1.1 Lenguajes y paradigmas

Introducción
En este trabajo de investigación se exponen los diferentes tipos de paradigmas de programación que existen según su clasificación, como se muestra en la Figura 1.1 extraída del libro Lenguajes de programación de los autores Appleby y Vandekopple la cual sirvió como eje central de investigación.

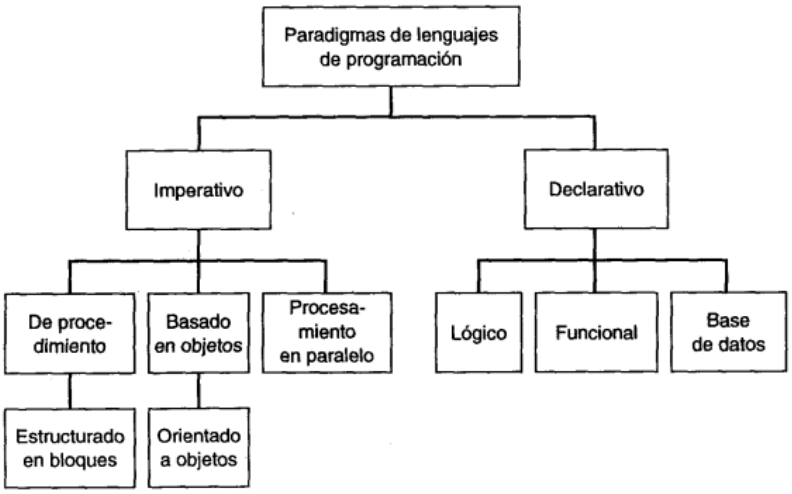

Figura 1.1 Clasificación de los paradigmas de programación. (Appleby, Vandekopple, 1998).

Como se observa en la figura, los paradigmas de los lenguajes de programación se dividen en dos principales partes, por un lado, el paradigma imperativo es aquel en el cual se llega a un resultado especificando el proceso a llevar a cabo, es decir el cómo se llega a ese resultado, mientras que en el paradigma declarativo no importa cómo es que se llega al resultado, sino el resultado en sí mismo, el qué es lo que se busca obtener.
A continuación se da un panorama más amplio de cómo es que funcionan estos paradigmas, sus conceptos, características sus clasificaciones y cuáles son los lenguajes más representativos de cada paradigma.

Paradigma imperativo

Es el paradigma en el cual se realizan cálculos u operaciones con los cuales se obtiene un estado de salida a partir de datos de entrada, generando cambios que producirán el estado final.
En este paradigma el programador especifica las secuencias de instrucciones que serán implementadas línea por línea modificando valores con el objetivo de llegar a un resultado deseado.

Paradigma estructurado en bloques

La programación estructurada surge de la necesidad de crear programas menos extensos, teniendo así fácil acceso y navegación en el código fuente, haciéndolo más compresible.

Los bloques son secciones de código donde se utilizan variables aisladas de otros bloques, es decir, son exclusivas del bloque donde se declaran.

Este paradigma se basa en el teorema estructural que establece que cualquier algoritmo con un único punto de inicio y de culminación se puede constituir como una composición de tres estructuras lógicas o de control:

- Estructura de secuencia son sentencias, operaciones o comandos que se ejecutan uno

tras otro.
- Estructura de decisión o condicional es la ejecución de sentencias que depende del valor de una variable.
- Iteración o estructura de ciclo es la sentencia o sentencias que se ejecutan repetitivamente mientras se cumple una condición.

Los bloques se implementan en un modelo de pila y pueden ser llamados o invocados desde otros bloques; sólo se ejecuta un bloque a la vez y la memoria del último bloque en ser llamado ocupa el tope de la pila. Cuando termina, se libera o extrae para activar la memoria asignada al bloque que hizo la invocación.

Los lenguajes representativos de este paradigma son Ada, ALGOL 60, Pascal, ALGOL 68 y C.

Algunas de las ventajas del paradigma estructurado son:
- Los cambios que se requieran se pueden hacer por bloques.
- Al declarar variables locales, éstas sólo afectan el bloque donde son visibles.
- El nombre de las variables locales se puede repetir en bloques distintos, algo que no implica cambios significativos a la lógica de ejecución de cierto programa, pero sí puede ser más cómodo para los programadores.
- Se puede llamar a distintos bloques las veces que sea necesario permitiendo la reutilización de código, haciéndolo más corto y permitiendo también la recursividad.
- Este paradigma permite una mejor organización y comprensión del código comparado con lenguajes de nivel más bajo.

Paradigma orientado a objetos

Un objeto es una colección de valores de datos y operaciones. Los objetos pertenecen a clases, que son la agrupación de objetos que comparten los mismos métodos.

En este paradigma se presenta una abstracción respecto a los objetos ya que no importa la parte interna del objeto, es decir, el cómo hace para realizar determinada acción, lo único que se muestra al usuario es la parte externa, la acción ya realizada.

Cuando se encubre la información o existe una separación estricta entre la implementación y la interfaz del objeto se trata de una encapsulación, que forma parte de las implementaciones de este paradigma.

Las clases forman parte de jerarquías en las cuales se relacionan por las propiedades heredadas de clases más altas en la jerarquía a una clase descendiente. A este comportamiento entre clases se le llama herencia, que es otra implementación de un lenguaje orientado a objetos.

Los objetos se comunican a través de mensajes (que son diferentes a las llamadas a funciones) en los cuales se solicita la realización de una acción. Ésta será realizada por el objeto receptor implementando sus propios métodos. Cada receptor puede interpretar un mismo mensaje de diferente manera.

Según Alan Kay, diseñador de Smalltalk (lenguaje orientado a objetos), hay seis características básicas de un lenguaje orientado a objetos:
1. Todo es un objeto.
2. Cada objeto es construido a partir de otros objetos.
3. Todo objeto es instancia de una clase.
4. Todos los objetos de la misma clase pueden realizar las mismas acciones. La clase es el lugar donde se define el comportamiento de los objetos y su estructura interna.
5. Las clases se organizan en una estructura arbórea de raíz única, llamada jerarquía de herencia.
6. Un programa es un conjunto de objetos que se comunican mediante el paso de

mensajes.

Los lenguajes representativos de este paradigma son Ada, Object Pascal, C ++, Java, C#, Python, Ruby, entre otros.

Entre las ventajas de este paradigma se encuentran las siguientes:

- Por medio de la herencia, se puede eliminar código redundante y extender el uso de clases existentes.
- El principio de ocultación de los datos ayuda al programador a construir programas seguros que no pueden ser invadidos por código de otras partes del programa.
- Es posible tener múltiples copias de un objeto que coexisten sin ninguna interferencia.
- Las técnicas de pase de mensajes para la comunicación entre objetos simplifican mucho las descripciones de las interfaces con sistemas externos.

Procesamiento en paralelo

La programación concurrente está dividida en dos categorías, sistemas acoplados fuerte y débilmente. En estos sistemas más de una CPU funciona simultáneamente en paralelo, compartiendo o no datos, pero con trabajo y resultaos compartidos.

Para los sistemas acoplados en forma débil las CPU puede trabajar en un mismo programa a la vez que se comunican mediante envío y recepción de mensajes. En estos sistemas no se necesita compartir memoria simultánea.

Para los sistemas acoplados fuertemente distintos procesos en ejecución pueden tener acceso a la misma ubicación de memoria bajo ciertas condiciones. Esto permite una mayor velocidad ya que no se necesitan mensajes.

Las características que distinguen este paradigma de la programación secuencial son:

- El uso de procesadores múltiples.
- Cooperación entre los procesadores.
- Uno o más procesos pueden fallar sin hacer peligrar todo el proyecto.

Sin importar el tipo de sistema, cuando varios procesadores están trabajando a la vez se puede aumentar la probabilidad de que ocurra una falla en uno de ellos. Los lenguajes en este paradigma incluyen mecanismos para continuar con los procesadores que aún trabajan o los que se recuperaron de la falla.

Las ventajas de este tipo de programación son las siguientes:

- Pueden acelerar programas al ejecutar diferentes procesos en paralelo.
- Son capaces de mejorar la confiabilidad si dos o más procesadores duplican el trabajo de cada uno de ellos.
- Proporcionan una avenida natural para el crecimiento del sistema cuando se agregan procesadores adicionales.
- Facilitan tareas naturalmente distribuidas.

Los lenguajes representativos de este paradigma son Ada, Pascal S, Occam y C-Linda.

Paradigma declarativo

En un lenguaje declarativo no se hacen asignaciones a variables del programa. El compilador es el que administra la memoria por nosotros. Estos lenguajes son de nivel más elevado que los lenguajes imperativos, en los que el programador opera más alejado del CPU.

Los tres paradigmas declarativos provienen de las matemáticas: la lógica, la teoría de funciones y

el cálculo relacional.

En este paradigma no es necesario definir algoritmos puesto que se detalla la solución del problema en lugar de cómo llegar a esa solución. En la programación declarativa no se especifica exactamente cómo es que se llega a esta solución.

Paradigma lógico
Un programa basado en la lógica se compone de una serie de axiomas o hechos, reglas de inferencia y un teorema o cuestión por demostrarse. La salida es verdadera si los hechos soportan o apoyan la cuestión, y es falsa en el caso contrario.

En la programación lógica se trabaja de forma descriptiva, estableciendo relaciones entre entidades, lo que obliga a que los lenguajes tengan un alto nivel de abstracción.

Este paradigma se basa en que un algoritmo se crea especificando conocimiento mediante axiomas (lógica) y el problema se resuelve mediante un mecanismo de inferencia que actúa sobre él mismo (control).

Las características de este paradigma son las siguientes:

- Verdad y deducción lógica
- Los programas lógicos se pueden descifrar a través de dos conceptos importantes: verdad y deducción lógica. La verdad sale si el cálculo en el programa es verdadero o no, bajo la implementación de los símbolos del programa. La deducción lógica determina si una cláusula lógica es una consecuencia del programa.
- Lógica de primer orden: considera si los objetos son verdaderos o falsos desde una perspectiva parcial del mundo, llamada dominio. La programación lógica se basa en una generalización de la lógica de primer orden.
- Empleada en Inteligencia artificial

La programación lógica es una metodología que es empleada para buscar que las computadoras deduzcan, porque resulta útil para representar el conocimiento. Se utiliza la lógica para representar el conocimiento y la inferencia para manejarlo.
Los lenguajes representativos de este paradigma son PROLOG, Lisp y Erlang.

Paradigma funcional

El paradigma funcional se caracteriza por utilizar funciones, de donde viene su nombre, pero lo que lo hace diferente a otros paradigmas es que no tiene efectos secundarios, es decir las variables no mutan en el llamado a funciones, asimismo en el paradigma funcional todo ocurre de forma interna, no hay interacción con el exterior, es decir, no se puede presentar nada a pantalla ni solicitar nada desde componentes externos como el teclado o una base de datos.

Dentro del concepto de efectos secundarios también se encuentra el hecho de que las funciones no tienen excepciones en el caso de que ocurra un error, siempre debe regresar un valor el cual debe ser el mismo para los mismos argumentos de entrada.

Entre los lenguajes representativos de este paradigma se encuentran Lisp, Haskell, Erlang y Scala.
Paradigma de base de datos

Las propiedades que distinguen a los lenguajes diseñados para tratar con bases de datos son la persistencia y la administración de cambios. Las entidades de base de datos no desaparecen después de que finaliza un programa, sino que permanecen activas durante tiempo indefinido como fueron estructuradas originalmente.

Puesto que la base de datos, una vez organizada, es permanente, estos lenguajes también deben soportar los cambios. Los datos pueden cambiar y así también pueden hacerlo las relaciones entre

objetos o entidades de datos.

Un sistema de administración de base de datos incluye un lenguaje de definición de datos para describir una nueva colección de hechos, o datos, y un lenguaje de manipulación de datos para la interacción con las bases de datos existentes.

Los lenguajes de base de datos pueden estar integrados en otros lenguajes de programación para mayor flexibilidad.

El lenguaje representativo de este paradigma es SQL.

Conclusiones

Dentro del paradigma de lenguajes imperativos se puede ver una mayor relación o características similares entre ciertos lenguajes de programación ya que todos se deben programar siguiendo una serie de instrucciones redactadas por el programador que buscan darle solución a un problema. Por otro lado, en los paradigmas declarativos existe mayores diferencias entre los lenguajes como se puede observar en esta investigación poniendo como ejemplo el paradigma lógico que implementa conceptos y características diferentes a los demás lenguajes.

La existencia de diversidad en cuanto a paradigmas ayuda a poder resolver problemas o llegar a un fin especifico con alternativas que ayudan al programador a elegir la mejor forma de obtener los resultados deseados.

Como conclusión personal, no me gusta realizar trabajos de investigación malos, pero por el tiempo que me queda para entregarlo no tuve otra alternativa que dejar a un lado la exposición en mis propias palabras de lo que leía e investigaba y escribir cada vez menos (algo que es poco deseado y favorable) de esta forma en las partes finales del trabajo donde se puede observar un trabajo menos presentable.

Arce Acevedo Javier 3CV4 ESCOM