TADs

Algoritmos y estructuras de datos

Elías Cerdeira

¿Qué vamos a ver hoy?

- Breve introducción a TADs
- Especificar algunos TADs simples
- **Consultas**

¿Qué es un TAD?

Define un **conjunto de valores** y las **operaciones** que se pueden realizar sobre ellos

Abstracto. Se enfoca en las operaciones. No necesita conocer los detalles de la representación interna o cómo se implementan sus operaciones

La especificación indica **qué** hacen las operaciones y no **cómo** lo hacen

¿Qué partes tiene un TAD?

1. Nombre (*y parámetros*)
2. Observadores
3. Operaciones
   a. Parámetros con sus tipos
   b. Requiere
   c. Asegura
4. *Predicados y funciones auxiliares (opcional)*

¿Cómo luce un TAD?

1

```
TAD Diccionario<K,V> {
```

2

```
    obs data: dict<K,V>

    proc diccionarioVacío(): Diccionario<K,V>
        asegura {res.data = {}}

    proc está(in d: Diccionario<K,V>, in k: K): bool
        asegura {res = true ↔ k ∈ d.data}

    proc definir(inout d: Diccionario<K,V>, in k: K, in v: V)
        requiere {d = D_0}
        asegura {d.data = setKey(D_0.data, k, v)}

    proc obtener(in d: Diccionario<K,V>, in k: K): V
        requiere {k ∈ d.data}
        asegura {res = d.data[k]}

    proc borrar(inout d: Diccionario<K,V>, in k: K)
        requiere {d = D_0 ∧ k ∈ d.data}
        asegura {d.data = delKey(D_0.data, k)}

    proc definirRápido(inout d: Diccionario<K,V>, in k: K, in v: V)
        requiere {d = D_0 ∧ k ∉ d.data}
        asegura {d.data = setKey(D_0.data, k, v)}

    proc tamaño(in d: Diccionario<K,V>): ℤ
        asegura {res = |d.data|}
}
```

3

En este caso **K** y **V** son tipos genéricos que se instanciarán según el caso. Por ej: Diccionario<Int,String>

¡Vamos a ver el apunte!

¡A resolver ejercicios!

1er TAD

Ejercicio 1. Especificar en forma completa el TAD `NumeroRacional` que incluya al menos las operaciones aritméticas básicas (suma, resta, división, multiplicación)

1) Piensen los observadores

2) Especificar las operaciones

3) Piensen ahora qué otros observadores podrían usar

¿Se les ocurre otro conjunto de observadores?

Más TADs

Ejercicio 4.

a) Especifique el TAD Diccionario$\langle K, V \rangle$ con las siguientes operaciones:

   a) *nuevoDiccionario*: que crea un diccionario vacío
   b) *definir*: que agrega un par clave-valor al diccionario
   c) *obtener*: que devuelve el valor asociado a una clave
   d) *está*: que devuelve true si la clave está en el diccionario
   e) *borrar*: que elimina una clave del diccionario

1) Pensemos más de un conjunto de observadores

2) ¿Dónde está la diferencia entre TADs en este caso?

b) Especifique el TAD DiccionarioConHistoria$\langle K, V \rangle$. El mismo permite consultar, para cada clave, todos los valores que se asociaron con la misma a lo largo del tiempo (en orden). Se debe poder hacer dicha consulta aún si la clave fue borrada.

¡Intervalo!

Aún más TADs...

Ejercicio 8. Un `caché` es una capa de almacenamiento de datos de alta velocidad que almacena un subconjunto de datos, normalmente transitorios, de modo que las solicitudes futuras de dichos datos se atienden con mayor rapidez que si se debe acceder a los datos desde la ubicación de almacenamiento principal. El almacenamiento en caché permite reutilizar de forma eficaz los datos recuperados o procesados anteriormente.

Esta estructura comunmente tiene una interface de diccionario: guarda valores asociados a claves, con la diferencia de que los datos almacenados en un cache pueden *desaparecer* en cualquier momento, en función de diferentes criterios.

Especificar caches genéricos (con claves de tipo K y valores de tipo V) que respeten las operaciones indicadas y las siguientes políticas de borrado automático. Si necesita conocer la fecha y hora actual, puede pasarla como parámetro a los procedimientos o bien puede asumir que existe una función externa $horaActual() : \mathbb{Z}$ que retorna la fecha y hora actual. Asuma que las fechas son números enteros (por ejemplo, la cantidad de segundos desde el 1 de enero de 1970).

```
TAD Cache⟨K, V⟩ {
proc esta(in c: Cache⟨K, V⟩, in k: K)  : Bool
proc obtener(in c: Cache⟨K, V⟩, in k: K)  : V
proc definir(inout c: Cache⟨K, V⟩, in k: K)
}
```

Aún más TADs...

a) `FIFO` o `first-in-first-out`:
El cache tiene una capacidad máxima (máximo número de claves). Si se alcanza esa capacidad máxima se borra automáticamente la clave que fue definida por primera vez hace más tiempo.

b) `LRU` o `last-recently-used`:
El cache tiene una capacidad máxima (máximo número de claves). Si se alcanza esa capacidad máxima se borra automáticamente la clave que fue accedida por última vez hace más tiempo. Si no fue accedida nunca, se considera el momento en que fue agregada.

c) `TTL` o `time-to-live`:
El cache tiene asociado un máximo tiempo de vida para sus elementos. Los elementos se borran automáticamente cuando se alcanza el tiempo de vida (contando desde que fueron agregados por última vez).

¿Qué sigue?

- Con la clase de hoy pueden empezar la **guía 4**
- La clase que viene tenemos más ejercicios de TADs

¡Terminamos!

¡Hagan consultas!