Paradigmas de Programacion

Programming Paradigms

Autor 1:Jhon Edinson David Zuñiga Mosquera Autor 2. Brayan Chiquito
Ingenieria en sistemas y computacion, Universidad Tecnologica de Pereira, Pereira, Colombia
Correo-e: jhon.zuniga@utp.edu.co

***Resumen*— Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis.**

https://www.4rsoluciones.com/blog/que-son-los-paradigmas-de-programacion-2/

***Palabras clave*— Términos- Paradigma, programacion, sintaxis.**

***Abstract*—**
A programming paradigm is a style of program development. That is, a model to solve computational problems. The programming languages, necessarily, are framed in one or several paradigms at the same time based on the type of orders they allow to implement, something that has a direct relationship with their syntax.

https://www.4rsoluciones.com/blog/que-son-los-paradigmas-de-programacion-2/

***Key Word* —About- Paradigm, Progamming, syntax.**

I. INTRODUCCIÓN

Los paradigmas de programación aborda distintos modelos de programación y son pilares que definen el desarrollo informatico de los programas y por extensión el desarrollo computacional. La aparición de cada uno de ellos ha marcado un hito en el modelo de desarrollo informatico. En este trabajo se verá reflejado la investigación de cada uno de los paradigmas de la programacion

II. INDICE

RECOMENDACIONES

Escoger la técnica de programación más adecuada para cada problema, e implementar su resolución.
Conocer diversos paradigmas de programación y tener criterio para seleccionar uno para cada problema.
Saber desarrollar aplicaciones informáticas con los diferentes paradigmas de programación.
Diseñar y estructurar los programas informáticos de forma modular y robusta usando la programación orientada a objetos..

REFERENCIAS

— http://www.slideshare.net/ARMANDO1022/m-o-d-u-l-a-r-i-d-a-d

— http://tareas.org/apuntes/EI/1/EDI/teoria/07-08/tad_-_ventajas.pdf

— http://paradimas/programación.org/apuntes/ventajas_desventajas

— https://sandramarramirez.fandom.com/es/wiki/Paradigmas_de_Programaci%C3%B3n

— https://www.4rsoluciones.com/blog/que-son-los-paradigmas-de-programacion-2/

III. CONTENIDO

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con un conjunto de normas específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc.

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.

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.

Hay multitud de ellos atendiendo a alguna particularidad metodológica o funcional Cuando un lenguaje refleja bien un paradigma particular, se dice que soporta el paradigma, y en la práctica un lenguaje que soporta correctamente un paradigma, es difícil distinguirlo del propio paradigma, por lo que se identifica con él.

Tipos de paradigmas de programación:

Podemos clasificar los paradigmas de programación en:

Paradigma imperativo, heuristico, concurrente, funcional, lógico, paradigma basado en objetos.

Paradigma imperativo: Son aquellos que facilitan los cálculos por medio de cambios de estado, entendiendo como estado la condición de una memoria de almacenamiento. Los lenguajes estructurados en bloques, se refieren a los ámbitos anidados, es decir los bloques pueden estar anidados dentro de otros bloques y contener sus propias variables.

Los programas escritos en un lenguaje funcional están constituidos únicamente por definiciones de funciones, entendiendo éstas no como subprogramas clásicos de un lenguaje imperativo, sino como funciones puramente matemáticas, en las que se verifican ciertas propiedades como la "transparencia referencial" (el significado de una expresión depende únicamente del significado de sus subexpresiones), y por tanto, la carencia total de "efectos laterales".

El objetivo es conseguir lenguajes expresivos y "matemáticamente elegantes", en los que no sea necesario bajar al nivel de la máquina para describir el proceso llevado a cabo por el programa, y evitando el concepto de "estado" del cómputo.

La secuencia de computaciones llevadas a cabo por el programa se regiría única y exclusivamente por la "reescritura" de definiciones más amplias a otras cada vez más concretas y definidas, usando lo que se denominan "definiciones dirigidas".

A este tipo de paradigma de programación se le suele llamar algorítmico

Otras características propias de estos lenguajes son la no existencia de asignaciones de variables y la falta de construcciones estructuradas como la secuencia o la iteración (lo que obliga en la práctica a que todas las repeticiones de instrucciones se lleven a cabo por medio de funciones recursivas).

Algunos de los lenguajes imperativos son:

- C
- C++
- C#
- Basic
- Java
- Perl

Paradigma heurístico: Define un modelo de resolución de problemas en el que se incorpora algún componente heurístico, sobre la base de una representación más apropiada de la estructura del problema, para su resolución con técnicas heurísticas.

Se puede definir como "aquel tipo de programación computacional que aplica para la resolución de problemas reglas de buena lógica (reglas del pulgar).

Denominadas heurísticas, las cuales proporcionan entre varios cursos de acción uno que presenta visos de ser el más prometedor, pero no garantiza necesariamente el curso de acción más efectivo."

La Programación Heurística implica una forma de modelizar el problema en lo que respecta a la representación de su estructura, estrategias de búsqueda y métodos de resolución, que configuran el Paradigma Heurístico.

Este tipo de programación se aplica con mayor intensidad en el campo de la

Inteligencia Artificial (IA), y en especial, en el de la Ingeniería del

Conocimiento.

La Programación Heurística se presenta y utiliza desde diferentes puntos de vista:

- Como técnica de búsqueda para la obtención de metas en problemas no algorítmicos, o con algoritmos que generan explosión combinatoria.

- Como un método aproximado de resolución de problemas utilizando funciones de evaluación de tipo heurístico.

- Como método de poda para estrategias de programas que juegan, aunque estos métodos no son realmente heurísticos.

Paradigma funcional: Sus orígenes provienen del Cálculo Lambda (o ?-cálculo), una teoría matemática elaborada por Alonzo Church como apoyo a sus estudios sobre computabilidad. Un lenguaje funcional es, a grandes rasgos, un azúcar sintáctico del Cálculo Lambda.

El paradigma funcional está basado en el modelo matemático de composición funcional. En este modelo, el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que una composición produce el valor deseado.

No existe el concepto de celda de memoria que es asignada o modificada. Más bien, existen valores intermedios que son el resultado de cálculos anteriores y las entradas a cálculos subsiguientes. Tampoco existen sentencias imperativas y todas las funciones tienen transparencia referencial.

La programación funcional incorpora el concepto de función como objeto de primera clase, lo que significa que las funciones se pueden tratar como datos (pueden pasar como parámetros, calculadas y devueltas como valores normales, y mezcladas en el cálculo con otras formas de datos).

En este paradigma el informático concibe la solución como una composición de

Funciones La forma en que se especifican las funciones puede variar. Se pueden especificar procedimentalmente o matemáticamente mediante su definición, sin secuencia de control.

Un lenguaje funcional es el Lisp.

Ventajas

- Ausencia de efectos colaterales
- Proceso de depuración menos problemático
- Pruebas de unidades más confiables
- Mayor facilidad para la ejecución concurrente

Desventajas

- Falta de estandarización
- Bajo rendimiento de los programas

Paradigma lógico: La Programación Lógica es un Paradigma de Programación basado en la Lógica.

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

Prolog, proveniente del inglés Programming in Logic, es un lenguaje lógico bastante popular en el medio de investigación en Inteligencia Artificial. Prolog es un lenguaje muy diferente, tanto de los imperativos como Fortran, Pascal, C etc., como de los funcionales como Lisp. En todos los mencionados, las instrucciones se ejecutan normalmente en orden secuencial, es decir, una a continuación de otra, en el mismo orden en que están escritas, que sólo varía cuando se alcanza una instrucción de control (un bucle, una instrucción condicional o una transferencia).

En Prolog, las cosas son distintas: 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.

Paradigma basado en objetos: La programación orientada a objetos (OOP, por las siglas inglesas de Object-Oriented Programming) es una forma de programar que proliferó a partir de los años ochenta.

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.

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).

Dada esta propiedad de conjunto de una clase de objetos, que al contar con una serie de atributos definitorios, requiere de unos métodos para poder tratarlos (lo que hace que ambos conceptos están íntimamente entrelazados), el programador debe pensar indistintamente en ambos términos, ya que no debe nunca separar o dar mayor importancia a los atributos en favor de los métodos, ni viceversa.

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

- La programación orientada a objetos es más moderna, es una evolución de la programación imperativa 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.

Lenguajes orientados a objetos

Ventajas

- Permite crear sistemas más complejos
- Agiliza el desarrollo de software
- Proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.
- Fomenta la reutilización y extensión del código.

Desventajas

- Complejidad para adaptarse
- Mayor cantidad de código

Entre los lenguajes orientados a objetos destacan los siguientes:

Ada C++ C# VB.NET Clarion Delphi Eiffel Java Léxico (en castellano)