Paradigmas de programación

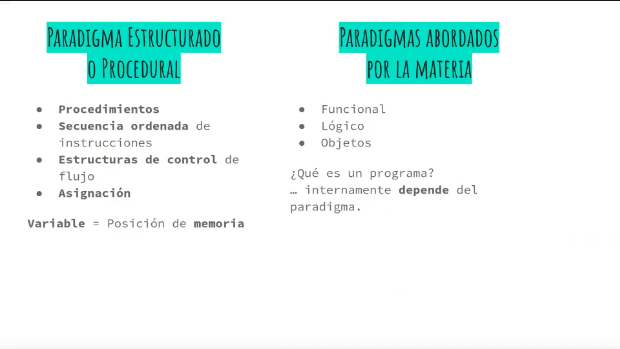
¿Que es programar?

La programación es el proceso por medio del cual se diseña y codifica el código fuente, con el objetivo de crear un software que sea capaz de realizar una acción o mejor dicho resolver el problema por el cual fue creado

¿Que es un paradigma de programación?

Vamos por lo primero. Un **paradigma de programación** es una forma de pensar para resolver un problema  de programación. Es decir es el conjunto de ideas, herramientas conceptuales y las posibles combinaciones entre ellas para resolver un problema.

Ustedes el año pasado en algoritmos y estructura de datos vieron el paradigma **Imperativo**. Este paradigma esta basado en determinar que datos son necesarios para la funcion a realizar, asociarlos a una direccion de memoria e ir transformandolos de manera tal de obtener el resultado esperado. Basicamente un algoritmo.



***DECLARATIVIDAD***

*La declaratividad tiene que ver con que nuestro programa va a estar mas enfocado en qué tiene que hacer y no en cómo hacerlo. Nos conviene delegar los detalles algoritmicos a un mecanismo externo que nos ayude a solucionar un problema.*

*Por ejemplo si nosotros queremos comprar un libro, es mas declarativo si vamos a una biblioteca y si se lo pedimos al bibliotecario. Estamos delegando la responsabilidad de buscarlo a otra persona, y no nos interesa como lo hace, mientas que lo haga para nosotros es suficiente.*

***EXPRESIVIDAD***

*La expresividad se podria definir como que tan lindo es nuestro codigo. ¿A que me refiero?. A que tan facil seria leer esto para otra persona o inclusive para nosotros. No es lo mismo que a una variable que sea un contador le pongamos "A", a que le pongamos "contador". Un ejemplo basico: No es lo mismo*

*comer (abuela,manzana); que*

*funcion1 (a,m);*

***EXPRESIVIDAD Y DECLARATIVIDAD***

*SON CONCEPTOS DIFERENTES.*

*La expresividad tiene que ver con la facilidad de entender lo que hace nuestro código.*

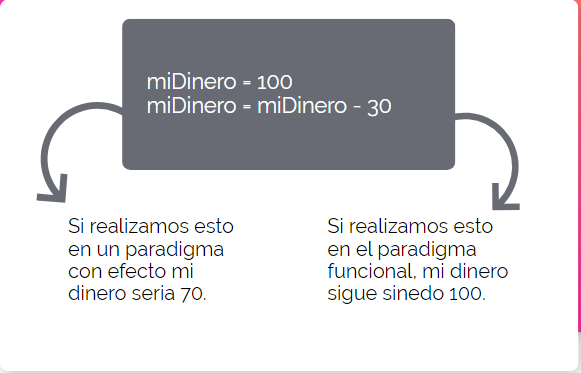
*La declaratividad tiene que ver con menor detalle algoritmico de como resolver y el orden de una secuencia pierde relevancia. De todas formas ambos conceptos van a compañarnos a lo largo de todo el año en los tres paradigmas.*

***Paradigma Funcional***

Este paradigma se basa en entender a los programas como un conjunto de funciones que pueden ser evaluadas para obtener un resultado. El paradigma funcional está basado en conceptos que vienen de la matemática.

***INMUTABILIDAD***

*Ya no existen mas las variables. En el mundo de funcional cuando alguien crea algo se le asigna un valor que no va a poder ser modificado. Es decir que la igualdad ( = ) en funcional es una expresion matematica, no una asignación. No hay Efecto. Olvidense de los contadores.*



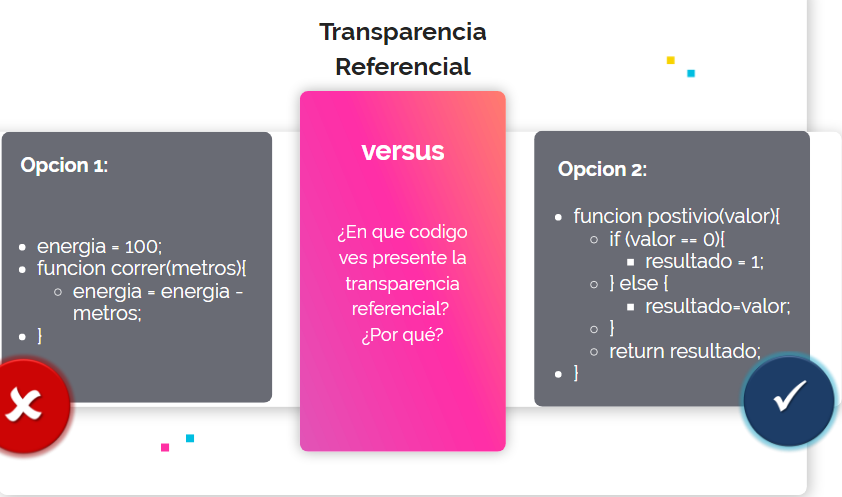
***Entonces aparecen los VALORES***

*En funcional ya no existen las variables como mencionamos anteriormente, existen los valores. A los valores los podemos igualar a otro valor. Con la diferencia que si hacemos por ejemplo miDinero = 2, miDinero es un valor creado por mi y 2 es un valor nativo (ya lo veremos mas adelante). Ahora veamos otro concepto a partir de esto.*

***TRANSPARENCIA REFERENCIAL***

*Hay transparencia referencial si al reemplazar una operación por su resultado se logra el mismo resultado. Una definición alternativa dice: Hay transparencia referencial cuando al realizar una operación con los mismos****valores****siempre da el mismo resultado. Esto se puede ver en el (=) que mencionamos anteriormente.*

*Es decir si tenemos en cuenta el igual(=) como es en matematica si miDinero = 2, entonces miDinero siempre va a ser equivalente a 2. Por lo cual en cualquier momento del programa que utilizemos este valor, el resultado va a ser el mismo.*



***TRANSPARENCIA REFERENCIAL(CONCLUSION)***

*De la inmutabilidad o el no efecto, surge la transperencia referencial.*

***Aplicación parcial****:* Es una función a la que le paso menos parámetros de los que espera pero al menos uno.

Si no le paso ninguno de sus parámetros decimos que no está aplicada y si le pasamos todos sus parámetros decimos que está totalmente aplicada.

La idea de la aplicación parcial es poder obtener nuevas funciones a partir de funciones ya existentes.

Pero… ¿A qué se debe que podamos aplicar parcialmente?

Esto se puede hacer gracias a la ***currificación***. ¿Y eso qué es?

Cuando hablamos de *currificación* nos referimos a que todas las funciones reciben como máximo un único parámetro. El hecho de que sea posible definir funciones de más de un parámetro se debe a que son funciones currificadas. Cuando evaluamos por ejemplo, max 4 5, lo que sucede es que se le aplica el número 5 a la función resultante de aplicarle el 4 a max, o sea que se transforma en (max 4) 5

Por eso, cuando escribimos el tipo de una función no hay una distinción entre lo que son los parámetros y el valor de retorno, se desdibuja un poco la diferencia. Por ejemplo, el tipo de max puede escribirse de dos formas:

max :: (Ord a) => a ->  a -> a --- Forma tradicional

max :: (Ord a) => a -> (a -> a)   --- Forma currificada

La segunda denota que max es una función de 1 parámetro que retorna una función que espera y retorna algo del mismo tipo que lo que ella espera.

Esto se llama ***orden superior*** y ocurre cuando **a una función se le manda como parámetro otra función.** De esta manera estamos creando una función que dependiendo de qué otra función le pasamos por parámetro cambiamos su comportamiento.

La composición, la aplicación parcial y el orden superior son formas de crear funciones a partir de funciones que ya teníamos y su posibilidad de combinarlas.