Segundo Parcial

Instituto de Computación - Facultad de Ingeniería

Noviembre 2011

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 sólo lado de la hoja y comience cada ejercicio en una nueva hoja.

Ejercicio 1

Escribir una función:

```
type
    Natural = 0..MaxInt;

function raiz_cuadrada_entera(numero: Natural): Natural;
```

que calcula la parte entera de la raíz cuadrada de un número natural:

El algoritmo se debe basar en la siguiente observación que surge de mirar la tabla de los primeros resultados:

Numero	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
Raíz	0	1	1	1	2	2	2	2	2	3	3	3	3	3	3	3	4	...

El 0 está repetido una vez. , el 1 tres veces. , el 2 cinco veces , el 3 siete veces.
El número de veces que el entero **n** se repite es **(2n+1)**

El código que se implemente sólo puede realizar operaciones aritméticas de suma. No se permite utilizar ninguna función auxiliar ni predefinida. No se puede utilizar arreglos ni listas.

Ejercicio 2

Dadas las siguientes declaraciones, donde cada constante tiene un valor apropiado:

```
const
   MAX_PALABRAS = ...;
   MAX_LARGO_PALABRA = ...;

type
   TipoPalabra =  record
                     cadena : array [1..MAX_LARGO_PALABRA] of char;
                     tope : 0 .. MAX_LARGO_PALABRA;
                  end;

   ListaPalabras= array [1 .. MAX_PALABRAS] of TipoPalabra;
```

Escribir la función *cantidad_con_letra* que dada una lista de palabras llamada *lista*, devuelva la cantidad de palabras que tienen el caracter *c*.

```
function cantidad_con_letra(lista : ListaPalabras; c : CHAR): Integer;
```

Ejercicio 3

Sea las siguientes declaraciones:

```
type
      TipoRango = 1..MAX;
      TipoArreglo = array [TipoRango] of real;
```

donde se considera la constante `MAX` previamente definida con algún valor apropiado.

a) Escribir un procedimiento:

```
procedure ObtenerIntervalo(arreglo: TipoArreglo;  inferior,superior: real;
                           var resultado: TipoIntervalo);
```

El arreglo se supone ordenado de forma creciente. Se desea obtener el intervalo del arreglo cuyos valores de celdas están entre los valores inferior y superior.

El `TipoIntevalo` se define así:

```
type
     TipoIntervalo = record
          case vacio : boolean of
             true:  ();
             false: (inicio,fin: TipoRango);
          end;
```

El campo `vacio` debe ser `true` en el caso que no haya ningún valor entre `inferior` y `superior`. En otro caso los campos inicio y fin deberán tener valores **M** y **N** tales que:

$$\text{inferior} \leq \text{arreglo}[i] \leq \text{superior}$$

para todos los valores de i comprendidos en el rango `M .. N`.

b)

Se considera la siguiente definición de lista:

```
type
      TipoLista = ^nodo;
      nodo = record
               info: real;
               sig: TipoLista;
             end;
```

Escribir una función:

```
function intervalo_en_lista(arreglo: TipoArreglo; intervalo: TipoIntervalo) : TipoLista;
```

que retorna en una lista todos los elementos del arreglo que se corresponden con el intervalo especificado por el parámetro correspondiente. Los valores de la lista resultante tienen que aparecer en el mismo orden que aparecían en el arreglo.