Paradigmas de programación

Programming paradigms

Autor 1: Julián Esteban Collazos Toro
Autor 2: Miguel Ángel López Fernández.
Ingeniería en sistemas y computación, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: j.collazos@utp.edu.co, miguel.lopez@utp.edu.co

***Resumen*-En el siguiente documento se hablará sobre los paradimas de programación, que son, que tipos existen, su ejemplificación, y por ende su asimilación en el mundo. Cabe resaltar que de muchas formas posibles que se puede dar a entender algo, y más a una computadora, sin embargo y como se mencionara a continuación, se dividirán en ciertas ramas, consideradas como maneras o métodos teóricos para el desarrollo de un programa o una actividad de computo.**

***Palabras clave*— Programación, cómputo, mundo, problema, formas.**

***Abstract*-In the following document we will talk about the programming paradigms, which are, what types exist, their exemplification, and therefore their assimilation in the world. It should be noted that in many possible ways that something can be implied, and more than one However, and as mentioned below, they will be divided into certain branches, considered as theoretical methods or methods for the development of a program or a computer activity.**

***Key Word* — Programming, computing, world, problem, forms.**

INTRODUCCIÓN

En nuestra cotidianidad se encuentran las computadoras, como pueden ser el celular o la calculadora, y estas vienen ya programadas para cumplir ciertas funciones, sin embargo, programar no es una tarea tan sencilla como lo hacen ver estas computadoras, aparte hay más de una manera de programarlas, dependiendo de la función que quiera que se haga y lo que se quiere lograr con tal programación, siendo así que hay lenguajes para enseñar a programar, lenguajes para desarrollar hasta cierto punto contenido digital, y cada vez se va "actualizando", por decirlo de alguna manera, estos lenguajes, que tienen su base en la programación lógica y la imperativa.

I. CONTENIDO

1. DEFINICION

Se define en ámbito general como una teoría que se acepta sin cuestionamiento, la cual es modelo para la resolución de problemas y organización de procesos.
Para la programación, según *Jesús Javier Rodríguez Sala* "se implementa como la manera, o patrones conceptuales de resolver problemas, efectuar soluciones o estructurar programas".
Con lo anterior, resumimos el concepto de paradigmas de programación como el conjunto de reglas de programación utilizados para llevar a cabo cómputos, la definición de su estructura y funcionamiento.

2. TIPOS DE PARADIGMAS

Los paradigmas difieren uno del otro por sus formas de solucionar un problema, estructurar la solución, o en otras palabras llevar a cabo la tarea en la computadora. Cada programa basa su funcionalidad en un paradigma especifico, y a veces, pueden dar a luz nuevos estilos, por lo que se puede considerar a la vez un estilo de programación.
Los paradigmas en programación se guían por dos enfoques que son el enfoque **imperativo** y el enfoque **declarativo**.

2.1 ENFOQUE IMPERATIVO

El enfoque imperativo describe como debe realizarse el cálculo o la tarea, mas no el porqué de su ejecución. Dicta una sentencia finita de instrucciones que se van ejecutando una tras otra y sus datos se almacenan en memoria y se reúsan en forma de variables. Es la forma de programación más usada y la más antigua, el ejemplo principal es el lenguaje de máquina.

```
leer(x)
leer(y)
resultado = x + y
escribir(resultado)
```

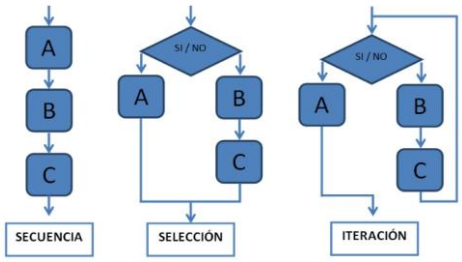
Otro ejemplo del enfoque imperativo, usado para calcular el N-ésimo término de la serie de Fibonacci:

```
function fibonacci(n) {
  var actual, ant1, ant2;
  if (n === 0) {
    actual = 0;
  } else if (n === 1) {
    actual = 1;
  } else {
    ant1 = ant2 = 1;
    for (i = 2; i < n; i++) {
      actual = ant1 + ant2;
      ant2 = ant1;
      ant1 = actual;
    }
  }
  return actual;
}
```

Algunos otros programas que se basan en esta estructura son **FORTRAN-77**, **COBOL**, **BASIC**, **Pascal**, **C**, **ADA**; otros como **C++**,**C#, Java**, **PHP** e **Eiffel** lo implementan en su interfaz de ejecución.
A este enfoque se asocian los paradigmas **procedurales, OOP** y la programación **estructurada**.

i. Estructurados

Es un paradigma de programación orientado al control de la ejecución, a la mejora de la claridad, calidad y tiempo de desarrollo de un programa de computo en donde solo se recurre y a subrutinas y tres estructuras secuenciales: secuencia, selección (if-switch) e iteración (bucles for-while).
Evita el uso de la instrucción GOTO , la cual siendo proveniente de Basic, pero existente en la mayoría de programas actuales. GOTO basa su ejecución en un punto determinado del código del programa, por lo cual es un tanto tedioso el hecho de compilar el código y no recorrerlo por completo, dificultando la limpieza de errores en el mismo.
Nació en la década de los 60, elaborado particularmente por Bohm y Jacopini, en compañía de un escrito de 1968 titulado "*La sentencia goto, considerada perjudicial".*



ii. Procedimental

Consiste en basarse de un número pequeño de expresiones repetidas englobándolas todas en un procedimiento o función y llamar a este cada vez que tenga que ejecutarse. Esta técnica es aplicable tanto para lenguajes de alto nivel como de bajo nivel, siendo esta para los altos niveles llamada como el paradigma o la programación funcional. Ofrece una muy buena respuesta con relación al tamaño de los programas, siendo poca la diferencia de ejecución de los mismos (siendo todo relacionado al encontrarse almacenado en memoria), aunque es difícil conseguir una programación funcional pura de base.
Algunos ejemplos como:

Dividir un numero entre 2,3,4,5,6,7,8,9 y 10:

```
funcion div(numero) {
    out(numero/2);
    out(numero/3);
    out(numero/4);
    out(numero/5);
    out(numero/6);
    out(numero/7);
    out(numero/8);
    out(numero/9);
    out(numero/10);
}

for( i = 0; i < tamaño(lista); i++) {
    div( lista[i] );
}
```

O mostrar el anterior, posterior y un propio numero de una lista :

```
funcion anterior_posterior( numero ) {
    out(numero-1);
    out(numero);
    out(numero+1);
}

for( i = 0; i < tamaño(lista); i++) {
    anterior_posterior( lista[i] );
}
```

Alguno lenguajes meramente funcionales como **Haskell**, **Miranda**, **Erlang** ; u otros que la soportan como **C++, c# Sharp, StarBasic, InfoBasic, Pascal, Python, As3,** entre otros.

iii. Orientado a objetos

Se basa en el hecho de que los objetos manipulan los datos de entrada para la obtención de datos de salida específicos donde cada objeto es funcional de los otros.
Se basa en técnicas como **herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento,** entre otros.
Según la UNA (Universidad de Antioquia, Colombia), en un paradigma de programación "Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas". Los "estados" y método tratan de manera general, las maneras o clases que necesitan para tratar sus atributos o características que se confieren al objeto.

Este surge como alternativa para la solución de problemas que presentaban otros tipos de paradigmas, mejorando así la eficiencia y acción de los programas.

Algunos programas que soportan la programación orientada a objetos son **Smalltalk**, **C++**, **Delphi (Object Pascal)**, **Java** y **C#**

```
//Definición de métodos públicos (interfaz pública de la clase).
class clase_cliente
{
     //Definición de variables de instancia.
     private string is_nombre;
     private string is_ape1;
     private string is_ape2;

     public void nombre (string nombre, string ape1, string ape2){
          is_nombre = nombre;
          is_ape1 = ape2;
          is_ape2 = ape2;
     }

     public string nombre(){
          return nombre_completo();
     }

     //Definición de métodos de soporte.
     private string nombre_completo(){
          return "Nombre: " + is_nombre + " | Primer ape.: " +
          is_ape1 + " | Segundo ape.: " + is_ape2 ;
     }
}
```

2.1 ENFOQUE DECLARATIVO

Este enfoque trata más de que el programador especifique qué debe computarse, a cómo debe hacerse, por lo que el programador deberá preocuparse más por la lógica que por el control.
Sus principales requisitos deben ser:

- Disponer de un lenguaje que sea lo suficientemente expresivo
- Un mecanismo de cómputo que permita ejecutar los programas
- Disponer de una semántica declarativa que permita dar un significado a los programas de forma independiente a su posible ejecución
- Resultados de corrección y completitud

i. Paradigma funcional

Este paradigma, al ser de enfoque declarativo, requiere de la lógica, pero en su caso específico requiere de la lógica ecuacional, es decir, de que algo lleve a otra cosa.
En este paradigma de programación no existe operación de asignación. Las variables almacenan definiciones o referencias a expresiones. La operación fundamental es la aplicación de una función a una serie de argumentos. La evaluación se guía por el concepto de sustitución.

```
predecesor(x)=x-1, si x>0
sucesor(x)=x+1
suma(x,0)=x
suma(x, y)=sucesor(suma(x, predecesor(y)))

?- suma(3,2)
```

Su principal problema es el mundo externo, puesto que, en un modelo funcional puro, las funciones no pueden tener efectos laterales, es decir, la salida solo puede depender de la entrada, no se pueden modificar entidades, solo crear otras nuevas y darlas como resultado; una función sin parámetros debe

devolver siempre el mismo resultado (es igual a una constante). Sin embargo, existen posibles soluciones a estos problemas, y las mónadas presentan una solución para tratar de mantener el modelo funcional puro.
Su principal virtud es que ayuda a desarrollar la lógica para así llegar a programar en otros paradigmas.
Un ejemplo de un programa de paradigma funcional es dr racket, aunque este no sea un modelo funcional puro.

ii. Paradigma lógico

En este paradigma un *programa* consiste en declarar una serie de hechos (elementos conocidos, relación de objetos concretos) y reglas (relación general entre objetos que cumplen unas propiedades) y luego preguntar por un resultado. Así como su nombre lo explica, funciona con el formalismo lógico, y este dio paso a la mayoría de los programas hoy en día, tanto así, que es la base de la inteligencia artificial. Su mayor ventaja es que solo necesita una fórmula para resolver cientos de problemas, solo es necesario ingresar los datos.

```
Mujer(Rosa)
Mujer(Marta)
Mujer(Laura)
Padres(Rosa, Carlos, Pilar)
Padres(Marta, Carlos, Pilar)
Padres(Laura, Carlos, Pilar)
Hermanas(X, Y):- mujer(X), mujer(Y), padres(X, P, M), padres(Y, P, M)

?- hermanas(Rosa, Marta)
?- hermanas(Rosa, X)
```

3. CONCLUSIONES

-La programación como ciencia abarca distintos campos de la lógica, y, por ende, relaciona muchos tipos de pensamiento para el entendimiento humano-maquina, por lo que su diversidad de acciones hace aún más compleja, pero a la vez innovadora el arte de la programación.

-La programación funcional es una buena manera de introducir a los interesados en la programación, puesto que funciona con conceptos básicos que deberíamos tener gracias a la secundaria, las ecuaciones.

-Para un programador es sumamente importante el desarrollo de la lógica, puesto que, sirve para dar solución a los problemas de la cotidianidad, que requieran o se puedan desarrollar a través de las computadoras.

-El enfoque imperativo por si solo engloba tres de los paradigmas que más se usan en el mundo tecnológico y social, por lo que, gracias a sus próximas evoluciones, junto con la aparición de otros, los cuales serán responsables de la llegada de una tecnología de 5 generación.

4. REFERENCIAS

-Programación Declarativa, Iranzo Julián (Universidad de Castilla-La Mancha,2007)
-Paradigmas De programación, 4. Paradigma Funcional (Universidad de Valladolid, 2010)
- Paradigmas De programación (Universidad de Valladolid, 2012)
-Lenguajes y Paradigmas de programación