TIC03

PARADIGMAS DE PROGRAMACIÓN: PROGRAMACIÓN IMPERATIVA Y PROGRAMACIÓN DECLARATIVA. PROGRAMACIÓN ESTRUCTURADA Y PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

1 **Introducción.**

2 **Paradigmas de programación. Programación imperativa y programación declarativa. Programación estructurada y programación orientada a objetos (POO).**

- **2.1 Paradigmas de programación. Programación imperativa y programación declarativa.**
- **2.2 Programación estructurada y programación orientada a objetos (POO).**

3 **La programación estructurada.**

- **3.1 Programación estructurada. Estructuras de control.**
- **3.2 El lenguaje C.**

4 **La Programación orientada a objetos (POO).**

- **4.1 Fundamentos de la programación orientada a objetos (Ampliación).**
- **4.2 Java.**

5 **Ejemplos**

1. INTRODUCCIÓN

La programación (programar) consiste en elaborar programas para su empleo en computadoras. Un programa es un conjunto de instrucciones que controlan una computadora.

Un programa esta formado por un conjunto de instrucciones o declaraciones (también conocidas como código) que debe ejecutar la CPU de una computadora. El orden en el cual se ejecutan las declaraciones de un programa se conoce como control de flujo del programa.

Los programadores utilizan los lenguajes de programación para crear el código fuente de los programas. Posteriormente, convierten el código fuente en código máquina (o código objeto) para la computadora.

A la hora de crear el código fuente de un programa, los programadores pueden utilizar diferentes métodos o paradigmas (ejemplo o modelo de algo) de programación.

2. PARADIGMAS DE PROGRAMACIÓN: PROGRAMACIÓN IMPERATIVA Y PROGRAMACIÓN DECLARATIVA. PROGRAMACIÓN ESTRUCTURADA Y PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

Según la RAE, un paradigma es una teoría o conjunto de teorías cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento.

Un paradigma (ejemplo o modelo de algo) de programación es una propuesta tecnológica adoptada por una comunidad de programadores y desarrolladores cuyo núcleo central es incuestionable en cuanto que únicamente trata de resolver uno o varios problemas claramente delimitados. (Para ampliar la información acceder a wikipedia: https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n).

Un paradigma de programación representa un enfoque particular o filosofía para diseñar soluciones. Los paradigmas difieren unos de otros en los conceptos y la forma de abstraer los elementos involucrados en un problema, así como en los pasos que integran la solución del problema, en otras palabras, el cómputo. 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. (Para ampliar la información acceder a wikipedia: https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n).

Existen muchos paradigmas de programación diferentes, cada uno de ellos tiene sus propias características y tratan de solucionar los problemas clásicos del desarrollo de software desde diferentes perspectivas y filosofías. Probablemente, el paradigma de programación más utilizado hoy en día sea el de la programación orientada a objetos (POO).

A continuación se van a analizar algunos de los paradigmas de programación más utilizados.

2.1. Paradigmas de programación. Programación imperativa y programación declarativa.

En general, la mayoría de los paradigmas son variaciones de uno de los dos tipos principales: programación imperativa y programación declarativa. La diferencia principal principal radica en que en el enfoque imperativo se indica el cómo se debe calcular y en el enfoque declarativo se indica el qué se debe calcular.

- **Programación imperativa o por procedimientos**: Es el paradigma más usado en general, se basa en dar instrucciones al ordenador de como hacer las cosas en forma de algoritmos. La programación imperativa es la más antigua y el ejemplo principal es el lenguaje máquina. Otros ejemplos de lenguajes puros de este paradigma serían C, BASIC o Pascal. Dentro del paradigma imperativo, existen otros paradigmas que se centran en la estructura y organización de los programas, y que son compatibles con los fundamentales. Algunos ejemplos son: la programación estructurada (C, Pascal), la programación modular, la programación orientada a objetos (Smaltalk, C++, Java), la programación orientada a eventos, etc.

- **Programación declarativa**: Está basado en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones. Hay lenguajes para la programación funcional (Lisp, Scheme (una variante de Lisp) o Haskell), la programación lógica (Prolog) o la programación con restricciones (Casi todos los lenguajes son variantes del Prolog). El enfoque declarativo tiene varias ramas diferenciadas: el paradigma funcional, el paradigma lógico, la programación reactiva y los lenguajes descriptivos.

Los paradigmas fundamentales están asociados a determinados modelos de cómputo y a un determinado estilo de programación. Si bien se puede seleccionar la forma pura de estos paradigmas a la hora de programar, en la práctica es habitual que se mezclen, dando lugar a la programación multiparadigma o lenguajes de programación multiparadigma. El lenguaje Lisp, por ejemplo, se considera multiparadigma.

Actualmente, el paradigma de programación más utilizado es el paradigma de la programación orientada a objetos (POO). El núcleo central de este paradigma es la unión de los datos y el procesamiento en una entidad llamada "objeto", que se puede relacionar a su vez con otras entidades "objeto".

2.2. Programación estructurada y programación orientada a objetos (POO)

En general, como se ha dicho anteriormente, la mayoría de los paradigmas son variaciones de uno de los dos tipos principales: programación imperativa (indicar el cómo se debe calcular) y programación declarativa (indicar el qué se debe calcular). No obstante, existen otros paradigmas que se centran en la estructura y organización de los programas, y son compatibles con los fundamentales. Entre ellos se pueden citar: la programación estructurada y la programación orientada a objetos (POO).

- **Programación estructurada**: Los programas se crean en base a módulos pequeños que son fáciles de leer y entender. Cada módulo cuenta con una sola entrada y salida y

realiza una sola tarea. Uno de los objetivos de la programación estructurada es crear código fácil de leer. Algunos ejemplos de lenguajes de programación que utilizan este paradigma son C y Pascal.

- **Programación orientada a objetos (POO)**: En la POO, la arquitectura software de un sistema se articula en base a una serie de abstracciones de datos (objetos) que colaboran entre si por medio del envío de mensajes. Los bloques de construcción de la POO, llamados objetos, son componentes reciclables y modulares. Algunos ejemplos de lenguajes de programación que utilizan este paradigma son C++ y Java.

3. LA PROGRAMACIÓN ESTRUCTURADA

La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de ordenador, utilizando únicamente subrutinas (funciones y procedimientos) y tres estructuras de control: secuencia, selección (if y switch) e iteración (bucles for y while), considerando innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), que podía conducir a un código que es mucho más difícil de seguir y de mantener (código espagueti), y que era la causa de muchos errores de programación. (Para ampliar la información puede acceder a wikipedia: https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada).

En la programación estructurada, el programador se concentra en el algoritmo requerido para llevar a cabo el cómputo deseado. Los lenguajes apoyan este paradigma proporcionando recursos para pasar argumentos a las funciones y devolviendo valores de las funciones.

En el paradigma de la programación estructurada, todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes: secuencia, instrucción condicional e iteración (bucle de instrucciones) con condición al principio. Solamente con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien

los lenguajes de programación tienen un mayor repertorio de estructuras de control, estas pueden ser construidas mediante las tres básicas citadas.

3.1. Programación estructurada. Estructuras de control

En el paradigma de la programación estructurada, los programas pueden ser escritos utilizando tres estructuras de control:

- **La estructura secuencial**: las instrucciones se ejecutan una después de la otra, en el orden en que fueron escritas.

- **Las estructuras de selección**: Las estructuras de selección permiten ejecutar un bloque de instrucciones u otro, o saltar a un subprograma o subrutina, según se cumpla o no una determinada condición. El resultado de evaluar la condición puede ser verdadero o falso. Cuando la evaluación de la condición es verdadera se ejecutan unas líneas de código y cuando la evaluación es falsa no se ejecutan esas líneas de código (sentencia if-then) o se ejecutan otras líneas de código (sentencia if-then-else).

- **Las estructuras de repetición o estructuras de ciclo (bucles)**: Las estructuras iterativas o de repetición permiten la ejecución de un bloque de instrucciones mientras se cumple una determinada condición. En una estructura de repetición, el programa evalúa una condición y ejecuta una serie de instrucciones basándose en esa condición. Mientras la condición es evaluada a verdadero, se repite la ejecución de una o más instrucciones. Cuando la condición es evaluada a falso se sale del bucle. Existen variaciones de la estructura básica que hacen que se repita el bucle de instrucciones mientras se cumple una condición (sentencia while), hasta que se cumpla una condición (sentencia do-while) o que permiten ejecutar un conjunto de sentencias un cierto número de veces (sentencia for).

  Algunos ejemplos de lenguajes de este paradigma serían C, BASIC o Pascal.

3.2. El lenguaje C

El lenguaje C es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos. (Puedes ampliar la información en el siguiente enlace: https://es.wikipedia.org/wiki/C_(lenguaje de programaci%C3%B3n).

4. LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

Frente al paradigma de la programación estructurada, que utiliza básicamente funciones y las estructuras secuencial, de selección (instrucciones if y switch) e iteración (bucles for y while), la programación orientada a objetos (POO) es un paradigma de programación que utiliza objetos y sus interacciones para diseñar y desarrollar aplicaciones.

En la POO, la arquitectura software de un sistema se articula en base a una serie de abstracciones de datos (objetos) que colaboran entre si por medio del envío de mensajes.

El paradigma de la programación orientada a objetos se articula a través de cuatro mecanismos fundamentales

- **Clases**: descritas internamente en términos de una colección de atributos y una serie de operaciones.
- **Métodos**: que definen las capacidades de una clase y permiten acceder o modificar los

atributos de la misma.

- **Objetos**: ejemplar de una clase que reside en memoria durante cierto tiempo en la fase de ejecución y que está definido por los valores de sus atributos en un determinado momento.
- **Mensajes**: acto de invocación de un método sobre un objeto en tiempo de ejecución.

Objetos: un objeto se define por sus atributos (color, tamaño, forma, etc.). Un atributo define las características de un objeto. Las funciones definen lo que un objeto puede hacer. Los atributos y las funciones definen al objeto. En la POO, todos los objetos tienen atributos y funciones que pueden encapsular (contener) otros objetos.

Debido a que la POO ofrece una manera más intuitiva de de modelar el mundo, los programas desarrollados con POO se vuelven más sencillos, la programación se hace más rápida y la carga de mantenimiento de un programa se reduce.

4.1. Fundamentos de la programación orientada a objetos (Ampliación)

La POO presenta una serie de características fundamentales: abstracción, encapsulación, herencia, polimorfismo, ligadura dinámica y genericidad.

- **Abstracción** (Abstracción de datos): La abstracción es el proceso mediante el cual describimos una realidad del dominio del problema para crear una representación simplificada dentro del dominio de solución.
- **Encapsulación**: La encapsulación es el proceso por el cual la colección de métodos de una clase protege el acceso ilegal al estado de un objeto por parte de otros objetos. De esta forma, los atributos sólo son modificables lo a a través de los métodos de la clase. Al utilizar la encapsulación, es posible definir diferentes modificadores de acceso a los atributos y métodos de una clase. Estos modificadores (prívate, protected y public) determinan la visibilidad de los elementos desde el exterior de la clase. Un elemento definido como privado sólo es accesible desde dentro de la clase donde se ha

declarado el elemento. Un elemento protegido es accesible desde la propia clase, desde las clases derivadas y desde las clases pertenecientes al paquete donde ésta está definida. Un elemento definido como público es accesible por todos los objetos del sistema.

- **Herencia**: La herencia es un mecanismo que permite organizar un conjunto de clases de acuerdo a una jerarquía. Mediante la herencia, se pueden compartir atributos y métodos entre clases y subclases. Una clase derivada tendrá acceso a todos los atributos y métodos públicos o protegidos de las clases de las que hereda. Por otra parte, las clases hijas pueden sobreescribir los métodos heredados para especializar su funcionalidad.
- **Polimorfismo**: La herencia hace que las instancias (objetos) de una clase puedan ser de múltiples tipos. Los objetos adquieren el tipo de la clase a la que pertenecen y el tipo definido por las clases de las que heredan su comportamiento. A esta propiedad se la llama polimorfismo. El polimorfismo también se puede conseguir mediante el uso de interfaces. Es posible convertir un objeto de un tipo a otro mediante casting.
- **Ligadura dinámica**: La ligadura dinámica es la capacidad de algunos lenguajes orientados a objetos de cambiar en tiempo de ejecución el objeto referenciado por una variable por otro objeto distinto, siempre que sea de un tipo polimórficamente compatible.
- **Genericidad o parametrización de tipos**: La genericidad permite abstraerse de los los tipos de objetos con los que trabaja internamente una clase por medio de la parametrización de los mismos.

Algunos ejemplos de lenguajes de este paradigma serían Java y C++.

Nota: La POO está basada en la programación imperativa, pero encapsula elementos denominados objetos que incluyen tanto variables como funciones. Los bloques de construcción de la POO, llamados objetos, son componentes reciclables y modulares. La POO se basa en la programación estructurada y la mejora. No se excluye la programación estructurada cuando se trabaja con objetos. Los objetos están compuestos de fragmentos de

programas estructurados y la lógica para manipular los objetos también es estructurada.

4.2. Java

En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos, siendo Java uno de los más populares.

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web. (Acceder a wikipedia: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))

Java es un lenguaje de programación moderno que presenta las siguientes características: gestión de la memoria dinámica automática (no se utilizan punteros), interpretado (un programa escrito en código java se compila y se obtiene un código objeto que es interpretado por la máquina virtual de Java), multiplataforma (el código se escribe y compila una única vez y se ejecuta igual en la máquina virtual Java de cualquier plataforma) y seguro (la máquina virtual Java sobre la que se ejecuta el programa controla que dicho programa no intente ejecutar operaciones no permitidas sobre los recursos del sistema).

Enlaces:

http://www.genbetadev.com/paradigmas-de-programacion/diferencias-entre-paradigmas-de-programacion

http://www.dccia.ua.es/dccia/inf/asignaturas/LPP/2010-2011/clases-cristina/s2.pdf

https://es.wikipedia.org/wiki/Lisp