Segundo Parcial
Instituto de Computación - Facultad de Ingeniería
Noviembre 2016

Leer con atención

- 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`.
  - Evaluación por circuito corto de las operaciones booleanas (`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 y escriba la cantidad total de hojas.
- Escriba de un solo lado de la hoja y comience cada ejercicio en una nueva hoja.

Ejercicio 1

Dado el siguiente programa, escribir cuál será su salida cuando la variable x se carga de la entrada estándar con **el último dígito** de su CI (antes del dígito verificador). Por ejemplo, si su CI es 1.234.567-8, el último dígito es 7:

```
program ejercicio;
var x: integer;

procedure primero (var a: integer; var b: integer);
var  x : integer;
begin
  a := a + 1;
  b := b * 2;
  x := a - b
end; (*primero*)

procedure segundo (a: integer; b: integer);
var x : integer;
begin
  a := a + 1;
  b := b * 2;
  x := a + b;
  writeln (x)
end; (*segundo*)

procedure tercero (a: integer; var b: integer);
begin
  a := a + 1;
  b := b * 2;
  segundo (a,b);
  writeln (x)
end; (*tercero*)

begin
  readLn (x);
  tercero (x,x);
  writeLn (x);
  primero (x,x);
  writeLn (x)
end.
```

Ejercicio 2

Dadas las siguientes declaraciones que permiten registrar información sobre una variante simplificada de las criaturas mitológicas popularmente conocidas como **pokemones**:

```
CONST
    MaxCad = ...;
    MaxPok = ...;

TYPE
    TipoPokemon = (comun, naturaleza, electrico);
    Natural = 0..Maxint;
    Cadena  = array [1..MaxCad] of char;
    Pokemon = record
                codigo: Natural;
                nombre: Cadena;
                case tipo: TipoPokemon of
                    comun:      (edad: Natural);
                    naturaleza: (acuatico: boolean);
                    electrico:  (voltaje: real);
              end;
    Pokedex = record
                arre: array [1..MaxPok] of Pokemon;
                tope: 0..MaxPok;
              end;
    Codigos = ^Celda;
    Celda   = record
                codigo: Natural;
                siguiente: Codigos;
              end;
```

Escribir los subprogramas que se piden a continuación:

Parte a)

Escribir la función *TerminaEnVocal* que, dada una cadena, determina si su primera palabra termina en una letra vocal. Asuma que la cadena contiene únicamente letras y espacios en blanco separando palabras (también puede haber espacios en blanco al inicio).

Ejemplos para MaxCad = 12:

- para **|** | | |P|i|k|a|c|h|u| | **|** la función devuelve TRUE
- para **|**M|R| | |S|q|u|i|r|t|l|e**|** la función devuelve FALSE
- para **|** | | | | | | | | | | **|** la función devuelve FALSE

```
    function TerminaEnVocal (cad: Cadena): boolean;
```

Parte b)

Escribir la función *HayCantPokemones* que, dada una cantidad, determina si en la pokedex hay al menos esa cantidad de pokemones tales que la primera palabra de su nombre termina en una letra vocal.

```
    function HayCantPokemones (cant: Natural; conj: Pokedex): boolean;
```

Parte c)

Escribir el procedimiento *EliminaMayorEdad* que devuelve en el parámetro poke el pokemon de tipo *comun* con la mayor edad de toda la pokedex y también lo elimina de la misma, manteniendo a los demás pokemones en el mismo orden en que están. Asuma que hay al menos un pokemon de tipo *comun* en la pokedex y que todos los que hay tienen diferentes edades.

```
    procedure EliminaMayorEdad (var conj: Pokedex; var poke: Pokemon);
```

Parte d)

Escribir el procedimiento *CodigosPorTipo* que, dado un tipo de pokemon, devuelve una lista conteniendo los códigos de todos los pokemones de ese tipo que hay en la pokedex. Deben quedar insertados en la lista en el mismo orden en que ocurren en la pokedex.

```
    procedure CodigosPorTipo (tipo: TipoPokemon; conj: Pokedex; var lista: Codigos);
```