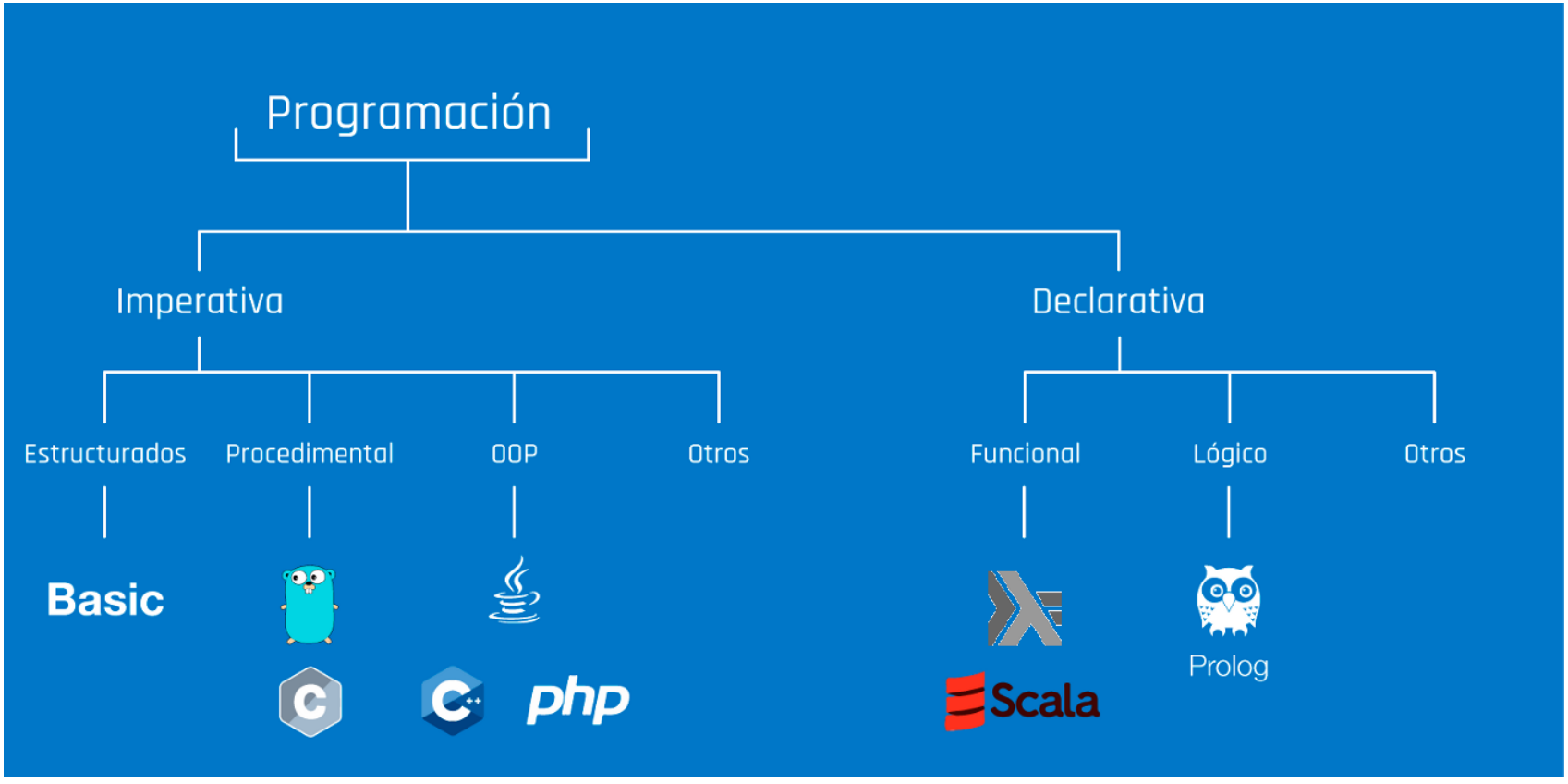
Introducción a la programación declarativa y a la programación funcional

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Estilos de programación

Programación Funcional: Una mirada diferente

¿Se puede liberar la programación del estilo de von Neumann?



Fuente

Imperativa vs Declarativa

La programación imperativa describe **qué pasos hay que dar (How?)** para obtener la solución de un problema mientras que la programación declarativa describe **qué/cuál es (What?)** la solución del problema.

Ejemplo

Problema: Llegar a casa desde la Escuela

- Imperativo

  Sal por la puerta, cruza a la acera de enfrente, gira a la izquierda, camina hasta el final de la calle, ...

- Declarativo

  La dirección es: calle ..., número ....

Programación funcional

- La **programación funcional** es un estilo de programación cuyo método básico de computación es la *aplicación de funciones a sus argumentos*.
- Un **lenguaje de programación funcional** es uno que soporta y potencia el estilo funcional.
- La **programación imperativa** es un estilo de programación en el que los programas están formados por instrucciones que especifican cómo se ha de calcular el resultado.

Ejemplo

Problema: Calcula la suma de una lista de números

- Imperativo

```
sumaLista l:
  asigna 0 a la variable suma
  para cada elemento n de la lista l:
    asigna (suma + n) a la variable suma
  devolver suma
```

- Declarativo

```
sumaLista l:
  si l es una lista vacía, devolver 0
  en otro caso, devolver (primero de l) + (sumaLista resto de l)
```

Ejemplo

Problema: Calcula la suma de una lista de números

- Imperativo (Python)

```
def sumaLista (l):
  suma = 0
  for n in l:
    suma += n
  return suma
```

- Declarativo (Prolog)

```
sumaLista([], 0).
sumaLista([N|L], S) :- sumaLista(L, SL), S is N + SL.
```

Ejemplo

Problema: Calcula la suma de una lista de números

- Imperativo (Java)

```
public static int sumaLista(List<Integer> l) {
  int i;
  int suma = 0;
  for(i = 1; i < l.size(); i++)
    suma += l.get(i);
  return suma;
}
```

- Declarativo (Haskell)

```
sumaLista :: (Num a) => [a] -> a
sumaLista [] = 0
sumaLista (n:l) = n + (sumaLista l)
```

Ejemplo

Problema: Calcula la suma de los números pares de una lista

- Imperativo

```
def sumaPares (l):
  pares = []
  for n in l:
    if (n % 2 == 0):
      pares.append(n)
  return sumaLista(pares)
```

- Declarativo

```
def sumaPares (l):
  return sumaLista(n for n in l if (n % 2 == 0))
```

Ejemplo

Problema: Calcula la suma de los números pares de una lista

- Imperativo

```
def sumaPares (l):
  pares = []
  for n in l:
    if (n % 2 == 0):
      pares.append(n)
  return sumaLista(pares)
```

- Declarativo

```
def sumaPares (l):
  return sumaLista(n for n in l if (n % 2 == 0))
```

Los dos escritos en Python

Ejemplo

Problema: Calcula la suma de los números del 1 al n

- Solución imperativa:

```
contador := 0
total := 0
repetir
        contador := contador + 1
        total := total + contador
hasta que contador = n
```

- Evaluación de *suma*4

contador	total
0	0
1	1
2	3
3	6
4	10

Ejemplo

Problema: Calcula la suma de los números del 1 al *n*

- Solución declarativa funcional

```
suma n = sum [1..n]
```

- Evaluación de *suma*4

```
suma 4
= sum [1..4] [def. de suma]
= sum [1, 2, 3, 4] [def. de [..]]
= 1 + 2 + 3 + 4 [def. de sum]
= 10 [def. de +]
```

Programación funcional

Características

- Evaluación de expresiones frente a ejecución de instrucciones (recursión frente a iteración)
- Las expresiones son llamadas a funciones
- El valor de una función sólo depende de sus argumentos (siempre se obtiene el mismo valor para los mismos argumentos: transparencia referencial).
- Las funciones pueden usarse como argumentos y como valores (funciones de primer orden).

Programación funcional

Ventajas

- Código más limpio, conciso y expresivo
- Sin efectos secundarios, al ser el estado inmutable
- Adecuado para sistemas concurrentes/paralelos, al no tener efectos laterales por ser inmutables
- Permite verificación formal y demostración automática

Leer:

- El resurgir de la programación funcional
- Qué es la programación funcional y por qué deberías usarla
- Importancia de la programación funcional en un mundo paralelo

Refactorización funcional

Si no ha quedado clara la filosofía, repasemos este enlace:

- Refactorización funcional

Fundamentalmente, ilustra el paso progresivo de una solución clásica imperativa a una solución funcional pura.

- Antes:

```
function SumPar(lista) {
        var resultado = 0;
        for (var i=0; i< lista.length ; i++) {
                if (lista[i] % 2 ==0) {
                        resultado += lista[i];
                }
        }
        return resultado;
}
```

- Después:

```
sumPar = (foldl (+) 0) . (filter even)
```

Adopción de programación funcional en lenguajes imperativos modernos

- Funciones lambda (anónimas) en C++11

```
[](int x, double y) -> double
{
  if (x<y)
    return x;
  return y;
}
```

- Funciones parciales (currificación) en C++11

```
int add(int first, int second)
{
  return first + second;
}
auto add_func = std::bind(&add, _1, _2);
add_func(4,5); // igual que add(4,5)
auto add_12_func = std::bind(&add, 12, _1);
add_12_func(5); // igual que add(12,5)
```

Adopción de programación funcional en lenguajes imperativos modernos

- Orden superior y funciones lambda en Java8

```
List nombres = Arrays.asList("Luis", "Miguel", "Angel");
Stream lengths = nombres.stream().map(name -> name.length());
```

- Orden superior y concatenación de funciones en Java8

```
double resultado=lista.stream()
.mapToDouble(gasto->gasto.getImporte()*1.21)
.filter(gasto->gasto<100)
.sum();
```

- Operaciones paralelas en Java8

```
ConcurrentMap<Person.Sex, List<Person>> byGender =
  roster.parallelStream().collect(
    Collectors.groupingByConcurrent(Person::getGender));
```

Bibliografía I

S. Thompson. Haskell: The Craft of Functional Programming, Second Edition. Addison-Wesley, 1999.
Chapter 1: Introducing functional programming

S. Akhmechet (trad. L. Mendoza). *Programación Funcional: Una mirada diferente*.
http://www.cs.us.es/~fsancho/?e=105

F. Sancho-Caparrini. *¿Se puede liberar la programación del estilo de von Neumann?*.
http://www.cs.us.es/~fsancho/?e=146

R. Fernández. *El resurgir de la programación funcional*.
https://www.genbeta.com/desarrollo/el-resurgir-de-la-programacion-funcional

R. García-Becerro. *Qué es la programación funcional y por qué deberías usarla*.
https://www.paradigmadigital.com/dev/la-programacion-funcional-deberias-usarla

Bibliografía II

A. Holderness. *Functional Programming: What Language Should You Be Talking?*.
https://hackernoon.com/functional-programming-what-language-should-you-be-talking-313dd8bc379b

A. Marzal. Charla Mayo/15: *Por qué aprender Programación Funcional ya*.
https://youtu.be/YU2i3L-euB0