidad de partidos perdidos y cantidad de puntos acumulados. El archivo maestro está ordenado por código de equipo.

Al finalizar la temporada (año), se reciben **12 archivos detalle** (uno por cada mes), que contienen información de los partidos jugados por los equipos durante cada mes del año. Cada archivo contiene múltiples registros con la siguiente información: código de equipo, fecha del partido, cantidad de puntos obtenidos en ese partido (3 si ganó, 1 si empató, 0 si perdió) y código del equipo rival. Los archivos detalle están ordenados por código de equipo. En cada archivo puede haber información de cualquier equipo, y un mismo equipo puede aparecer más de una vez o no aparecer en absoluto.

Defina los tipos de datos necesarios para representar la información del archivo maestro y los archivos detalles, e implemente un procedimiento (y los procedimientos que utilice) que permita actualizar el archivo maestro a partir de los archivos detalle, recalculando los valores estadísticos de cada equipo: partidos jugados, ganados, empatados, perdidos y puntos acumulados. Además, durante el mismo recorrido en que se actualiza la información, se debe determinar e informar por pantalla el equipo que más puntos sumó a lo largo de la temporada (año). Se debe mostrar el nombre del equipo junto con la cantidad de puntos obtenidos en el año.

Notas:

1. Los nombres de los archivos deben pedirse por teclado. Se puede suponer que los nombres ingresados corresponden a archivos existentes.
2. Los archivos se deben recorrer una única vez.

Si un **equipo A jugó un partido contra un equipo B**, en un archivo detalle **aparecerán dos registros separados**: uno para el equipo A, y otro para el equipo B. **Los partidos se registran de forma independiente para cada equipo.**

Resolución

Una posible solución al ejercicio planteado se presenta a continuación:

```
const
       VA = 9999;
       N = 12;
type
       str30 = string[30]
       TEquipo = record
              codEquipo: integer;
              nombreEquipo: str30;
              cantJ: integer;
              cantG: integer;
              cantE: integer;
              cantP: integer;
              cantPuntos: integer;
       end;
       TPartido = record
              codEquipo: integer;
              fechaPartido: longInt;
              cantPuntos: integer;
              codRival: integer;
       end;

       maestro = file of TEquipo;
       detalle = file of TPartido;

       detalles = array [1..N] of detalle;
       regDetalle = array [1..N] of TPartido;

procedure leer (var a: detalle; var reg: TPartido);
begin
       if (not(EOF(a))) then
           read(a, reg)
       else
           reg.codEquipo := VA;
end;

procedure minimo (var vDet: detalles; var rDet: regDetalle; var min: TPartido);
var
      i, posMin: integer;
begin
      posMin := -1; min.codEquipo := VA;
      for i:= 1 to N do begin
          if (rDet[i].codEquipo < min.codEquipo) then begin
            min := rDet[i];
            posMin := i;
          end;
```

```
      end;
      if (min.codEquipo <> VA) then
          leer(vDet[posMin], rDet[posMin]);
end;

procedure asignarArchivos (var m: maestro, var vDet: detalles);
var
    i: integer;
    nombreArchivo: str30;
begin
    writeln('Ingrese el nombre del archivo maestro: ');
    readln(nombreArchivo);
    assign(m, nombreArchivo);
    for i := 1 to N do begin
        writeln('Ingrese el nombre del archivo detalle: ', i);
        readln(nombreArchivo);
        assign(vDet[i], nombreArchivo);
    end;
end;

procedure ejercicio (var m: maestro; var vDet: detalles);
var
    i: integer; min: TPartido; total: integer;
    puntos, maxPuntos: integer; nombreMax: str30;
    regM: TEquipo;
    partidos: regDetalle;
    nombreArchivo: str30;
begin
    maxPuntos := -1;
    asignarArchivos(m, vDet);
    reset(m);
    for i := 1 to N do begin
        reset(vDet[i]);
        leer(vDet[i], partidos[i]);
    end;
    minimo(vDet, partidos, min);
    while (min.codEquipo <> VA) do begin
        read(m, regM);
        while (regM.codEquipo <> min.codEquipo) do
            read(m, regM);
        puntos := 0;
        while (regM.codEquipo = min.codEquipo) do begin
            puntos := puntos + min.cantPuntos;
            regM.cantJ := regM.cantJ + 1;
            if (min.cantPuntos = 3) then
                regM.cantG := regM.cantG + 1;
            else
                if (min.cantPuntos = 1) then
                    regM.cantE := regM.cantE + 1;
                else
                    regM.cantP := regM.cantP + 1;
```

```
            minimo(vDet, partidos, min);
        end;
        regM.cantPuntos := regM.cantPuntos + puntos;
        if (puntos > maxPuntos) then begin
            maxPuntos := puntos;
            nombreMax := regM.nombreEquipo;
        end;
        seek(m, filePos(m) - 1);
        write(m, regM);
    end;
    writeln('El equipo que sumó más puntos durante la temporada fue: ', nombreMax, ' con ',
maxPuntos, ' puntos.');
    close(m);
    for i := 1 to N do
        close(vDet[i]);
end;
```

Errores comunes:

1. No abrir los archivos o abrirlos con rewrite.
2. No asignar los archivos.
3. No leer los nombres de los archivos por teclado.
4. No realizar el procedimiento leer. Este error genera pérdida de información.
5. No pasar el archivo por referencia al procedimiento leer o al procedimiento de actualización.
6. No pasar el registro por referencia al procedimiento leer.
7. Recorrer múltiples veces el archivo maestro o los detalles.
8. Recorrer por el maestro para comparar por todos los equipos cuando se pedía solamente de la temporada (año).
9. Comparar el máximo con los datos históricos del maestro.
10. Anidar la condición de EOF en el bucle en el que buscan el mínimo en el maestro.
11. No inicializar/resetear variables.
12. Falta de actualización de algún dato (por ejemplo: partidos jugados).
13. Pisar los datos del maestro.
14. No posicionarse con seek(arch, filepos(arch)-1) para escribir en la posición correcta.
15. No cerrar los archivos.
16. Calcular mal el mínimo registro de los detalles.
17. Leer en todas las posiciones del vector de registros cuando solo se debía leer en una (la que tiene el mínimo).

EJERCICIO 2: HASHING

Enunciado

Dado un archivo que utiliza dispersión extensible, con bloques de datos con capacidad para 2 registros, para cada operación: dibujar la tabla resultante y justificar las decisiones tomadas. Las operaciones a realizar son: +Catley (01101011), +Little (01101000) -Zinsberger(00000001). El signo “-” indica una baja, el signo “+” indica una alta.

Tabla: Bits de dispersión **2**
0(00): 3
1(01): 2
2(10): 1
3(11): 0
Bloques Libres: -

Claves de registros en Bloques:
0: 2 | Blackstenius(01100111) Russo(11110111)
1: 2 | Mead(01010110) Williamson(10110010)
2: 2 | Zinsberger(00000001)
3: 2 | Foord(00101000) Van Domselaar(00110100)

Operación 1: +Catley (01101011)

Objetivo:

- Evaluar el alta de una clave que provoca desborde en un bloque que involucra la duplicación de direcciones en la tabla.

Explicación:

- Como se indica en los bits de dispersión de la tabla, se toman los dos bits menos significativos de la clave a insertar (11).
- Se busca en la tabla la entrada correspondiente.
- Se debería insertar en el bloque 0. Como el bloque 0 tiene el máximo de registros, se produce colisión y desborde.
- Se debe aumentar en uno la cantidad de bits de dispersión locales (bloque 0), pasando de 2 a 3.
- Se crea un nuevo bloque (bloque 4) con la misma cantidad de bits de dispersión.
- Se compara con los bits de dispersión de la tabla. Como los bits de dispersión locales (3) son mayor a los bits de dispersión globales (2), se debe aumentar en uno la cantidad de bits de dispersión globales, llegando a 3, y se duplica la cantidad de entradas de la tabla, quedando 8 entradas.
- Una vez duplicada se deben reacomodar los enlaces de las entradas intervinientes.
- Hay dos explicaciones posibles, en la explicación práctica se menciona que la entrada que apuntaba al bloque en el que se generó overflow (entrada 3) ahora apuntará al nuevo bloque (bloque 4). La entrada 7 (111), apunta al bloque en el cual se produjo el desborde (bloque 0).

- Se redispersan las claves intervinientes teniendo en cuenta ahora los últimos 3 bits menos significativos.

Tabla: Bits de dispersión **3**
0(000): 3
1(001): 2
2(010): 1
3(011): **4**
4(100): 3
5(101): 2
6(110): 1
7(111): **0**

Claves de registros en Bloques:
0: **3** | ***Blackstenius(01100111) Russo(11110111)***
1: 2 | Mead(01010110) Williamson(10110010)
2: 2 | Zinsberger(00000001)
3: 2 | Foord(00101000) Van Domselaar(00110100)
4: **3** | ***Catley (01101011)***

Bloques Libres: -

Operación 2: +Little (01101000)

Objetivo:

- Evaluar el alta de una clave que provoca desborde en un bloque sin la duplicación de direcciones en la tabla.

Explicación:

- Como se indica en los bits de dispersión de la tabla, se toman los tres bits menos significativos de la clave a insertar (000).
- Se busca en la tabla la entrada correspondiente.
- Se debería insertar en el bloque 3. Como el bloque 3 tiene el máximo de registros, se produce colisión y desborde.
- Se debe aumentar en uno la cantidad de bits de dispersión locales (bloque 3), pasando de 2 a 3.
- Se crea un nuevo bloque (bloque 5) con la misma cantidad de bits de dispersión.
- Se compara con los bits de dispersión de la tabla. Como los bits de dispersión locales (3) no son mayores a los bits de dispersión globales (3) no es necesario aumentar los bits de dispersión globales ni duplicar las entradas de la tabla.
- Se deben reacomodar los enlaces de las entradas intervinientes.
- Hay dos explicaciones posibles, en la explicación práctica se menciona que la entrada que apuntaba al bloque en el que se generó overflow (entrada 0) ahora apuntará al nuevo bloque (bloque 5). La entrada 4 (100), apunta al bloque en el cual se produjo el desborde (bloque 3). La otra explicación sería tomar los 3 bits menos significativos de la clave que genera el desborde y hacer apuntar la entrada que coincide con esos 3 bits menos significativos al nuevo bloque. En este caso, quedaría igual.
- Se redispersan las claves intervinientes teniendo en cuenta ahora los últimos 3 bits menos significativos.

Tabla: Bits de dispersión **3**
0(000): **5**
1(001): 2
2(010): 1
3(011): 4
4(100): **3**
5(101): 2
6(110): 1
7(111): 0

Claves de registros en Bloques:
0: 3 | Blackstenius(01100111) Russo(11110111)
1: 2 | Mead(01010110) Williamson(10110010)
2: 2 | Zinsberger(00000001)
3: **3** | ***Van Domselaar(00110100)***
4: 3 | Catley (01101011)
5: **3** | ***Foord(00101000) Little(01101000)***

Bloques Libres: -

Operación 3: -Zinsberger(00000001)

Objetivo:

- Evalúa la baja de una clave en un bloque que no se puede liberar, quedando vacío y referenciado.

Explicación:

- Como se indica en los bits de dispersión de la tabla, se toman los tres bits menos significativos de la clave a dar de baja (001).
- Se busca en la tabla la entrada correspondiente.
- La clave a dar de baja se encuentra en el bloque 2.
- Se da de baja la clave.
- Al quedar el bloque vacío, se debe analizar si es posible liberarlo o no.
- Se buscan los sufijos hermanos. Aquellas direcciones de la tabla que comparten con el bloque en el que se realiza la baja el mismo sufijo de k-1 bits. Siendo k el número de bits de dispersión local. En este caso, k=2.
- Serían los que terminan en 1 pero no se tendrán en cuenta aquellas entradas que ya apuntan al bloque 2, por lo tanto, miraremos a dónde apunta el resto. Nos encontramos con que “011” apunta al bloque 4 y “111” al bloque 0. Como no apuntan a un mismo bloque, no podemos liberar el bloque vacío. Las direcciones que lo apuntaban, seguirán apuntando al mismo. No se reflejarán cambios en la tabla de dispersión.

Tabla: Bits de dispersión **3**
0(000): 5
1(001): 2
2(010): 1
3(011): 4
4(100): 3
5(101): 2
6(110): 1
7(111): 0

Claves de registros en Bloques:
0: 3 | Blackstenius(01100111) Russo(11110111)
1: 2 | Mead(01010110) Williamson(10110010)
2: 2 | (vacío)
3: 3 | Van Domselaar(00110100)
4: 3 | Catley (01101011)
5: 3 | Foord(00101000) Little(01101000)

Bloques Libres: -

Errores comunes:

1. No escribir ninguna explicación.
2. No explicar las decisiones tomadas (por ejemplo: no explicar por qué se duplican o no se duplican las direcciones de la tabla o no mencionar que se produce desborde) o explicar incorrectamente/en desorden.
3. Usar conceptos de árboles en hashing (por ejemplo: underflow, balancear, fusionar, etc).
4. Hacer todas las operaciones sobre una misma tabla.
5. No ser consistentes con definir qué entrada apunta a qué bloque luego de una operación de alta con desborde.
6. Realizar mal una operación (ej: liberar el bloque en una baja).
7. Marcar al bloque vacío como “bloque libre”.
8. Disminuir en 1 los bits locales del bloque que queda vacío.
9. No enumerar los bloques.
10. No indicar los bits de dispersión de cada bloque o la tabla.