Lenguajes de programación y sus paradigmas

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

21 de agosto de 2024

Paradigmas

— ¿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

Temas

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

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 cómputos 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 " 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

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

Paradigmas

Imperativo

Fortran

Estructurado

C

Orientado a objetos

C++ Python

Java Ruby

PHP >5

Javascript

Declarativo

Funcional

LISP

Scheme

Lógico

Prolog

Historia

Antecedentes

Tabla 1: Antecedentes

Fecha	Lenguaje	Contribución
1950s		El objetivo era obtener programas eficientes que explotaran al máximo el potencial del hardware.
~1960s		Uso de *mainframes*, primeras computadoras de uso general. Procesaban datos en lotes. Programadas con tarjetas perforables. Un programa se ejecutaba tras otro. Sus entradas y salidas estaban en archivos.
1957	Fortran	(*FORmula TRANslation*) Fue diseñado para aplicaciones de cálculo numérico científico. Sigue en uso hoy en día.

Antecedentes

Tabla 2: Antecedentes

Fecha	Lenguaje	Contribución
1956	ALGOL	(*ALGORithmic Languaje*) Diseñado como lenguaje universal. Primer lenguaje en usar la gramática BNF de Chomsky para expresar su sintaxis. Comandos estructurados para control de secuencia usados hoy en día: `if then else`, `while` y `case`.
1960	LISP	(*LISt Processor*) Lenguaje funcional diseñado por el grupo de John McCarthy para manipular expresiones simbólicas, para Inteligencia Artificial. Usa recolector de basura.

Antecedentes

Tabla 3: Antecedentes

Fecha	Lenguaje	Contribución
1960	COBOL	(*COmmon Busines Oriented Languaje*). Aún en uso hoy en día.
1962	Simula	Primer lenguaje orientado a objetos que introduce los conceptos de *clase*, *objeto*, *subtipo* y *despacho dinámico de métodos*.
1970s	Aparición del microchip y las microcomputadoras.	
1970	C	Permite acceder la funcionalidad de bajo nivel de la máquina y programar sistemas que interactúan más con el usuario.

Antecedentes

Tabla 4: Antecedentes

Fecha	Lenguaje	Contribución
1970	Pascal	Usa el P-code con la finalidad de ser portable, concepto que usa ahora el *bytecode* de Java. Permite anidar bloques y funciones con complejidad arbitraria.
1970	Smalltalk	Reglas de visibilidad: métodos públicos, variables de instancia privadas.
1970	Prolog	1er lenguaje de programación lógico. En uso hoy en día. Utiliza los algoritmos de unificación y resolución restringida para calcular los valores que satisfacen las restricciones lógicas, así la demostración provee el resultado.

Antecedentes

- En 1977, Ken Olson, presidente de Digital Equipment Corp (empresa productora de minicomputadoras) afirmó que "*no veía ninguna razón por la cual alguien pudiera querer una computadora en su casa*".
- En 1981 IBM saca `Lotus` el primer programa tipo hoja de cálculo y la opinión del público empieza a cambiar.
- 1984 Apple saca la Macintosh, con el primer sistema operativo con interfaz gráfica basada en ventanas, iconos y ratón.
- Aparecen los primeros sistemas *embebidos*. Son sistemas compuestos de computadoras conectadas a sistemas físicos que realizan tareas de control como motores, maquinaria industrial, aparatos electrodomésticos, aviones, etc. Por lo que aparecen los lenguajes para respuesta en *tiempo real*.

Antecedentes

Tabla 5: Antecedentes

Fecha	Lenguaje	Contribución
1980s	Dominados por el desarrollo de la computadora personal (PC).	
1986	C++	Stroustrup añade orientación a objetos a C, casi sin afectar la compatibilidad con C. Uso de plantillas y herencia múltiple.
80s	Ada	Concepto de tarea, temporizadores, ejecución concurrente de tareas.
80s	CLP	(*Constraint Logic Programming Languages*) Permiten la manipulación de relaciones sobre dominios apropiados: ecuaciones, desigualdades, restricciones sobre cadenas de caracteres, booleanos, reales o dominios finitos (lenguaje `CHIP`).

Antecedentes

Tabla 6: Antecedentes

Fecha	Lenguaje	Contribución
1990s	Aparición de Internet y la *World Wide Web* con sus lenguajes de marcado: HTML y XML.	
1990s	Java	Diseñado para distribuir mejor el trabajo entre servidores y clientes usando `Applets`. Requiere portabilidad para ejecutar código en varios tipos de cliente. Uso de hilos para ejecución concurrente.

Bibliografía

Bibliografía I

- Maurizio Gabbrielli, Simone Martini (2010). *Programming Languages: Principles and Paradigms*. Springer. 440 págs. ISBN: 978-607-707-211-9. DOI: 10.1007/978-1-84882-914-5.
- Viso, Elisa y Canek Peláez V. (2012). *Introducción a las ciencias de la computación con Java*. 2a. Temas de computación. Las prensas de ciencias. 571 págs. ISBN: 978-607-02-3345-6.