UAEM | Universidad Autónoma del Estado de México

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN

UNIDAD DE APRENDIZAJE: ANÁLISIS DE LOS LENGUAJES DE PROGRAMACIÓN.

Créditos institucionales de la UA: 6
Material visual: Diapositivas

Unidad de competencia II

A N Á L I S I S D E L O S P A R A D I G M A S D E P R O G R A M A C I Ó N .

Elaboró M. en C. Selene Palacios Astudillo.

Período 2015-A

¿Cómo emplear este material?

El presente material tiene como finalidad facilitar la exposición gráfica de la Unidad de Competencia “Análisis de los Paradigmas de Programación” que se aborda en la Unidad de Aprendizaje “Análisis de los lenguajes de programación”.

La presentación deberá ir acompañada de una explicación oral del docente, ya que la aportación que pueda hacer mediante ejemplos y situaciones cotidianas brindará la oportunidad de que los estudiantes comprendan la importancia de construir argumentos sólidos, creíbles y bien soportados.

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

ÍNDICE

Tema	Diapositiva
Concepto de Paradigma.	5
Tipos de Paradigma.	14
Evolución, Características, Ventajas, Desventajas y lenguajes representativos de los siguientes paradigmas:	22
Imperativo	
Funcional	
Lógico	
Orientado a Objetos	
Bibliografía	35

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Objetivo de la Unidad Temática.

Al término de la unidad temática, los estudiantes tendrán la capacidad, de comprender lo que refiere a cada paradigma, así como las principales características que distinguen a cada uno de ellos.

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Un grupo de científicos, colocó cinco monos en una jaula

PARADIGMA

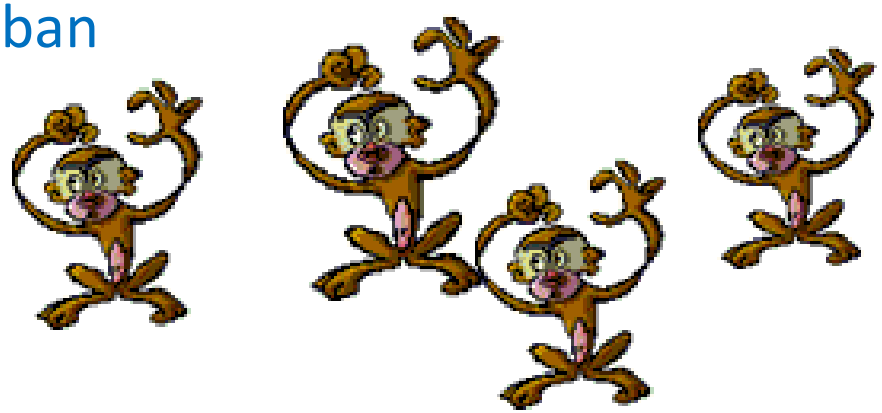
Al centro colocaron una escalera, y sobre ella una penca de plátanos.

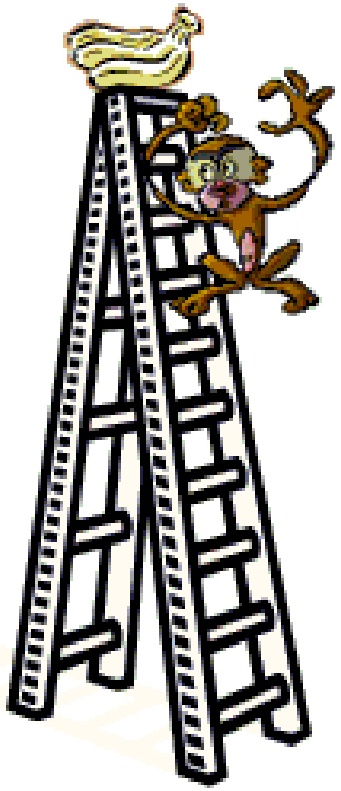
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Cuando un mono, subía la escalera para agarrar plátanos…

PARADIGMA

Los científicos lanzaban un chorro de agua fría, sobre los que quedaban en el suelo.

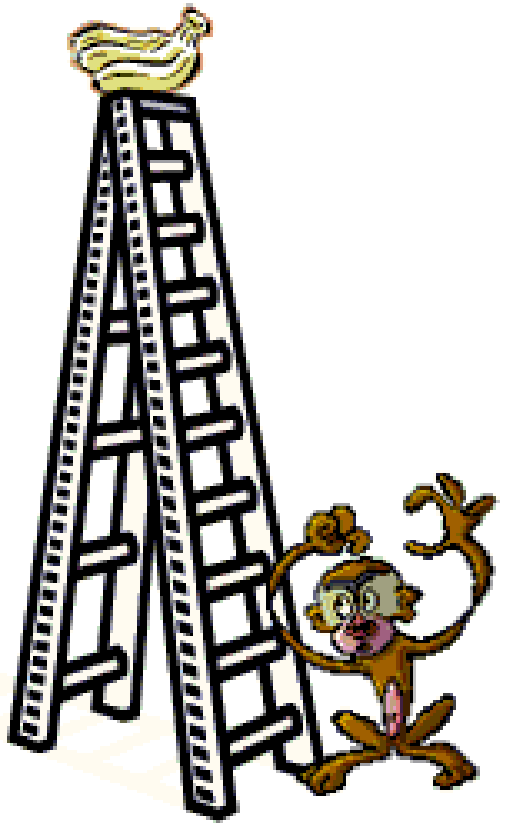




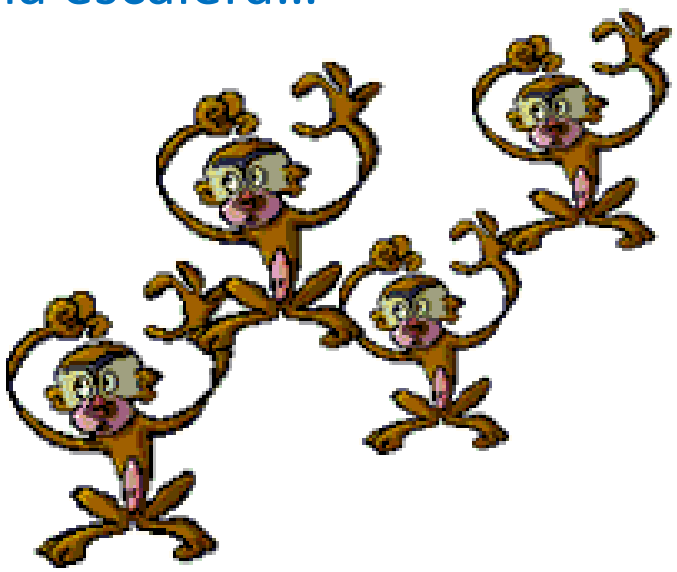
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Después de algún tiempo…

PARADIGMA



Cuando un mono iba a subir la escalera…



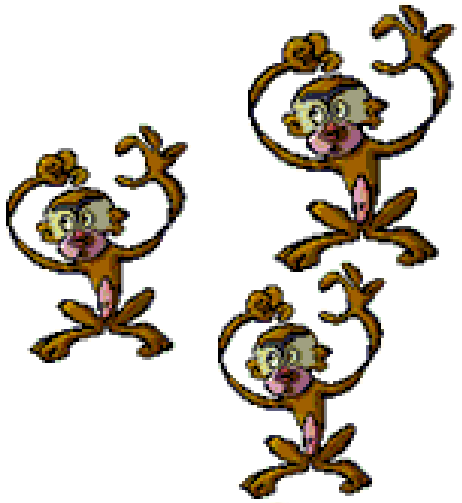
los otros lo golpeaban

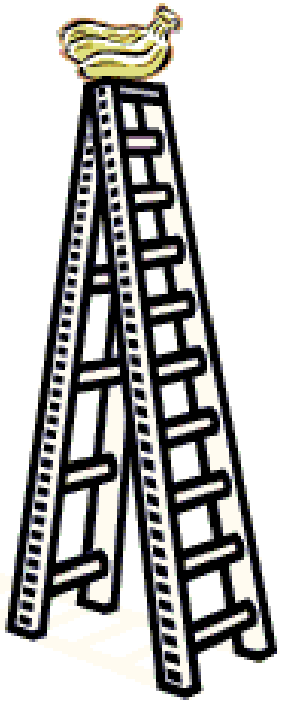
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

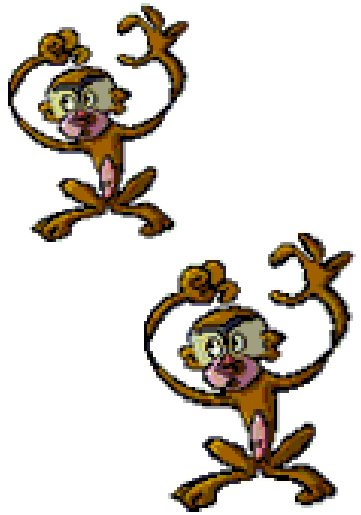
Con el paso del tiempo…

PARADIGMA

Ningún mono subía la escalera, a pesar de la tentación de los plátanos







Entonces, los científicos, sustituyeron uno de los monos.

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

PARADIGMA

Lo primero que hizo, fue subir la escalera… siendo rápidamente bajado por los otros, quienes le acomodaron tremendos golpes.

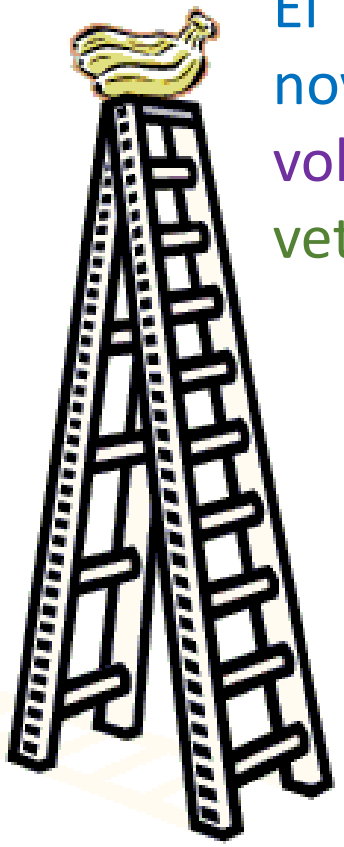
Después de algunos golpes, el nuevo integrante del grupo ya no subió más la escalera. Aunque nunca supo el porqué de los golpes.

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

PARADIGMA

Un segundo mono fue sustituido, y ocurrió lo mismo.

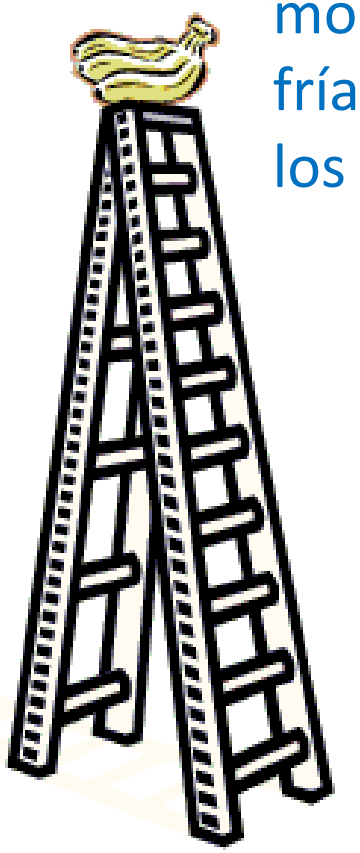
El primer sustituto participó con entusiasmo de los golpes al novato. Un tercero fue cambiado y se repitió el hecho, lo volvieron a golpear. El cuarto y, finalmente, el quinto de los veteranos fue sustituido.

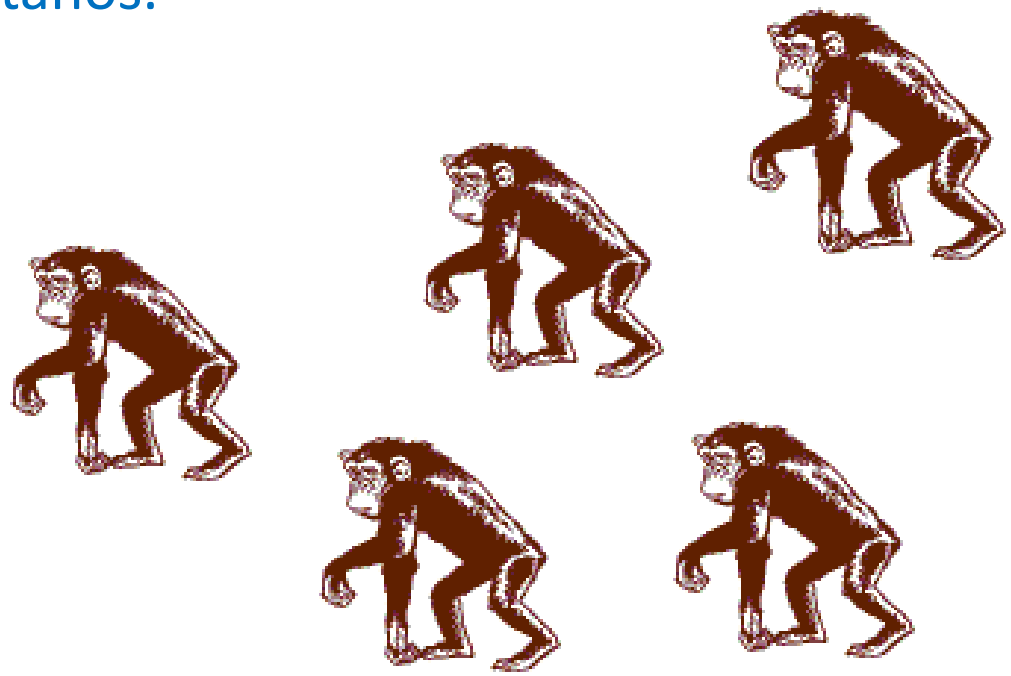


ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

PARADIGMA

Los científicos quedaron entonces, con un grupo de cinco monos que, aún cuando nunca recibieron un baño de agua fría, continuaban golpeando a aquel que intentase llegar a los plátanos.

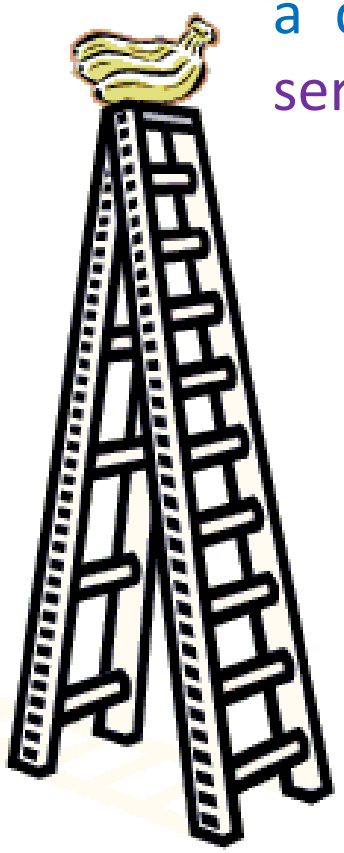


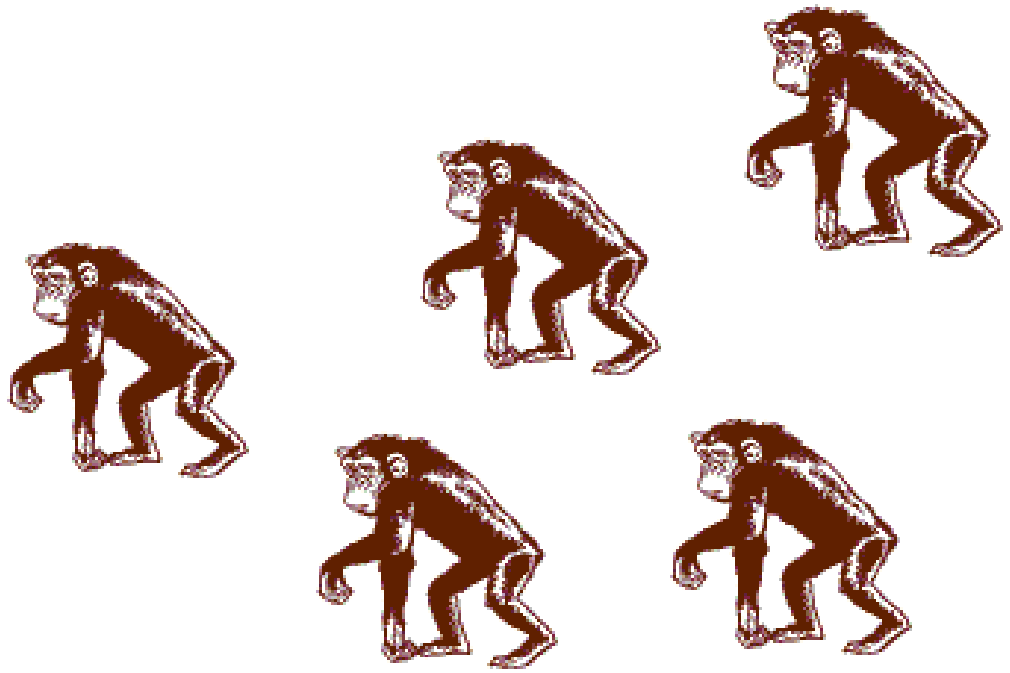


ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

PARADIGMA

Si fuese posible preguntar a algunos de ellos ¿Por qué le pegaban a quien intentaba subir la escalera?, con certeza la respuesta sería: “No sé, aquí las cosas siempre se han hecho así”.





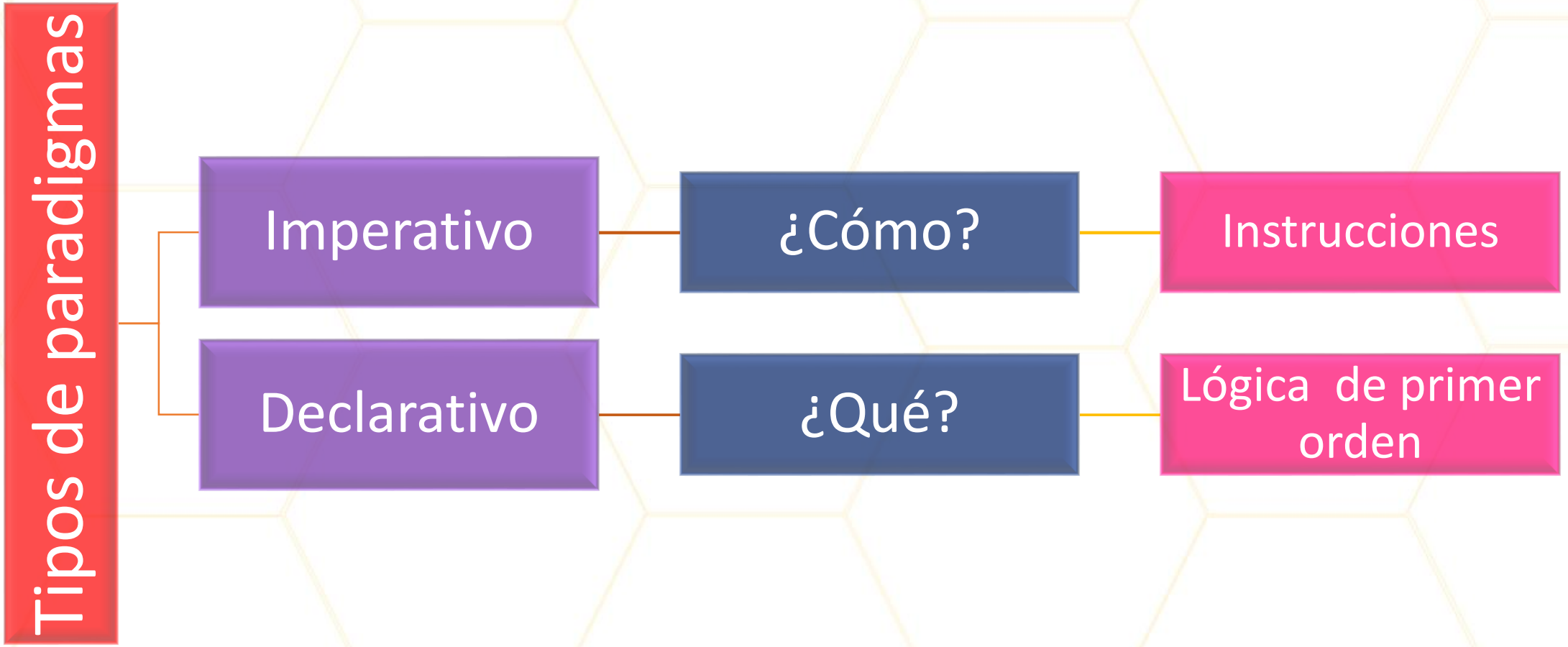
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

PARADIGMA

- **Thomas Kuhn, “ logros científicos universalmente reconocidos, que durante un tiempo proporcionan problemas y soluciones modelo, para una comunidad profesional”.**

- **Colección de características abstractas que categorizan un grupo de lenguajes que son aceptados y utilizados por un grupo de profesionales.**

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.



ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

- Imperativo
  - Estructurado
  - Orientado a Objetos
  - Modular
- Declarativo
  - Funcional
  - Lógico
- Dirigida por eventos
- Orientada a Aspectos

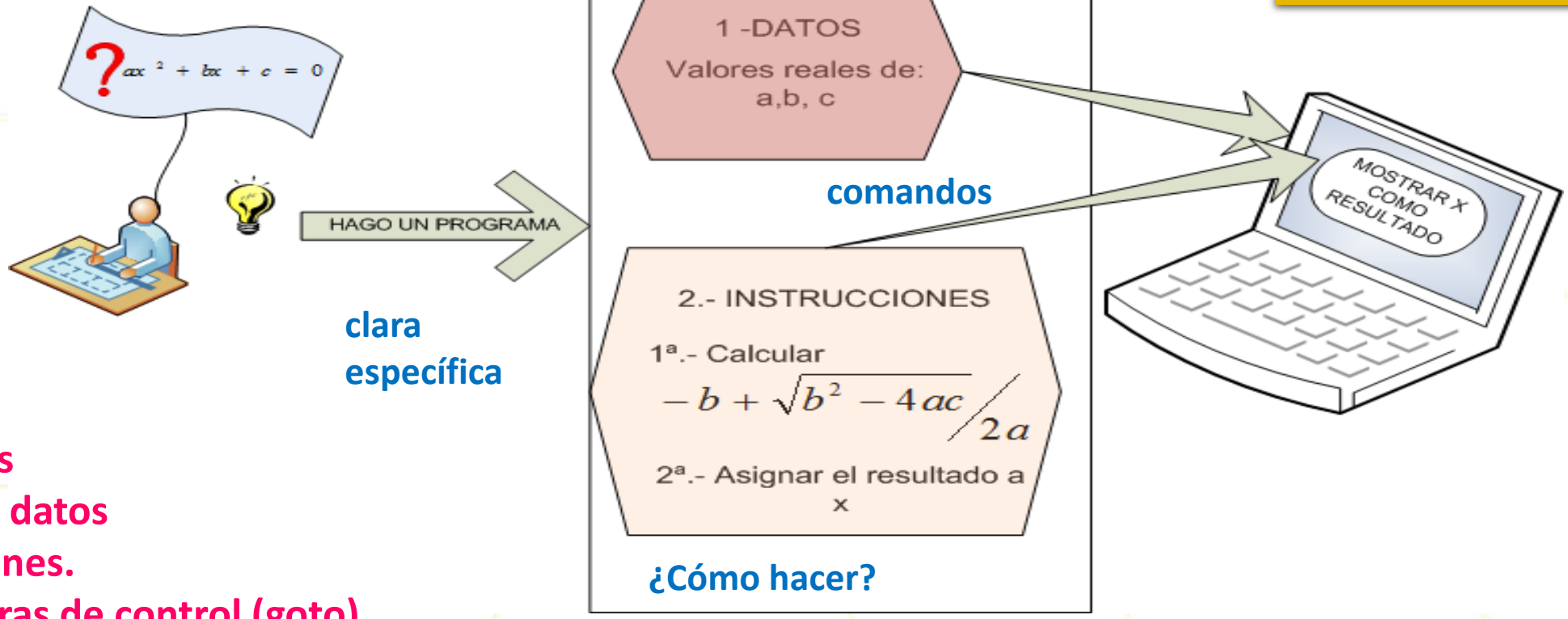
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Paradigma Imperativo

- Imperar: mandar, ordenar.
- Estructura y Módulo (análisis descendente).
- Celda de memoria
- Operaciones de asignación (aritméticas, lógicas, cadena de caracteres).
- Operaciones de repetición.
- Secuencia de transformación de datos

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Paradigma Imperativo



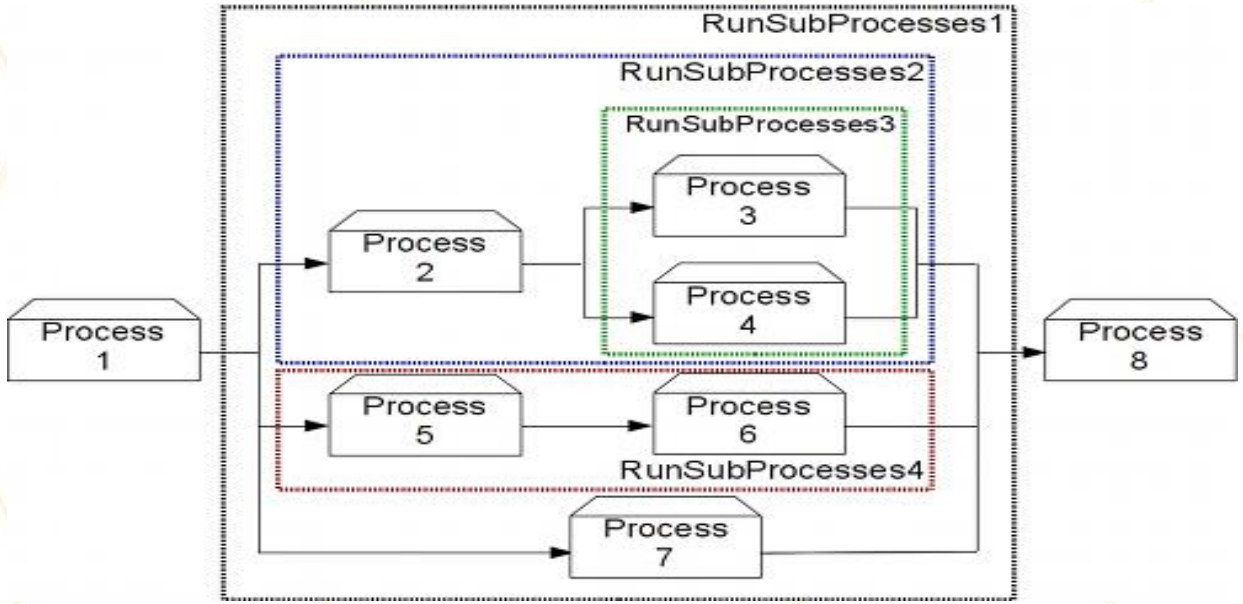
Variables
Tipos de datos
Expresiones.
Estructuras de control (goto).

Basic, C, C++, Java, Clipper, Dbase, Pascal, C#, Perl.

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Paradigma Imperativo

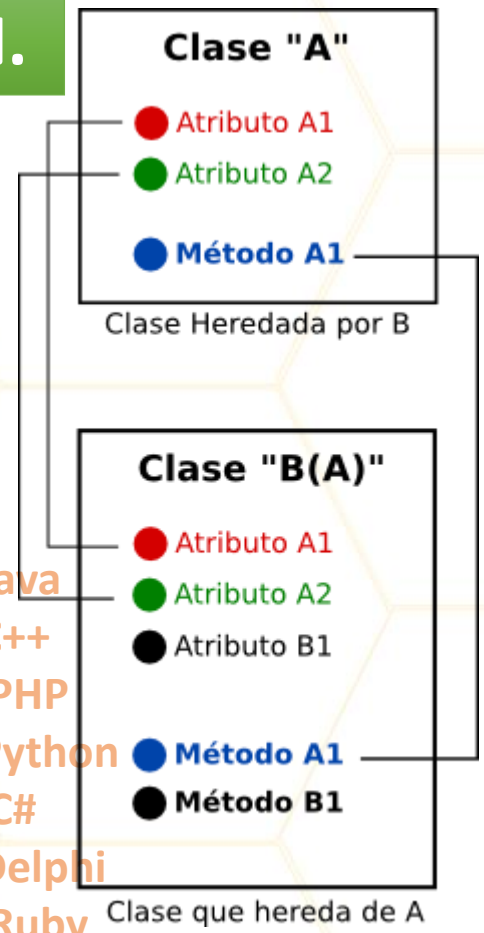
- Pueden ser llamados desde otra parte del programa.
- Toda estructura posee un solo punto de entrada y uno solo de salida de la misma.
- Varios caminos desde la entrada hasta la salida.
- Instrucciones pueden ser ejecutables en algún momento y no existen bucles infinitos.



ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Paradigma Imperativo Orientado a Objetos

- Clase (Tipo de dato)
- Objeto.
- Encapsulamiento (código o datos, ocultos para cualquier entidad externa a él).
- Herencia (nuevos objetos a partir de definición de otros).
- Polimorfismo(actuar modo diferente en función del objeto aplicado).
- Reutilización de códigos
- Facilidad para pensar soluciones a determinados problemas.
- Facilitar el desarrollo y comprensión de programas de gran porte.



ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

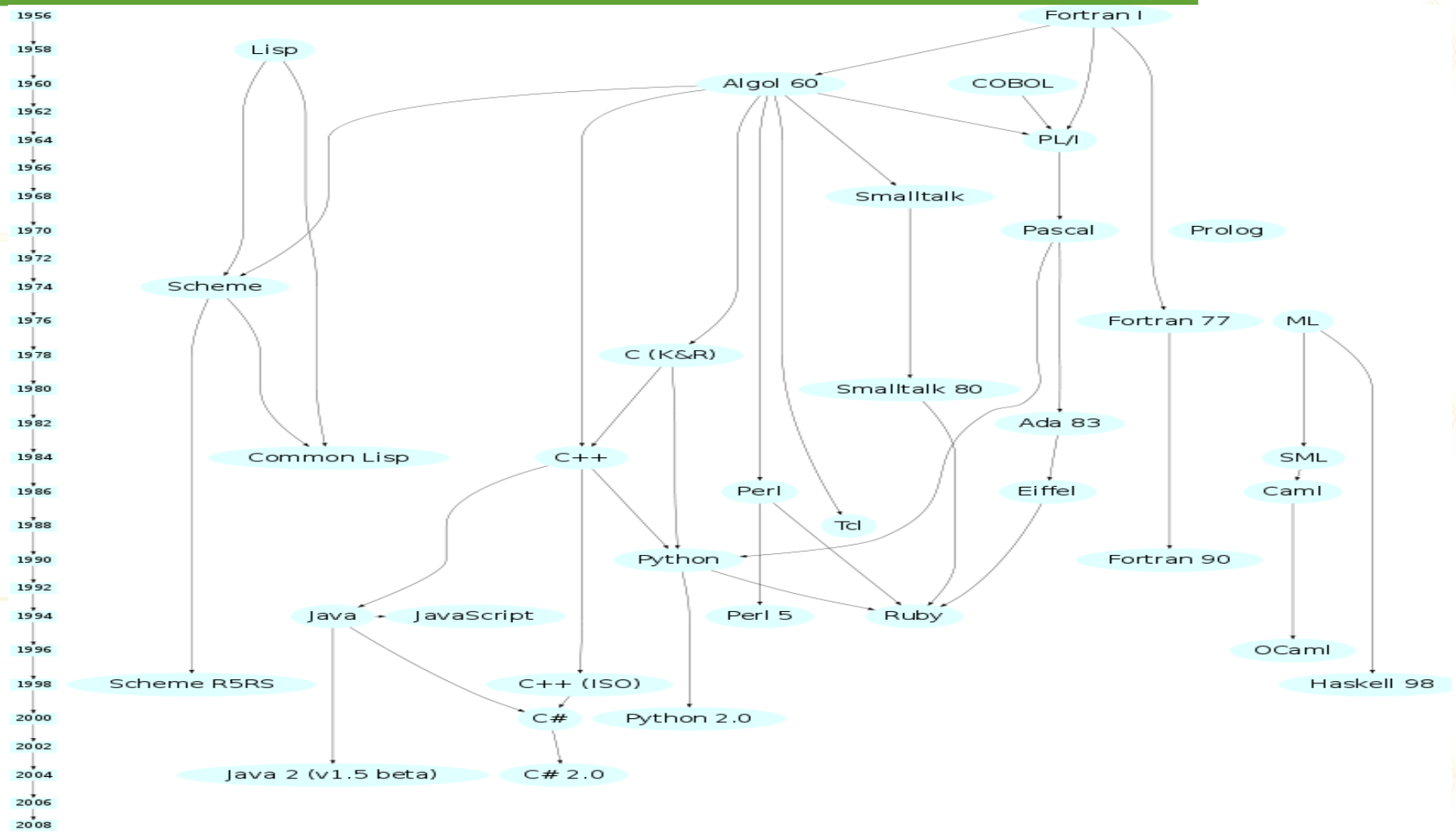
Paradigma Declarativo	Describe que se debe calcular.
	Las variables son nombres asociados a definiciones y una vez instanciadas son inmutables.
	El control de flujo, suele estar asociado a la composición funcional, la recursividad y/o técnicas de reescritura y unificación.
	No existe sentencia de asignación
	Lisp, Scheme, Haskell, Prolog.

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Sep 2015	Sep 2014	Change	Programming Language	Ratings	Change
1	2	↑	Java	19.565%	+5.43%
2	1	↓	C	15.621%	-1.10%
3	4	↑	C++	6.782%	+2.11%
4	5	↑	C#	4.909%	+0.56%
5	8	↑	Python	3.664%	+0.88%
6	7	↑	PHP	2.530%	-0.59%
7	9	↑	JavaScript	2.342%	-0.11%
8	11	↑	Visual Basic .NET	2.062%	+0.53%
9	12	↑	Perl	1.899%	+0.53%
10	3	⇊	Objective-C	1.821%	-8.11%
11	29	⇈	Assembly language	1.806%	+1.22%
12	13	↑	Ruby	1.783%	+0.50%
13	15	↑	Delphi/Object Pascal	1.745%	+0.59%
14	14		Visual Basic	1.532%	+0.26%
15	17	↑	Pascal	1.298%	+0.40%
16	18	↑	Swift	1.188%	+0.34%

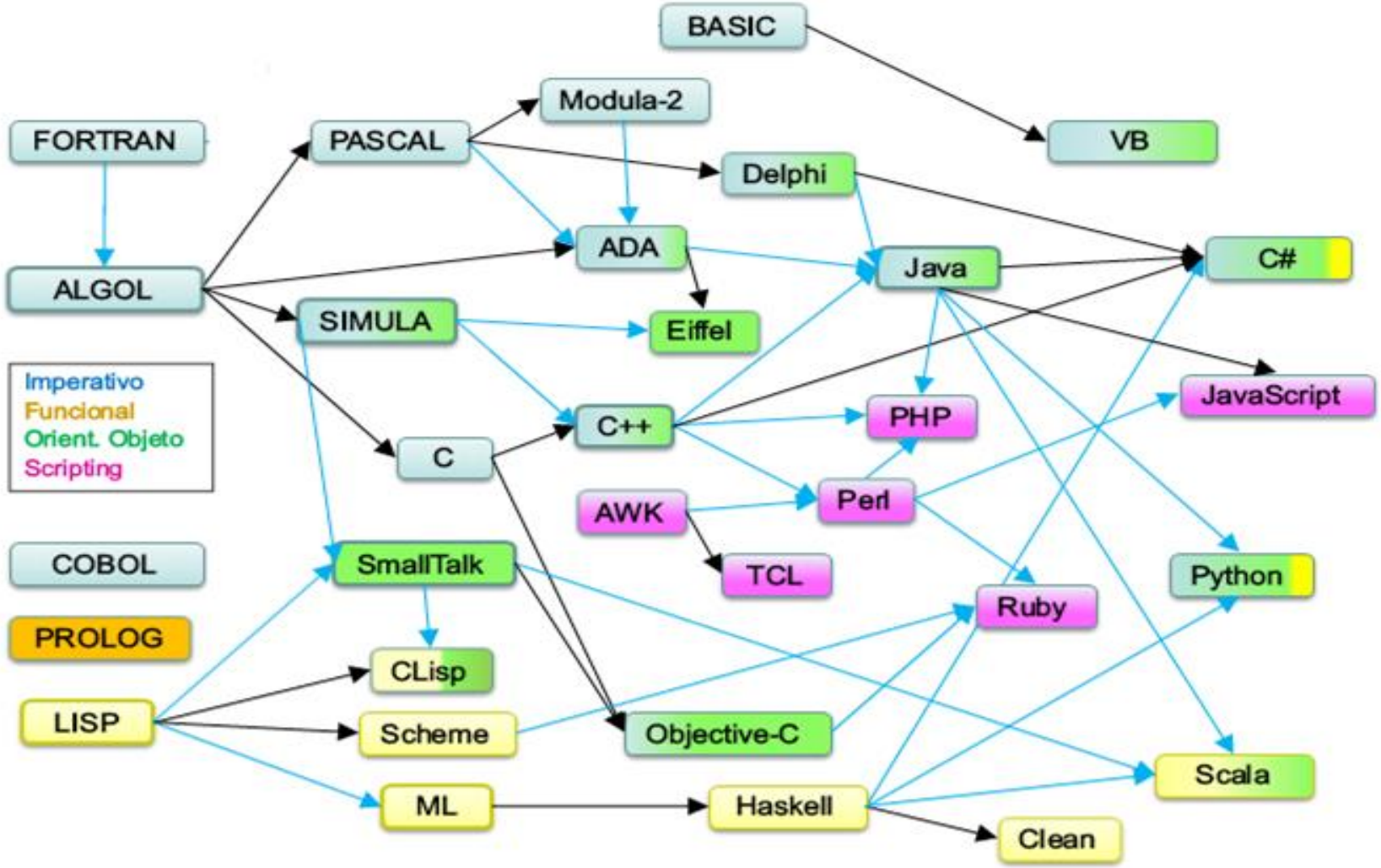
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Evolución.



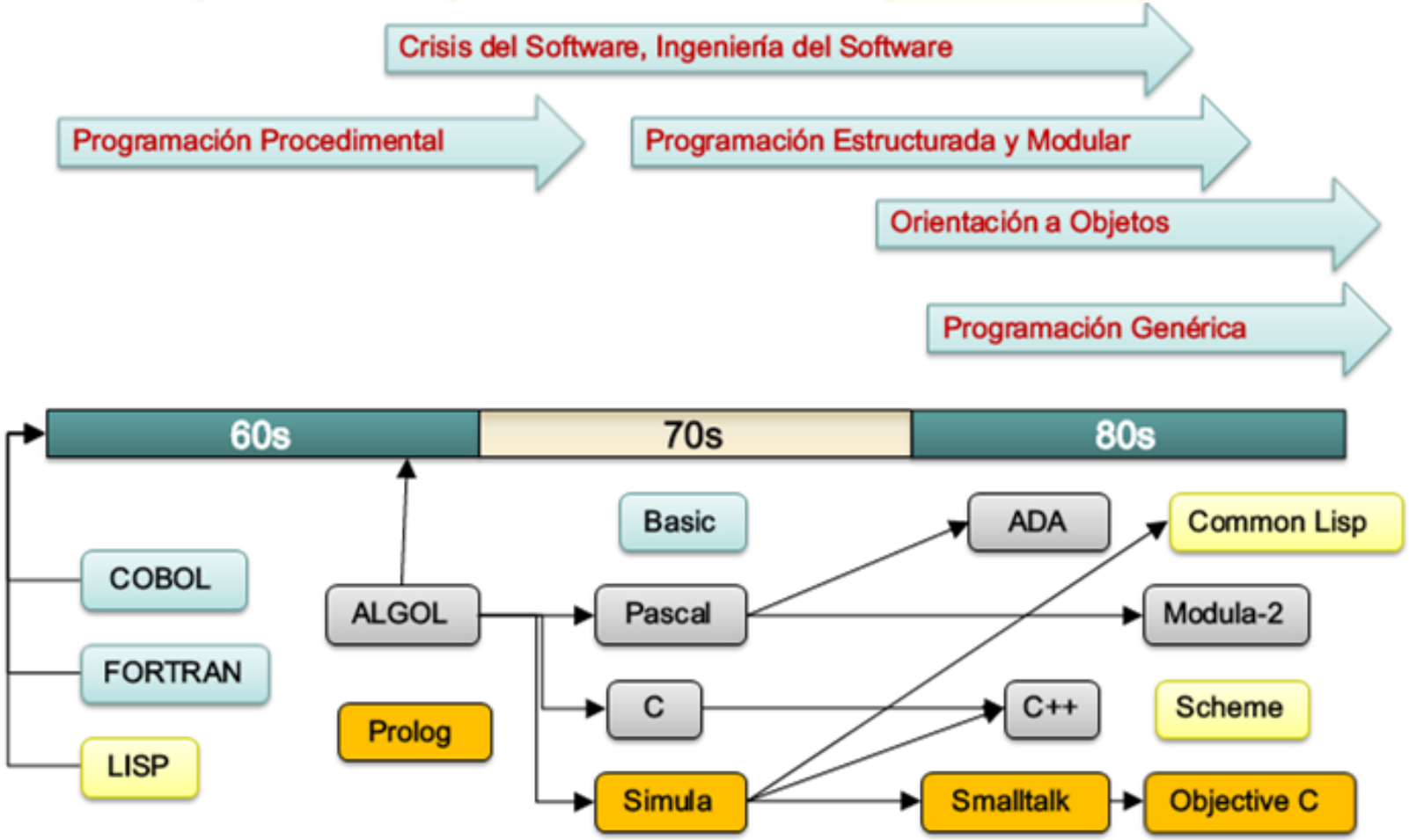
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Evolución.



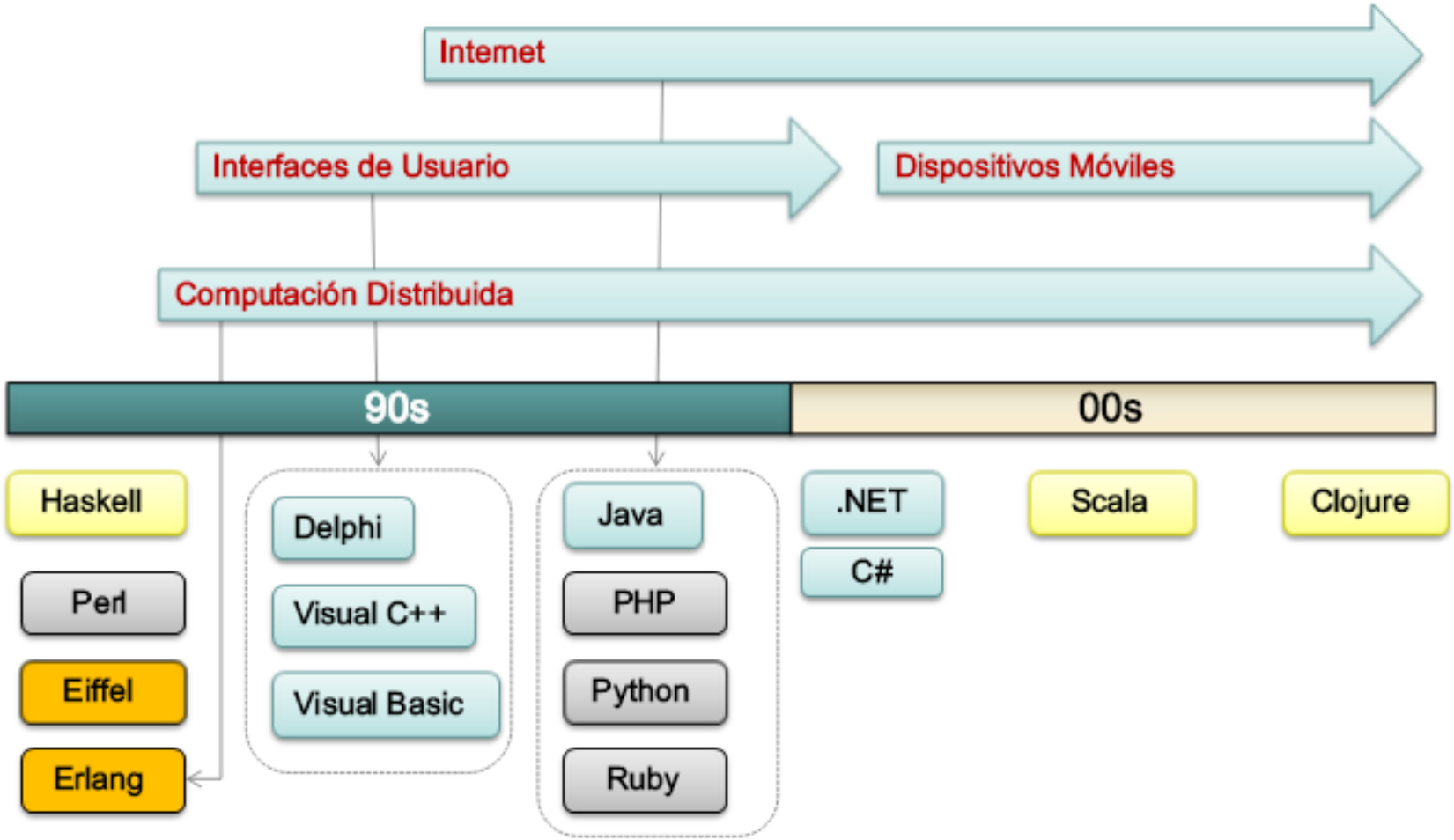
ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Características .



ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Características .



ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Características .

Generación	Lenguajes	Hardware	Movimientos
Primera (1945-55)	Código Máquina	Relés, Válvulas de vacío	
Segunda (1955-68)	FORTRAN COBOL LISP	Transistores, Memorias de ferrita	Prog. Estructurada y Modular Proceso por Lotes
Tercera (1968-1980)	ALGOL PASCAL C BASIC ADA	Circuitos integrados, Memorias de transistores	Ingeniería de Software Orientación a Objetos Bases de Datos
Cuarta (1980-)	C++ JAVA HASKELL PYTHON	VLSI MultiCore Flash	Comp. Distribuida Interfaces Gráficas Multimedia Internet

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Características .

- FORTRAN (Formula Translating) 1957
  - Orientado al cálculo científico
  - Proceso de arrays
  - GOTO asignado
  - Versiones II, III, IV, 66 (subrutinas), 77, 90 (array slicing, recursividad, modularidad, sobrecarga, TADs), 96-03-08 (paralelismo, orientación a objeto)
- COBOL (Common Business Oriented Languaje) 1959:
  - Orientado al mundo empresarial
  - Sintaxis basada en lenguaje natural: 400 palabras reservadas, con verbos, nombres, modificadores, etc.
  - Código auto-modificable: ALTER X TO PROCEED TO Y
  - Especificación detallada de valores numéricos (PIC)
  - Modularidad mediante Copybooks

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Características .

- LISP (List Processing) 1958, McCarthy
  - Orientado a la investigación (Inteligencia Artificial)
  - Basado en *s-expresiones*
  - Código y datos intercambiables
  - Tipado débil
- Familia Lisp:
  - Common Lisp , 1984 (Generalización, Orientación a Objeto)
  - Scheme, 1975 (Simplificación, Closures)
  - Clojure, 2007
- Familia ML:
  - Sistema de tipado Hindler-Millner
  - Haskell, 1990
  - Clean (1987), Scala (2003)

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Ejemplo Programa COBOL.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  PerformFormat4.
AUTHOR.  Michael Coughlan.
* An example program using the PERFORM..VARYING format.
* Pay particular attention to the values produced by the
* WITH TEST BEFORE and WITH TEST AFTER loops.
* Note that the PERFORM within a PERFORM produces the same
* results as the PERFORM..VARYING..AFTER

DATA DIVISION.
WORKING-STORAGE SECTION.
01  LoopCount           PIC 9  VALUE ZEROS.
01  LoopCount2          PIC S9 VALUE ZEROS.

PROCEDURE DIVISION.
Begin.
    DISPLAY "Start WHILE Iteration of LoopBody"
    PERFORM LoopBody WITH TEST BEFORE
        VARYING LoopCount FROM 1 BY 2
        UNTIL LoopCount GREATER THAN 5.
    DISPLAY "Finished WHILE iteration.  LoopCount = " LoopCount.
    ...
```

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Ejemplo Programa LISP .

```
(defun simplify (expression)
    (simplify-2 (rules) expression))

(defun simplify-2 (rules expression)
    (cond
    ((null rules)  expression)
    (T (simplify-2 ( cdr rules) (apply-rule ( car rules) expression)))))

(defun apply-rule (rule expression)
    (substitute ( car rule) (cadr rule) expression))

(defun substitute (pattern replacement expression)
    (cond
    ((null expression)  ())
    ((occurs-at-front pattern expression)
        (substitute-at-front pattern replacement expression))
    (T (cons (car expression)
             (substitute pattern replacement (  cdr expression))))))

(defun occurs-at-front (pattern expression)
    (cond ((null pattern) T) ((null expression) nil)
    ((matches ( car pattern) ( car expression))
        (occurs-at-front ( cdr pattern) (cdr expression)))
    (T  nil)))
```

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Ventajas.

- ALGOL (Algorithmic Languaje) 1958/60/68 N.Wirth
  - Familia de lenguajes, diseñados por un comité de expertos
  - Bloques de código, recursividad, funciones internas, paso de parámetros
  - Arrays dinámicos, paralelismo, definición de operadores
- Familia ALGOL
  - PASCAL, 73 → Modula-2, 80
    - → Delphi/ Free-Pascal, 93
    - → ADA, 83
  - C, 74 → C++, 80
    - → Objective-C, 86
    - → C#, 95
    - → Java, 95

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Ventajas.

- SIMULA, 67
- Ortodoxa:
  - SmallTalk, 80 → Objective-C, 86
  - Eiffel, 86
- Parcial
  - ADA, 83 (Genericidad)
  - C++, 80 (Templates)
  - Java, 95 (Interfaces)
  - C# , 2001 (.NET)
- Basada en prototipos
  - JavaScript, 96
  - Python, 91

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Ventajas.

- Cliente (navegador):
  - HTML / CSS
  - ACMEScript: JavaScript, ActionScript (Flash), 1995
  - Java (applets), 1995
- Servidor
  - PHP, ASP, 1995
  - Java (servlets), 2000
  - Ruby, 1995
- Propósito general
  - Perl, 1987
  - Tcl/Tk, 1989
  - Python, 1991

ANÁLISIS DE LOS PARADIGMAS DE PROGRAMACIÓN.

Ventajas.

- Programación lógica:
  - PROLOG, 1972
- Concurrencia
  - Erlang, 1986
  - Oz , 1991
- Bases de Datos
  - SQL (No es Turing-completo)
- Minimalistas
  - Forth, 1970
  - APL (1964) → J (1990)
  - BrainFuck

Bibliografía

- Pratt T.W. Zelkowitz, M-V. (2000). Programming Languages: Design and Implementation. 4th Edition. Prentice Hall.

- Louden K,. Lambert K.(2011). Programming languages. Principles and Practice. 3nd Edition. Course Technology.

- Tucker, A.; Noonan, R. 2006. Programming languages.. McGraw Hill

- Tucker, A.; Noonan, R. 2001. Programming languages.Principles and Paradigms. McGraw Hill