Paradigmas de Programación

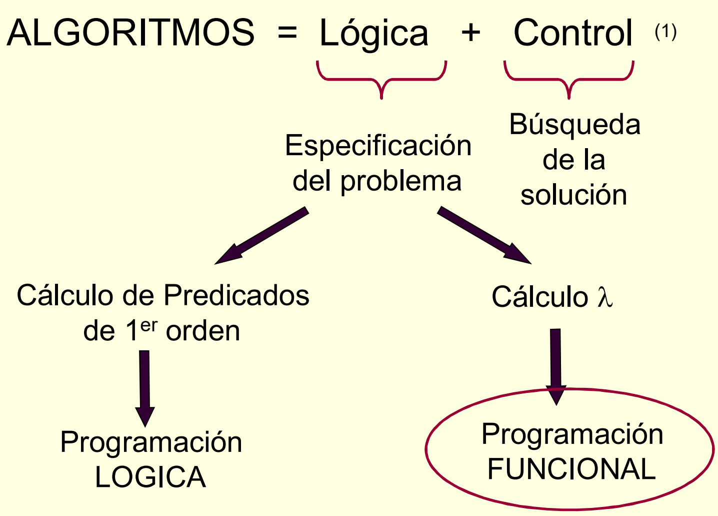
2do Cuatrimestre 2025

Programación Funcional

PARADIGMAS DE PROGRAMACIÓN

	Marco Teórico	**Programa**	**Ejecución de un prog.**	**Ejemplo Lenguaje**
Imperativo	Máquina de Von Neuman	Secuencia de sentencias que ejecutan acciones	Ejecución paso a paso de las sentencias	C, Pascal
Objetos	Modelo de Objetos	Secuencia de mensajes intercambiados entre objetos	Intercambio de mensajes entre objetos	Smalltalk
Lógico	Cálculo de predicados de primer orden	Conjunto de cláusulas	Demostración de una fórmula (teorema)	Prolog
Funcional	Cálculo Lambda	Expresión funcional que puede ser evaluada	Evaluar una función	Scheme

Programación Declarativa



(1) Kowalski, R. Algorithm = Logic + control. CACM, 22(7) 424-436

3

Programación Funcional

- Aparece como un paradigma independiente a principio de los sesenta para dar respuesta a problemas en el área de la inteligencia artificial.
- Toda computación es el resultado de evaluar una función matemática
- Enfatiza la evaluación de expresiones y no la ejecución de instrucciones

4

Programación Funcional

- Programa = Función matemática
- Ejecutar un programa = Evaluar una función
- No hay noción de variable como posición de memoria
  - No hay instrucción de asignación.
  - No existen efectos colaterales
  - Hay transparencia referencial (la salida depende solamente de las entradas)
- Permite definir funciones de orden superior

5

Lenguajes Funcionales

- El primero
  - LISP (*John McCarthy*)
    - **Scheme**
    - Common LISP
- En los '80
  - Miranda
  - Erlang
  - Alfil
  - …
- En los '90
  - Haskell
  - F# en Microsoft .NET
- En los '00
  - Lenguajes multiparadigma
    - **Python**
    - JavaScript
    - Ruby
    - Scala
  - Clojure
    - LISP concurrente sobre JVM
- Más recientes
  - Elixir, Rust, Kotlin, Swift…

6

Función Matemática

Una función **f** entre dos conjuntos **A** y **B** es una correspondencia que a cada elemento de un subconjunto de **A**, llamado "**Dominio de f**", le hace corresponder uno y sólo un elemento de un subconjunto **B** llamado "**Imagen de f**".

f: A → B
f(x) = <expresion>

f: Z → Z
f(x) = 2 * x + 1

x argumento nominal que representa a los elementos del Dominio
f nombre de la función
< expresion> representa el elemento del codominio o la forma de obtenerlo.

7

Función Matemática

g: Z → Z
g(x) = 2 + f(x)

para calcular la imagen de x a partir de g, deben usarse otras funciones, las cuales también deberán definirse.

- Definición de funciones:
  - Es una estructura jerárquica en la cual funciones más simples aparecen en la definición de funciones más complejas
  - Las funciones utilizadas a la derecha del signo de igualdad se referencian sólo por su nombre. El significado puede obtenerse de reemplazar el nombre por su definición.
  - Esta sucesión se clausura con funciones primitivas, cuyo significado se da por sobreentendido.

8

¿Cómo pensar un problema en este paradigma?

- Definir una función que determine si un número es perfecto. Un número es perfecto si la suma de sus divisores es igual al doble de si mismo. Es decir:

$$\text{perfecto(x)} = \begin{cases} \text{True} & \text{, si sumaDivisores(x) == doble(x)} \\ \text{False} & \text{, sino} \end{cases}$$

perfecto(x) = (sumaDivisores(x) == doble(x))

9

¿Cómo pensar un problema en este paradigma?

doble(x) = 2 * x

sumaDivisores(x) = buscar(x, 1)

buscar(x, y)

$$= \begin{cases} 0 & \text{, si } y > x \\ y + \text{buscar}(x, y+1) & \text{, si } y \leq x \text{ and esDivisor}(y, x) \\ \text{buscar}(x, y+1) & \text{, sino} \end{cases}$$

$$\text{esDivisor(a, b)} = \begin{cases} \text{True} & \text{, si resto(b, a) == 0} \\ \text{False} & \text{, sino} \end{cases}$$

10

Expresiones permitidas

- Expresiones que definen funciones

```
f(x) = 2 * x + g(x)
g(x) = 3 * x
```

```
perfecto(x) =
 (sumaDivisores(x) == 2 * x)
```

ABSTRACCIÓN FUNCIONAL

- Expresiones que aplican una función a un argumento

```
f(5) = 2 * 5 + g(5)
      = 10 + 15 = 25
```

```
perfecto(6)
= … = True
```

APLICACIÓN FUNCIONAL

- Expresiones atómicas

```
x, 2 ,5..
```

```
n,2 …
```

11