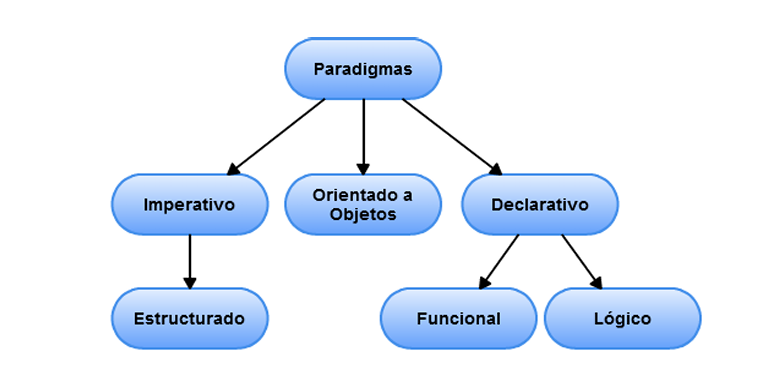
**Un paradigma de programación** es un modelo de diseño e implementación de software que sigue principios específicos aceptados.

**Paradigmas fundamentales:**

* Paradigma Lógico
* Paradigma Funcional
* Paradigma Imperativo o procedural
* Paradigma de Objetos

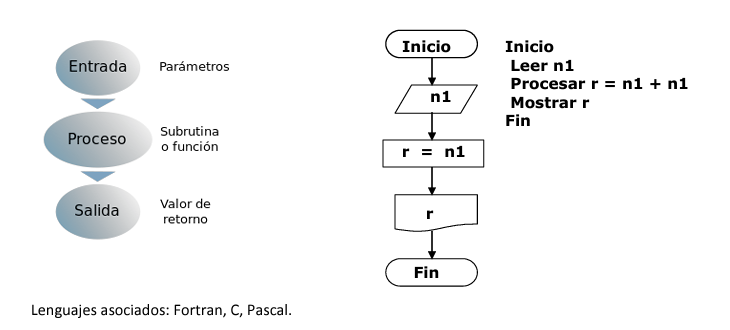


**Paradigma Imperativo:**

Describe paso a paso las instrucciones para resolver un problema. Dentro de este paradigma surgieron diferentes Tipos:

* **Lineal**: Programas como un solo bloque, uso de etiquetas y banderas.
* **Estructurado**: Combina algoritmos y estructuras de datos para modularidad y claridad.

**Algoritmo + Estructura de Datos = programa**



**Ventajas**:

* Cercanía al código máquina (ejecución rápida).
* Modularidad facilita mantenimiento y corrección.

**Desventajas**:

* Difícil modularizar sistemas complejos.
* No adecuado para todos los problemas.

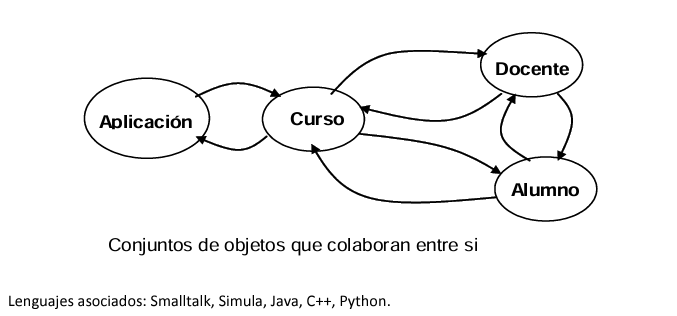
Aplicaciones:

* Sistemas de tiempo real
* Programación de microcontroladores
* Desarrollo de videojuegos de consola

**Orientado a Objetos:**

**Objetos + Mensajes = Programa**

Basado en encapsular datos y comportamientos en objetos que interactúan mediante mensajes. Sus pilares son la abstracción, herencia, polimorfismo y encapsulamiento.



Ventajas: 

* Reusable
* Mantenible
* Fiable

Desventajas:

* Programas mas extensos y lentos
* Curva de aprendizaje mas lenta para principiantes

Aplicaciones:

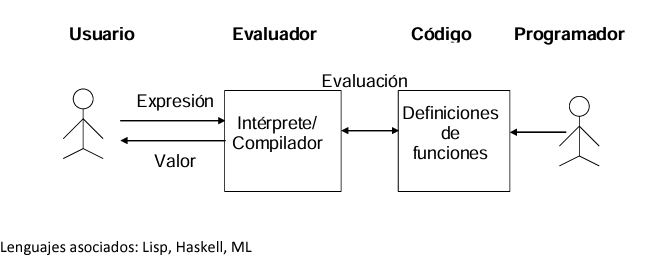
* aplicaciones de negocios
* Modelado y simulación
* Sistemas complejos

**Paradigma Declarativo:**

* Funcional:

**Funciones + Control = programa**

Computación mediante funciones matemáticas, evita cambios de estado



Ventajas:

* Altos niveles de abstracción
* Códigos cortos y expresivos
* Memoria administrada automáticamente.

Desventajas:

* Limitida portabilidad
* Baja eficiencia

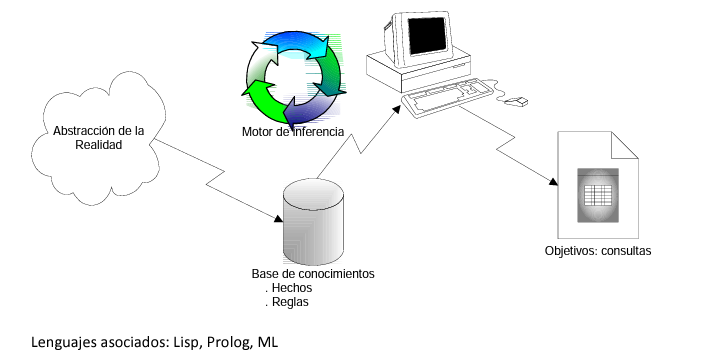
Aplicaciones:

* Programas de Inteligencia Artificial
* Sistemas distribuidos tolerante a fallos de software
* Telecomunicaciones

**Paradigma Lógico:**

**Lógica + Control = programa**

Define reglas lógicas y usa inferencias para resolver problemas.



Ventajas:

* Simplicidad.
* Potencia en la búsqueda de soluciones.
* Sencillez en la implementación de estructuras complejas.

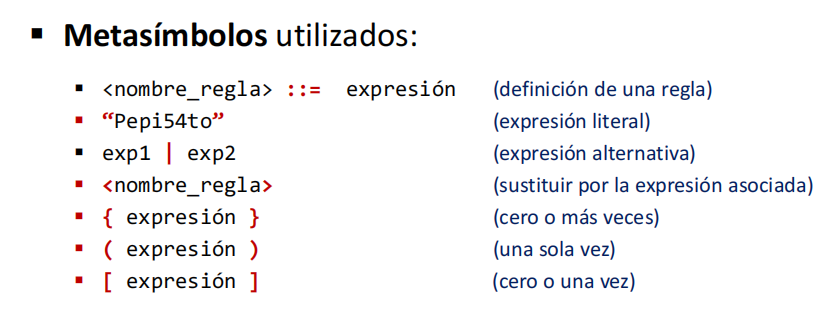
Desventajas:

* Poco utilizado en aplicaciones de gestión.
* Los programas son pocos eficientes por la lentitud de ejecución de los intérpretes

Aplicaciones:

* Sistemas expertos.
* Sistemas de soporte a decisiones
* Procesamiento de lenguaje natural.
* Robótica.

**NOTACION BNF (Backus‐Naur)**



es una manera de describir las reglas (sintaxis) de cómo se deben organizar los símbolos y las secuencias de símbolos (frases) en un lenguaje de programación. Básicamente, nos dice cómo construir las frases que forman un programa.

**Algunos tipos de símbolos:**

* Identificadores
* Literales constantes (enteros, reales, caracteres, secuencias de caracteres, booleanos)
* Operadores
* Separadores

Características Lenguajes Funcionales:

* + - * + Uso de calculo lambda
        + Se los considera de tipo declarativos
        + Las funciones son elementos de primer orden

Características Lenguajes Imperativos Estructurados:

* Dividido en bloques
* Secuencia, selección e iteración

Gramática para describir los lenguajes de programación:

* Gramática tipo 2; son gramáticas libres de contexto

S (símbolo inicial), P (producciones), VT(símbolo terminales), VN (símbolo no terminales)