Parcial - Programación 1
Diciembre 2021

Leer con atención:

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje Pascal tal como fue dado en el curso.
- 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.
- Entregue solamente las hojas de solución escritas a lápiz.

Ejercicio 1 (16 puntos)

Se consideran las siguientes declaraciones de constantes y tipos:

```
const
   MAX = ...;   {valor apropiado}
type
   Rango = 1 .. MAX;

   Matriz = array [Rango,Rango] of Rango;
```

Escribir una función:

```
function  esUnoMax(i: Rango; M: matriz) : boolean;
```

que retorna `true` si la fila i-ésima `M[i]` es de la forma: [1, 2, ... , `MAX`]. Es decir: `M[i,k]` = `k` para todo valor de `k` en `Rango`.

Ejemplos:

Consideremos `MAX` = 5 y la matriz `M`:

2	3	2	1	5
1	2	3	4	5
2	2	3	4	2
2	3	4	5	4
1	2	3	4	3

La siguiente tabla muestra los valores que retorna la función `esUnoMax` con la matriz `M` dada, para algunos valores de `i`:

i	esUnoMax(i,M)
1	false
2	true
5	false

Solución

```
function esUnoMax (i : Rango; M : matriz) : boolean;
var k : integer;
begin
  k := 1;
  while (k <= MAX) and (M[i,k] = k) do
    k := k + 1;
  esUnoMax := k > MAX
end;
```

Ejercicio 2 (16 puntos)

Sea la siguiente definición de *palíndromo*:

> Un palíndromo es una secuencia de caracteres que es igual si se lee de izquierda a derecha que de derecha a izquierda

Se considera una representación de *cadena de caracteres* como un *array con tope*:

```
const N = ...; (* valor mayor que 1*)
type
   TCadena = record
                cars : array [1 .. N] of Char;
                tope : 0 .. N
             end;
```

Escribir el procedimiento:

```
procedure crearPalindromo(var cadena: TCadena);
```

que crea un palíndromo con el doble de largo que `cadena`, agregando a la misma los caracteres que sean necesarios. Si el arreglo no tiene celdas suficientes para tal operación, el procedimiento no hace nada.

Ejemplos: Para `N` = 8. En la siguiente tabla se muestra el arreglo con tope antes y después de la invocación de `crearPalindormo(cadena)`:

Antes	Después
cadena.cars: `|'a'|'b'|'c'|?|?|?|?|?|` cadena.tope: 3	cadena.cars: `|'a'|'b'|'c'|'c'|'b'|'a'|?|?|` cadena.tope: 6
cadena.cars: `|'a'|'b'|?|?|?|?|?|?|` cadena.tope: 2	cadena.cars: `|'a'|'b'|'b'|'a'|?|?|?|?|` cadena.tope: 4
cadena.cars: `|'a'|'b'|'c'|'d'|'e'|?|?|?|` cadena.tope: 5	cadena.cars: `|'a'|'b'|'c'|'d'|'e'|?|?|?|` cadena.tope:5

Solución

```
procedure crearPalindromo(var cadena : TCadena);
var i : 1..N;
begin
  if cadena.tope <= N div 2 then
  begin
    for i := 1 to cadena.tope do
      cadena.cars[cadena.tope+i] := cadena.cars[cadena.tope-i+1];
    cadena.tope := cadena.tope * 2
  end
end;
```

Ejercicio 3 (22 puntos)

Dadas las siguientes declaraciones:

```
const
   MAX = ...;    {valor apropiado}
type
   FormaGeometrica = (triangulo, rectangulo, circulo);

   Figura = record case forma: FormaGeometrica of
      triangulo :  (base, altura   : integer);
      rectangulo:  (largo, ancho   : integer);
      circulo   :  (radio          : integer);
   end;

   ArregloFiguras = array [1..MAX] of Figura;

   ListaFiguras = ^celda;
   celda = record
      elemento  : Figura;
      siguiente : ListaFiguras
   end;
```

donde: `Figura` representa una figura geométrica que puede ser un triángulo, un rectangulo o un círculo, `ArregloFiguras` es un *array* de figuras, mientras que `ListaFiguras` es una lista encadenada de figuras.

Escribir una función con encabezado:

```
function ObtenerFiguras (arrFiguras : ArregloFiguras;
                         n          : integer;
                         cotaArea   : integer       ) : ListaFiguras;
```

que retorna una lista de las `n` primeras figuras del arreglo cuya área sea menor o igual que `cotaArea`. Si el arreglo tiene menos de `n` figuras que cumplan con lo pedido, se retornará la lista con todas las figuras que lo cumplan. El orden de las figuras en la lista es irrelevante.

Se puede asumir que `n` es mayor o igual que 0 y `cotaArea` es mayor que 0.

Se recuerda que:

- El área del triángulo es: base $\times$ altura / 2.
- El área del rectángulo es: largo $\times$ ancho.
- El área del círculo es: $\Pi \times \text{radio}^2$
- En Pascal se dispone de la constante `Pi` predefinida.

Solución

```
procedure AgregarAlPrincipio (var lst : ListaFiguras; fg : Figura);
var p : ListaFiguras;
begin
  new(p);
  p^.elemento  := fg;
  p^.siguiente := lst;
  lst := p
end;

function ObtenerFiguras (arrFiguras : ArregloFiguras;
                         n          : integer;
                         cotaArea   : integer       ) : ListaFiguras;
var i,k  : integer;
    area : real;
    lst  : ListaFiguras;
begin
  i := 1;
  k := 0;
  lst := NIL;
  while (i <= MAX) and (k < n) do
  begin
    with arrFiguras[i] do
      case forma of
        triangulo : area := base * altura / 2;
        rectangulo: area := largo * ancho;
        circulo   : area := Pi * sqr(radio);
      end;
    if area <= cotaArea then
    begin
      AgregarAlPrincipio(lst,arrFiguras[i]);
      k := k + 1
    end;
    i := i + 1
  end;
  ObtenerFiguras := lst
end;
```

Ejercicio 4 (6 puntos)

Indicar cuál es la salida que produce el siguiente programa cuando en la entrada se ingresa el último dígito de su cédula de identidad (nos referimos a la cédula del estudiante). Por ejemplo, si su cédula es 1234567-8 deberá ingresar el número 7.

```
program segundoParcial;
var w,z : integer;

procedure primera (var a:integer;var b:integer);
var z : integer;
begin
  z:= a;
  a:= z;
  b:= z * 2;
end; {primera}

procedure segunda (a: integer; b: integer);
begin
  a:= a + 1;
  b:= b * 2;
  writeln(a,' ',b,' ',z);
end; {segunda}

function tercera (a: integer) : integer;
var b : integer;
begin
  a:= a + 1;
  primera (a,b);
  b:= b * 2;
  writeln(a,' ',b,' ',z);
  tercera:= b;
end; {tercera}

begin
  readln(z);
  w:= tercera(z);
  segunda(w,z);
end.
```

Solución

0	1	2	3	4	5	6	7	8	9
1 4 0	2 8 1	3 12 2	4 16 3	5 20 4	6 24 5	7 28 6	8 32 7	9 36 8	10 40 9
5 0 0	9 2 1	13 4 2	17 6 3	21 8 4	25 10 5	29 12 6	33 14 7	37 16 8	41 18 9