UNIVERSIDADE DA CORUÑA

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

Práctica 0b: Recursividad

Ejercicios

1. Comparar el rendimiento entre las implementaciones recursivas e iterativas del problema de la serie de Fibonacci para valores de $5, 10, 20, 30, \ldots$

```
program fibonacci;
{Solucion recursiva}
uses sysutils;
var
    pos: longint;

function Fibo (pos:integer): longint;
begin
    if pos = 1
    then Fibo := 1
    else if pos = 2
    then Fibo := 1
    else Fibo := Fibo(pos -1) + Fibo (pos -2);
end;

begin
    writeln ('Serie de Fibonacci: que posicion?');
    readln (pos);
    writeln('El tiempo actual es ', FormatDateTime('hh:nn:ss',Time));
    writeln ('El resultado es ', Fibo(pos));
    Writeln ('The current time is : ', FormatDateTime('hh:nn:ss',Time));
end.
```

```
program fibonacci;
{Solucion iterativa}
uses sysutils;
var
    pos: integer;

function Fibo (pos:integer): longint;
var
    sig, previo, actual: longint;
    i: integer;
begin
    if pos = 1
    then Fibo := 1
    else if pos = 2
    then Fibo := 1
    else begin
```

```
            previo:= 1;
            actual:= 1;
            for i:=3 to pos do begin
                    sig:= previo+actual;
                    previo:= actual;
                    actual:= sig;
            end;
    Fibo := actual;
    end;
end;

begin
    writeln ('Serie de Fibonacci: que posicion?');
    readln (pos);
    writeln('El tiempo actual es ', FormatDateTime('hh:nn:ss',Time));
    writeln ('El resultado es ', Fibo(pos));
    writeln('El tiempo actual es ', FormatDateTime('hh:nn:ss',Time));
end.
```

2. Fallos por agotamiento de pila o montículo. Ejecuta el siguiente código, y obtendrás un error de ejecución con código 203 que significa *Agotamiento de Montículo*. ¿Por qué?

```
program overflow;
type
    tArray = array[1..2550] of string;
    PtArray = ^tArray;
var
    i: longint;

{Version recursiva}
procedure reserva (i:longint);
var
    p: PtArray;
begin
    i:= i + 1;
    writeln (i);
    new(p);
    {dispose(p);}
    reserva(i);
end;

begin
    i:= 1;
    reserva(i);
end.
```

   Descomenta la sentencia `dispose(p)` y ejecútalo de nuevo. Ahora se obtiene un error 202 que indica *Agotamiento de la Pila*. ¿Por qué?

3. Prueba ahora la versión iterativa, comentando y descomentando la sentencia `dispose(p)`. ¿Qué errores se obtienen y por qué?

```
program overflow;
type
    tArray = array[1..2550] of string;
    PtArray = ^tArray;

{Version iterativa}
procedure reserva;
```

```
var
    p: PtArray;
    i: longint;
begin
    for i:= 1 to maxlongint do
    begin
        writeln (i);
        new(p);
        dispose(p);
    end;
end;

begin
    reserva;
end.
```

4. La siguiente función implementa la suma de los dígitos de un número entero. Desarrolla una versión recursiva de la misma y pruébala.

```
function sumaiterativa (n: integer): integer;
var
    suma: integer;
begin
    suma := 0;
    while n>9 do begin
        suma := suma + (n mod 10);
        n := n div 10;
    end;
    sumaiterativa := suma + n;
end;
```

Soluciones

1. Como has podido ver las versiones recursivas son más lentas que las versiones iterativas. Además, en este caso, se realizan cálculos repetidos.

2. Las variables dinámicas consumen memoria del *Montículo*. Esta memoria se agota al hacer continuas reservas de memoria con `New` sin hacer ningún `Dispose`. En el segundo caso el *Montículo* se usa correctamente pero no así la *Pila*. Con cada llamada a una función o procedimiento se reserva una parte de la *Pila*, llamada *registro de activación* para guardar datos de esa función. Al generar de forma recursiva múltiples llamadas a la función `Reserva` se acaba agotando la *Pila*.

3. En este caso sólo se necesita un *registro de activación* (sólo hay una llamada) y por lo tanto sólo existe el problema del *Montículo* cuando no se hace el `Dispose`.

4. La solución podría ser la siguiente:

```
function sumarecursiva (n: integer): integer;
begin
    if n<=9
    then sumarecursiva := n
    else sumarecursiva := sumarecursiva (n div 10) + (n mod 10);
end;
```