Segundo Parcial. Programación 1

Instituto de Computación
Diciembre 2023

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 (25 puntos)

Se definen estructuras que representan datos relativos a una prueba de múltiple opción. La cantidad de preguntas está dada por el valor de una constante **MaxPregunta**.

```
const
   MAXPREGUNTA = ...; { valor mayor que 0 }
```

Cada pregunta tiene cinco opciones de las cuales sólo una es correcta. El estudiante puede elegir una de las 5 opciones o dejar sin responder la pregunta.

```
type
   TOpcion = 'A' .. 'E';

   TRespuesta = record case responde : boolean of
                   true : (opcion : TOpcion);
                   false: ()
                end
```

El tipo **TRespuestas** es un arreglo con todas las respuestas de un estudiante. El tipo **TCorrectas** es un arreglo que contiene las respuestas correctas de todas las preguntas.

```
   RangoPregunta =   1 .. MAXPREGUNTA;
   TRespuestas = array [RangoPregunta] of TRespuesta;
   TCorrectas = array [RangoPregunta] of TOpcion
```

Parte a)

Considerando que una respuesta correcta suma 2 puntos, una respuesta incorrecta resta 0.5 puntos y una pregunta sin responder implica 0 puntos,escribir un subprograma:

```
        function puntaje(respuestas : TRespuestas; correctas : TCorrectas) : real;
```

que recibe en **respuestas** las respuestas dadas por un estudiante, en **correctas** las respuestas correctas para cada pregunta y retorna el puntaje correspondiente.

Parte b)

Escribir un subprograma:

```
        function AlMenosN (n : Integer; opcion : TOpcion; respuestas : TRespuestas) : boolean;
```

que retorna **true** si hay **n** preguntas o más que fueron respondidas con la opción dada por el parámetro **opcion**.

Solución:

```
function puntaje(respuestas : TRespuestas; correctas : TCorrectas) : real;
var
   i : integer;
   puntos : real;
begin
   puntos:= 0;
   for i:= 1 to MAXPREGUNTA do
      if respuestas[i].responde then
         if respuestas[i].opcion = correctas[i] then
            puntos:= puntos + 2
         else
            puntos:= puntos - 0.5;
   puntaje:= puntos;
end;

{ determinar si hay  n preguntas o más que están respondidas con la opción dada }
function AlMenosN (n : Integer; opcion : TOpcion; respuestas : TRespuestas) : boolean;
```

```
var i, cant : integer;
begin
   cant:= 0;
   i:= 1;
   while (cant + MAXPREGUNTA - i >= n) and (cant < n) do
   begin
       if respuestas[i].responde and (respuestas[i].opcion = opcion) then
          cant:= cant + 1;
      i:= i + 1
   end;
   AlMenosN := cant = n
end;

{ Esta solución verifica que las respuestas restantes alcancen para cumplir la condición. Una solución un
    poco menos eficiente pero que cumple con la estructura de ser una búsqueda (y por lo tanto se considerara
     correcta) sería sustituir la condición (cant + MAXPREGUNTA - i >= n) por (i <= MAXPREGUNTA) }
```

Ejercicio 2 (10 puntos)

Se considera la siguiente definición de lista:

```
type
   Lista = ^TipoCelda;
   TipoCelda = record
                  dato: real;
                  sig: Lista
               end
```

Escribir el procedimiento:

```
        procedure NumeroAlCuadrado(var lis : Lista; r : real);
```

que recibe en **lis** una lista de números reales y remplaza la primera ocurrencia del número **r** por su cuadrado (es decir por r^2). Si **r** no pertenece a lis, agrega **r** al final de la lista. La solución debe realizar **una única recorrida** de la lista.

Solución:

```
procedure NumeroAlCuadrado (var lis : Lista; r : real);
var p, q : Lista;
begin
   p := lis;
   if p <> nil then
      while (p^.sig <> nil) and (p^.dato <> r) do
     p := p^.sig;
   if (p <> nil) and (p^.dato = r) then p^.dato := r*r
   else
   begin
      new (q);
      q^.dato := r;
      q^.sig := nil;
      if p = nil then lis := q
      else p^.sig := q
   end
end;
```

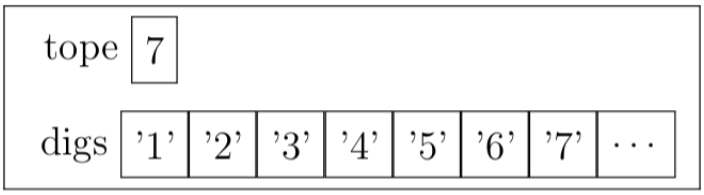
Ejercicio 3 (16 puntos)

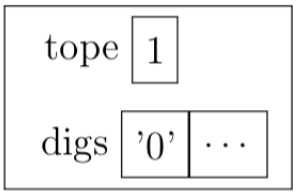
Una forma de representar números muy grandes consiste en abandonar el tipo **integer** y usar en su lugar un arreglo con tope de caracteres.

Consideremos la siguiente definición de tipos y constantes:

```
const MAX = ... ; { valor mayor que 1 }
type
    TEntero = record
        digs : array [1..MAX] of '0'..'9';
        tope : 0..MAX
    end
```

Las siguientes variables almacenan el valor 1234567 y 0.





Escriba la función que indica si dos **TEnteros** son iguales, usando el siguiente cabezal.

```
function iguales (n, m: TEntero): boolean;
```

Asuma que ninguno de estos **TEnteros** tiene ceros a la izquierda.

Solución:

```
function iguales (n, m : TEntero): boolean;
var k : integer;
begin
   if n.tope <> m.tope then iguales := false
   else
   begin
      k := 1;
      while (k <= n.tope) and (n.digs[k] = m.digs[k]) do k := 1 + k;
      iguales := k > n.tope
   end
end;
```

Ejercicio 4 (9 puntos)

Dado el siguiente programa, indicar qué despliegan las instrucciones *writeln* cuando se lee el dígito de su cédula de identidad ANTERIOR al guión. Por ejemplo si su cédula es 1234567-8, se ingresa 7.

```
program cultivos;
var a,b,c,u : integer;
function maiz(a: integer) : integer;
var b: integer;
   procedure trigo(a: integer; var b: integer);
   begin
      b := a + c
   end;
begin
   trigo(a,b);
   c := b;
   writeln(c);
   maiz := b - a
end;
function soja(var a: integer; b: integer) : integer;
   function cebada(c : char) : integer;
   begin
      cebada := 5
   end;
begin
   a := c + cebada('*');
   writeln(a);
   soja := b - c
end;
begin
   readln(a);
```

```
   b := 9 - a;
   c := b div 2;
   u := soja(a,b);
   writeln(maiz(u))
end.
```

Solución:

0	1	2	3	4	5	6	7	8	9
9	9	8	8	7	7	6	6	5	5
9	8	7	6	5	4	3	2	1	0
4	4	3	3	2	2	1	1	0	0