Segundo Parcial - Programación 1
Instituto de Computación - Facultad de Ingeniería
Noviembre 2017

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

Considere las siguientes declaraciones:

```
const MAX  = ...; (* MAX > 0 *)

type  Rango = 1 .. MAX;
      Tipo = (entero,caracter);

      IntChar = record
                  case tipoElem : Tipo of
                    entero   : ( intElem : Integer; );
                    caracter : ( chrElem : Char;    );
                end;

      ArrIntChar = array [Rango] of IntChar;

      ArrChar = record
                   elems : array [Rango] of Char;
                   tope  : 0..MAX;
                end;

      Resultado = record
                     sumPares : Integer;
                     chars    : ArrChar;
                  end;
```

Escribir los subprogramas que se piden a continuación:

Parte a)

Implementar la función:

```
      function HayKConsecutivosTipo (arr : ArrIntChar; k : Rango; t : Tipo) : boolean;
```

que, dados un arreglo *arr* de enteros y caracteres, un número *k* mayor que cero y un tipo *t*, devuelve True si existen al menos *k* elementos consecutivos del tipo *t* en el arreglo *arr.* En caso contrario se devuelve False.

Ejemplos:

- con arr = [2, 34, 'a', 'b', 'c', 7, 4], k = 3 y t = carácter, la función devuelve True
- con arr = [2, 34, 'a', 'b', 'c', 7, 4], k = 3 y t = entero, la función devuelve False

Parte b)

Implementar el procedimiento:

```
procedure ArrToResult (arr : ArrIntChar; var res : Resultado);
```

que, dados un arreglo *arr* de enteros y caracteres, devuelve en el record *res*:

- en el campo *sumPares* la suma de los enteros pares contenidos en el arreglo *arr;*
- en el campo *chars* todos los elementos de tipo carácter contenidos en *arr* y en el mismo orden en que allí aparecen.

Ejemplo:

- con arr = [2, 34, 'a', 'b', 'c', 7, 4], se devuelve res = 40, ['a', 'b', 'c']
- con arr = [1, 'a', '&', 5, 7, 'c', 3], se devuelve res = 0, ['a', '&', 'c']

Ejercicio 2

Considere las siguientes declaraciones:

```
const MAX  = ...; (* MAX > 0 *)

      TArr = array [1..MAX] of Integer;

      Lista    = ^TCelda;
      TCelda   = record
                    val : Integer;
                    sig : Lista;
                 end;
```

Escribir los subprogramas que se piden a continuación:

Parte a)

Implementar la función:

```
function ArrToList (arr : TArr) : Lista;
```

que, dado un arreglo *arr,* devuelve una lista conteniendo los valores de *arr* en el mismo orden en que aparecen en el arreglo.

Ejemplo:

- con arr = [2, 34, 7, 4], la función devuelve la lista <2, 34, 7, 4>

Parte b)

Implementar el procedimiento:

```
procedure ListToArr (l : Lista; var arr : TArr);
```

que, dada una lista *l*, devuelve en *arr* los elementos contenidos en *l* en el mismo orden en el que figuran en la lista, teniendo en cuenta las siguientes consideraciones:

- En el caso de que *l* tenga más de *MAX* elementos, se deben almacenar en *arr* solamente los primeros *MAX* elementos.
- Si *l* contiene menos de *MAX* elementos, se completarán con el valor 0 las restantes celdas del arreglo hasta llegar a *MAX*.

Ejemplos (asumiendo *MAX* = 4):

- con l = <3, 1, 9, 8>, devuelve el array [3, 1, 9, 8]
- con l = <3, 1, 9, 8, 4, 6>, devuelve el array [3, 1, 9, 8]
- con l = <3, 1>, devuelve el array [3, 1, 0, 0]

Ejercicio 3

Dado el siguiente programa, escribir cuál será su salida cuando la variable **a** 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 ejercicio3;
var a,b: integer;

procedure proc0 (var a: integer; b: integer);
begin
  a := b + 1;
  b := a + 2;
  writeln(a)
end; (*proc0*)

procedure proc1 (a: integer; var b: integer);
var x : integer;

  procedure proc11 (var a: integer; b: integer);
  begin
    a := a + b;
    x := a + 2;
    writeln(x)
  end; (*proc11*)

begin
  proc11 (a,b);
  x := x - b;
  b := x * 2
end; (*proc1*)

function func (a: integer; b: integer): integer;
begin
  proc0 (a,a);
  func := a + b
end;(*func*)

begin
  readln(a);
  b := succ(a);
  proc1(a,b);
  b := a div 2;
  b := func(a,b);
  writeln(b)
end.
```