Algoritmos y Estructuras de Datos

Guía Práctica 4
Especificación de TADs

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

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

Ejercicio 2. Especificar TADs para las siguientes figuras geométricas. Tiene que contener las operaciones rotar, trasladar y escalar y una más propuestas por usted.

a) `Rectangulo` (2D)

b) `Esfera` (3D)

Ejercicio 3. Especifique el TAD `DobleCola<T>`, en el que los elementos pueden insertarse al principio o al final y se eliminan por el medio. Ejemplo:

```
c := DobleCola<int>.NuevaDobleCola()

encolarAdelante(c, 1)       // c = <1>
encolarAdelante(c, 2)       // c = <2,1>
encolarAtrás(c, 3)          // c = <2,1,3>

desencolar(c)               // devuelve 1, c = <2,3>
desencolar(c)               // devuelve 2, c = <3>
desencolar(c)               // devuelve 3, c = <2>
```

Ejercicio 4. Especifique el TAD `DiccionarioConHistoria`. El mismo guarda, para cada clave, todos los valores que se asociaron con la misma a lo largo del tiempo (en orden).

Ejercicio 5. Modifique el TAD `ColaDePrioridad<T>` para que, si hay muchos valores iguales al máximo, la operación `desapilarMax` los desapile a todos.

Ejercicio 6. Especifique los TADs indicados a continuación pero utilizando los observadores propuestos:

a) `Diccionario<K,V>` observado con conjunto (de tuplas)

b) `Conjunto<T>` observado con funciones

c) `Pila<T>` observado con diccionarios

d) `Punto` observado con coordenadas polares

Ejercicio 7. Especificar TADs para las siguientes estructuras:

a) `Multiconjunto<T>`
También conocido como `multiset` o `bag`. Es igual a un conjunto pero con duplicados: cada elemento puede agregarse múltiples veces. Tiene las mismas operaciones que el TAD `Conjunto`, más una operación que indica la multiplicidad de un elemento (la cantidad de veces que ese elemento se encuentra en la estructura). Nótese que si un elemento es eliminado del multiconjunto, se reduce en 1 la multiplicidad.

Ejemplo:

```
c := MultiConjunto<int>.nuevoMultiConjunto()

agregar(c, 1)
agregar(c, 1)
pertenece(c, 1)          // devuelve true
multiplicidad(c, 1)      // devuelve 2

sacar(c, 1)
pertenece(c, 1)          // devuelve true
multiplicidad(c, 1)      // devuelve 1
```

b) `Multidict<K,V>`
Misma idea pero para diccionarios: Cada clave puede estar asociada con múltiples valores. Los valores se definen de a uno (indicando una clave y un valor), pero la operación `obtener` debe devolver todos los valores asociados a una determinada clave.

Nota: En este ejercicio deberá tomar algunas decisiones. ¿Se pueden asociar múltiples veces un mismo valor con una clave? ¿Qué pasa en ese caso? Qué parámetros tiene y cómo se comporta la operación `borrar`? Imagine un caso de uso para esta estructura y utilice su sentido común para tomar estas decisiones.

Ejercicio 8. Especifique el TAD `Contadores` que, dada una lista de eventos, permite contar la cantidad de veces que se produjo cada uno de ellos. La lista de eventos es fija. El TAD debe tener una operación para incrementar el contador asociado a un evento y una operación para conocer el valor actual del contador para un evento.

- Modifique el TAD para que sea posible preguntar el valor del contador *en un determinado momento del pasado*. Si necesita conocer la fecha y hora actual, puede pasarla como parámetro a los procedimientos. Asuma que las fechas son números enteros (por ejemplo, la cantidad de segundos desde el 1 de enero de 1970).

Ejercicio 9. Supongamos que queremos utilizar el TAD contador en un sistema que procesa millones de eventos por segundo y no damos abasto para procesar todos los eventos. Una posible solución es hacer *sampling*: en lugar de contar cada evento, contamos un porcentaje (configurable) de ellos, por ejemplo, un 1 %. Es decir que de todas las llamadas al proc `incrementar`, sólo el 1 % de ellas efectivamente incrementa el contador.

- ¿Es correcta esta especificación? ¿Cuándo falla? ¿Cuál cree que es el problema?

  `proc incrementar(inout c: Contador, in e: Evento)`
  `requiere` $\{c = C_0\}$
  `asegura` $\{c.valor[e] = C_0.valor[e] \vee c.valor[e] = C_0.valor[e] + 1\}$

- ¿Cómo cree que se podría mejorar esta especificación?

Ejercicio 10. 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)
}
```

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).

Ejercicio 11. Especifique tipos para un robot que realiza un camino a través de un plano de coordenadas cartesianas (enteras), es decir, tiene operaciones para ubicarse en un coordenada, avanzar hacia arriba, hacia abajo, hacia la derecha y hacia la izquierda, preguntar por la posición actual, saber cuántas veces pasó por una coordenada dada y saber cuál es la coordenada más a la derecha por dónde pasó. Indique observadores y precondición/postcondición para cada operación:

```
Coord es struct <x: ℤ, y: ℤ>
TAD Robot {
      proc arriba(inout r: Robot)
      proc abajo(inout r: Robot)
      proc izquierda(inout r: Robot)
      proc derecha(inout r: Robot)
      proc másDerecha(in r: Robot): ℤ
      proc cuantasVecesPaso(in r: Robot, in c: Coord): ℤ
}
```

Ejercicio 12. Queremos modelar el funcionamiento de un vivero. El vivero arranca su actividad sin ninguna planta y con un monto inicial de dinero.

Las plantas las compramos en un mayorista que nos vende la cantidad que deseemos pero solamente de a una especie por vez. Como vivimos en un país con inflación, cada vez que vamos a comprar tenemos un precio diferente para la misma planta. Para poder comprar plantas tenemos que tener suficiente dinero disponible, ya que el mayorista no acepta fiarnos.

A cada planta le ponemos un precio de venta por unidad. Ese precio tenemos que poder cambiarlo todas las veces que necesitemos. Para simplificar el problema, asumimos que las plantas las vendemos de a un ejemplar (cada venta involucra un solo ejemplar de una única especie). Por supuesto que para poder hacer una venta tenemos que tener *stock* de esa planta y tenemos que haberle fijado un precio previamente. Además, se quiere saber en todo momento cuál es el balance de caja, es decir, el dinero que tiene disponible el vivero.

- Indique las operaciones (procs) del TAD con todos sus parámetros.
- Describa el TAD en forma completa, indicando sus observadores, los requiere y asegura de las operaciones. Puede agregar los predicados y funciones auxiliares que necesite, con su correspondiente definición
- ¿qué cambiaría si supiéramos a priori que cada vez que compramos en el mayorista pagamos exactamente el 10 % más que la vez anterior? Describa los cambios en palabras.