Departamento de Informática. I. T. Informática
Solución Examen Febrero 2003 – Programación I

1. Búsqueda en un vector

Este ejercicio se puede resolver de varias maneras, en función del tipo de datos utilizado para almacenar los índices, que puede ser un vector de índices, un conjunto de índices o un vector de lógicos.

A continuación se detallan dos posibles soluciones:

```
type ...
     TIndices = set of TyIndice;
procedure superbusq (var v : Tyvector; buscado : Tipobase;
                     var num : integer; var indices: TIndices);
var i : Tyindice;
begin
      num := 0;
      indices := [];
      for i := a1 to am do
          if v[i] = buscado then
              begin
                    num := num +1;
                    indices := indices + [i]
              end
end;
```

```
Tyresul=array[tyindice] of tyindice;

Procedure  Buscar (var V:tyvector; Buscado:Tipobase; var Cuantos:integer, var Resul:Tyresul);
{busca los índices de V en los que se encuentra BUSCADO, y los devuelve en RESUL, y en Cuantos
devuelve cuantos coincidían. Se supone V no ordenado}
Var i,j: tyindice;
Begin  {buscar}
        Cuantos:=0; j:=a1;
        For i:=a1 to am do
                Begin
                        If  V[i] = Buscado then
                                Begin
                                        Resul[j]:=i;
                                        j:=succ(j);
                                        Cuantos:= Cuantos+1;
                                End
                End;
End;    {buscar}
```

2. Puntero

Un puntero es una variable estática, ya que se declara en la zona de declaración de variables, se crea al iniciar la ejecución del programa y existe durante toda la ejecución.

3. Función cuadrado

Solución recursiva:

```
Function cuadrado(n:integer):integer;
{calcula de forma recursiva el cuadrado de n, utilizando las sumas de impares adecuados.}
{Precondición: n>0 }

begin {cuadrado}
        if n=1 then
          cuadrado :=1
        else
          cuadrado:= cuadrado(n-1) + 2*n-1;
end;{cuadrado}
```

Solución iterativa: varias posibilidades, entre ellas planteamos dos:

```
function cuadrado_iter(n:integer):integer;
{calcula el cuadrado de n, utilizando las sumas de impares adecuados.}
{Precondición: n>0 }
var suma,i:integer;

begin {cuadrado_iter}
        suma := 0 ;
        for i :=1 to 2*n-1 do
          if odd(i) then suma:= suma + i;
        cuadrado_iter:=suma;
end; {cuadrado_iter}
```

```
function cuadrado_iter(n:integer):integer;
{calcula el cuadrado de n, utilizando las sumas de impares adecuados}
{Precondición: n>0 }
var impar,suma,i:integer;

begin {cuadrado_iter}
        suma := 0 ;
        impar := 1;
        for i :=1 to n do
        begin
          suma := suma + impar;
          impar := impar + 2;
        end
        cuadrado_iter:=suma;
end; {cuadrado_iter}
```

4. Estructuras de datos

1)
```
CONST LongTE = 9;
      LongNS = 12;
TYPE fecha = RECORD
        dia: 1..31;
        mes: 1..12;
        anyo: INTEGER;
END;
TYPE enfermo = RECORD
        FN: fecha;
        HO: STRING;
        DI: STRING;
        TE: STRING[LongTE];
        NS: STRING[LongNS];
        IN: BOOLEAN;
END;
```

2)
```
CONST k = 4;
TYPE jugador = RECORD
                    Posicion: 0..81;
                    Estado: (gana,pierde,sinAcabar);
               END;
TYPE partida = ARRAY[1..k] OF jugador;
```

3) Dos posibles soluciones, según se interprete el enunciado:
   - **Solución 1**: Para ambos casos, no sabemos el número máximo de enfermos y debemos emplear un fichero:
     FILE OF enfermo;
   - **Solución 2**:
     i. Se indica el número máximo de hospitales, pero no se indica el número máximo de pacientes por hospital. Por tanto, debemos suponer un número máximo de pacientes por hospital:
     ```
     CONST MAXCAMAS = 600;
     TYPE Hospial = ARRAY[1..MAXCAMAS] OF enfermo;
     TYPE TodosHosp = ARRAY[1..200] OF Hospital;
     ```
     También es posible:
     ```
     TYPE TodosHosp = ARRAY[1..200,1..MAXCAMAS] OF enfermos;
     ```
     ii. no sabemos el número máximo de enfermos y debemos emplear un fichero:
     FILE OF enfermo;

4) Este punto depende de la solución adoptada para el punto 3.
   i. Debemos acceder al array en la posición adecuada, dependiendo de cómo hayamos declarado el vector:
   ```
   paciente[hospital][cama].FN;
   paciente[hospital,cama].FN;
   ```

   ii. Para identificar un enfermo emplearemos el número de SACYL. Deberemos recorrer el fichero hasta localizar el que buscamos. Una vez encontrado (en la variable paciente), accedemos a la fecha:

   ```
   paciente.FN
   ```

5. Fichero glosario

```
Program fichero (input, output, f);
{extrae de un fichero el significado de una abreviatura si existe,  en otro caso la anterior y
posterior si existen. El fichero esta ordenado alfabéticamente}
CONST LONGABR = 10;
VAR   f:txt;
      AbrevUs,AbrevF, Anter, Poster:string[LONGABR];
      Signi:string;
      Encontrado:Boolean;
Begin
      Signi:=''; Anter:=''; Poster''; AbrevUs:=''; Encontrado:= false;  AbrevF:='';
      Assign (f,'glosario.dat');
      Reset (f); {fichero preparado para leer}
      Writeln ('de una abreviatura, de no mas de 10 caracteres');
      Readln(AbrevUs);

      {búsqueda en el fichero del significado de abrev}
      while not eof(f) and not encontrado do
            begin
                  readln(f, AbrevF);
                  if AbrevF = AbrevUs then
                        begin {se ha encontrado}
                          encontrado:=true;
                          Readln(f, Signi);
                        end
                  else
                        if AbrevF > AbrevUs then
                              begin {no esta en el fichero}
                                    Poster:=AbrevF;
                                    Encontrado:=true;
                              end
                        else
                              begin {no esta pero puede estar}
                                    Anter:=AbrevF;
                                    Readln(f, Signi);
                              end;
            end; {del while}
      if (Encontrado) and (AbrevUs =AbrevF) then
            writeln ('abreviatura encontrada', AbrevUs , '=  ', signi)
      else begin
            writeln ('abreviatura no encontrada');
            IF Anter <> '' THEN
                  writeln ('anterior = ', Anter);
            IF Poster <> '' THEN
                  writeln('Posterior = ', Poster);
      end;
      Close (f);
      End; {programa}
```

PRINCIPAL			
a	b	c	Instruccin
3	?	?	a := 3;
3	2	?	b := 2;
3	2	2	c := 2;
1	51	51	a:=f1(b,a);

Principal			F1					
a	b	c	d	b	a	i	f1	Instruccin
3	2	2	2	3	?	?	?	
3	2	2	2	3	4	?	?	a := d + 2;
3	2	2	2	6	4	1	?	b := b * c;
3	2	2	2	12	4	2	?	b := b * c;
3	2	2	2	24	4	3	?	b := b * c;
3	2	2	2	48	4	4	?	b := b * c;
3	2	3	2	48	4	4	?	c := c + 1;
3	52	3	52	48	4	4	?	d := b + a;
3	51	51	51	48	4	4	?	f2(b,c,d);
3	51	51	51	48	4	4	1	f1 := a+b-c;

Principal			F2				
a	b	c	b	c	a	d	Instruccin
3	52	3	48	3	52	?	
3	52	3	48	3	52	-1	d := b+c-a;
3	51	3	48	3	51	-1	a := b + c;
3	51	3	50	3	51	-1	b := a + d;
3	51	51	50	51	51	-1	c := b - d;