Segundo Parcial. Programación 1

Instituto de Computación
Noviembre 2022

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 (12 puntos)

En el método de cifrado llamado **César** (por el emperador romano), cada letra del alfabeto se desplaza, cíclicamente, a lo largo de cierto número de lugares. Por ejemplo, en un cifrado César de **desplazamiento 3 a la derecha**, la letra A se convertirá en D, la B se convertirá en E, y así sucesivamente, finalizando con la conversión de la letra Y en B y la letra Z en C.

Dada la siguiente definición en Pascal del tipo `TAlfabeto`:

```
type
  TAlfabeto = array [1..26] of Char;
```

Defina la función:

```
function desplazamiento(alf: TAlfabeto): Integer;
```

que, dado el alfabeto `alf`, retorna el desplazamiento del alfabeto. En caso de no cumplir con la condición de alfabeto con desplazamiento, es decir, en caso de no mantener siempre el mismo desplazamiento entre los caracteres, retorna -1.

Ejemplos:

Alfabeto	Salida
A B C ··· W X Y Z	0
D E F ··· Z A B C	3
B C F ··· X Y Z A	−1

Solución:

```
function siguiente(c: Char): Char;
begin
  if c = 'Z' then
    siguiente := 'A'
  else
    siguiente := succ(c)
end;

function desplazamiento(alf: TAlfabeto): Integer;
var i,desp,res: Integer;
begin
  res := -1;
  desp := ord(alf[1]) - ord('A');
  if (0 <= desp) and (desp <= 25) then
  begin
    i := 2;
    while (i <= 26) and
          (alf[i] = siguiente(alf[i-1])) do
      i := i+1;
    if i > 26 then
      res := desp
  end;
  desplazamiento := res
end;
```

Solución alternativa con precondición: en el arreglo solamente ocurren letras en el rango `'A'..'Z'`.

```
function desplazamiento(alf: TAlfabeto): Integer;
var i: Integer;
begin
  i := 2;
  while (i <= 26) and
        (alf[i] = siguiente(alf[i-1])) do
    i := i+1;
  if i <= 26 then
    desplazamiento := -1
  else
    desplazamiento := ord(alf[1]) - ord('A')
end;
```

Ejercicio 2 (28 puntos)

Se tiene un robot encargado de fumigar y detectar plagas en un campo de frutales.

Dada la siguiente definición:

```
const
  MAX = ... {valor entero mayor estricto que 0};

type
  TEspecie = (manzano, peral, naranjo);

  TArbol =  record
              identificador: Integer;
              especie: TEspecie;
              case plaga: Boolean of
                true : (fumigado: Boolean);
                false: ()
            end;

  TCultivo =  record
                arboles: array [1..MAX] of TArbol;
                tope: 0..MAX;
              end;

  TIDArbol =  record
                ids: array [1..MAX] of Integer;
                tope: 0..MAX;
              end;
```

a) (10 puntos)

Defina el procedimiento `arbolesAFumigar` que retorna en `af` los identificadores de los árboles de la especie `esp` que tienen una plaga y no fueron fumigados.

```
procedure arbolesAFumigar(cultivo: TCultivo;
                          esp: TEspecie;
                          var af: TIDArbol);
```

Solución:

```
procedure arbolesAFumigar(cultivo: TCultivo;
                          esp: TEspecie;
                          var af: TIDArbol);
var i: Integer;
begin
  af.tope := 0;
  for i := 1 to cultivo.tope do
    if (cultivo.arboles[i].especie = esp) and
       cultivo.arboles[i].plaga and
       not cultivo.arboles[i].fumigado then
    begin
      af.tope := af.tope + 1;
      af.ids[af.tope] :=
        cultivo.arboles[i].identificador
    end
end;
```

b) (18 puntos)

El robot realiza recorridos periódicos por el cultivo para monitorear el estado de los frutales, guardando los identificadores de los árboles que no tienen más plagas.

Implemente el procedimiento `actualizarCultivo`, que recibe el arreglo `curados` con los identificadores de los árboles que ya se curaron, y actualiza en `cultivo` el estado de cada árbol curado indicando que ya no tiene plaga.

```
procedure actualizarCultivo(curados: TIDArbol;
                            var cultivo: TCultivo);
```

Obs: asuma que los identificadores de los árboles a actualizar existen en el cultivo.

Solución:

```
{ precondición: los identificadores de los árboles a
  actualizar existen en el cultivo }
procedure actualizarCultivo(curados: TIDArbol;
                            var cultivo: TCultivo);
var i, j: Integer;
begin
  for i := 1 to curados.tope do
  begin
    j := 1;
    while curados.ids[i] <>
          cultivo.arboles[j].identificador do
      j := j + 1;
    cultivo.arboles[j].plaga := false
  end
end;
```

Ejercicio 3 (10 puntos)

Se definen los siguientes tipos:

```
type
  ListaBin = ^NodoBin;
  NodoBin  =  record
                bin: 0..1;
                sig: ListaBin;
              end;
```

donde los valores del tipo `ListaBin` son listas que permiten representar un número expresado en base 2. El dígito binario menos significativo está almacenado en la primera celda y así sucesivamente hasta el dígito más significativo que está almacenado en la última celda de la lista. Por ejemplo, el número binario 11010 se representa como la lista $[0, 1, 0, 1, 1]$.

Se pide implementar una función `calcularNum` que, dado un número expresado en base 2 pasado en el parámetro `lista`, devuelve su representación en base decimal (base 10). Si la lista es vacía la función retorna el valor `0`.

```
function calcularNum(lista: ListaBin): Integer;
```

La representación decimal de un número binario de n dígitos

$$d_{n-1} \dots d_3 d_2 d_1 d_0$$

se calcula como:

$$d_0 * 2^0 + d_1 * 2^1 + \cdots + d_{n-1} * 2^{n-1}$$

Ejemplos:

- El valor resultante de invocar `calcularNum` sobre el número binario 110, representado por la lista $[0, 1, 1]$, es el decimal 6.
- El valor resultante de invocar `calcularNum` sobre el número binario 11010, representado por la lista $[0, 1, 0, 1, 1]$, es el decimal 26.

Solución:

```
function calcularNum(lista: ListaBin): Integer;
var
  acum, pot: Integer;
begin
  acum := 0;
  pot  := 1;
  while lista <> nil do
  begin
    acum  := acum + lista^.bin * pot;
    pot   := pot * 2;
    lista := lista^.sig
  end;
  calcularNum := acum
end;
```

Ejercicio 4 (10 puntos)

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

```
program segundoParcial;
var z: Integer;

procedure procUno(var a, b: Integer);
var z: Integer;
begin
  z := a + 1;
  a := z;
  b := b * 2;
end;

function funcUno(a: Integer; b: Integer): Integer;
begin
  z := a + 1;
  b := b * 2;
  funcUno := a
end;

procedure procDos(a: Integer; var b: Integer);
begin
  a := a + 1;
  b := b * 2;
  writeln(a, ' ', b, ' ', z);
  z := funcUno(a, b)
end;

begin
  readln(z);
  procDos(z, z);
  writeln(z);
  procUno(z, z);
  writeln(z);
end.
```

Solución:

Entrada	Salida		
0	1	0	0
	1		
	4		
1	2	2	2
	2		
	6		
2	3	4	4
	3		
	8		
3	4	6	6
	4		
	10		
4	5	8	8
	5		
	12		

Entrada	Salida		
5	6	10	10
	6		
	14		
6	7	12	12
	7		
	16		
7	8	14	14
	8		
	18		
8	9	16	16
	9		
	20		
9	10	18	18
	10		
	22		