REPÚBLICA BOLIVARIANA DE VENEZUELA

UNIVERSIDAD NORORIENTAL PRIVADA

“GRAN MARISCAL DE AYACUCHO”

FACULTAD DE INGENIERÍA

NUCLEO BARCELONA

ESCUELA DE INFORMÁTICA

PROGRAMACIÓN FUNCIONAL

PROFESOR: **BACHILLER:**

MANUEL CARRASQUERO **OSCAR DÍAZ C. I: 27.455.444**

BARCELONA, JUNIO DEL 2020

INTRODUCCIÓN

Los lenguajes de programación funcional, especialmente los puramente funcionales, han sido enfatizados en el ambiente académico y no tanto en el desarrollo comercial o industrial. Sin embargo, lenguajes de programación funcional como Lisp (Scheme, Common Lisp, etc.), Erlang, Rust, Objective Caml, Scala, F# y Haskell, han sido utilizados en aplicaciones comerciales e industriales por muchas organizaciones. La programación funcional también es utilizada en la industria a través de lenguajes de dominio específico como R (estadística), Mathematica (cómputo simbólico), J y K (análisis financiero).

Los lenguajes de uso específico usados comúnmente como SQL y Lex/Yacc, utilizan algunos elementos de programación funcional, especialmente al procesar valores mutables. Las hojas de cálculo también pueden ser consideradas lenguajes de programación funcional.

La programación funcional también puede ser desarrollada en lenguajes que no están diseñados específicamente para la programación funcional. En el caso de Perl, por ejemplo, que es un lenguaje de programación imperativo, existe un libro que describe como aplicar conceptos de programación funcional. JavaScript, uno de los lenguajes más ampliamente utilizados en la actualidad, también fue diseñado con capacidades de programación funcional. Python también incorpora particularidades de los lenguajes funcionales como listas de comprensión y funciones de tratamiento de listas, como teoría de conjuntos. Versiones recientes de lenguajes originalmente sin capacidades funcionales, como C++ y Java, han ido incorporando la programación funcional así como el uso de las expresiones lambda.

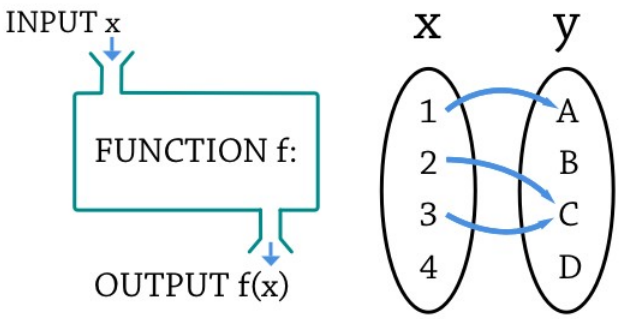
Programación Funcional

En informática, la programación funcional es un paradigma de programación declarativa basado en el uso de verdaderas funciones matemáticas. En este estilo de programación las funciones son ciudadanas de primera clase, porque sus expresiones pueden ser asignadas a variables como se haría con cualquier otro valor; además de que pueden crearse funciones de orden superior.

Utilidad

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 evitar el concepto de estado del cómputo. La secuencia de computaciones llevadas a cabo por el programa se rige única y exclusivamente por la reescritura de definiciones más amplias a otras cada vez más concretas y definidas.

La programación funcional puede ayudarnos a crear software más robusto, mantenible y fácil de testear. Quizás hayas empezado a oír hablar de lenguajes de programación funcional como Scala, Haskell o Lisp, pero quizá no sepas todavía que Java en su versión 8 permite usar la potencia de la programación funcional sin abandonar su orientación a objetos.



Funciones de primera clase y de orden superior

Funciones de orden superior son funciones que pueden tomar otras funciones como argumentos o devolverlos como resultados. En cálculo , un ejemplo de una función de orden superior es el operador diferencial d / dx , que devuelve la derivada de una función f .

Las funciones de orden superior permiten la aplicación parcial, una técnica en la que se aplica una función a sus argumentos uno a la vez, con cada aplicación devolver una nueva función que acepta el siguiente argumento. Esto le permite a uno expresar, por ejemplo, la función sucesor como el operador de suma aplicada parcialmente al número natural uno.

Funciones puras

Las funciones puramente funcionales (o expresiones) no tienen efectos secundarios (memoria o E/S). Esto significa que las funciones puras tienen varias propiedades útiles, muchas de las cuales pueden ser utilizadas para optimizar el código:

- ✓ Si no se utiliza el resultado de una expresión pura, se puede eliminar sin afectar a otras expresiones.
- ✓ Si una función pura se llama con parámetros que no causan efectos secundarios, el resultado es constante con respecto a la lista de parámetros (a veces llamada transparencia referencial), es decir, si la función pura se llama de nuevo con los mismos parámetros, el mismo resultado será devuelto (esto puede habilitar las optimizaciones de almacenamiento en caché).
- ✓ Si no hay una dependencia de datos entre dos expresiones puras, entonces su orden puede ser invertido, o pueden llevarse a cabo en paralelo y que no pueda interferir con los otros.

- ✓ Si el lenguaje no permite efectos secundarios, entonces cualquier estrategia de evaluación se puede utilizar, lo que da la libertad al compilador para reordenar o combinar la evaluación de expresiones en un programa (por ejemplo, usando la poda).

Recursividad

Iterar en los lenguajes funcionales es normalmente llevado a cabo mediante recursividad. Las funciones recursivas se invocan a sí mismas, permitiendo que una operación se realice una y otra vez hasta alcanzar el caso base. Aunque algunas recursividades requieren el mantenimiento de una pila, la recursividad mediante una cola puede ser reconocida y optimizada mediante un compilador dentro del mismo código utilizado, para implementar las iteraciones en un lenguaje imperativo. El estándar del esquema del lenguaje requiere implementaciones para conocer y optimizar la recursividad mediante una cola. La optimización de la recursividad mediante una cola puede ser implementada transformando el programa a un estilo de pase de continuidad durante la compilación, entre otros enfoques.

Ventajas de usar un paradigma funcional

Entre las ventajas que suelen citarse de usar un paradigma funcional en la programación de computadoras, están las siguientes:

1. Ausencia de efectos colaterales.
2. Proceso de depuración menos problemático.
3. Pruebas de unidades más confiables.
4. Mayor facilidad para la ejecución concurrente.

Lenguajes funcionales

Entre los lenguajes funcionales puros, cabe destacar a Haskell y Miranda. Los lenguajes funcionales híbridos más conocidos son Scala, Lisp, Clojure, Scheme, Ocaml, SAP y Standard ML (estos dos últimos, descendientes del lenguaje ML). Erlang es otro lenguaje funcional de programación concurrente. Mathematica permite la programación en múltiples estilos, pero promueve la programación funcional. R también es un lenguaje funcional dedicado a la estadística.4 Recientemente Microsoft Research está trabajando en el lenguaje F# (Functional#).

Entre otros lenguajes que se podrían utilizar para programación funcional se podrían incluir a Perl, pues, aunque es un lenguaje de propósito muy general, se pueden realizar programas usando exclusivamente funciones definidas por el usuario; así como Python, como lenguaje que incorpora el paradigma funcional; o Ruby.

PARA ENTENDER MEJOR EL TEMA DE PROGRAMACIÓN FUNCIONAL SE MOSTRARAN EJEMPLOS EN TRES LENGUAJES DE PROGRAMACIÓN DISTINTOS EN SINTAXIS (JAVA, JAVASCRIPT, PYTHON) LOS CÓDIGOS SE ENCUENTRAN ADJUNTOS EN LA CARPETA

CONCLUSIÓN

Uno de los principios del paradigma es hacer que las funciones sean lo más específicas posible. De esta manera se cumple otro de los principios de este paradigma: la reutilización de código -pues, como veremos, las funciones retornarán lo mismo siempre a lo largo de toda la ejecución del programa.

La Inmutabilidad es la Característica existente en los lenguajes funcionales en la cual que un objeto no puede cambiar su estado. Como consecuencia, esta característica aporta muchas facilidades al momento de razonar sobre el código que estemos creando, y a que no tenemos que preocuparnos por cambios que puedan sufrir los objetos a lo largo del programa. Como ventaja adicional, los objetos que son inmutables, se vuelven automáticamente seguros en el manejo de hilos (o thread-safe) de manera en que pueden ser accedidos de manera concurrente sin presentar ningún tipo de "side effects" debido a que no pueden modificarse. Manejo de errores:Debido a la característica principal de este paradigma (programar mediante funciones),Es relativamente mas fácil detectar errores en el código, ya que al separarlo por funciones, no es necesario revisar todo el código como habría que hacerlo si se hubiera programado de forma imperativa.