Segundo Parcial. Programación 1

Instituto de Computación

Noviembre 2007

Observaciones:

- Todos los programas o fragmentos de programas, deben ser escritos en el lenguaje **Pascal** tal como fue dado en el curso. A grandes rasgos este es el Pascal estándar con algunos agregados a saber:
  - Utilización de `else` en la instrucción `case`. Si no se especifica `else` se ejecutará la sentencia siguiente al `case` en caso de no coincidir con ninguna de las etiquetas.
  - Evaluación por circuito corto de los operadores booleanos `AND` y `OR`.
- 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. De esta manera se restarán puntos entre otros conceptos por: mala o nula indentación, mala utilización de las estructuras de control, código confuso e innecesariamente largo, programas ineficientes, utilización de variables globales, pasaje incorrecto de parámetros, etc.

  No obstante, por razones prácticas no exigimos que incluya comentarios en los códigos que escriba en la prueba.
- Escriba su nombre completo y cédula en todas las hojas. Numere todas las hojas entregadas y escriba en todas la cantidad total de hojas.
- Escriba de un solo lado de cada hoja, un ejercicio por hoja.

Ejercicio 1

Se define la función: *f* tal que

$$f : R - \{0\} \times N \rightarrow R$$

$f(x,n) = n! / x^n$, *con x distinto de 0*

que corresponde al enésimo término de una secuencia de términos.

a) Se pide implementar esa función en pascal en forma **recursiva**:

```
function enesimo_termino(N: Natural; x: real): real;
```

donde el tipo `Natural` se define como:

```
type Natural = 0..MaxInt;
```

b) Implementar en pascal la misma función en forma **iterativa**.

Ejercicio 2

Dadas las siguientes declaraciones:

```
const
   Max_personas = . . .;  {algún valor apropiado}
   Max_grupos = . . .;  {algún valor apropiado}

type
   TNombre = ... ;  {algún tipo apropiado}

   TFecha = record
               mes: 1 .. 12;
               dia: 1 .. 31;
            end;

   TPersona = record
                  nombre: TNombre;
                  edad:   0 .. 100;
                  fecha : TFecha;
              end;

   TGrupo = record
               personas: array [1..Max_personas] of TPersona;
               tope: 0 .. Max_personas;
            end;

   TCon_Responsable = record
                        responsable : TNombre;
                        grupo: TGrupo
                      end;

   TLista = array [1 .. Max_grupos] of TCon_Responsable;
```

Se dispone de la función `comparaNombres` que dados dos nombres determina si son iguales o no, devolviendo `True` en caso afirmativo y `False` si no. El cabezal de la función es:

```
function comparaNombres(nom1,nom2:TNombre):Boolean;
```

Esta función **no** hay que escribirla, si la necesita asuma que ya está escrita.

La lista de grupos está completa, o sea existen `Max_grupos` grupos en ella. No está ordenada.

Dentro de cada grupo, las personas están ordenadas por fecha. Se asume que no hay fechas iguales. Las fechas son todas del mismo año (solo importa el mes y el día).

Se pide:

a) Escribir un procedimiento

`procedure ingresar(per: TPersona, resp: TNombre, var list: TLista);`

que dada una persona (`per`), el nombre del responsable del grupo (`resp`), y una lista de grupos (`list`), ingrese a la persona (per) en el grupo

correspondiente al responsable (resp), y en el lugar correcto según el orden dado por la fecha.
Si el nombre de responsable no se encuentra en la lista, la persona se insertará en el último grupo de la lista, respetando el orden de las fechas. Se asume también que cada responsable tiene asignado un solo grupo.

b) Escribir un procedimiento

```
procedure mayorEdad(list:TLista; var resp:TNombre; var per:TPersona);
```

que dada una lista de grupos (list), devuelve la persona (per) de mayor edad de la lista y el nombre (resp) del responsable del grupo al que pertenece la persona (per).
Se supone que no hay edades repetidas.

Por ejemplo, dada la siguiente lista de grupos:

Responsable	**Grupo**
Luis	Pedro, 35, 11/05 Ricardo, 28, 11/06 Sofia, 10, 23/06 Mario, 66, 12/07
Santiago	Miriam, 47, 27/04 Oscar, 33, 12/09 Carlos, 24, 10/11
Carlos	Juan, 50, 22/03 Estela, 34, 30/03 Mirta, 30, 05/12

Al ingresar la persona Esteban, de 45 años de edad, con fecha 14/10, al grupo cuyo responsable es Santiago, se obtiene la siguiente lista:

Responsable	**Grupo**
Luis	Pedro, 35, 11/05 Ricardo, 28, 11/06 Sofia, 10, 23/06 Mario, 66, 12/07
Santiago	Miriam, 47, 27/04 Oscar, 33, 12/09 Esteban, 45, 14/10 Carlos, 24, 10/11
Carlos	Juan, 50, 22/03 Estela, 34, 30/03 Mirta, 30, 05/12

Donde Esteban queda en el grupo de Santiago, y dentro de ese grupo después de Oscar y antes de Carlos, según el orden por fecha.

Si no hubiera ningún responsable con nombre Santiago en la lista, la persona debe ingresarse en el último grupo respetando el orden de las fechas, en este caso en el grupo de Carlos antes de Mirta.

Dada esta última lista, el procedimiento `mayorEdad` debe dar como resultado la persona Mario, 66, 12/07 y en responsable Luis.

Ejercicio 3

Determinar la salida del siguiente programa, considerando que como entrada se ingresa el dígito de unidad de su cédula de identidad.

```
program alcance(input,output);

VAR var1:INTEGER;
    var2:INTEGER;

Function f1(a:INTEGER):INTEGER;
BEGIN
    var1:= a + 2;
    f1:= var1 * 2;
END;

Procedure p1(var var3:INTEGER);
BEGIN
  var3:= f1(var3);
  var1:=var3 * 2;
END;

Procedure p2(var2:INTEGER);
VAR var1: INTEGER;
BEGIN
  var1:=2;
  var2:= f1(var1) + var1;
  WriteLn(var1);
  WriteLn(var2);
END;

BEGIN
  Read(var1);
  var2:= var1 + 2;
  p2(var1);
  WriteLn(var2);
  p1(var2);
  WriteLn(var1);
  WriteLn(var2);
end.
```