UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA
Departamento de Computación
Estructura de Datos e da Información

Ejercicios de recursividad

1. Determinar qué operación realiza:

```
//x>=0, y>0
function funcion_x (x,y: integer):integer;
begin
    if (x =0) or (x<y) then funcion_x:= 0
    else   funcion_x := 1+ funcion_x (x-y, y);
end;
```

Solución: La división entera

2. Considere el siguiente procedimiento P:

```
procedure P(num:integer);
begin
    if (num>=1) and (num <=8)
    then begin
        P(num-1);
        write (num:1);
    end else
        writeln
end {P};
```

¿Qué salida producen las llamadas P(3), P(7), P(10):

a) en el procedimiento original,

Solución:
`P(3)`=< *salto de linea* > 123
`P(7)`=< *salto de linea* > 1234567
`P(10)`=< *salto de linea* >

b) si reemplazamos `Num-1` por `Num+1`,

Solución:
`P(3)`=< *salto de linea* > 876543
`P(7)`=< *salto de linea* > 7654321
`P(10)`=< *salto de linea* >

c) si intercambiamos el `write` y la llamada recursiva, y

Solución:
`P(3)`=321 < *salto de linea* >
`P(7)`=7654321 < *salto de linea* >
`P(10)`=< *salto de linea* >

d) si insertamos una copia de la instrucción `write` antes de la llamada recursiva.

Solución:
`P(3)`=321 < *salto de linea* > 123
`P(7)`=7654321 < *salto de linea* > 1234567
`P(10)`=< *salto de linea* >

3. Dada la siguiente función trace la secuencia de llamadas y retornos de las funciones para `F(1,5)` y `F(8,3)`.

```
Function F (Num1, Num2: integer): integer;
begin
    if num1 > num2 then
        f:= 0
    else if num2 = num1 + 1 then
        f:= 1
    else
        f:= f (num1 + 1, num2 - 1) + 2
end;
```

Solución:
`F(1,5)`→ Llamadas `F(2,4)`, `F(3,3)`, `F(4,2)`
Retornos `F(4,2)=0`,`F(3,3)=F(4,2)+2=0+2=2`,
`F(2,4)=F(3,3)+2=2+2=4`,`F(1,5)=F(2,4)+2=4+2=6`
`F(8,3)`→ Retorna directamente 0

4. ¿Qué calculan las siguientes funciones recursivas (suponer `n>=0` para los apartados a-d)?

a)

```
Function F (n: integer): integer;
begin
    if n = 0 then
        F:= 3
    else
        F:= n * F (n-1)
end;
```

Solución: $3 * n!$

b)

```
Function F (x: real; n: integer): real;
begin
    if n = 0 then
        F:= 0
    else
        F:= x + F (x, n-1)
end;
```

Solución: $n * x$

c)

```
Function F (n: integer): integer;
begin
    if n < 2 then
```

```
        F:= 0
    else
        F:= 1 + F (n div 2)
end;
```

Solución: La potencia de 2 más cercana a n

d)

```
Function F (n: integer): integer;
begin
    if n = 0 then
        F:= 0
    else
        F:= F (n div 10) + (n mod 10)
end;
```

Solución: Suma de las cifras que componen n

e)

```
Function F (n: integer): integer;
begin
    if n < 0 then
        F:= F(-n)
    else if n < 10 then
        F:= n
    else
        F:= F (n div 10)
end;
```

Solución: La primera cifra de n

5. Escribe versiones no recursivas (iterativas) de las funciones del ejercicio anterior.

6. Considera el siguiente procedimiento:

```
procedure Q (num1, num2: integer);
begin
    if num2 <= 0 then
        writeln
    else
        begin
            Q (num1-1, num2-1);
            write (num1:1);
            Q (num1+1, num2-1);
        end
end;
```

a) ¿Qué salida produce `Q(14,3)`

Solución:

```
<retorno de carro>
12<retorno de carro>
13<retorno de carro>
14<retorno de carro>
14<retorno de carro>
14<retorno de carro>
15<retorno de carro>
16<retorno de carro>
```

b) Si movemos la instrucción `write` antes de la llamada recursiva, ¿qué salida produce `Q(14,4)`?

Solución:

```
141312<retorno de carro>
<retorno de carro>
14<retorno de carro>
<retorno de carro>
1514<retorno de carro>
<retorno de carro>
16<retorno de carro>
<retorno de carro>
```

7. Realiza una función recursiva en Pascal que realice la operación potencia(x,n)= x^n, siendo $x, n \geq 0$

Solución:

```
function potencia (x,n: integer): integer;
begin
      if n=0 then potencia:=0
      else potencia:=x*potencia(x,n-1);
end;
```

8. Desarrolla la función recursiva `Par (n: integer): boolean` que devuelve true si n es par y false en caso contrario, siendo $x \geq 0$. Puedes basarte en la idea de que 0 es par, y que la paridad de cualquier otro entero positivo es la opuesta que la del entero anterior.

Solución:

```
function par (n: integer): boolean;
begin
      if n=0 then par:=true
      else par:=not(par(n-1));
end;
```

9. ¿Es correcto el siguiente código desde el punto de vista de la *verificación*? (suponemos n ≥ 0).

```
procedure recursivo(n: integer);
begin
    if n=0
    then writeln('Llegamos al final')
    else recursivo(n div 2+1);
end;
```

Solución:
Es incorrecto: si $n > 0$ se generan llamadas recursivas infinitas del tipo recursivo(1)

10. Implementar una función recursiva que devuelva la suma de los elementos de un vector v comprendidos entre la posición ini y la posición fin.

Solución:
```
function suma (v:vector ; ini ,fin : integer): integer;
Precond: ini <= fin son posiciones válidas en el array
begin
      if ini=fin then suma := v[ini]
      else suma := v[ini] + suma (v,ini+1,fin);
end;
```

11. Implementar un procedimiento recursivo que imprima los dígitos de un número entero positivo en orden inverso.

Solución:
```
procedure imprimirInverso (n: integer);
var resto:integer;
begin
     resto:=n div 10;
     if resto=0 (*solo un dígito*)
     then write(resto:1)
     else begin
          imprimirInverso(resto);
          write(n mod 10);
     end;
end;
```

12. Implementar una función recursiva a la cual se le pasa un string por parámetro y devuelve dicho string invertido (ejemplo: `Invertir('hola')` = 'aloh').

    Puedes usar las siguientes funciones:

    - `Copy(S:string;Index,Count:Integer):string` que permite extraer parte de un string, siendo `S` el string original, `Index` el índice en donde empieza el substring a extraer y `Count` el número de caracteres de dicho substring.
    - `Length(S:string):integer`, que devuelve el tamaño de un string
    - `Concat(S1,S2[,S3,...,Sn]:string):string` que devuelve un único string resultado de concatenar S1,S2, ... Sn.

Solución:

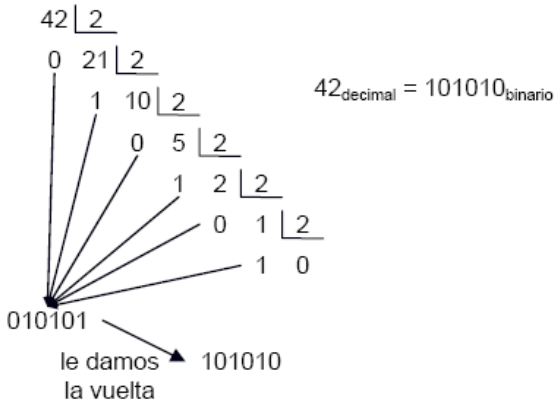
```
procedure InvertirCadena (input: string; var output:string);
var tamano:integer;
begin
     tamano:=length(input);
     if (tamano=0) or (tamano=1) (*solo un caracter*)
     then output:=input
     else begin
          InvertirCadena(copy(input,2,tamano), output);
          output:=concat(output,input[1]);
     end;
end;
```

13. Realiza un procedimiento recursivo `procedure ConvertirBinario(n:integer)` que, siendo $n \geq 0$ imprima por pantalla su representación en binario.

    Para convertir un número entero decimal (en base 10) a un número entero binario (en base 2) hay que seguir los siguientes pasos:

    a. Dividir el número entre dos (usando la división entera, operador `div`) y repetir el proceso hasta obtener un cociente cero

    b. El número binario resultante se obtiene de los restos (operador `mod`) de las divisiones pero en sentido inverso



Solución:

```
procedure ConvertirBinario(n:integer);
begin
     if (n div 2 = 0)
     then write(n mod 2)
     else begin
          ConvertirBinario(n div 2);
          write(n mod 2);
     end;
end;
```