Pilas

Comentarios sobre la guía

Al ejercicio 16, al menos en la versión que está en el repo github, le falta un renglón. Podría ser “(... una con sólo) los pares y otra con sólo los impares”.

Otra cosa: tratemos de no denominar “función” a subprogramas que tienen efecto. P.ej. el ejercicio 9, “escribir una función que coloque en el fondo de una pila un nuevo elemento”. El subprograma modifica la pila que recibe como parámetro, no devuelve una pila nueva. Por eso, creo que conviene llamarlo “procedimiento” para respetar la nomenclatura de Intro.

En el ejercicio 11, no entiendo qué espera como resultado de “duplicar el contenido de una pila”. Si la pila tiene (de tope a la base) a-b-c, ¿se espera a-b-c-a-b-c, a-a-b-b-c-c, dos pilas?

Propuestas de ejercicios.

Definir un procedimiento `volcarPila(pilaOrigen, pilaDestino, cantidad)` que trasvase la cantidad indicada de elementos de la pila origen a la destino. Si la cantidad es mayor al tamaño de la pila origen, o no se especifica, entonces vuelca la pila origen completa, de forma tal que queda vacía.
Realizar implementaciones alternativas de los ejercicios 6, 7, 9 y 16 usando este procedimiento.

Definir un procedimiento
`volcarPilaEnDos(pilaOrigen, pilaDestino1, pilaDestino2, cantidad)`
similar al anterior, que copie los elementos a dos pilas destino.
Este procedimiento podría ser útil en algún ejercicio de los que siguen.

Agregar al TDA Pila una operación que devuelva una lista con los mismos elementos, respetando el orden con el tope como primer elemento.
Realizar una implementación alternativa de los ejercicios 12 y 13 usando esta operación.

Definir un TDA que represente a un oficinista que maneja dos pilas de expedientes, una de trámites urgentes y otra de trámites normales. De cada expediente se conoce: su prioridad que puede ser normal o urgentes, y un número que lo identifica.
El TDA oficinista debe incluir las siguientes operaciones:

- recibirExpediente(exp) - lo agrega a la pila que corresponde a su prioridad
- primerExpedienteATratar() - si hay trámites urgentes, el tope de la pila correspondiente, caso contrario, el tope de la pila de los normales.
- tratarPrimerExpediente() - lo saca del tope de la pila en la que está, y lo devuelve.
- cantidadDeExpedientes() - sumando las dos pilas.
- estaComplicado() - indica si tiene más de 10 expedientes por tratar.

Agregar una operación al TDA Pila que cree y devuelva un clon, o sea otra pila con los mismos elementos en el mismo orden.

Definir una función que reciba dos pilas de números ordenados de mayor a menor, y devuelva una pila que contenga los elementos de ambas pilas, ordenados de menor a mayor. Las pilas originales no deben modificarse. Sugerencia: usar la operación que devuelve un clon, del ejercicio anterior.

Definir una función similar a la anterior, que reciba dos pilas de números ordenados de mayor a menor, y devuelva una pila que contenga los elementos de ambas pilas, ordenados de mayor a menor.

Hay que acomodar un montón de remeras rojas, amarillas y verdes. Primero se hacen tres pilas, una por color, y luego se vuelcan a una única pila formando la bandera de Bolivia (rojo / amarillo / verde) tantas veces como se pueda.
De cada remera se conoce: talle, color, número que la identifica
Armar un TDA que debe soportar las siguientes operaciones

- Incorporar una remera, la agrega a la pila correspondiente al color
- Obtener la pila de remeras rojas. Idem con amarillas y verdes.
- Consolidar las remeras apiladas, esto debe devolver una estructura con dos pilas. En una están las remeras organizadas de acuerdo a la bandera de Bolivia. En la otra las remeras que sobraron.

Definir un TDA que modele la operación de organizar una montaña de remeras en pilas organizadas por talle y color. Debe soportar las siguientes operaciones:

- Incorporar una remera.
- Dados un talle y un color, devolver la pila de remeras que corresponde.

Definir una función take(pila,n), que devuelve una pila que contiene los primeros n elementos de la pila.

Colas

Comentarios sobre la guía

En el ejercicio 4, indicar como sugerencia que se puede usar una pila en la implementación.

Propuestas de ejercicios.

Modelar una sucursal de Pago Fácil que tiene dos colas. De cada cliente se conoce el importe que va a pagar de facturas.
El TDA que modela a la sucursal debe soportar estas operaciones:

- recibirCliente(cliente): se suma a la cola más corta. Si tienen el mismo tamaño, elegir a cuál se suma.
- atenderCliente(cliente): se obtiene de la cola más larga. Si tienen el mismo tamaño, elegir de qué cola se obtiene el cliente a atender.
- cantidadDeClientesAtendidos()
- cantidadDeClientesEsperando()
- importeRecaudado()
- ultimaPersonaAtendida()

Variantes de este TDA

- En una de las colas se admiten solamente pagos hasta una cierta cantidad de dinero, p.ej. hasta 10000 pesos. Entonces, al recibir a un cliente, si el importe a pagar supera la cantidad aceptada por una cola, va a la otra sin importar las longitudes. Al recibir a un cliente, si las dos colas tienen el mismo tamaño y el importe a pagar no supera el límite de la cola con límite, entonces se lo asigna a esa cola.
- Imaginar la siguiente situación
    - Llega un cliente a pagar por un importe mayor al que soporta la cola con límite, se lo envía a la cola sin límite.
    - Llegan dos clientes con importes pequeños, de acuerdo a lo especificado se los ubica en la cola con límite.
    - Se atiende un cliente, se lo obtiene de la cola con límite porque es la más larga.
    - Llega un cliente con un importe pequeño, va a la cola con límite.
    - Se atiende un cliente, se vuelve a obtener de la cola con límite porque en este momento es la más larga.
    - Se repite la secuencia llega un cliente con un importe pequeño / se atiende un cliente.

  En este escenario, el cliente de la cola sin límite de importe nunca es atendido. Este es un caso del fenómeno conocido como *inanición*.
  Evitar este fenómeno, haciendo que luego de 4 clientes consecutivos que se obtienen de la cola con límite, se atienda uno de la cola sin límite sin importar los tamaños de las colas.

- Agregar una cola prioritaria, p.ej. para embarazadas o personas con problemas de salud. A cada cliente agregarle como dato si debe ser atendido en forma prioritaria o no. Cuando se recibe un cliente que debe ser atendido en forma prioritaria, se lo ubica en la nueva cola. Al atender un cliente, si hay alguien en la cola prioritaria se lo atiende.

Modelar un supermercado con una cantidad arbitraria de cajas. Algunas de estas son cajas rápidas, que sólo reciben clientes con hasta 15 productos. Por otro lado, algunas cajas (ya sean rápidas o no) aceptan pago con tarjeta, y otras no. Finalmente, una caja puede estar abierta o cerrada. Cada caja tiene un número.
Para cada cliente se conoce: cuántos productos está comprando, y si paga con tarjeta.
El TDA caja debe contemplar estas operaciones:

- abrir(): la caja queda abierta.
- cerrar(): la caja queda cerrada. Los clientes que están en la cola al momento de cerrarla se pueden atender.
- numero(): el número de caja.
- aceptaCliente(cliente): indica si el cliente puede pagar en esta caja, de acuerdo a la cantidad de productos y a la forma de pago. Una caja cerrada no puede atender a ningún cliente.
- recibirCliente(cliente): si no lo puede atender, debe lanzarse un error.
- recibirClientePrioritario(cliente): lo ubica al principio de la cola de espera.
- seVaCliente(cliente): el cliente deja la cola de espera. Si el cliente indicado no está en la cola de espera, debe lanzarse un error.
- primerClienteAAtender(): devuelve el primer cliente en la cola de espera.
- atenderCliente(): saca al primer cliente de la cola de espera.
- cantidadDeClientesAtendidos()

El TDA supermercado debe contemplar estas operaciones:

- agregarCaja(caja): si ya hay una caja con el número de la caja a agregar, debe lanzarse un error.
- recibirCliente(cliente): se agrega a la cola más corta (si hay varias, una cualquiera entre ellas) considerando sólo las que aceptan al cliente.
- cajaAtiendeCliente(numero): la caja con el número indicado atiende al primer cliente en su cola de espera.
- volcarColaDeEspera(cajaOrigen, cajaDestino): agregar todos los clientes en espera de la caja origen, en la cola de la caja destino.
- cantidadDeClientesAtendidos()
- cantidadDeClientesAtendidosEnCajasRapidas()
- cantidadDeClientesEnEspera()