Diseño

```
Función validaNIF (    p.d. DNI ES ENTERO,
                       p.d.r.. letra ES CARÁCTER,
                       p.r. NIF ES CADENA DE CARACTERES)
DEVUELVE booleano
```

La función lee un DNI y su letra; si la letra es correcta, construye el NIF y devuelve ‘cierto’. Si la letra es incorrecta, devuelve ‘falso’, asigna a ‘letra’ el valor correcto y no construye el NIF.

Implementación en PL/SQL

PL/SQL permite implementar los tres tipos de parámetro: dato, resultado y dato-resultado:.

```
CREATE OR REPLACE

FUNCTION VALIDANIF  ( DNI    IN          INTEGER,
                      letra   IN OUT CHAR,
                      NIF    OUT      VARCHAR2)
RETURN BOOLEAN AS
  letrasValidas CHAR(23) := 'TRWAGMYFPDXBNJZSQVHLCKE';
  letraCorrecta CHAR;
  letraLeida CHAR := SUBSTR (letra, 1);
  resto INTEGER;
BEGIN

 resto := DNI MOD 23;
 letraCorrecta := SUBSTR(letrasValidas, resto+1, 1);
 NIF := letraCorrecta;

 IF   (letraCorrecta = letraLeida) THEN
       NIF := TO_CHAR(DNI) || letraLeida ;
       RETURN true;
 ELSE
       letra := letraCorrecta;
       RETURN false;
 END IF;

END VALIDANIF;
```

Implementación en C

En C no existe el paso por referencia: todos los parámetros se pasan por valor. El paso por referencia puede simularse utilizando punteros.

```
typedef unsigned int boolean;

#define false 0
#define true (!false)

boolean validaNIF (int DNI, char*  letra, char NIF[])
{
   char letraCorrecta, letrasValidas[23] =
         {'T','R','W','A','G','M','Y','F','P','D','X','B','N','J','Z','S','Q','V','H','L','C','K','E'};
   int resto;

   resto = DNI % 23;
   letraCorrecta = letrasValidas[resto];

   if ( letraCorrecta == letra[0])
   {
     sprintf (NIF, "%d%c", DNI,letra[0]);
     return true;
   }
   else
   {
      *letra = letraCorrecta;
      return false;
   }

}
```

Implementación en C++

En C ++ sí existe el paso por referencia, utilizando el operador &. No existe una diferencia específica entre parámetros resultado y dato-resultado.

```
bool validaNIF (int DNI, char  &letra, char NIF[])
{
   char letraCorrecta, letrasValidas[23] =
                  {'T','R','W','A','G','M','Y','F','P','D','X','B','N','J','Z','S','Q','V','H','L','C','K','E'};
   int resto;

   resto = DNI % 23;
   letraCorrecta = letrasValidas[resto];

   if ( letraCorrecta == letra)
   {
     sprintf (NIF, "%d%c", DNI,letra);
     return true;
   }
   else
   {
      letra = letraCorrecta;
      return false;
   }

}
```

Implementación en Java:

En Java los parámetros se pasan siempre por valor. El único modo de tener acceso a un objeto desde dentro de un método es pasar como parámetro una referencia al objeto.
En el ejemplo, sería necesario utilizar objetos de la clase Integer, envolvente del tipo de datos int.

```
public class Letra
{
        private char valor;

        public Letra(char valor) { this.valor = valor;}

        public char getValor() { return valor; }

        public void setValor(char valor) { this.valor = valor;}

}

static boolean validaNIF (int DNI, Letra letra, StringBuffer NIF)
{
   String letrasValidas = "TRWAGMYFPDXBNJZSQVHLCKE";
   int resto = DNI % 23;

   char letraBuena = letrasValidas.charAt(resto);

   if ( letra.getValor() == letraBuena )
   {
      NIF.delete(0, NIF.length());
      NIF.append(DNI + String.valueOf(letraBuena) );
      return true;
   }
   else
   {
      letra.setValor(letraBuena);
      return false;
   }

}
```