Examen Final (26.1.2011)

Solo voy a mostrar posibles respuestas al 3º problema (el que yo propuse y corrijo).

Es conveniente que veais las posibles respuestas y la interpretación del problema para evitar tener que repetir a todos los que vengais a revisión las mismas cosas. Lo mismo os digo de los comentarios que están al final de esta página.

I. T. Xestión

Nombre	Prob1	Prob2	Prob3	Nota
Allegue Ameneiro, Lucia	9	1	5	5
Castelo Vazquez, Sergio	10	0	2	4
Iglesias Fernandez, Victor	8	1	9	6
Rios Maalvido, Alexandre	7	0	2	3
Salgado López, Alejandro	0	0	0	0

Problema 3º

Suponiendo que existe la función *diasMes(mes: tMes; anio: tAnio): integer* de significado obvio y los tipos siguientes

```
type
    tAnio = 1800 .. 2400;
    tMes  = 1 .. 12;
    tFecha = record
        anio: tAnio;
        mes: tMes;
        dia: 1 .. 31
    end;
```

programar la función que calcula el número de días que hay entre dos fechas del mismo año.

La cabecera de la función podría ser así, ya que debe recibir dos fechas y devolver un entero que representa el número de días que hay entre ambas. Del enunciado no se puede deducir que las fechas están ordenadas, o sea que f1 <= f2. Sin embargo, sabemos que serán del mismo año.

```
function difDias(f1, f2: tFecha): integer;
```

La resolución del problema admite varias aproximaciones. Todas se pueden resumir en tres, como vemos a partir de este esquema.

```
--+--------+---+----+--------+--------+--+-----+--------+------
  0        1   A    2        3        4  B     5
```

Podemos medir la longitud del intervalo AB de tres formas

- Sumando al intervalo 24, el A2 y el 4B
- Restando al intervalo 15, el 1A y el B5
- Restando del intervalo 0B, el intervalo 0A

Comentarios

Algunos habeis usado una cabecera como la siguiente (o parecida, que cada cual se aplique lo que le corresponda) que está llena de inconvenientes.

```
type
   tAFechas = packed array [ 1 .. 2 ] of tFecha)
function difDias (protected var a:tAFechas): integer;
```

Donde están los inconvenientes

- Empaquetar las fechas no ahorra memoria porque el tipo tFecha no es empaquetado. Para que la palabra packed tenga efecto tienen que ser empaquetadas todas las estructuras internas o bien ser tipos simples.
- El acceso a las fechas almacenadas en el array exige cálculo adicional que hace la computación más costosa.
- El paso por referencia, cuando se trata de una fecha ahorra poca memoria. Esto no es un registro de miles o millones de bytes. A cambio, ocurren dos cosas
  - para garantizar que los originales no cambien hay que pasarlos protected,

- el pasarlos protected nos limita a la hora de ordenar las fechas si se necesita.

Otros, han comprobado dentro de la función si las dos fechas son del mismo año. Sin embargo eso debe tomarse como una precondición ya que lo exige el enunciado del problema. Esta condición se debe comprobar antes de usar la función.

Por el contrario, algunos habeis supuesto que la primera fecha es anterior a la segunda, lo que no se dice en el enunciado y no se puede suponer ya que sería una limitación a la especificación. Es como si para simplificar el problema se supone que las dos fechas son del mismo mes sin que el enunciado lo exija (como exige que sean del mismo año).

Algunos usais cabeceras de la función que no incluyen el tFecha, por ejemplo esta que sigue. Esta forma de hacer la función no usa el tFecha, lo que es contradictorio con el enunciado. Se pretende saber cómo manejais registros.

```
function difDias(var anio1, anio2: tAnio; var mes1, mes2: tMes; var dia1, dia2: tDia): integer;
```

Estas cosas restan puntos aunque no invalidan el problema.

Soluciones

1ª Solución

```
--+--------+---+----+--------+--------+--+-----+--------+------
  0        1   A    2        3        4  B     5
```

La primera solución : AB = 24 + A2 + 4B

Hay que tener en cuenta la posibilidad de que ambos meses sean iguales y solo haya que restar dias.

```
si (f1.mes = f2.mes)
    nDias <-- abs(f1.dia - f2.dia)
si no
    nDias := diasMesesCompletos + diasMesIni + diasMesFin
fin si
```

Pero, si (f1.mes <> f2.mes) entonces para ordenar las fechas, si no lo están, basta considerar los meses.

```
if f1.mes > f2.mes then begin
   f  := f1;
   f1 := f2;
   f2 := f
end;
```

Ahora, contar los días de los meses completos con las fechas ordenadas. Observa que si los meses son seguidos no se entra en el bucle.

```
nDias := 0;  {contador de días}
for m := f1.mes + 1 to f2.mes - 1 do  {fechas ya ordenadas}
   nDias := nDias + diasMes(m, f1.anio);
```

A los días de los meses completos hay que agregar los del primer mes y los del último.

```
    nDias := nDias + diasMes(f1.mes, f1.anio) - f1.dia; {dias del primer mes}
    nDias := nDias + f2.dia;                            {dias del último mes}
```

Poniendo todo junto podría hacerse así

```
function difDias(f1, f2: tFecha): integer;
var nDias, m: integer; f: tFecha;
```

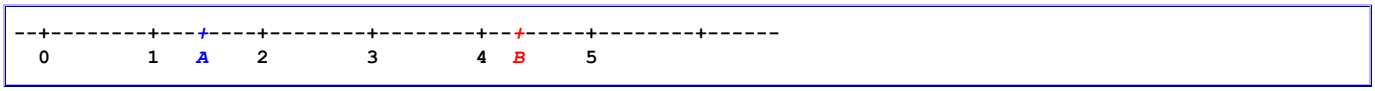
```
begin
    if f1.mes = f2.mes
    then nd := abs(f1.dia - f2.dia)
    else begin
        if f1.mes > f2.mes then begin   {primero ordenar las fechas}
           f  := f1;
           f1 := f2;
           f2 := f
        end;

        nDias := 0;  {acumular de días de meses completos}
        for m := f1.mes + 1 to f2.mes - 1 do
            nDias := nDias + diasMes(m, f1.anio);

        nDias := nDias + diasMes(f1.mes, f1.anio) - f1.dia; {ultimos dias del primer mes}
        nDias := nDias + f2.dia;                            {primeros dias del último mes}
    end;

    difDias := nDias
end;
```

Unas variantes de esta solución que recorren todos los meses con el for.

```
function difDias(f1, f2: tFecha): integer;
var      m: integer; f: tFecha;
     nDias: integer value 0;
begin
    if f1.mes = f2.mes
    then difDias := abs(f2.dia - f1.dia)
    else begin
         if f1.mes  > f2.mes then begin
            f := f1; f1 := f2; f2 := f;
         end;

         for m := f1.mes  to f2.mes do
            if m = f1.mes
            then  nDias := nDias + diasMes(m, f1.anio) - f1.dia
            else if m = f2.mes
                 then nDias := nDias + f2.dia
                 else nDias := nDias + diasMes(m, f1.anio);

        difDias := nDias;
    end
end;
```

```
function difDias(f1, f2: tFecha): integer;
var      m: integer; f: tFecha;
     nDias: integer value 0;
begin
    if f1.mes  > f2.mes then begin
      f := f1; f1 := f2; f2 := f;
    end;

    for m := f1.mes  to f2.mes do
        if (m = f1.mes) and (m = f2.mes)
        then nDias := abs(f1.dia - f2.dia)
        else if m = f1.mes
             then  nDias := nDias + diasMes(m, f1.anio) - f1.dia
             else if m = f2.mes
                  then nDias := nDias + f2.dia
                  else nDias := nDias + diasMes(m, f1.anio);

    difDias := nDias;
end;
```

2ª Solución



Esta es una solución más intuitiva aunque algo menos eficiente. Se basa en convertir las dos fechas en los días que han pasado desde principio de año y luego dar la respuesta acorde a esos valores. AB := 0B - 0A

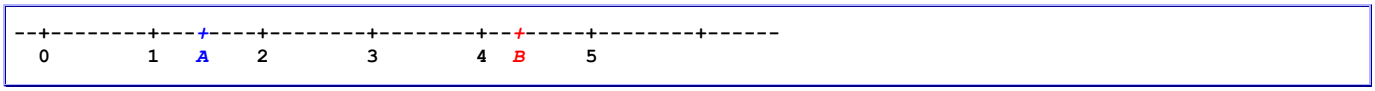
```
function difDias(f1, f2: tFecha): integer;
var nd1, nd2, m: integer value 0;
begin
    for m := 1 to f1.mes-1 do nd1 := nd1 + diasMes(m, f1.anio);
    nd1 := nd1 + f1.dia;

    for m := 1 to f2.mes-1 do nd2 := nd2 + diasMes(m, f2.anio);
    nd2 := nd2 + f2.dia;

    if nd1 >= nd2
    then difDias := nd1 - nd2
    else difDias := nd2 - nd1
    {difDias := abs(nd1 - nd2)}
end;
```

Observa que de esta manera, la comparación de fechas se reduce a la comparación de días pasados desde principio de año y que podría no ser necesaria si se usa la función estandar abs.

Una variante más eficiente que la anterior, aunque más prolija de escritura es la que usa un case para contar los días. Esta es quizás la función más eficiente de todas porque da todo el trabajo hecho y solo falta hacer un par de selecciones, dos sumas y una resta.

```
function difDias(f1, f2: tFecha): integer;
var nd1, nd2: integer value 0;
    feb: integer value diasMes(2. f1.anio);
begin
    case f1.mes of
         1: nd1 := f1.dia;
         2: nd1 := f1.dia +  31;
         3: nd1 := f1.dia +  31 + feb;
         4: nd1 := f1.dia +  62 + feb;
         5: nd1 := f1.dia +  92 + feb;
         6: nd1 := f1.dia + 123 + feb ;
         7: nd1 := f1.dia + 153 + feb;
         8: nd1 := f1.dia + 184 + feb;
         9: nd1 := f1.dia + 215 + feb;
        10: nd1 := f1.dia + 245 + feb;
        11: nd1 := f1.dia + 276 + feb;
        12: nd1 := f1.dia + 306 + feb;
    end;
    case f2.mes of
         1: nd2 := f2.dia;
         2: nd2 := f2.dia +  31;
         3: nd2 := f2.dia +  31 + feb;
         4: nd2 := f2.dia +  62 + feb;
         5: nd2 := f2.dia +  92 + feb;
         6: nd2 := f2.dia + 123 + feb ;
         7: nd2 := f2.dia + 153 + feb;
         8: nd2 := f2.dia + 184 + feb;
         9: nd2 := f2.dia + 215 + feb;
        10: nd2 := f2.dia + 245 + feb;
        11: nd2 := f2.dia + 276 + feb;
        12: nd2 := f2.dia + 306 + feb;
    end;

    if nd1 >= nd2
    then difDias := nd1 - nd2
    else difDias := nd2 - nd1
    {difDias := abs(nd1 - nd2)}
end;
```

3ª solución



Contar los dias de los meses involucrados como si fueran completos y luego restar a ese total los dias de los meses extremos que caen fuera de intervalo. Exige ordenar primero las fechas. AB := 15 - 1A - B5

```
function difDias(f1, f2: tFecha): integer;
var   m, nd: integer value 0; f: tFecha;
begin
     if (f1.mes > f2.mes) or_else ((f1.mes = f2.mes) and (f1.dia > f2.dia)) then begin
        f := f1; f1 := f2; f2 := f;
     end;

     for m := f1.mes to f2.mes do nd := nd + diasMes(m, f1.anio);

     difDias := nd - f1.dia - (diasMes(f2.mes, f2.anio) - f2.dia)
end;
```