Examen - Programación 1

Instituto de Computación
Febrero 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.
- Escribir las respuestas de un solo lado de la hoja.
- Entregar solamente las hojas de solución escritas a lápiz.
- En cada hoja entregada se debe incluir nombre, cédula y qué número de hoja es del total entregadas.

Ejercicio 1 (40 puntos)

a) (20 puntos)

Una ficha de dominó es una pareja de naturales, y una cadena de fichas es un arreglo con tope de fichas de dominó. Lo implementamos como:

```
const
  MAX = ...; { MAX > 0 }

type
  TFicha  = record
              left, right: Integer
            end;
  TCadena = record
              tope: 0..MAX;
              arr : array [1..MAX] of TFicha
            end;
```

Una cadena de fichas de dominó es válida cuando cada componente derecho de una ficha coincide con el componente izquierdo de la ficha siguiente. Por ejemplo,

- `(1,3), (3,4), (4,1)` es una cadena válida
- `(1,3), (3,4), (1,4)` NO es una cadena válida
- `(1,3), (3,3), (4,1)` NO es una cadena válida

Escribir la función:

```
function esValida(a: TCadena): Boolean;
```

que indica si una cadena es válida o no. La cadena vacía y la que tiene una única ficha son válidas.

Solución:

```
function esValida(a: TCadena): Boolean;
var
  i : Integer;
begin
  i := 1;
  while (i < a.tope) and
        (a.arr[i].right = a.arr[i+1].left) do
    i := i + 1;
  esValida := i >= a.tope
end;
```

b) (20 puntos)

Otra forma de implementar una cadena de dominó es mediante una lista simple.

```
type
  TLista = ^TNodo;
  TNodo  = record
             info: TFicha;
             sig : TLista
           end;
```

Escribir el procedimiento:

```
procedure insertar(f: TFicha; var lis: TLista);
```

que agrega una ficha de dominó, `f`, al final de una *cadena válida*, `lis`, de forma que:

- si la cadena `lis` es vacía, la devuelve como la cadena cuyo único elemento es la ficha `f`;
- si la cadena `lis` es, por ejemplo, `(1,3), (3,4), (4,1)`, entonces

f	**lis a la salida de insertar**
`(1,5)`	`(1,3), (3,4), (4,1), (1,5)`
`(6,1)`	`(1,3), (3,4), (4,1), (1,6)`
`(1,1)`	`(1,3), (3,4), (4,1), (1,1)`
`(2,3)`	`(1,3), (3,4), (4,1)`

Debe considerar que la ficha a insertar se podrá "rotar", como se muestra en el segundo ejemplo, donde la ficha `(6,1)` debe convertirse en `(1,6)`.

Solución:

```
procedure insertar(f: TFicha; var lis: TLista);
var
  p  : TLista;
  aux: Integer;
begin
  if lis = nil then
    begin
      new(lis);
      lis^.info := f;
      lis^.sig  := nil;
    end
  else
    begin
      p := lis;
      while p^.sig <> nil do
        p := p^.sig;
      if p^.info.right = f.right then
      begin
        aux     := f.left;
        f.left  := f.right;
        f.right := aux
      end;
      if p^.info.right = f.left then
      begin
        new(p^.sig);
        p       := p^.sig;
        p^.info := f;
        p^.sig  := nil;
      end;
    end
end;
```

Ejercicio 2 (45 puntos)

Dadas las siguientes declaraciones:

```
const
  MAX = ...; { MAX > 0 }
type
  TCadena = array [1..MAX] of Char;
```

a) (25 puntos)

Escribir la función:

```
function estanCharIncluidos(
  cad1, cad2: TCadena
): Boolean;
```

que devuelva `true` si y solo si todos los caracteres de `cad1` están en `cad2`.

A modo de ejemplo:

cad1	**cad2**	**Resultado**
`[a,b,c,d,e]`	`[e,d,c,b,a]`	`true`
`[a,b,c,d,e]`	`[f,g,h,i,j]`	`false`
`[a,a,a,b,b]`	`[e,d,c,b,a]`	`true`

Solución:

```
{ función auxiliar }
function pertenece(c: Char; cad: TCadena): Boolean;
var i: Integer;
begin
  i := 1;
  while (i <= MAX) and (cad[i] <> c) do
    i := i + 1;
  pertenece := i <= MAX;
end;

function estanCharIncluidos(
  cad1, cad2: TCadena
): Boolean;
var i: Integer;
begin
  i := 1;
  while (i <= MAX) and (pertenece(cad1[i], cad2)) do
    i := i + 1;
  estanCharIncluidos := i > MAX;
end;
```

Solución alternativa más eficiente utilizando el tipo conjunto para evitar recorrer la segunda cadena más de una vez.

```
function estanCharIncluidos(
  cad1, cad2: TCadena
): Boolean;
var
  memo: set of char;
  i, j: integer;
begin
  memo := [];
  i := 1;
  j := 1;
  while (i <= MAX) and (j <= MAX) do
    if cad1[i] in memo then
      i := i + 1
    else
      begin
        memo := memo + [cad2[j]];
        j := j + 1
      end;
  while (i <= MAX) and (cad1[i] in memo) do
    i := i + 1;
  estanCharIncluidos := i > MAX
end;
```

b) (20 puntos)

Se agrega un nuevo tipo para guardar la cantidad de dígitos pares e impares contenidos en una cadena de caracteres:

```
TDigitos =
  record case hayDigitos: Boolean of
    true : (cantPares, cantImpares: Integer);
    false: ()
  end;
```

Escribir el procedimiento:

```
procedure cantDigitos(
  cad: TCadena;
  var cant: TDigitos
);
```

que dada una cadena de caracteres, `cad`, si esta contiene algún dígito, devuelve la cantidad de pares e impares.

Solución:

```
procedure cantDigitos(
  cad: TCadena;
  var cant: TDigitos
);
var
  i: Integer;
  cp, ci: Integer;
begin
  cp := 0;
  ci := 0;
  for i := 1 to MAX do
    case cad[i] of
      '0', '2', '4', '6', '8': cp := cp + 1;
      '1', '3', '5', '7', '9': ci := ci + 1
    end;
  cant.hayDigitos := cp + ci > 0;
  if cant.hayDigitos then
  begin
    cant.cantPares   := cp;
    cant.cantImpares := ci
  end
end;
```

Ejercicio 3 (15 puntos)

Determinar la salida del siguiente programa, cuando se le da como entrada el dígito de las unidades (el último dígito antes del guion) de su número de cédula. Por ejemplo, si su cédula es 1234567-8, la entrada será 7.

```
program alcance;

var
  a, b, c: Integer;

procedure proc1(var y: Integer; c: Integer);

  function f1(x, y: Integer): Boolean;
  begin
    a  := x + c;
    f1 := y mod b = 0;
  end;

begin
  c := c + 1;
  if f1(y, c) then
    begin
      y := a + 1;
      writeln(y, b);
    end
  else
    begin
      y := a - 1;
      writeln(y, c)
    end
end;

procedure proc2(var x, y: Integer);
var
  c: Integer;

begin
  c := y + 4;
  if c < b then
    x := x - 1
  else
    x := x + 1;
  writeln(c, x)
end;

begin
  read(a);
  b := a + 2;
  c := b * 3;
  proc1(a, b);
  proc2(b, c);
  writeln(a, b, c)
end.
```

Solución:

Entrada	Salida		
0	2	3	
	10	3	
	2	3	6
1	4	4	
	13	4	
	4	4	9
2	6	5	
	16	5	
	6	5	12
3	8	6	
	19	6	
	8	6	15
4	10	7	
	22	7	
	10	7	18

Entrada	Salida		
5	12	8	
	25	8	
	12	8	21
6	14	9	
	28	9	
	14	9	24
7	16	10	
	31	10	
	16	10	27
8	18	11	
	34	11	
	18	11	30
9	20	12	
	37	12	
	20	12	33