Programación Avanzada

Tarea 01: Paradigmas de Programación, Funcional y Lógico

2023-1

Valeria Jimeno Villegas - vjimenov@ciencias.unam.mx

24 de agosto de 2022

Para poder establecer un contacto entre dos entes, es necesario establecer un canal de comunicación entre estos. Entre personas, el lenguaje funge como la principal herramienta para poder lograr ese contacto. Podemos definir al lenguaje como la capacidad que permite al ser humano expresar pensamientos y sentimientos por medio de las palabras. No obstante se tuvo la necesidad de establecer un contacto con otro ente que no fuese una persona, si no una computadora. Como resultado a este problema se crearon los lenguajes de programación, los cuales nos permiten, como programadores, comunicarnos con las computadoras.

Los lenguajes de programación son los mecanismos que nos permite a nosotros, como programadores, comunicarnos con la computadora por medio de ideas algorítmicas que corresponden a especificaciones de cómputos que la máquina debe llevar a cabo[1].

Actualmente existen muchos lenguajes de programación que comparten características en común, lo que ha promovido su clasificación en diferentes categorías, una de estas es por su **paradigma**.

El paradigma es el estilo en el que el lenguaje define la forma en la que se va a solucionar el problema dado. Este se divide en imperativo y declarativo. El paradigma imperativo se centra en dar una sucesión instrucciones de **cómo** se va a resolver un problema. Por otro lado, con el paradigma declarativo no es necesario definir un algoritmo, sino describir las propiedades del resultado al que se desea llegar, es decir a **qué** se desea llegar.

En este trabajo nos centraremos en dos vertientes del paradigma declarativo, el paradigma funcional y el lógico, los cuales se verán con detalle a lo largo del documento. La clasificación del paradigma imperativo está fuera del alcance de este trabajo. Para el desarrollo del archivo se tomó como referencia: [1], [2],

Paradigma Funcional

En este paradigma, se especifica el problema a resolver por medio de programas definidos como una colección de funciones matemáticas. Si se requiere un comportamiento más complejo, se combinan funciones usando la composición de estas.

Características

- Un programa es declarativo, es decir que no va a importar cómo se llega, sino qué es a lo que se quiere llegar.
- En estos lenguajes no se tiene un manejo explícito de la memoria, los cómputos están expresados por aplicación y composición de funciones, en donde los valores intermedios son pasados directamente a otras funciones.
- La recursión es la herramienta de control principal[2].
- Al no tener estado y memoria, las estructuras de datos son persistentes.

Como ejemplos de lenguajes puramente funcionales se tienen a: Haskell, Miranda, Gofer, etcetera. También existen lenguajes funcionales impuros como lo son: Lisp, Scheme, Racket, ML entre otros. Una vez dicho lo anterior, se muestra el Algoritmo de ordenamiento `Quick Sort` en el lenguaje de programación puramente funcional Haskell[1].

```haskell
quicksort :: (Ord a) => [a] -> [a]
quicksort [] = []
quicksort (x:xs) = quicksort less ++ x:(quicksort more)
where less = filter (<x) xs
        more = filter (>=x) xs
```

Figura 1: Algoritmo de ordenamiento `Quicksort` en el lenguaje Haskell.

A continuación se muestra una tabla con las posibles ventajas y desventajas que se le encuentran al paradigma funcional.

Ventajas	Desventajas
Cálculos representables algebraicamente	Difícil de escalar
Algoritmos demostrablemente correctos	Tipo de seguridad a menudo falta
Representaciones teóricas	Limitado a abstracciones matemáticas
Procesos sin estado	

Paradigma Lógico

A partir de un problema, dentro del paradigma lógico, se especifican características esenciales que debe de cumplir la solución de este mediante declaraciones lógicas. Es decir se parte de la solución del problema y de ahí se lleva a cabo un proceso de *backtracking* con el cual el lenguaje encuentra la solución que cumple con las reglas o características definidas inicialmente haciendo una búsqueda de pruebas. Análogo al paradigma de programación funcional, en el lógico no es posible la manipulación explícita de la memoria[1].

Características

- Un programa es declarativo, no importa el cómo sino el qué.
- La orientación es a pruebas formales[2].
- El control basado en recursión.
- No hay memoria implícita.
- Un cómputo se expresa mediante la búsqueda de pruebas o por definición de predicados recursivos.

Como ejemplos de lenguajes puramente lógicos se tienen a: Prolog, Datalog, entre otros, en donde los predicados lógicos se escriben en forma de cláusulas:

$$\mathtt{H} : -\mathtt{B_1}, \ldots, \mathtt{B_n},$$

y se leen en forma de implicaciones lógicas:

$$\texttt{H if } \texttt{B}_1 \texttt{ and } \ldots \texttt{ and } \texttt{B}_n,$$

en donde `H` es un hecho y $\texttt{B}_1, \ldots, \texttt{B}_n$ son las reglas que componen un campo de conocimiento definido como `C`.

Análogo al paradigma funcional, se muestra a continuación el algoritmo de ordenamiento *Quicksort* en el lenguaje de programación *Prolog*[1].

```
quicksort([X|Xs],Ys) :-
        partition(Xs,X,Left,Right),
        quicksort(Left,Ls),
        quicksort(Right,Rs),
        append(Ls,[X|Rs],Ys).
quicksort([],[]).
```

Figura 2: Algoritmo de ordenamiento `Quicksort` en el lenguaje Prolog.

A continuación se muestra una tabla con las posibles ventajas y desventajas que se le encuentran al paradigma lógico.

Ventajas	Desventajas
Algoritmos demostrablemente correctos	Es difícil ver lo que realmente está sucediendo computacionalmente
Problemas basados en la lógica	Se requieren modelos no intuitivos
Problemas caracterizados por definiciones y/o restricciones	

Conclusión

Desde mi punto de vista el ver las ventajas y desventajas entre paradigmas no es lo más viable al momento de querer llevar a cabo un programa, creo que de primera mano es necesario conocer a fondo el problema, sus características y de igual forma

definir cual es a la solución a la que se quiere llegar, para de ahí partir y saber con cual paradigma abordar el problema y ya de ahí sí tomar en cuenta las ventajas y desventajas que tienen ciertos lenguajes de programación sobre de otros. Como se mencionó en clase, cada paradigma tiene su razón de ser y esta se dio a partir de los problemas a los que se enfrentaron sus creadores, es por eso que debemos analizar detalladamente cada problema y cuál sería la mejor forma de abordarlo.

Una vez dicho lo anterior, podemos concluir que los paradigmas son simplemente la manera o el estilo que toman los lenguajes para poder definir la forma en la que van a solucionar un problema. Como se mencionó anteriormente, el paradigma declarativo define características de la solución del problema, y dado que de de ahí se parte, este utiliza la recursividad como su herramienta de control principal. Por otro lado, el paradigma imperativo define una serie de instrucciones o pasos de cómo se debe llegar a la solución del problema dado. El primero parte de la solución, mientras que el otro camina hacia esta mediante una serie de pasos.

Referencias

[1] J. E. Mendoza, "Notas de clase 1: Estudio de lenguajes," 2022, [Online; accessed 21-08-22].

[2] F. E. M. Perea and L. C. G. Huesca, "Nota de clase 1: Introducción," 2020, [Online; accessed 21-08-22].