Segundo Parcial - Programación 1
Instituto de Computación - Facultad de Ingeniería
Noviembre 2018

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
  MaxDatos = ...; (* algún valor apropiado *)

type
  RangoInfo = 1..MaxDatos;
  RangoTope = 0..MaxDatos;
  Natural   = 0..MaxInt;
  Valores   = record
                info : ARRAY [RangoInfo] OF Natural;
                tope : RangoTope;
              end;
```

Donde *Valores* es un arreglo con tope que contiene números naturales.

Parte a)

Escribir un procedimiento que lea una secuencia de naturales de la entrada estándar y los guarde en un arreglo con tope de tipo *Valores*. La lectura se termina cuando el usuario ingresa un número negativo o cuando se completa el arreglo (con *MaxDatos* naturales). Tenga en cuenta que la secuencia de naturales ingresada puede ser vacía (si el primer valor que se ingresa es negativo).

```
      procedure LeoValores (var arr : Valores);
```

Ejemplo (asumiendo MaxDatos = 4):

- con entrada 10 3 5 -10, retorna (arr.info = [10,3,5,?], arr.tope = 3)
- con entrada 10 3 5 8 9 -1, retorna (arr.info = [10,3,5,8], arr.tope = 4)
- con entrada -15 2 4, retorna (arr.info = [?,?,?,?], arr.tope = 0)
- con entrada 4 -15 2 4, retorna (arr.info = [4,?,?,?], arr.tope = 1)

Parte b)

Escribir una función que, dado un arreglo con tope *arr*, devuelva true si existe una secuencia no vacía de valores consecutivos en el arreglo cuya suma sea igual a un natural *num* dado, y false en caso contrario.

```
     function consecutivosSumanN (arr : Valores; num : Natural) : Boolean;
```

Ejemplo (asumiendo MaxDatos = 4):

- con (arr.info = [10,3,5,4], arr.tope = 4) y num = 8, retorna true
- con (arr.info = [10,3,5,4], arr.tope = 4) y num = 19, retorna false
- con (arr.info = [10,3,5,4], arr.tope = 4) y num = 22, retorna true
- con (arr.info = [10,3,5,4], arr.tope = 4) y num = 0, retorna false
- con (arr.info = [?,?,?,?], arr.tope = 0) y num = 10, retorna false

Ejercicio 2

Dada la siguiente definición de tipos, que modela a un Parking diario:

```
const MAX_AUTOS = ... (* algún valor apropiado *)

type

  Hora = record
              hora : 0..23;
              min  : 0..59;
              seg  : 0..59
         end;

  Lugar = record
          matricula: Integer;
            case ocupado : boolean of
               true  : (hora_ingreso : Hora);
               false : ()
           end;

  Parking = array [1 .. MAX_AUTOS] of Lugar;
```

Donde cada celda del arreglo *Parking* representa un lugar reservado para un vehículo. Para cada lugar se tiene la matrícula (de tipo *Integer*) del vehículo asignado y, en caso de estar actualmente ocupado, la hora en que ingresó. Como la entrada al Parking es estrecha, no pueden ingresar dos vehículos exactamente a la misma hora.

Parte a)

Escribir la función *posteriorHora*, que dadas dos horas *h1* y *h2*, determina si *h1* es posterior a *h2*.

```
      function posteriorHora(h1, h2 : Hora) : Boolean;
```

Parte b)

Escribir la función *ultimoIngreso*, que devuelve la matrícula del último auto que ingresó al parking. Si todos los lugares están libres retorna -1.

```
     function ultimoIngreso(park : Parking) : Integer;
```

Ejercicio 3

Considere la siguiente declaración de tipos:

```
type
    Lista = ^Nodo;

    Nodo = record
        elem : char;
        sig  : Lista
    end;
```

Escribir un procedimiento que, dada una lista *l* de tipo *Lista*, un carácter *val* y una posición *pos* (empezando en 1), inserta el elemento en esa posición de la lista.

```
      procedure InsPos(val : char; pos : integer; var l : Lista );
```

Si la posición excede al largo de la lista, el elemento se inserta al final.

Ejemplo:

- con val = 'a', pos = 1 y l = <'x','y','z'>, retorna l = <'a','x','y','z'>
- con val = 'a', pos = 2 y l = <'x','y','z'>, retorna l = <'x','a','y','z'>
- con val = 'a', pos = 4 y l = <'x','y','z'>, retorna l = <'x','y','z','a'>
- con val = 'a', pos = 10 y l = <'x','y','z'>, retorna l = <'x','y','z','a'>
- con val = 'a', pos = 10 y l = <>, retorna l = <'a'>

Ejercicio 4

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

  procedure proc(a,b : integer; var c : integer);

    function func(a: integer) : integer;
    var b: integer;
    begin
      b := c + a;
      func := b + c;
    end;

  begin
    c := b + c;
    writeln(func(c));
    writeln(a+b);
  end;

begin
  readln(dig);
  a := dig;
  b := dig + 2;
  proc(dig,a,b);
  writeln(a);
  writeln(b);
end.
```