TALEN

Anexo de Programación

Apunte Teórico

ÍNDICE

1. Programación
   1.1. Resolución de problemas con computadora
   1.2. El paradigma de la programación estructurada
2. Programación estructurada
3. Evolución a la Programación modular
4. Editores de programación
   4.1. Diferencias entre un compilador y un intérprete
   4.2. Tipos de errores
   4.3. La evolución histórica hacia el compilador
5. Lenguajes de bajo y alto nivel

1) Paradigmas de Programación

1 Programación

Es el punto de partida de un modo específico de razonar, desde la cual se inicia, para luego deducir las preguntas y respuestas subsiguientes. Dicho en otras palabras: según sea el paradigma, se partirá de modelos y de preguntas distintas. Por ejemplo, los distintos paradigmas entre la teoría geocéntrica y heliocéntrica conllevan distintas preguntas, lo cual implicó un cambio esencial en la forma de razonar (ver Figura 1).

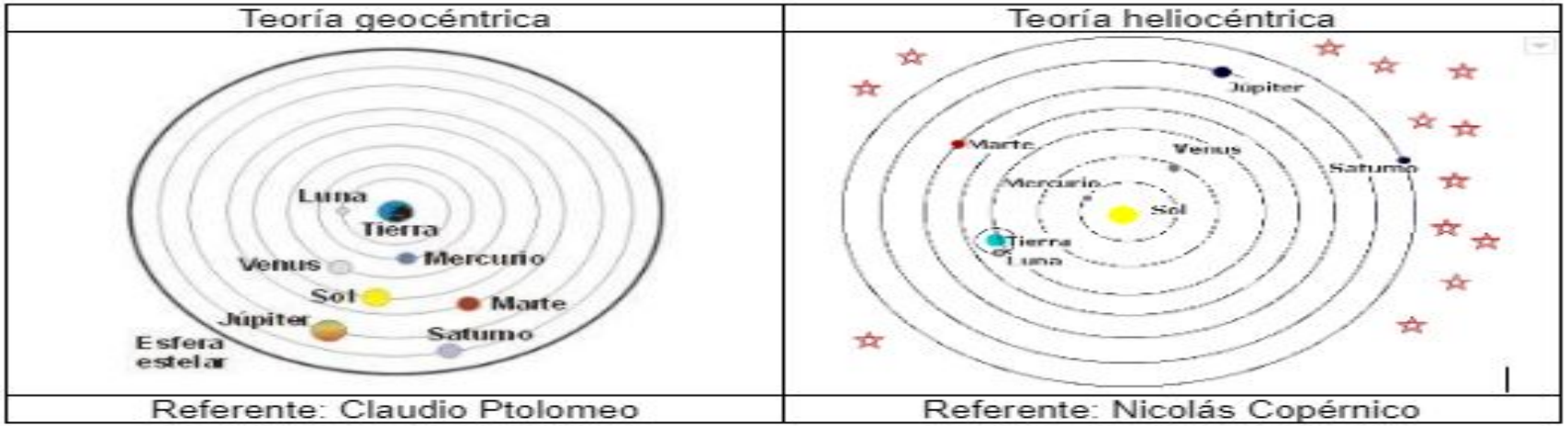


Figura 1: Distintos paradigmas que conllevaron a distintas preguntas

Resolución de problemas con computadora

Observación: si bien la metodología general de resolución de problemas con computadoras, es la misma para todos los paradigmas (en pocas palabras): análisis, diseño e implementación (ver Figura 2). Son distintas las preguntas que cada etapa conlleva, según sea el paradigma.

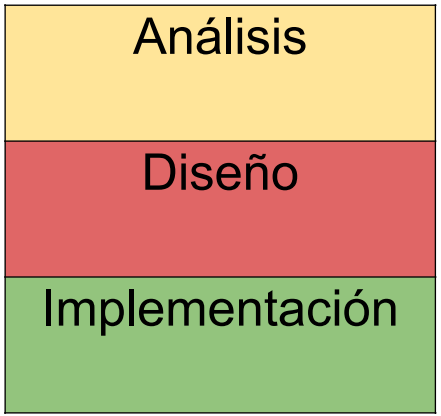


Figura 2: Misma metodología en la resolución de problemas con computadoras

Paradigma de la programación estructurada

Se parte de preguntarnos cuáles son los datos y cuáles las instrucciones (algoritmo). De ello se deduce la conocida ecuación: Datos + Instrucciones = Programa (ver Figura 3).

Datos + Instrucciones = Programa
Figura 3: Ecuación de la programación estructurada

Observación

En el ámbito de la programación orientada a objetos, que se desarrollará , estas preguntas no se aplicarán y dicha ecuación será incorrecta.

Esto conlleva un cambio de paradigma y metodología, como se verá posteriormente.

Programación Estructurada

La programación estructurada es un paradigma de programación, que utiliza la técnica de desarrollo de programas de la forma más clara posible, haciendo uso de tres estructuras de control:

- Secuencial.
- Selectiva.
- Repetitiva.

(ver Figura 4). Estas estructuras pueden combinarse para crear programas que manejen cualquier necesidad de procesamiento de datos.

Programación Estructurada

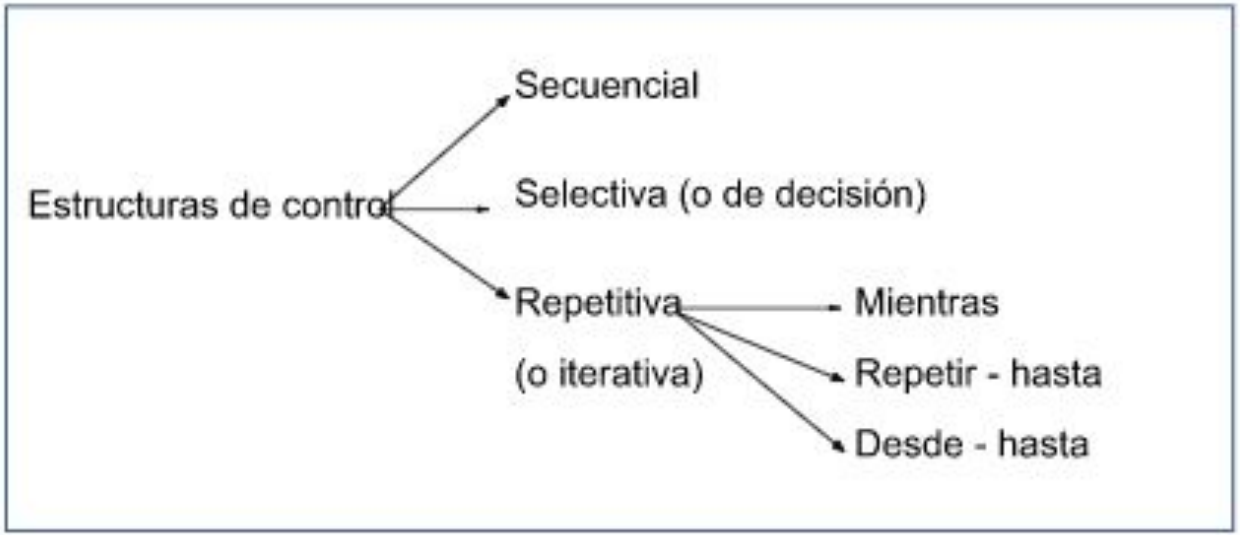


Figura 4: Estructuras de control en la programación estructurada

3) Evolución a la Programación Modular

Posteriormente a la programación estructurada se ha creado un nuevo paradigma conocido como la programación modular. La programación modular consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable.

3) Evolución a la Programación Modular

Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente. Esta técnica se llama, “divide y vencerás” ó análisis descendente Top-Down (ver Figura 5 y Figura 6).

Programación Modular

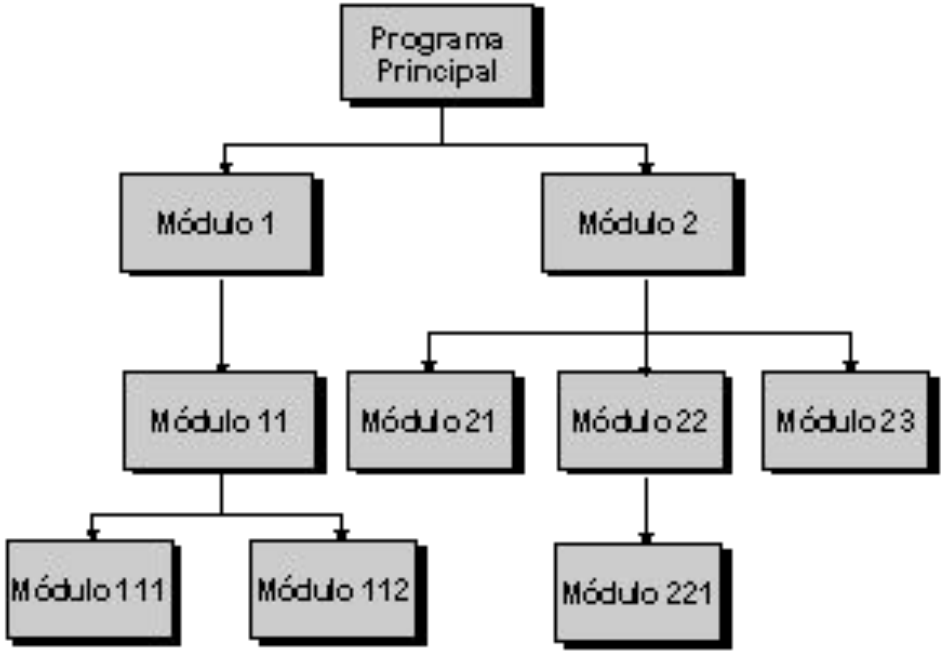


Figura 5: Programación modular

Programación Modular

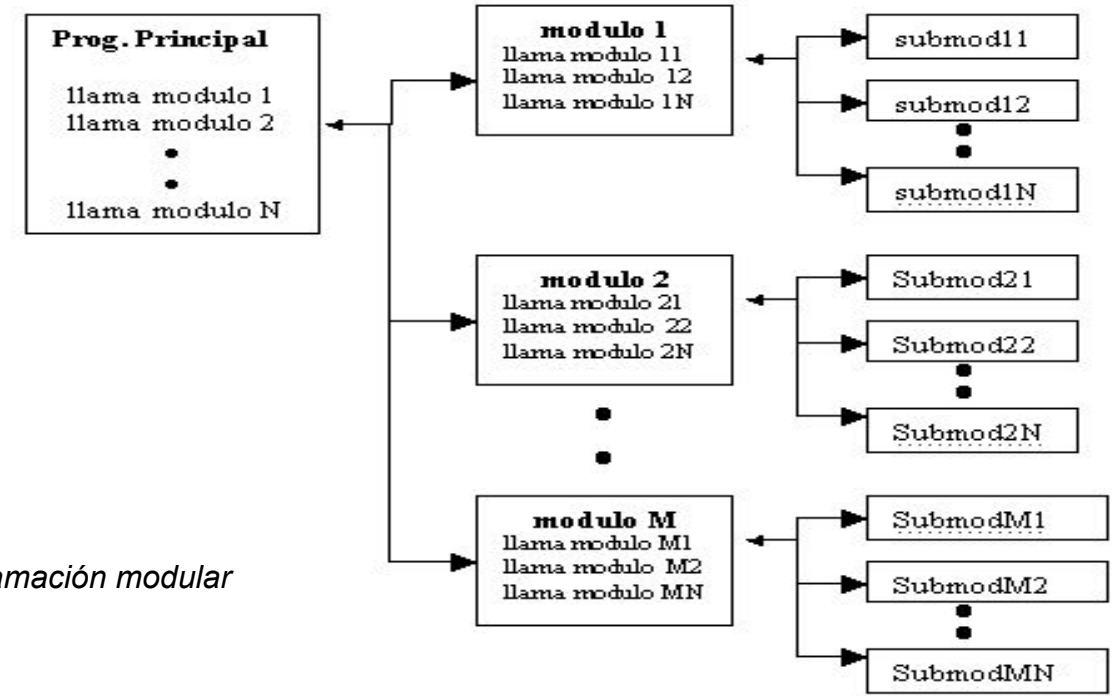


Figura 6: Programación modular

Programación Modular

Asimismo cada módulo o subprograma puede estar implementado con las estructuras de control de la programación estructurada. Si este fuera el caso, en términos de conjuntos: implica que toda implementación modular es estructurada, pero que sea estructurada no implica que se modular (ver Figura 7).

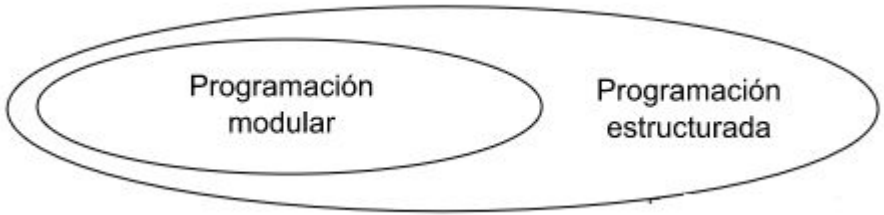


Figura 7: Relación entre la programación estructurada y su posterior evolución modular

4) Editores de Programación (Compiladores e Intérpretes)

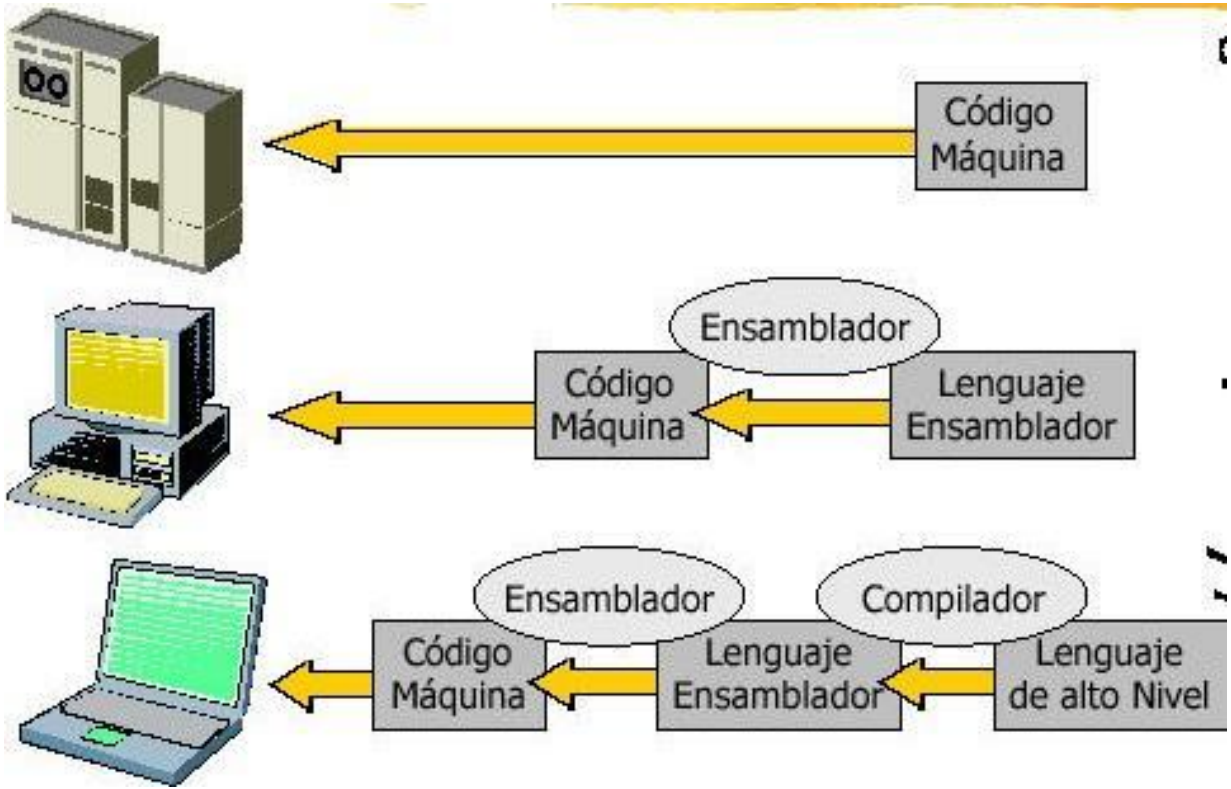
Los programas para implementar (codificar) programas suelen recibir el nombre de Editores de Programación, Entornos de Desarrollo, o equivalentes. Estos pueden ser compiladores o intérpretes.

Diferencia de un compilador y un Intérprete

Diferencias de un Compilador y un Intérprete

Compilador	Intérprete
Tiene como salida un único lenguaje objeto.	Tiene como salida instrucciones traducidas
No requiere del programa fuente porque el programa objeto es ejecutable. **Puede ser secreto**(El programa fuente)	Se requiere del lenguaje fuente para su ejecución.
Los errores sintácticos y semánticos se detectan antes de la ejecución del programa objeto.	Se detectan los errores en la ejecución del programa.

La evolución hacia el Compilador en particular



5) Lenguajes de bajo y alto nivel