

Paradigmas y Lenguajes

Introducción



Lic. Ricardo Monzón



PARADIGMAS Y LENGUAJES

LICENCIATURA EN SISTEMAS DE INFORMACION
FACULTAD DE CIENCIAS EXACTAS Y NATURALES Y AGRIMENSURA-UNNE

2DO. AÑO PRIMER CUATRIMESTRE

PROFESOR TEORIA:

- LIC. RICARDO MONZON

PROFESORES DE PRACTICO

- LIC. SONIA CORRALES
- LIC. MONICA RINGA



AYUDANTES DE PRACTICA

- LIC. CARLOS ROMERO
- LIC. LEANDRO RODRIGUEZ

PARADIGMAS Y LENGUAJES

LICENCIATURA EN SISTEMAS DE INFORMACION
FACULTAD DE CIENCIAS EXACTAS Y NATURALES Y AGRIMENSURA-UNNE

HORARIOS

Teoria

Lunes de 14:00 a 17:00 Aula Magna
Sabados de 09:00 a 13:00 Aula Magna

Practica

Lunes de 17:00 a 20:00 Aula Magna (Letras A-L)
Miercoles de 14:00 a 17:00 Aula Magna (Letras M-Z)

Laboratorios

Jueves 14:00 a 16:00, 16:00 a 18:00, 20:00 a 22:00.
Viernes de 18:00 a 20:00



PARTE I: INTRODUCCION.

TEMA 1: PARADIGMAS y Paradigmas de Programación





Definición Teórica - Paradigma

«Un paradigma está constituido por los supuestos teóricos generales, las leyes y las técnicas para su aplicación que adoptan los miembros de una determinada comunidad científica.»

Podemos decir que, los paradigmas son marcos de referencia que imponen reglas sobre cómo se deben hacer las cosas, indican qué es válido dentro del paradigma y qué está fuera de sus límites.



Un paradigma distinto implica nuevas reglas, elementos, límites y maneras de pensar, o sea implica un cambio.

Definición Teórica – Paradigma de Programación

Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los lenguajes de programación, según nuestro interés de estudio.

Un paradigma de programación es entonces una forma de representar y manipular el conocimiento. Representa un enfoque particular o filosofía para la construcción del software. No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Tipos de Paradigmas de Programación

1. **Paradigma Imperativo:** Modelo abstracto que consiste en un gran almacenamiento de memoria donde la computadora almacena una representación codificada de un cálculo y ejecuta una secuencia de comandos que modifican el contenido de ese almacenamiento. Algoritmos + Estructura de Datos = Programa.
2. **Paradigma Procedimental:** Modelos de Desarrollo: Orientado a Eventos y a Agentes. Secuencia computacional realizada etapa a etapa para resolver el problema. Su mayor dificultad reside en determinar si el valor computado es una solución correcta del problema.

Tipos de Paradigmas de Programación

3. Paradigma Declarativo. - Modelos de Desarrollo: Funcional y Lógico. Se construye señalando hechos, reglas, restricciones, ecuaciones, transformaciones y otras propiedades derivadas del conjunto de valores que configuran la solución.

3.1 Paradigma Funcional: Modelo matemático de composición funcional donde el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que una composición produce el valor deseado. Como su nombre lo dice operan solamente a través de funciones. Cada función devuelve un solo valor, dada una lista de parámetros. La programación funcional proporciona la capacidad para que un programa se modifique así mismo, es decir que pueda aprender.

Tipos de Paradigmas de Programación

3.2. Paradigma Lógico: Esta programación se basa en un subconjunto del cálculo de predicados, incluyendo instrucciones escritas en formas conocidas como cláusulas de Horn (sentencias de lógica de primer orden). Este paradigma puede deducir nuevos hechos a partir de otros hechos conocidos. Permite un método particularmente mecánico de demostración llamado resolución. Representa conocimiento mediante relaciones (predicados) entre objetos (datos).

Un programa lógico consiste en un conjunto de relaciones, y su ejecución demuestra que una nueva relación se sigue de las que constituía el programa. Las relaciones se especifican con reglas y hechos. La ejecución consiste en la demostración de hechos sobre las relaciones por medio de preguntas.

Tipos de Paradigmas de Programación

4. Paradigma Demostrativo.- Modelos de Desarrollo: Genético. Cuando se programa bajo un paradigma demostrativo (también llamada programación por ejemplos), el programador no especifica procedimentalmente cómo construir una solución sino que presentan soluciones de problemas similares.

5. Paradigma Orientado a Objeto: disciplina de ingeniería de desarrollo y modelado de software que permite construir más fácilmente sistemas complejos a partir de componentes individuales. Objetos + Mensajes = Programa.

Describen los lenguajes que soportan objetos en interacción. Un objeto es un grupo de procedimientos que comparten un estado.

Ejemplos de paradigmas de programación:

- El paradigma imperativo es considerado el más común y está representado, por ejemplo, por el C o por BASIC.
- El paradigma funcional está representado por la familia de lenguajes LISP, en particular Scheme.
- El paradigma lógico, un ejemplo es PROLOG.
- El paradigma orientado a objetos. Un lenguaje completamente orientado a objetos es Smalltalk.

PARADIGMA DECLARATIVO

Es un paradigma de programación que está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla.

Diferencia entre imperativo y declarativo

En el paradigma imperativo se describe paso a paso un conjunto de instrucciones que deben ejecutarse para variar el estado del programa y hallar la solución, es decir, un algoritmo en el que se **describen los pasos necesarios para solucionar el problema.**

En el declarativo las sentencias que se utilizan lo que hacen es **describir el problema** que se quiere solucionar, pero no las instrucciones necesarias para solucionarlo. Esto último se realizará mediante mecanismos internos de inferencia de información.



PARTE I: INTRODUCCION.

TEMA 2: Lenguajes de Programación






Definición Teórica – Lenguaje de Programación

Un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador.

Un lenguaje le da la capacidad al programador de especificarle al computador, qué tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano.




Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, interpretación o intermedio, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador.



Clasificación de Lenguajes de Programación

Una primera clasificación puede ser según el tipo:

- **Lenguajes interpretados** (Interpretes) como Basic, Dbase.
 - **Lenguajes compilados** (Compiladores) como C, C++, Clipper.
 - **Lenguajes interpretados con recolectores de basura** (Maquina Virtual) como Smalltalk, Java, Ocaml.
 - **Lenguajes Scripts** (Motor de ejecución) como Perl, PhP.
- 

Clasificación de Lenguajes de Programación

Declarativos: en ciencias computacionales son aquellos lenguajes de programación en los cuales se le indica a la computadora que es lo que se desea obtener o que es lo que se esta buscando. Por ejemplo: *Obtener los nombres de todos los empleados que tengan más de 32 años*. Algunos ejemplos de lenguajes declarativos son el LISP, SQL, Prolog.

Imperativos: En ciencias de la computación se llama lenguajes imperativos a aquellos en los cuales se le ordena a la computadora cómo realizar una tarea siguiendo una serie de pasos o instrucciones, por ejemplo: *Paso 1, solicitar número. Paso 2, multiplicar número por dos. Paso 3, imprimir resultado de la operación. Paso 4, etc.* Algunos ejemplos de lenguajes imperativos son: BASIC, C, C++, Java, Clipper, Dbase, C# y Perl.

GENERACIONES DE LOS LENGUAJES

Los equipos de ordenador (el hardware) han pasado por cuatro generaciones, de las que las tres primeras (ordenadores con válvulas, transistores y circuitos integrados) están muy claras, la cuarta (circuitos integrados a gran escala) es más discutible.

Algo parecido ha ocurrido con la programación de los ordenadores (el software), que se realiza en lenguajes que suelen clasificarse en cinco generaciones, de las que las tres primeras son evidentes, mientras no todo el mundo está de acuerdo en las otras dos. Estas generaciones no coincidieron exactamente en el tiempo con las de hardware, pero sí de forma aproximada, y son las siguientes:

GENERACIONES DE LOS LENGUAJES

Primera generación: Los primeros ordenadores se programaban directamente en código binario, que puede representarse mediante secuencias de ceros y unos sistema binario. Cada modelo de ordenador tiene su propio código, por esa razón se llama lenguaje de máquina.

Segunda generación: Los lenguajes simbólicos, así mismo propios de la máquina, simplifican la escritura de las instrucciones y las hacen más legibles.

Tercera generación: Los lenguajes de alto nivel sustituyen las instrucciones simbólicas por códigos independientes de la máquina, parecidas al lenguaje humano o al de las Matemáticas.

GENERACIONES DE LOS LENGUAJES

Cuarta generación: se ha dado este nombre a ciertas herramientas que permiten construir aplicaciones sencillas combinando piezas prefabricadas. Hoy se piensa que estas herramientas no son, propiamente hablando, lenguajes. Algunos proponen reservar el nombre de cuarta generación para la programación orientada a objetos.

Quinta generación: se llama así a veces a los lenguajes de la inteligencia artificial, aunque con el fracaso del proyecto japonés de la quinta generación el nombre ha caído en desuso.

ESTILOS DE PROGRAMACIÓN

La **Programación Imperativa**, en contraposición a la programación declarativa es un paradigma de programación que describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican al computador cómo realizar una tarea.

La implementación de hardware de la mayoría de computadores es imperativa; prácticamente todo el hardware de los computadores está diseñado para ejecutar código de máquina, que es nativo al computador, escrito en una forma imperativa.

Los primeros lenguajes imperativos fueron los lenguajes de máquina de los computadores originales.

Ej: Lenguaje C.

ESTILOS DE PROGRAMACIÓN

La **Programación Funcional** es un paradigma de programación declarativa basado en la utilización de funciones matemáticas.

Sus orígenes provienen del Cálculo Lambda (o λ -cálculo), una teoría matemática elaborada por Alonzo Church como apoyo a sus estudios sobre computabilidad.

Existen dos grandes categorías de lenguajes funcionales: los funcionales *puros* y los *híbridos*. La diferencia entre ambos estriba en que los lenguajes funcionales híbridos son menos dogmáticos que los puros, al admitir conceptos tomados de los lenguajes imperativos, como las secuencias de instrucciones o la asignación de variables.

Entre los lenguajes funcionales puros, cabe destacar a Haskell y Miranda. Los lenguajes funcionales híbridos más conocidos son Lisp, Scheme, Ocaml y Standard ML (estos dos últimos, descendientes del lenguaje ML).

ESTILOS DE PROGRAMACIÓN

Ejemplo de Código LISP para el Cálculo de una Potencia (utilizando recursividad).

```
(defun potencia-aux (b e)
  (if (= e 0) 1
      (* b (potencia-aux b (1- e)))))
```

```
(potencia-aux '3 '4)
81
```

ESTILOS DE PROGRAMACIÓN

La **Programación Lógica** consiste en la aplicación del *corpus* de conocimiento sobre lógica para el diseño de lenguajes de programación; no debe confundirse con la disciplina de la lógica computacional. Gira en torno al concepto de **predicado**, o relación entre elementos.

La programación lógica encuentra su *hábitat* natural en aplicaciones de inteligencia artificial o relacionadas:

Sistemas expertos, donde un sistema de información imita las recomendaciones de un experto sobre algún dominio de conocimiento.

Demostración automática de teoremas, donde un programa genera nuevos teoremas sobre una teoría existente.

Reconocimiento de lenguaje natural, donde un programa es capaz de comprender (con limitaciones) la información contenida en una expresión lingüística humana.

ESTILOS DE PROGRAMACIÓN

El lenguaje de programación lógica por excelencia es **Prolog**. Los programas construidos en un lenguaje lógico están contruidos únicamente por expresiones lógicas, es decir, que son ciertas o falsas, en oposición a una expresión interrogativa (una pregunta) o expresiones imperativas (una orden).

Prolog, proveniente del inglés **Programming in Logic**, es un lenguaje lógico bastante popular en el medio de investigación en Inteligencia Artificial.

Las instrucciones de Prolog se llaman **"reglas o cláusulas de Horn"**. En Prolog el orden de ejecución de las instrucciones no tiene nada que ver con el orden en que fueron escritas. Tampoco hay instrucciones de control propiamente dichas. Para trabajar con este lenguaje, un programador debe acostumbrarse a pensar de una manera muy diferente a la que se utiliza en los lenguajes clásicos.

ESTILOS DE PROGRAMACIÓN

%% declaraciones

padrede('juan', 'maria'). % juan es padre de maria

padrede('pablo', 'juan'). % pablo es padre de juan

padrede('pablo', 'marcela').

padrede('carlos', 'debora').

% A es hijo de B si B es padre de A

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

% A es abuelo de B si A es padre de C y C es padre B

abuelode(A,B) :- padrede(A,C), padrede(C, B).

% A y B son hermanos si el padre de A es tambien el padre de B y si
A y B no son lo mismo

hermanode(A,B) :- padrede(C,A) , padrede(C,B), A <> B.

ESTILOS DE PROGRAMACIÓN

%% CONSULTAS

% juan es hermano de marcela?

?- **hermanode('juan', 'marcela').**

si

% carlos es hermano de juan?

?- **hermanode('carlos', 'juan').**

no

% pablo es abuelo de maria?

?- **abuelode('pablo', 'maria').**

si

% maria es abuelo de pablo?

?- **abuelode('maria', 'pablo').**

no

ESTILOS DE PROGRAMACIÓN

La **programación estructurada** es una forma de escribir programación de ordenador de forma clara, para ello utiliza únicamente tres estructuras: secuencial, selectiva e iterativa; siendo innecesario y no permitiéndose el uso de la instrucción o instrucciones de transferencia incondicional (GOTO).

Surge del Teorema de Dijkstra, demostrado por Edsger Dijkstra en los años sesenta, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

- Secuencia
- Instrucción condicional.
- Iteración, o bucle de instrucciones.

ESTILOS DE PROGRAMACIÓN

La **programación dirigida por eventos** es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema o que ellos mismos provoquen.

El creador de un programa dirigido por eventos debe definir los eventos que manejará su programa y las acciones que se realizarán al producirse cada uno de ellos, lo que se conoce como el manejador de evento. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo programador.

Un ejemplo es el Visual Basic, en el que a cada elemento del programa (objetos, controles, etcétera) se le asignan una serie de eventos que generará dicho elemento, como la pulsación de un botón del ratón sobre él o el redibujado del control.

ESTILOS DE PROGRAMACIÓN

La **Programación Orientada a Aspectos (POA)** es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

Gracias a la POA se pueden capturar los diferentes conceptos que componen una aplicación en entidades bien definidas, de manera apropiada en cada uno de los casos y eliminando las dependencias inherentes entre cada uno de los módulos.

De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables.

ESTILOS DE PROGRAMACIÓN

La **Programación Orientada a Aspectos (POA)** es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos.

Gracias a la POA se pueden capturar los diferentes conceptos que componen una aplicación en entidades bien definidas, de manera apropiada en cada uno de los casos y eliminando las dependencias inherentes entre cada uno de los módulos.

De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables.

ESTILOS DE PROGRAMACIÓN

La **Programación Orientada a Objetos** (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan **estado** (es decir, datos), **comportamiento** (esto es, procedimientos o métodos) e **identidad** (propiedad del objeto que lo diferencia del resto).

La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

ESTILOS DE PROGRAMACIÓN

De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases (e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos).

A su vez, dispone de mecanismos de interacción (los llamados métodos) que favorecen la comunicación entre objetos (de una misma clase o de distintas), y en consecuencia, el cambio de estado en los propios objetos.

Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan (ni deben separarse) información (datos) y procesamiento (métodos).

ESTILOS DE PROGRAMACIÓN

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida.

La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan.

En la programación estructurada se escriben funciones y después les pasan datos. Los programadores que emplean lenguajes orientados a objetos definen objetos con datos y métodos y después envían mensajes a los objetos diciendo que realicen esos métodos en sí mismos.

ESTILOS DE PROGRAMACIÓN

Las principales diferencias entre la programación estructurada y la orientada a objetos son:

- La programación orientada a objetos es más moderna, es una evolución de la programación estructurada que plasma en el diseño de una familia de lenguajes conceptos que existían previamente con algunos nuevos.
- La programación orientada a objetos se basa en lenguajes que soportan sintáctica y semánticamente la unión entre los tipos abstractos de datos y sus operaciones (a esta unión se la suele llamar clase).
- La programación orientada a objetos incorpora en su entorno de ejecución mecanismos tales como el polimorfismo y el envío de mensajes entre objetos.

ESTILOS DE PROGRAMACIÓN

Lenguajes orientados a objetos

Entre los lenguajes orientados a objetos destacan los siguientes:

Ada , C++ , C# , VB.NET , Visual FoxPro , Clarion , Delphi , Harbour , Eiffel , Java , Lexico (en castellano) , Objective-C , Ocaml , Oz , Perl (soporta herencia múltiple) , PHP (en su versión 5) , PowerBuilder , Python , Ruby , Smalltalk , Magik (SmallWorld) .

ESTILOS DE PROGRAMACIÓN

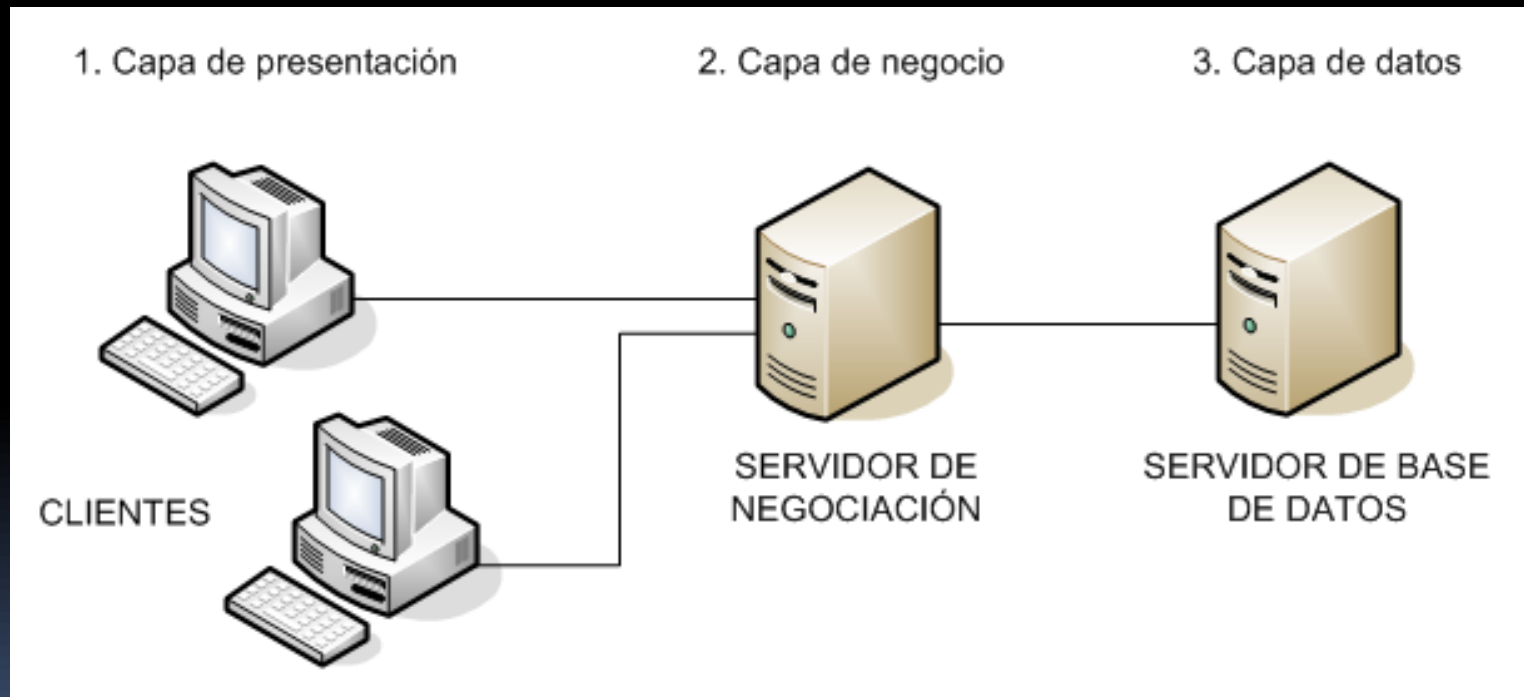
La **Programación con restricciones** es un paradigma de la programación en informática, donde las relaciones entre las variables son expresadas en términos de restricciones (ecuaciones). Actualmente es usada como una tecnología de software para la descripción y resolución de problemas combinatorios particularmente difíciles, especialmente en las áreas de planificación y programación de tareas (calendarización).

Este paradigma recientemente fue identificado por la ACM (Asociación de Maquinaria Computacional) como una dirección estratégica en la investigación en computación.

Se trata de un paradigma de programación basado en la especificación de un conjunto de restricciones, las cuales deben ser satisfechas por cualquier solución del problema planteado, en lugar de especificar los pasos para obtener dicha solución. Se relaciona mucho con la programación lógica y con la investigación operativa.

ESTILOS DE PROGRAMACIÓN

La **programación por capas** es un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.



ESTILOS DE PROGRAMACIÓN

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería: Modelo de interconexión de sistemas abiertos

Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, simplemente es necesario conocer la API que existe entre niveles.