UNIVERSIDAD FRANCISCO DE VITORIA
VINCE IN BONO MALUM

Estructuras de Datos y Algoritmos

Especificación algebraica de TADs

Ejemplos: Pilas, Colas, Listas

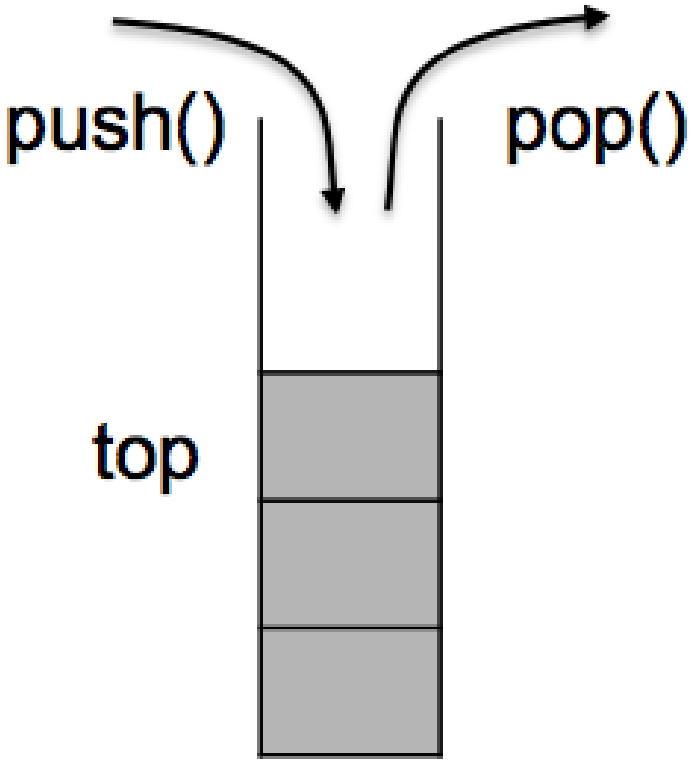
Prof. Dr. P. Javier Herrera

Índice

Ejemplos de especificación algebraica de TADs.

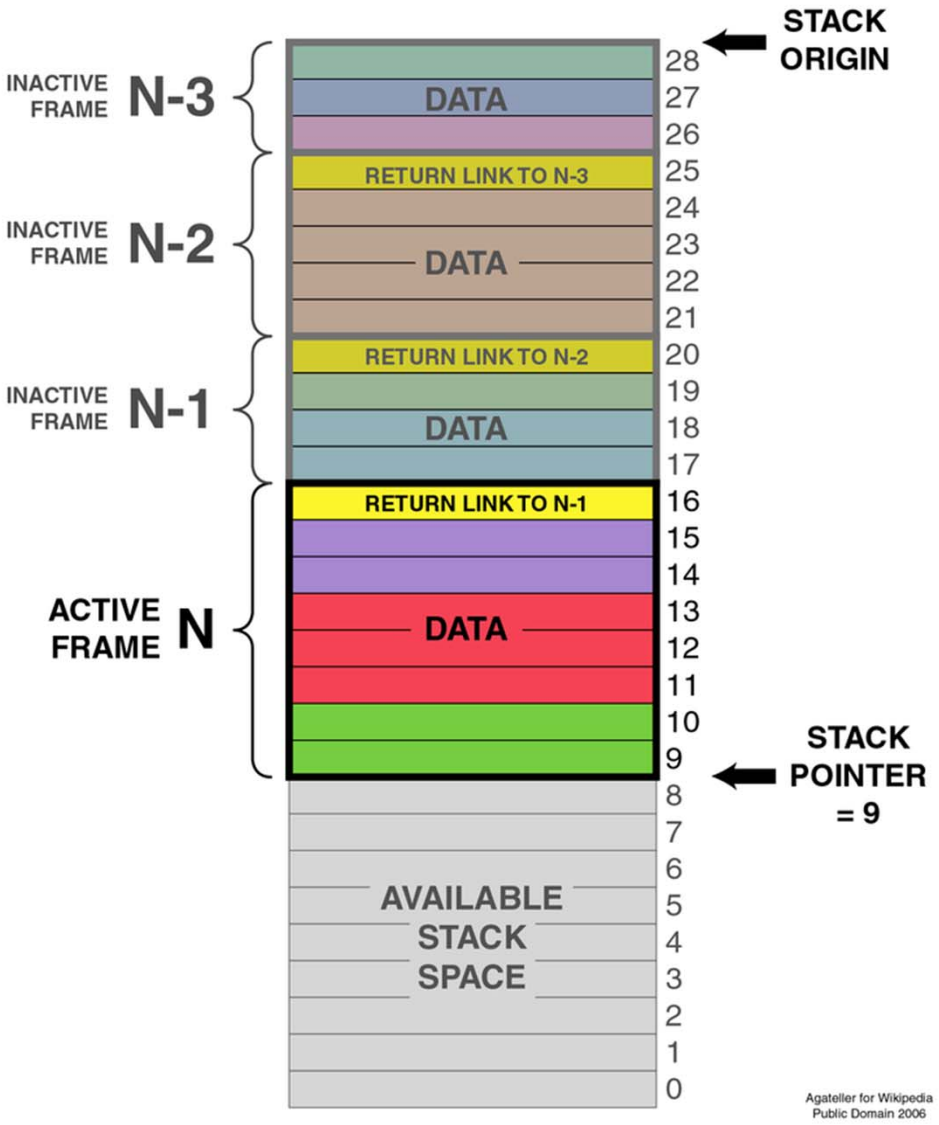
- Pila
- Cola
- Lista

Pila (*Stack*)



Pila - Aplicaciones

- Almacenamiento de datos locales y la información de llamadas para las llamadas a procedimientos anidados



Pila - Aplicaciones

- Evaluación de la expresión y el análisis sintáctico.
  - Por ejemplo, el cálculo **((1 + 2) * 4) + 3**, puede ser anotado como en notación postfija con la ventaja de no prevalecer las normas y los paréntesis necesario: **1 2 + 4 * 3 +**

- Backtracking (Vuelta atrás): es una estrategia para encontrar soluciones a problemas que satisfacen restricciones

- Gestión de memoria en tiempo de ejecución

Pila - Operaciones básicas

- El TAD de las pilas cuenta con las siguientes operaciones:
  - crear la pila vacía,
  - apilar un elemento,
  - desapilar el elemento en la cima,
  - consultar el elemento en la cima, y
  - determinar si la pila es vacía.

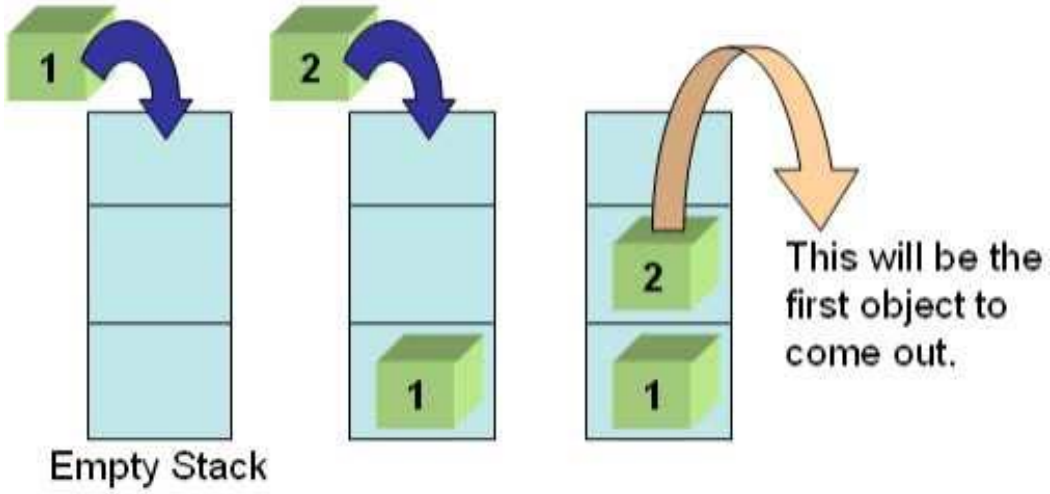
Pila - Especificación algebraica

especificación *PILAS*[*ELEM*]
usa *BOOLEANOS*
tipos *pila*
operaciones

pila-vacía	:		→ *pila*	{ constructora }
apilar	:	*elemento pila*	→ *pila*	{ constructora }
desapilar	:	*pila*	$\rightarrow_p$ *pila*	
cima	:	*pila*	$\rightarrow_p$ *elemento*	
es-pila-vacía?	:	*pila*	→ *bool*	

- Como el orden de apilación es fundamental para la posterior consulta y eliminación, las constructoras son **libres** (no son necesarias ecuaciones de equivalencia).

Pila - Especificación algebraica



variables

e : elemento

p : pila

ecuaciones

desapilar(pila-vacía) = error

desapilar(apilar(*e*, *p*)) = *p*

cima(pila-vacía) = error

cima(apilar(*e*, *p*)) = *e*

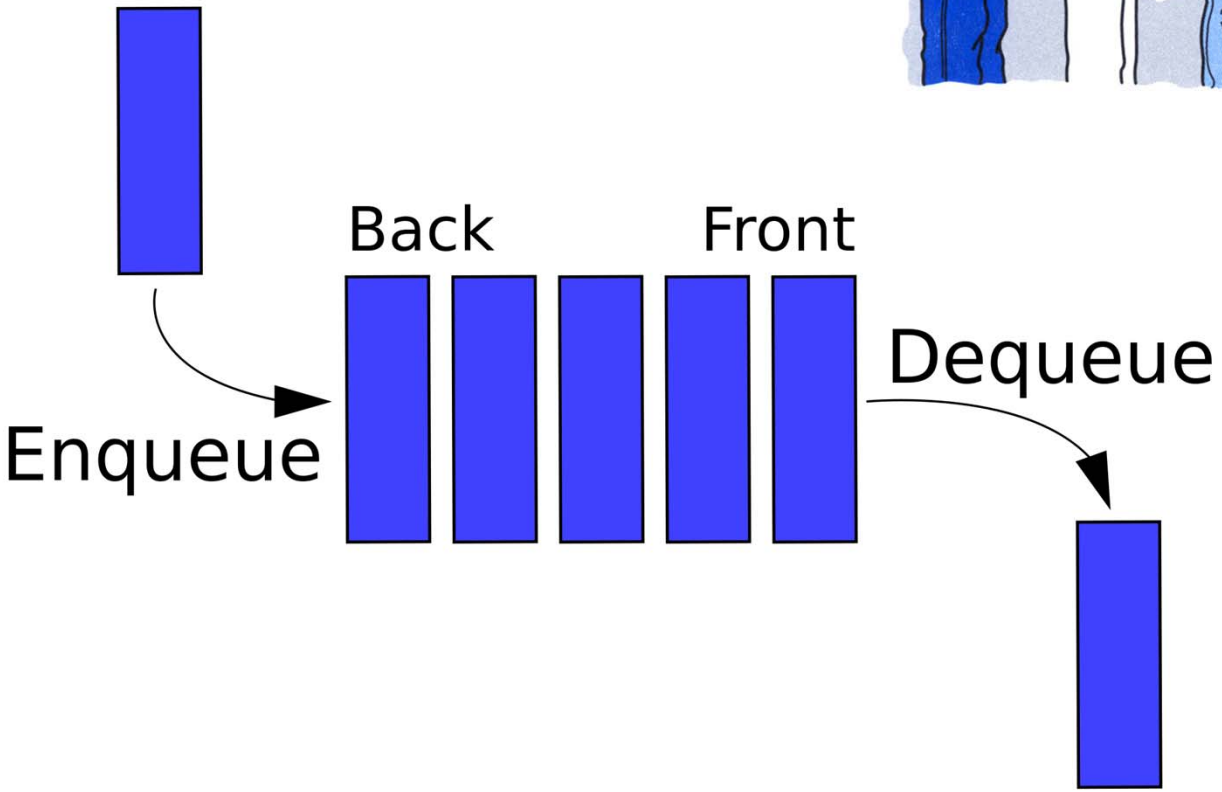
es-pila-vacía?(pila-vacía) = cierto

es-pila-vacía?(apilar(*e*, *p*)) = falso

fespecificación

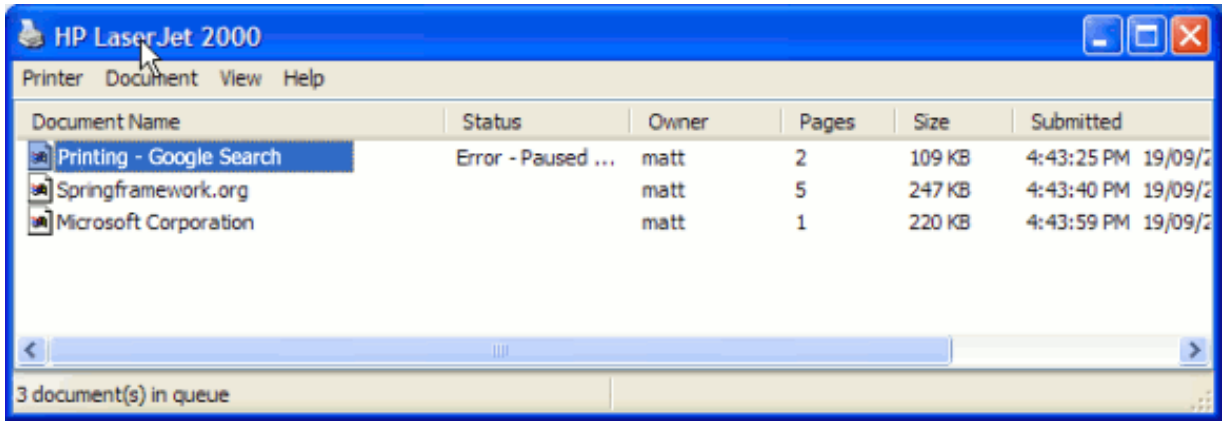
Cola (*queue*)

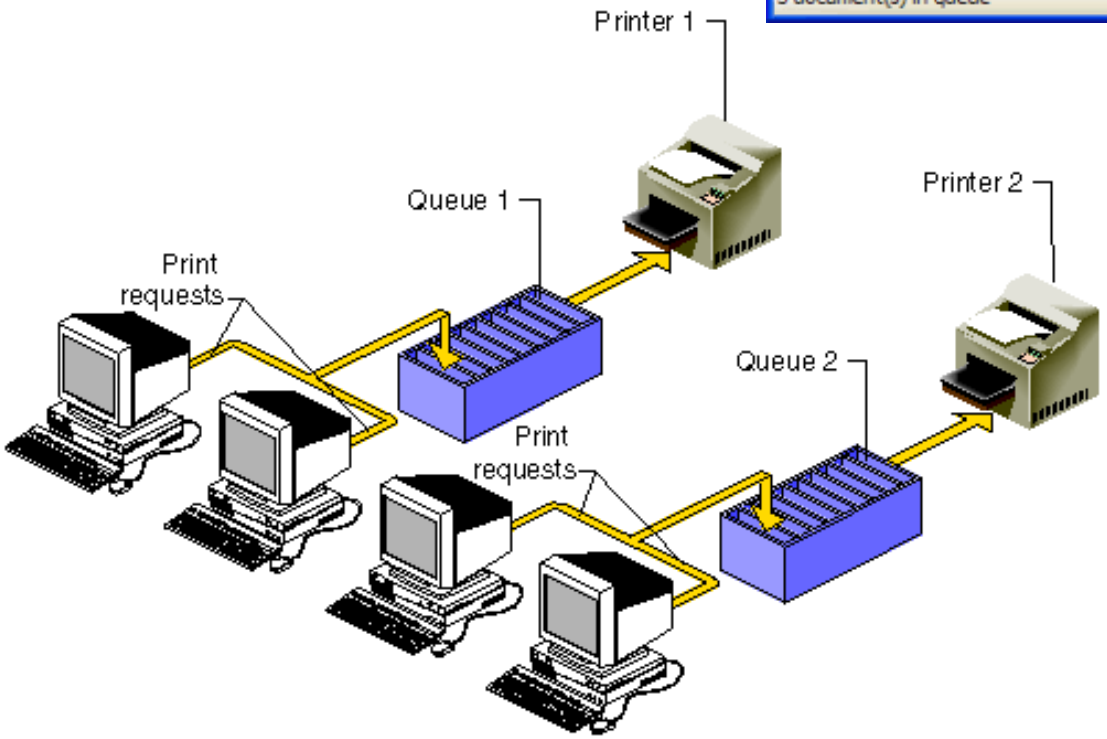

Rear of Queue
Front of Queue
next...
PAY HERE


Back
Front
Enqueue
Dequeue

Cola - Aplicaciones

- Cola de impresión





Cola - Operaciones básicas

- El TAD de las colas cuenta con las siguientes operaciones:
  - crear la cola vacía,
  - añadir un elemento al final de la cola,
  - eliminar el primer elemento en la cola,
  - consultar el primer elemento, y
  - determinar si la cola es vacía.

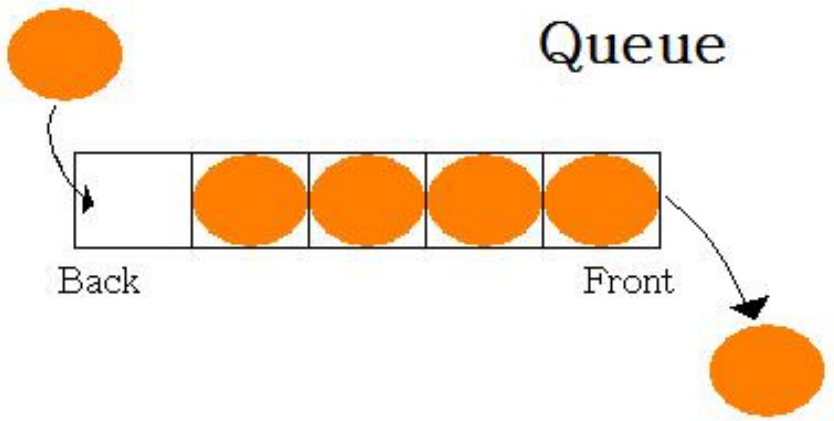
Cola - Especificación algebraica

especificación *COLAS*[*ELEM*]
usa *BOOLEANOS*
tipos *cola*
operaciones

cola-vacía	:		→ *cola*	{ constructora }
pedir-vez	:	*cola elemento*	→ *cola*	{ constructora }
avanzar	:	*cola*	$\rightarrow_p$ *cola*	
primero	:	*cola*	$\rightarrow_p$ *elemento*	
es-cola-vacía?	:	*cola*	→ *bool*	

- El orden de inserción de los elementos en la cola determina la cola, por lo que las constructoras son **libres**. No son necesarias ecuaciones de equivalencia.

Cola - Especificación algebraica



variables

e : elemento

c : cola

ecuaciones

avanzar(cola-vacía) = error

avanzar(pedir-vez(*c*, *e*)) = cola-vacía ⇐ es-cola-vacía?(*c*)

avanzar(pedir-vez(*c*, *e*)) = pedir-vez(avanzar(*c*), *e*) ⇐ ¬es-cola-vacía?(*c*)

primero(cola-vacía) = error

primero(pedir-vez(*c*, *e*)) = *e* ⇐ es-cola-vacía?(*c*)

primero(pedir-vez(*c*, *e*)) = primero(*c*) ⇐ ¬es-cola-vacía?(*c*)

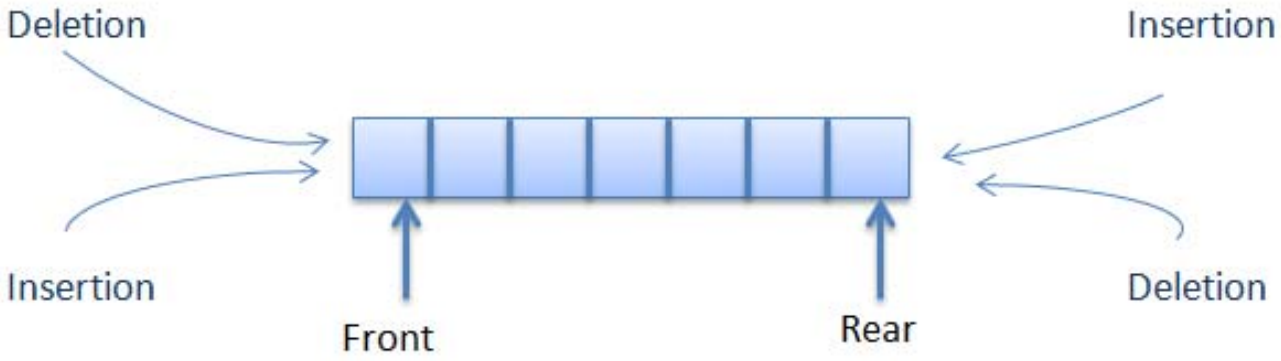
es-cola-vacía?(cola-vacía) = cierto

es-cola-vacía?(pedir-vez(*c*, *e*)) = falso

fespecificación

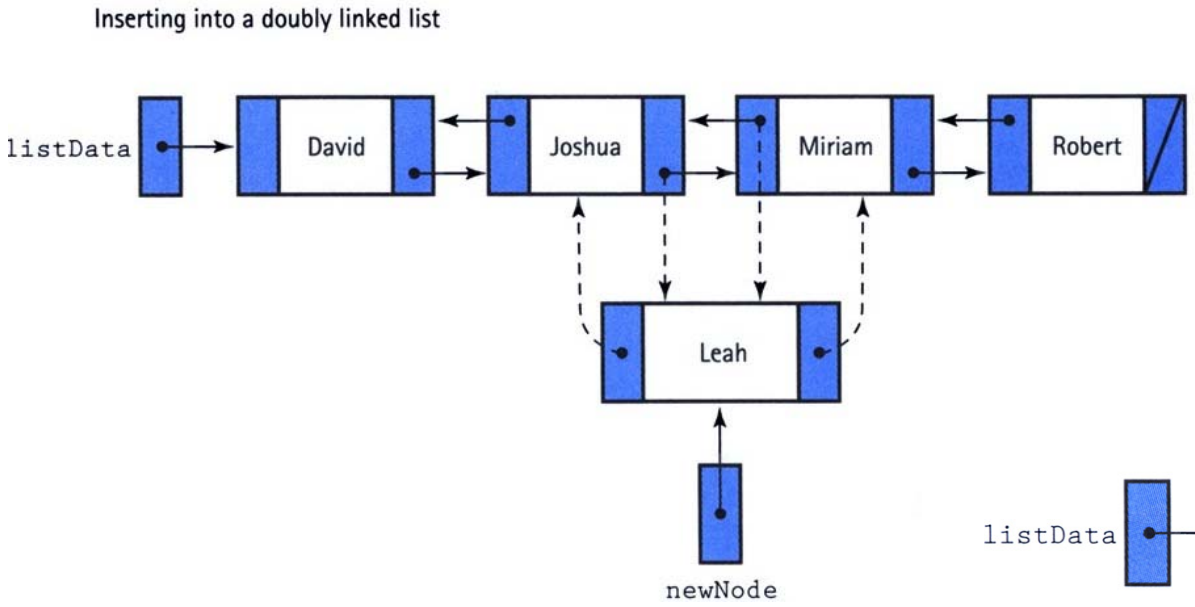
Lista (*List*)

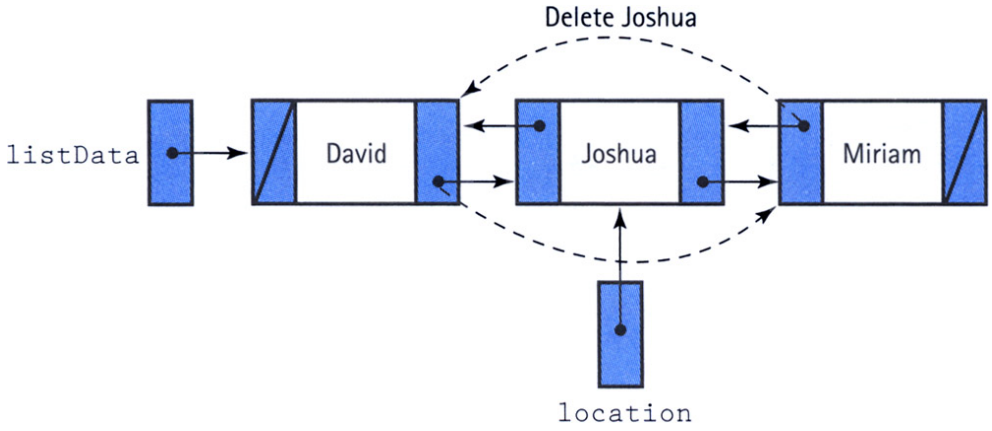

Bread
Milk
Eggs
Fruit
Rice
Pasta
Butter
Juice


Deletion
Insertion
Insertion
Deletion
Front
Rear

Lista - Aplicaciones

- Generalización de las pilas y las colas. Puede ser usada para implementar otras estructuras de datos.





Lista - Operaciones básicas

- Un posible TAD de las listas cuenta con las siguientes operaciones:
  - crear la lista vacía,
  - generar una lista unitaria formada por un elemento dado,
  - añadir un elemento por la izquierda,
  - añadir un elemento por la derecha,
  - consultar el elemento más a la izquierda,
  - consultar el elemento más a la derecha,
  - eliminar el elemento más a la izquierda,
  - eliminar el elemento más a la derecha,
  - determinar si una lista es vacía,
  - concatenar dos listas, y
  - calcular la longitud de una lista.

Lista - Especificación algebraica

especificación *LISTAS*[*ELEM*]
usa *BOOLEANOS, NATURALES*
tipos *lista*
operaciones

[ ]	:		→ *lista*	{ constructora }
_ : _	:	*elemento lista*	→ *lista*	{ constructora }
[_]	:	*elemento*	→ *lista*	
_ # _	:	*lista elemento*	→ *lista*	
izquierdo	:	*lista*	$\rightarrow_p$ *elemento*	
elim-izq	:	*lista*	$\rightarrow_p$ *lista*	
derecho	:	*lista*	$\rightarrow_p$ *elemento*	
elim-der	:	*lista*	$\rightarrow_p$ *lista*	
es-lista-vacía?	:	*lista*	→ *bool*	
_ ++ _	:	*lista lista*	→ *lista*	
longitud	:	*lista*	→ *nat*	

Lista - Especificación algebraica

variables

e, f : *elemento*

x, y, z : *lista*

ecuaciones

$[e] = e : [\,]$

$x \# e = x +\!+ [e]$

izquierdo([ ]) = error

izquierdo($e : x$) = e

elim-izq([ ]) = error

elim-izq($e : x$) = x

derecho([ ]) = error

derecho(e : [ ]) = e

derecho($e : f : x$) = derecho($f : x$)

Lista - Especificación algebraica

elim-der([ ]) = error
elim-der(e : [ ]) = [ ]
elim-der(e : f : x) = e : elim-der(f : x)

es-lista-vacía?([ ]) = cierto
es-lista-vacía?(e : x) = falso

[ ] ++ y = y
(e : x) ++ y = e : (x ++ y)

longitud([ ]) = 0
longitud(e : x) = 1 + longitud(x)

fespecificación

- También podemos considerar como constructoras la operación que crea la lista vacía y la que añade un elemento por la derecha (situación simétrica a la anterior); o bien elegir la lista vacía, la operación de construcción de la lista unitaria y la operación de concatenación.

Bibliografía

- Martí, N., Ortega, Y., Verdejo, J.A. *Estructuras de datos y métodos algorítmicos*. Ejercicios resueltos. Pearson/Prentice Hall, 2003. Capítulos 1, 3, 4, 5

- Peña, R.; *Diseño de programas. Formalismo y abstracción*. Tercera edición. Prentice Hall, 2005. Capítulos 5, 6