Examen - Programación 1

Instituto de Computación

Julio 2024

Leer con atención:

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje Pascal tal como fue dado en el curso.
- En todos los problemas se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso.
- Escribir las respuestas de un solo lado de la hoja. Entregar solamente las hojas de solución escritas a lápiz.
- En cada hoja entregada se debe incluir nombre, cédula y qué número de hoja es. En la primera hoja se debe incluir además la cantidad total de hojas entregadas.

Ejercicio 1 (49 puntos)

Dadas las siguientes declaraciones:

```
const
      N = ...  { N > 0 y par };
      F = N - 1;
      M = N div 2;

type
    NEquipo = 1..N;
    Partido = record
            equipo1 : NEquipo; goles1 : integer;
            equipo2 : NEquipo; goles2 : integer;
            end;
    Fecha = array [1..M] of Partido;

    Puntaje = record
            equipo: NEquipo;
            puntos: integer
            end;
    TablaPosiciones = array [1..N] of Puntaje;

    Disputados = record
            fechas : array [1..F] of Fecha;
            tope : 0..F
            end;
    PrimeraVictoria = record
            case victoria : boolean of
                true: (fec: 1..F);
                false: ()
            end;
```

y la siguiente función (asuma que ya está implementada y puede ser invocada donde se considere necesario):

```
function puntosEnPartido (equipo: NEquipo; part: Partido) : integer;
```

que dado un número de `equipo` retorna la cantidad de puntos obtenido por ese equipo en un partido `part` disputado.

a) (14 puntos)

Escribir la función:

```
function puntosEnFecha (equipo: NEquipo; nfecha: Fecha) : integer;
```

que dado un número de `equipo` retorna la cantidad de puntos obtenidos por ese equipo en una `fecha` disputada. En una fecha se juegan M partidos. En cada partido se enfrentan dos equipos, `equipo1` y `equipo2`, y se registran los goles anotados por los equipos, `goles1` y `goles2`, respectivamente. Si la cantidad de goles anotados por un equipo es mayor que la de su oponente obtiene 3 puntos, si es igual, 1 punto y si es menor, 0 puntos. Cada equipo participa en un partido, y solamente uno, por fecha.

Solución:

```
function puntosEnFecha (equipo: NEquipo; nfecha: Fecha) : integer;
var
    i:integer;
begin
    i := 1;
    while (nfecha[i].equipo1 <> equipo) and (nfecha[i].equipo2 <> equipo) do
        i := i +1;
    puntosEnFecha := puntosEnPartido(equipo, nfecha[i])
end;
```

b) (25 puntos) Escribir el procedimiento:

```
procedure actualizarTabla (equipo: NEquipo; puntos: integer; var tab: TablaPosiciones);
```

que actualiza la tabla de posiciones del campeonato con los `puntos` obtenidos por el `equipo` en una fecha. La tabla de posiciones registra los puntos obtenidos por cada equipo, en orden decreciente de puntos. En caso de que un grupo de equipos tenga la misma puntuación, el último en alcanzarla quedará ubicado último dentro de dicho grupo. **Ejemplo**: Dadas las tabla ordenada $equipo1 : 12, equipo2 : 11, equipo3 : 10, equipo4 : 9$, si el $equipo4$ obtiene 3 puntos, la nueva tabla será $equipo1 : 12, equipo4 : 12, equipo2 : 11, equipo3 : 10$.

Solución:

```
procedure actualizarTabla (equipo: NEquipo; puntos: integer; var tab: TablaPosiciones);
var
    i,j,ptosequipo:integer;
begin
    i := 1;
    while tab[i].equipo <> equipo do i:=i+1;
    ptosequipo := puntos + tab[i].puntos;
    j := i;
    while (1 < j) and (ptosequipo > tab[j-1].puntos) do
    begin
        tab[j] := tab[j-1];
        j := j-1;
    end;
    tab[j].equipo := equipo; tab[j].puntos := ptosequipo
end;
```

c) (10 puntos)

Escribir el procedimiento:

```
procedure primerPartidoGanado (equipo: NEquipo; disp: Disputados; var posibleVictoria : PrimeraVictoria);
```

que retorna en `posibleVictoria`, en caso de haber ganado al menos un partido de los disputados, el número de la fecha en el que ganó el primero de ellos. En un campeonato se disputan, como máximo, `F` fechas.

Solución:

```
procedure primerPartidoGanado (equipo: NEquipo; disp: Disputados; var posibleVictoria : PrimeraVictoria);
var i:integer;
begin
    i := 1;
    while (i <= disp.tope) and (puntosEnFecha(equipo,disp.fechas[i]) <> 3) do
        i := i+1;
    posibleVictoria.victoria := i <= disp.tope;
    if posibleVictoria.victoria then
        posibleVictoria.fec := i
end;
```

Ejercicio 2 (35 puntos)

Una empresa se encuentra instalando antenas, para lo cual se definen los siguientes tipos:

```
const
    N = .. ; { N > 0 }
    M = .. ; { M > 0 y M par }

type
    RangoAntenas = 1..N;
    TDistAntenas = array [1..M] of RangoAntenas;
    TEstado = array [RangoAntenas] of boolean;

    ListaPares = ^Nodo;
    Nodo = record
        ant1,ant2 : RangoAntenas;
        sig : ListaPares;
    end;
```

Las antenas están identificadas de 1 hasta N. Es de interés agrupar M antenas, de a pares, en una lista. Los identificadores de cada antena pueden repetirse en más de un par, estar en un único par o no estar en ninguno.

a) (25 puntos) Escribir la función:

```
function crearListaAntenas(antenas : TDistAntenas) : ListaPares;
```

que toma una distribución de antenas en el parámetro `antenas` y retorna la lista correspondiente a agrupar las antenas de a pares consecutivos. **Ejemplo**: si $N = 4$, $M = 6$ y $antenas = [1, 2, 3, 4, 1, 4]$ la función `crearListaAntenas` retorna la lista $[(1, 2), (3, 4), (1, 4)]$.

Solución:

```
function crearListaAntenas(antenas : TDistAntenas) : ListaPares;
var
   i : integer;
   res, aux : ListaPares;
begin
(*  Creo el primer nodo *)
    new(res);
    res^.ant1 := antenas[1];
    res^.ant2:= antenas[2];
    aux := res;
(* Creo resto   (notar que si M = 2 el for no se ejecuta) *)
    for i := 1 to (M div 2) - 1 do
    begin
        new(aux^.sig);
        aux := aux^.sig;
        aux^.ant1 := antenas[2*i+1];
        aux^.ant2 := antenas[2*i+2];
    end;
(*  Completo expresión booleana y retorno  *)
    aux^.sig := NIL;
    crearListaAntenas := res
end;
```

b) (10 puntos) Luego de instaladas, se observan fallas en varias antenas, por lo que se desea comprobar si la conexión sigue siendo estable. La conexión es estable si existe al menos un par en la lista tal que ambas antenas funcionan correctamente. Escribir la función:

```
function conexionEstable(estadoAntenas : TEstado; listaAntenas : ListaPares) : boolean;
```

que toma como entrada una asignación `estadoAntenas` de valores booleanos que indican el estado de cada antena (`true` si funciona, `false` si no funciona) y una lista de pares de antenas en el parámetro `listaAntenas`, y retorna `true` si la conexión es estable o `false` si no lo es. Si la lista es vacía la función retorna el valor `false`.

Solución:

```
function conexionEstable(estadoAntenas : TEstado; listaAntenas: ListaPares): boolean;
begin
    while (listaAntenas <> NIL) and (not (estadoAntenas[listaAntenas^.ant1] and estadoAntenas[listaAntenas^.
        ant2])) do
        listaAntenas := listaAntenas^.sig;
    conexionEstable := listaAntenas <> NIL
end;
```

Ejercicio 3 (16 puntos)

Considere el siguiente programa

```
program alcance;
var
   n : integer;

   procedure p (a : integer; var b : integer);

      function f (n : integer ): integer;
      ...

   begin
     a := a + 1;
     b := f(a)
   end;

begin
   read(n);
   p (n,n);
   writeln('El doble del valor leido es:', n);
end.
```

en el que no se incluye la implementación de la función `f`. Implemente la función `f` para que el programa imprima el doble del valor leído.

Solución:

```
     function f (n : integer ): integer;
     begin
        f := (n-1)*2
     end;
```