Unidad 1: Conceptos Fundamentales

Elaboró:

María Elena Cárdenas Mosqueda

Introducción

Lenguaje de Programación?

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

Programa?

Es un conjunto de instrucciones que una vez ejecutadas realizarán una o varias tareas en una computadora..

paradigma.

(Del lat. *paradigma,* y este del gr. παράδειγμα).

1. m. Ejemplo o ejemplar.

2. m. Teoría cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento; p. ej., en la ciencia, las leyes del movimiento y la gravitación de Newton y la teoría de la evolución de Darwin.

- "*Un **paradigma de programación** indica un método de realizar cómputos y la manera en que se deben estructurar y organizar las tareas que debe llevar a cabo un programa*"
- Los paradigmas fundamentales están asociados a determinados **modelos de cómputo.**
- Tambien se asocian a un determinado **estilo de programación**
- Los lenguajes de programación suelen implementar, a menudo de forma parcial, **varios** paradigmas.

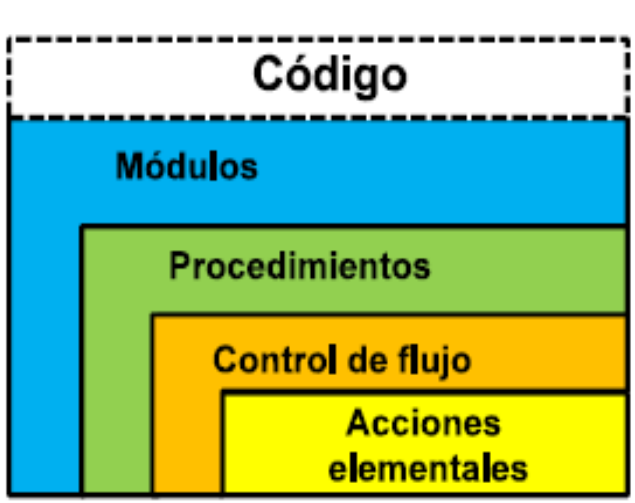
Tipos de paradigmas

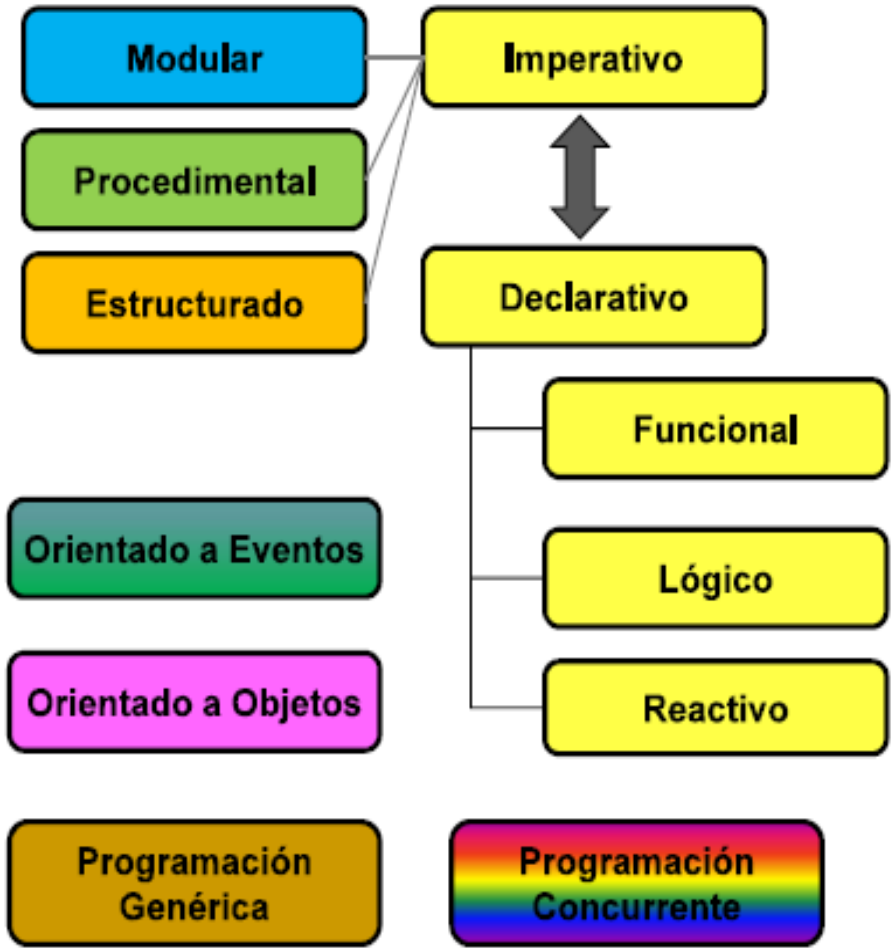
- Los **paradigmas fundamentales** están basados en diferentes **modelos de cómputo** y por lo tanto afectan a las construcciones más básicas de un programa.
- La división principal reside en el enfoque **imperativo** (indicar el **cómo** se debe calcular) y el enfoque **declarativo** (indicar el **qué** se debe calcular).
  - El enfoque declarativo tiene varias ramas diferenciadas: el paradigma **funcional**, el paradigma **lógico**, la programación **reactiva** y los lenguajes **descriptivos**.
- Otros paradigmas se centran en la estructura y organización de los programas, y son compatibles con los fundamentales:
  - Ejemplos: Programación estructurada, modular, **orientada a objetos**, **orientada a eventos**, programación genérica.
- Por último, existen paradigmas asociados a la **concurrencia** y a los **sistemas de tipado**.

Paradigmas de programación

- Un paradigma define un conjunto de reglas, patrones y estilos de programación que son usados por un grupo de lenguajes de programación
  - Paradigma funcional
  - Paradigma lógico
  - Paradigma imperativo o procedural
  - Paradigma orientado a objetos

El zoo de los paradigmas





Paradigma Imperativo

- Describe **cómo** debe realizarse el cálculo, no el porqué.
- Un cómputo consiste en una serie de sentencias, ejecutadas según un control de flujo **explícito**, que **modifican el estado** del programa.
- Las variables son **celdas de memoria** que contienen datos (o referencias), pueden ser modificadas, y representan el **estado** del programa.
- La sentencia principal es la **asignación**.
- Es el estándar 'de facto'.
  - Asociados al paradigma imperativo se encuentran los paradigmas **procedural**, **modular**, y la programación **estructurada**.
  - El lenguaje representativo sería FORTRAN-77, junto con COBOL, BASIC, PASCAL, C, ADA.
  - También lo implementan Java, C++, C#, Eiffel, Python, ...

Paradigma imperativo

- Definición de procedimientos
- Definición de tipos de datos
- Chequeo de tipos en tiempo de compilación
- Cambio de estado de variables
- Pasos de ejecución de un proceso

```
type
   tDimension = 1..100;
   eMatriz(f,c: tDimension) = array [1..f,1..c] of real;

   tRango = record
      f,c: tDimension value 1;
   end;

   tpMatriz = ^eMatriz;


procedure EscribirMatriz(var m: tpMatriz);
var filas,col : integer;
begin
   for filas := 1 to m^.f do begin
      for col := 1 to m^.c do
         write(m^[filas,col]:7:2);
      writeln(resultado);
      writeln(resultado)
     end;
end;
```

Paradigma Declarativo

- Describe **que** se debe cálcular, sin explicitar el **cómo**.
- No existe un orden de evaluación prefijado.
- Las variables son **nombres** asociados a **definiciones**, y una vez instanciadas son **inmutables**.
- No existe sentencia de asignación.
- El control de flujo suele estar asociado a la composición funcional, la recursividad y/o técnicas de reescritura y unificación.
  - Existen distintos grados de **pureza** en las variantes del paradigma.
  - Las principales variantes son los paradigmas **funcional**, **lógico**, la programación **reactiva** y los lenguajes descriptivos.

Programación Funcional

- Basado en los modelos de cómputo **cálculo lambda** (Lisp, Scheme) y **lógica combinatoria** (familia ML, Haskell)
- Las funciones son elementos de **primer orden**
- Evaluación por **reducción funcional.** Técnicas: recursividad, parámetros acumuladores, CPS, Mónadas.
- Familia LISP (Common-Lisp, Scheme):
  - Basados en **s-expresiones**.
  - Tipado debil.
  - Meta-programación
- Familia ML (Miranda, Haskell, Scala):
  - Sistema estricto de tipos (**tipado algebraico**)
  - Concordancia de patrones.
  - Transparencia referencial
  - Evaluación perezosa (estruct. de datos infinitas)

Paradigma funcional

- La computación se realiza mediante la evaluación de expresiones
- Definición de funciones
- Funciones como datos primitivos
- Valores sin efectos laterales, no existe la asignación
- Programación declarativa
- Lenguajes: LISP, Scheme, Haskell, Scala

```
(define (factorial x)
   (if (= x 0)
      1
      (* x (factorial (- x 1)))))

(factorial 8)
40320
(factorial 30)
265252859812191058636308480000000
```

Programación Lógica

- Basado en la **lógica de predicados** de primer orden
- Los programas se componen de hechos, predicados y relaciones.
- Evaluación basada en resolución SLD: unificación + backtracking.
- La ejecución consiste en la resolución de un problema de decisión, los resultados se obtienen mediante la instanciación de las variables libres.
- Lenguaje representativo: PROLOG

Paradigma lógico

- Definición de reglas
- Unificación como elemento de computación
- Programación declarativa
- Lenguajes: Prolog, Mercury, Oz.

```
padrede('juan', 'maria'). % juan es padre de maria
padrede('pablo', 'juan'). % pablo es padre de juan
padrede('pablo', 'marcela').
padrede('carlos', 'debora').

hijode(A,B) :- padrede(B,A).
abuelode(A,B) :-  padrede(A,C), padrede(C,B).
hermanode(A,B) :- padrede(C,A) , padrede(C,B), A \== B.

familiarde(A,B) :- padrede(A,B).
familiarde(A,B) :- hijode(A,B).
familiarde(A,B) :- hermanode(A,B).

?- hermanode('juan', 'marcela').
yes
?- hermanode('carlos', 'juan').
no
?- abuelode('pablo', 'maria').
yes
?- abuelode('maria', 'pablo').
no
```

Paradigma orientado a objetos

- Definición de clases y herencia
- Objetos como abstracción de datos y procedimientos
- Polimorfismo y chequeo de tipos en tiempo de ejecución
- Ejemplo en Java

```
public class Bicicleta {
    public int marcha;
    public int velocidad;

    public Bicicleta(int velocidadInicial, int marchaInicial) {
        marcha = marchaInicial;
        velocidad = velocidadInicial;
    }

    public void setMarcha(int nuevoValor) {
        marcha = nuevoValor;
    }

    public void frenar(int decremento) {
        velocidad -= decremento;
    }

    public void acelerar(int incremento) {
        velocidad += incremento;
    }
}

public class MountainBike extends Bicicleta {
    public int alturaSillin;

    public MountainBike(int alturaInicial, int velocidadInicial, int
marchaInicial) {
        super(velocidadInicial, marchaInicial);
        alturaSillin = alturaInicial;
    }

    public void setAltura(int nuevoValor) {
        alturaSillin = nuevoValor;
    }
}

public class Excursion {

        public static void main(String[] args) {
           MountainBike miBicicleta = new MoutainBike(10,10,3);
           miBicicleta.acelerar(10);
           miBicicleta.setMarcha(4);
           miBicicleta.frenar(10);
        }
}
```

Programación Reactiva (Dataflow)

- Basado en la **teoria de grafos.**
- Un programa consiste en la especificación del flujo de datos entre operaciones.
- Las variables se encuentran **ligadas** a las operaciones que proporcionan sus valores. Un cambio de valor de una variable se **propaga** a todas las operaciones en que participa.
- Las hojas de cálculo se basan en este modelo.
- Lenguajes representativos: Simulink, Oz, Clojure.

```
A := 10
B := A + 1
print B  ---> 11
A := 3
print B  ---> 4
```

Ejemplo: Algoritmo de Euclides

- Cálculo del máximo común divisor
  - Primer algoritmo no trivial. Euclides, año 300 A.C.

$$\mathrm{mcd}(a,b) = \max\{d : a \mid d,\ b \mid d\}$$

  - donde *a* | *d* significa que *a* es divisible exactamente por *d*:

$$a \mid d \Leftrightarrow \exists n : a = n \cdot d \qquad b \mid d \Leftrightarrow \exists m : b = m \cdot d$$

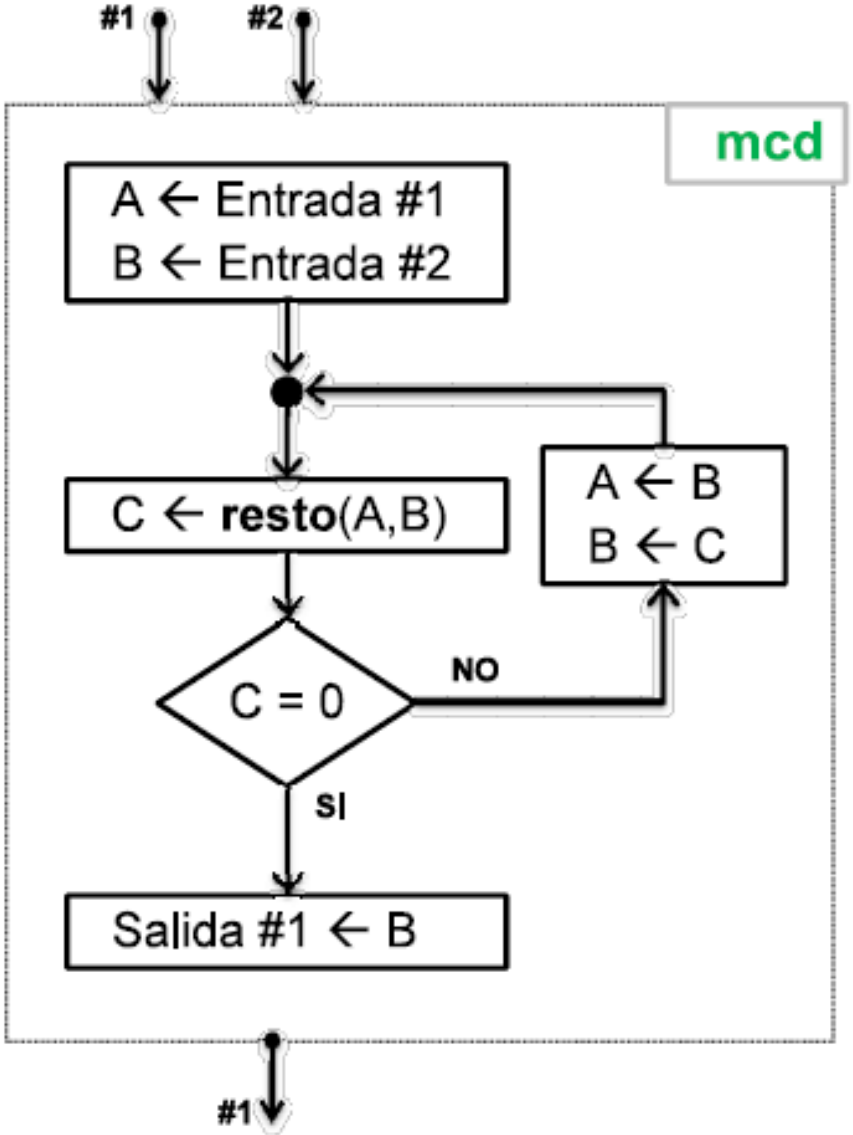
  - si *a* y *b* son divisibles por *d*, y *a* > *b*, entonces:

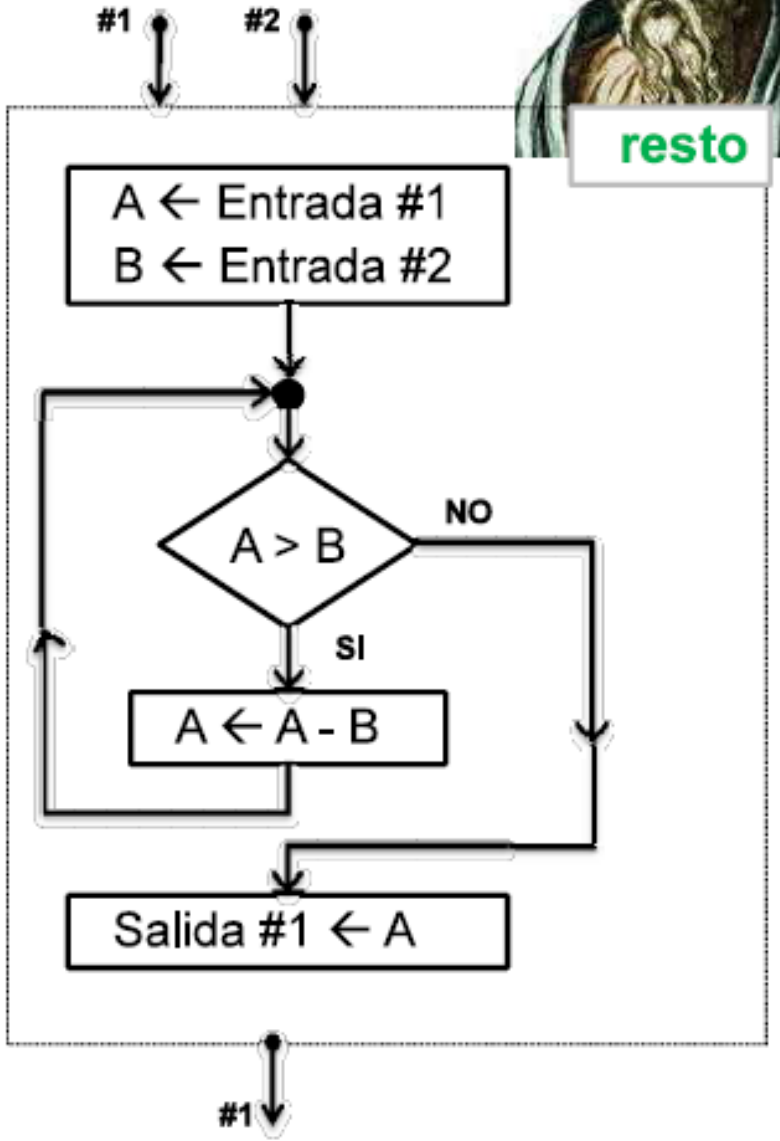
$$a - b = (n - m) \cdot d \Rightarrow (a - b) \mid d$$

  - y por lo tanto (restar sucesivamente *b* es equivalente a hallar el resto de dividir *a* por *b*):

$$a > b \Rightarrow \mathrm{mcd}(a,b) = \mathrm{mcd}(a-b,b) = \mathrm{mcd}(a \bmod b, b)$$

Diagrama de flujo





FORTRAN-77

- Imperativo, procedural, no estructurado

```
  FUNCTION MCD(NA, NB)
     IA = NA
     IB = NB
1    IF (IB.NE.0) THEN
        ITMP = IA
        IA = IB
        IB = MOD(ITMP, IB)
        GOTO 1
     END IF
     MCD = IA
     RETURN
  END
```

Paso por referencia

Tipado implícito

Saltos

PASCAL

- Imperativo, procedural, estructurado

```
function MCD(a,b: integer): integer;
var c: integer;
begin
  while b <> 0 do
  begin
    c := a;
    a := b;
    b := c mod b
  end;
  MCD := b
end;
```

Paso por valor

Tipado explícito

SCHEME, HASKELL, PROLOG

- Lenguajes funcionales y lógicos (recursividad)
  - Scheme

```
(define (mcd a b)
  (if (= b 0) a
      (mcd b (modulo a b))))
```

s-expresiones

  - Haskell

```
mcd :: Int -> Int -> Int
mcd a 0 = a
mcd a b = mcd b (rem a b)
```

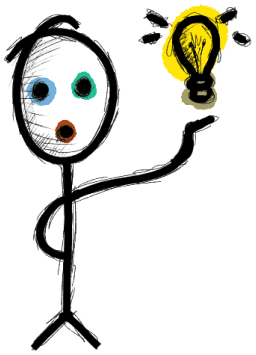
tipado estricto

concordancia de patrones

  - Prolog

```
mcd(A,0,D) :- A = D.
mcd(A,B,D) :- B > 0, C is A mod B, mcd(B,C,D).
```

predicados, unificación



Declarative vs imperative languages

	Imperative	Declarative
Paradigm	Describe HOW TO solve the problem	Describe WHAT the problem is
Program	A sequence of commands	A set of statements
Examples	C, Fortran, Ada, Java	Prolog, Pure Lisp, Haskell, ML
Advantages	Fast, specialized programs	General, readable, correct(?) programs.

Programación declarativa

- La programación declarativa es un estilo de programación en el que el programador especifica ***qué*** debe computarse más bien que ***cómo*** deben realizarse los cómputos.
- La programación declarativa provee el "qué", pero deja el "cómo" liberado a la implementación particular del intérprete.
- Por lo tanto se puede ver que la programación declarativa tiene dos fases bien diferenciadas, ***la declaración y la interpretación.***

- Según la clase de lógica que empleemos como fundamento del lenguaje declarativo obtenemos los diferentes estilos de programación declarativa.

Clase de lógica	Estilo
Ecuacional	Funcional
Clausal	Relacional
Heterogenea	Tipos
Géneros ordenados	Herencia
Modal	Sistemas Basados en Conocimiento
Temporal	Concurrencia

Programación Funcional

- Los *lenguajes funcionales* se basan en el concepto de *función* (matemática) y su definición mediante ecuaciones (generalmente recursivas), que constituyen el programa.
- La programación funcional se centra en la evaluación de expresiones (funcionales) para obtener un *resultado*.
- La composición de funciones es la técnica por excelencia de la programación funcional;
- Permite la construcción de programas mediante el empleo de funciones primitivas o previamente definidas por el usuario;
- La composición de funciones refuerza la modularidad de los programas.

- Mecanismo de cómputo que permite una *búsqueda indeterminista* (*built-in search*) de soluciones =>

  1. El programa puede responder a diferentes cuestiones (*objetivos*) sin necesidad de efectuar ningún cambio en el programa,
  2. Permite computar con *datos parcialmente definidos*,
  3. La *relación de entrada/salida* no está fijada de antemano.

- A pesar de ser un área de trabajo relativamente nueva la programación declarativa ha encontrado una gran variedad de aplicaciones:
  - Representación del Conocimiento, Resolución de Problemas, Desarrollo de Sistemas de Producción, Sistemas Expertos y Procesamiento del lenguaje natural.
  - Metaprogramación.
  - Prototipado de aplicaciones, Bases de Datos Deductivas, Servidores y buceadores de información inteligentes.
  - Química y biología molecular.
  - Diseño de sistemas VLSI.

Más generalmente, la programación declarativa se ha aplicado en todos los campos de la computación simbólica (lenguajes de computación simbólica).

```
function length (L: list): nat
B:bool;
aux:list;

B:= is_empty(L);
case B of
     true:  return 0;
     false: aux:=tail(L);
            return 1+length(aux)
end case
end function;
```

PROGRAMA IMPERATIVO

```
length(nil) = 0
length(E:L) = length(L)+1
```

PROGRAMA FUNCIONAL

```
length(nil,0)
length(E.L,N) ← length(L,M) ^ N = M+1
```

PROGRAMA LÓGICO