

# Lenguajes de programación y sus paradigmas

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

15 de agosto de 2024



# Paradigmas

- 1 Paradigmas
- 2 Paradigma Imperativo vs Declarativo
- 3 Paradigma orientado a objetos
- 4 Los 4 paradigmas
- 5 Bibliografía

— ¿Cuántos lenguajes de programación existen?

Para el 2017 la página  
<https://www.azulweb.net/estos-todos-los-lenguajes-programacion-existen-la-actualidad/>  
reportaba una lista de aproximadamente 650 lenguajes.

— ¿En qué lenguaje programa un computólogo?

— *¡En el que le pidan!*

— ¿Cómo le hace?

## — Estudiando paradigmas



# Paradigmas de programación

## Definición

Cada *paradigma de programación* describe:

- una filosofía y metodología para crear programas para la computadora.
- Esta filosofía define la forma de conceptualizar a la información que será procesada, así como
- el formalismo y reglas para operar con esa información, permitiendo calcular resultados que satisfagan las condiciones establecidas.

Recientemente se han propuesto varios paradigmas, pero mencionaremos aquí los referentes fundamentales.

# Paradigma Imperativo vs Declarativo

- 1 Paradigmas
- 2 Paradigma Imperativo vs Declarativo
- 3 Paradigma orientado a objetos
- 4 Los 4 paradigmas
- 5 Bibliografía

# Temas

## 2 Paradigma Imperativo vs Declarativo

- Imperativo
- Declarativo

# Lenguajes imperativos

- Especifican qué y cómo se debe hacer.
- Están inspirados en la estructura física de la computadora.
- La *memoria* o *estado* se visualiza como un conjunto de asociaciones entre *posiciones de memoria* y los *valores almacenados* en esas posiciones.
- Un programa consiste en una serie de *comandos* que indican cómo y cuándo almacenar y procesar valores en las posiciones de memoria.
- **Ejemplos:** Fortran, Pascal, C, Java, Python.

# Ejemplo

Código: datos.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int num;
6      printf("Dame un entero:\n");
7      scanf("%d", &num);
8      if (num < 0)
9      {
10         printf("El número fue negativo.\n");
11     }
12     else
13     {
14         int sum = num * (num + 1) / 2;
15         printf("La suma de 1 a %d es %d\n", num, sum);
16     }
17 }
```

```
$ gcc datos.c -o datos      # Compila
$ ./datos                  # Ejecuta
Dame un entero:
4
La suma de 1 a 4 es 10
```

# Temas

## 2 Paradigma Imperativo vs Declarativo

- Imperativo
- Declarativo

# Lenguajes declarativos

- Se basan en el principio de que *la programación debe enfocarse en indicar lo que se debe hacer, mientras que el intérprete del lenguaje se encarga de resolver el cómo llegar al resultado deseado.*
- Este ideal, en su forma pura, produce programas menos eficientes.



# Lenguajes declarativos

- Están inspirados en las notaciones matemática y lógica.
- En sus versiones *puras*, no hay variables modificables ni comandos para modificar su estado.
- Un programa consiste en un conjunto de *declaraciones de funciones* o *relaciones* que definen valores nuevos.
- Se dividen en dos clases:
  - *Funcionales*: consiste en la evaluación de funciones siguiendo reglas como la *composición* y *aplicación* (o *evaluación*) en forma semejante a las funciones de cálculo.  
**Ejemplos:** Scheme, ML, Haskell.
  - *Lógicos*: los cálculos están basados en *deducciones* según las reglas de la *lógica de primer orden*.  
**Ejemplo:** Prolog.

# Ejemplo en un lenguaje funcional

Código: hola.lisp

```
1 (write-line "Operemos: (+ (* (/ 9 5) 60) 32) ")
2 (write (+ (* (/ 9 5) 60) 32))
3 (write-line "")
4 (write-line "Fin")
5
6 (defun factorial (num)
7   (if (<= num 0)
8       (return-from factorial 1)
9       (* num (factorial (- num 1)))))
10 )
11 )
12 (terpri)
13 (princ "Dame un número natural pequeño: ")
14 (setq n (read))
15 (format t "El factorial de ~d es ~d" n)
16 (write (factorial 5))
17 (write-line "")
```

```
$ sudo apt install clisp          # Instala intérprete
$ clisp hola.lisp                 # Ejecuta
Operemos: (+ (* (/ 9 5) 60) 32)
140
Fin

Dame un número natural pequeño: 6
El factorial de 6 es 120
```

<https://tutoriales.edu.lat/pub/lisp?alias=tutorial-de-lisp>

# Ejemplo en un lenguaje lógico

Código: socrates.pl

```
1 human(socrates).  
2 mortal(X) :- human(X).
```

## Código: Intérprete: swipl

```
$ sudo apt install swi-prolog-core # Instalar
$ swipl socrates.pl                # Ejecutar intérprete
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)

1 ?- human(socrates).
true.

2 ?- mortal(socrates).
true.

3 ?- member(X, [socrates,platon,aristoteles]).
    X = socrates;
    X = platon;
    X = aristoteles
4 ?- member(platon, [socrates, platon, aristoteles]).
    true .
5 ?- halt.
```

<https://blog.adrianistan.eu/supertutorial-prolog/>

# Paradigma orientado a objetos

- 1 Paradigmas
- 2 Paradigma Imperativo vs Declarativo
- 3 Paradigma orientado a objetos**
- 4 Los 4 paradigmas
- 5 Bibliografía

# Lenguajes orientados a objetos

- Se trata de un paradigma orientado hacia lograr la correcta *organización* de sistemas vastos y complejos mediante el uso de *clases* y *objetos*.
- *Abstraen* el concepto de *tipo de dato* a manipular según:
  - ① El conjunto de datos admitibles dentro de cada tipo.
  - ② Las operaciones que se pueden realizar con ellos.
- *Encapsulamiento*, delimitando estrictamente las fronteras entre operaciones permitidas entre tipos de datos distintos.
- Reutilizamiento del código mediante el mecanismo de *herencia*.
- **Ejemplos:** Java, Python, Ruby.

# Los 4 paradigmas

- 1 Paradigmas
- 2 Paradigma Imperativo vs Declarativo
- 3 Paradigma orientado a objetos
- 4 Los 4 paradigmas**
- 5 Bibliografía



# Paradigmas

## Imperativo

Fortran

Estructurado

C

Orientado a objetos

C++

Python

Java

Ruby

PHP >5

Javascript

## Declarativo

Funcional

LISP

Scheme

Lógico

Prolog

# Bibliografía

- 1 Paradigmas
- 2 Paradigma Imperativo vs Declarativo
- 3 Paradigma orientado a objetos
- 4 Los 4 paradigmas
- 5 Bibliografía**

