Abstracción de datos

Ricardo Pérez López

IES Doñana, curso 2019/2020

Índice general

1. Tipos abstractos de datos

1.1. Concepto, terminología y ejemplos

Al considerar el amplio conjunto de cosas en el mundo que nos gustaría representar en nuestros programas, descubrimos que la mayoría de ellas tienen una estructura compuesta.

Por ejemplo, una posición geográfica tiene coordenadas de latitud y longitud. Para representar posiciones, nos gustaría que nuestro lenguaje de programación tenga la capacidad de juntar una latitud y longitud para formar un par, un valor de datos compuesto que nuestros programas pueden manipular

como una sola unidad conceptual, pero que también tiene dos partes que pueden ser considerados individualmente por separado.

El uso de datos compuestos nos permite aumentar la modularidad de nuestros programas.

Si podemos manipular posiciones geográficas como valores completos, entonces podemos proteger partes de nuestro programa que calculan usando posiciones de los detalles de cómo se representan esas posiciones.

La técnica general de aislar las partes de un programa que se ocupan de cómo se representan los datos de las partes que se ocupan de cómo se manipulan los datos es una poderosa metodología de diseño llamada abstracción de datos.

La abstracción de datos hace que los programas sean mucho más fáciles de diseñar, mantener y modificar.

La abstracción de datos es de carácter similar a la abstracción funcional.

Cuando creamos una abstracción funcional, los detalles de cómo se implementa una función pueden ser suprimidos, y la función particular en sí misma puede ser reemplazada por cualquier otra función con el mismo comportamiento general.

- En otras palabras, podemos hacer una abstracción que separe la forma en que se utiliza la función de los detalles de cómo se implementa la función.

De manera análoga, la abstracción de datos aísla cómo se usa un valor de datos compuesto de los detalles de cómo se construye.

La idea básica de la abstracción de datos es estructurar programas para que operen con datos abstractos. Es decir, nuestros programas deben usar los datos de tal manera que hagan la menor cantidad posible de suposiciones sobre los datos. Al mismo tiempo, una representación de datos concretos se define como una parte independiente del programa.

Estas dos partes de un programa, la parte que opera en datos abstractos y la parte que define una representación concreta, están conectadas por un pequeño conjunto de funciones que implementan datos abstractos en términos de la representación concreta. Para ilustrar esta técnica, consideraremos cómo diseñar un conjunto de funciones para manipular números racionales.

1.1.1. Ejemplo: números racionales

Un número racional es una relación de enteros, y los números racionales constituyen una subclase importante de números reales. Un número racional como 1/3 o 17/29 generalmente se escribe como:

/

donde tanto como son marcadores de posición para valores enteros. Ambas partes son necesarias para caracterizar exactamente el valor del número racional. La división de enteros en realidad produce una aproximación flotante , perdiendo la precisión exacta de los enteros.

> 1 / 3 ,3333333333333333 1 / 3 == 0.333333333333333300000 # La división de números enteros, se obtiene una aproximación verdadera

Sin embargo, podemos crear una representación exacta para números racionales combinando juntos el numerador y el denominador.

Por el uso de abstracciones funcionales, sabemos que podemos comenzar a programar productivamente antes de implementar algunas partes de nuestro programa. Comencemos suponiendo que ya tenemos una manera de construir un número racional a partir de un numerador y un denominador. También suponemos que, dado un número racional, tenemos una forma de seleccionar su numerador y su componente denominador. Supongamos además que el constructor y los selectores están disponibles como las siguientes tres funciones:

racional (n, d) devuelve el número racional con numerador n y denominador d . numer (x) devuelve el numerador del número racional x . denom (x) devuelve el denominador del número racional x . Estamos utilizando una estrategia poderosa para diseñar programas: ilusiones . Todavía no hemos dicho cómo se representa un número racional, o cómo se deben implementar las funciones numer , denom y racional . Aun así, si definimos estas tres funciones, podríamos sumar, multiplicar, imprimir y probar la igualdad de números racionales:

1.2. Programación con tipos abstractos de datos

1.2.1. Modularidad

1.2.2. Refinamientos sucesivos

1.2.3. Programación a gran escala

1.2.4. Programación genérica

2. Especificación

2.1. Especificaciones algebraicas

2.1.1. Signatura de un TAD

2.1.1.1. Géneros

2.1.1.2. Operaciones

2.1.1.2.1. Constructoras

2.1.1.2.2. Selectoras