Segundo Parcial
Instituto de Computación - Facultad de Ingeniería
Julio 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
   MAX_LETRAS = ...;
   TOTAL_PALABRAS = ...;

type
   TLetras = 'a' .. 'z';
   Texto = record
              letras: array[1 .. MAX_LETRAS] of TLetras;
              tope: 0 .. MAX_LETRAS;
           end;
   Diccionario = array[1 .. TOTAL_PALABRAS] of Texto;
```

Parte a)

Implementar la función:

```
     function prefijo(t1,t2: Texto): Boolean;
```

que, dados dos textos, devuelve TRUE si el primero es un prefijo del segundo.

Ejemplos para MAX_LETRAS = 6 (se omiten las comitas de los chars por simplicidad)

```
t1.letras:[p,a,?,?,?,?], t1.tope:2 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve TRUE
t1.letras:[p,a,p,e,l,?], t1.tope:5 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve TRUE
t1.letras:[p,a,d,?,?,?], t1.tope:3 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve FALSE
t1.letras:[?,?,?,?,?,?], t1.tope:0 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve TRUE
```

Parte b)

Implementar la función:

```
       function prediccion(t: Texto; dic: Diccionario): Integer;
```

que, dados un texto t y un diccionario, devuelve el índice de la primera palabra del diccionario que tiene al texto t como prefijo. Si no existe ningún elemento que cumpla con lo pedido la función devuelve el valor 0.

Ejemplos para MAX_LETRAS = 6 y TOTAL_PALABRAS = 4 (se omiten las comitas de los chars por simplicidad)

```
dic: [ [p,a,p,e,l,?],tope:5 | [c,a,s,a,?,?],tope:4 | [d,e,?,?,?,?],tope:2 | [c,a,o,s,?,?],tope:4 ]
t: letras: [c,a,?,?,?,?],tope:2 → prediccion devuelve el índice 2 (palabra casa)
t: letras: [d,e,?,?,?,?],tope:2 → prediccion devuelve el índice 3 (palabra de)
t: letras: [c,a,l,?,?,?],tope:2 → prediccion devuelve el valor 0
```

Parte c)
Implementar la función:

```
     function distancia(t1,t2: Texto): Integer;
```

que, dados dos textos, devuelve la cantidad de letras diferentes entre ellos, comparando posición a posición, más la diferencia entre los topes. Es decir, calcula la cantidad de elementos que cumplen `t1.letras[i] <> t2.letras[i]` más `abs(t1.tope - t2.tope)`.

Ejemplos para MAX_LETRAS = 6 (se omiten las comitas de los chars por simplicidad)
```
t1.letras:[p,s,d,e,l,?], t1.tope:5 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve 2
t1.letras:[p,s,d,?,?,?], t1.tope:3 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve 4
t1.letras:[p,a,p,e,l,?], t1.tope:5 y t2.letras:[p,a,p,e,l,?], t2.tope:5, la función devuelve 0
```

Ejercicio 2

Dada una lista de enteros definida como:

```
type
   ListaEnt = ^Celda;
   Celda = record
              elem: Integer;
              sig: ListaEnt
           end;
```

Escribir el procedimiento:

```
      procedure duplicarN(n: Integer; var l : ListaEnt);
```

que agrega una celda con el valor **n** inmediatamente después de la primera celda de la lista original que contenga **n**. Si **n** no está en la lista, esta no se modifica.

Ejemplos:
```
n= 3, l = [12, 3, 4, 0] → después de ejecutar el procedimiento: l = [12, 3, 3, 4, 0]
n= 3, l = [ 3, 3, 5, 3] → después de ejecutar el procedimiento: l = [ 3, 3, 3, 5, 3]
n= 3, l = [12, 2, 4, 1] → después de ejecutar el procedimiento: l = [12, 2, 4, 1]
n= 3, l = []            → después de ejecutar el procedimiento: l = []
```

Ejercicio 3

Considere las siguientes declaraciones para representar dígitos, en donde cada dígito puede ser representado como un `Integer` o como un `Char`:

```
type
   DigitoC = '0' .. '9';
   DigitoN = 0 .. 9;
   TipoDato = (num, car);
   Valor = record
              case tipo: TipoDato of
                 num: (valorN: DigitoN);
                 car: (valorC: DigitoC)
           end;
```

Escribir la función:

```
     function sumaValores(v1, v2 : Valor) : Integer;
```

que, dados dos valores de tipo Valor, calcula su suma y la devuelve como entero.
Ejemplos
```
v1: tipo=num, valorN=5 y v2: tipo=car, valorC='2'   → la función retorna el Integer 7.
v1: tipo=car, valorN='9' y v2: tipo=car, valorC='2' → la función retorna el Integer 11.
```