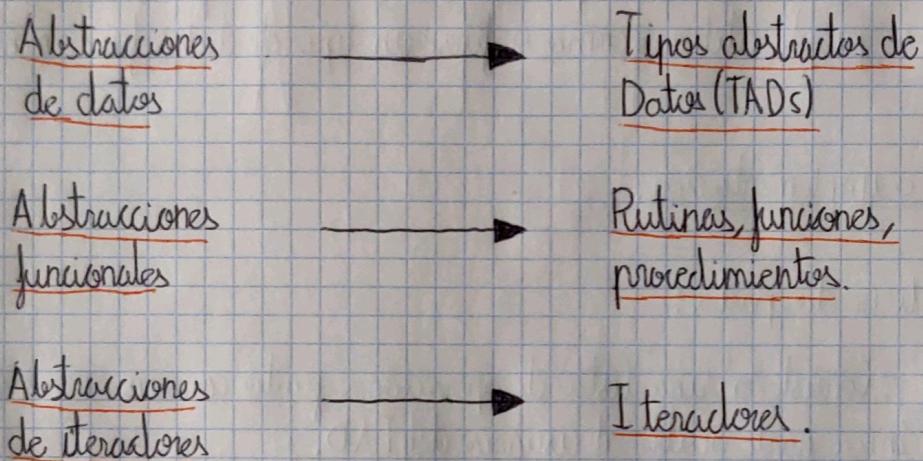


Tema 2. Abstracciones y Especificaciones.

① Mecanismos de abstracción.

- Abstracción por especificación: Sólo necesitamos conocer qué va a hacer el procedimiento y no cómo funciona.
- Abstracción por parametrización: Un algoritmo, un tipo, o una variable se definen en base a unos parámetros.

② Tipos de abstracciones.



③ Concepto de TAD.

Se llama abstracción de datos o Tipo Abstracto de Datos (T.A.D.) al conjunto de valores que pueden tomar los datos de ese tipo, juntamente con las operaciones que los manipulan que se definen mediante una especificación que es independiente de cualquier implementación.

- Tipo Abstracto de Datos: dominio abstracto de valores junto con las operaciones aplicables sobre el mismo.
- Tipo de Datos: conjunto de valores que puede tomar una variable, un parámetro o una expresión.
- Estructura de Datos: disposición en memoria de los datos necesarios para almacenar valores de un tipo.

② Ventajas de la abstracción funcional.

- Generalización del concepto de operador.
- Encapsulación.

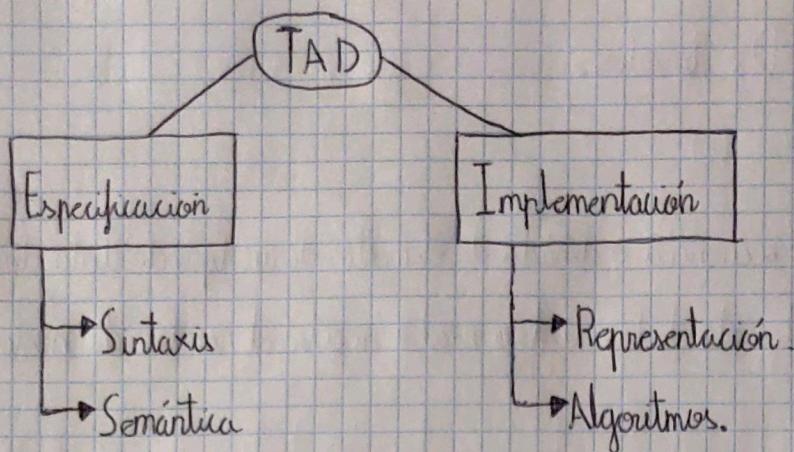
③ Ventajas del TAD.

- Generalizar los tipos de datos fundamentales.
- Encapsular cierto tipo de datos.
- Fuera de la sección en la que está codificado el tipo abstracto de dato, éste se puede utilizar como si fuese un tipo de datos fundamental.

④ Interfaz e implementación.

El TAD tiene dos partes:

- Interfaz: Consiste en una lista de operaciones junto con sus argumentos, y es la única parte visible al usuario del TAD.
- Implementación: Es el lugar en donde se aclara cómo realizar las operaciones del TAD, es por ello que sólo es visible al diseñador y al programador.



- La parte de Especificación define cómo se puede utilizar el tipo de dato y está formada por la sintaxis y la semántica.
- La parte de implementación define una posible realización del tipo de dato y está formada por la representación y los algoritmos correspondientes a las operaciones en términos de la misma.
- La Especificación de un TAD consiste en establecer las propiedades que lo definen. Sirve al usuario para conocer detalladamente el comportamiento del tipo.

Debe cumplir 4 propiedades:

- Ser precisa.
- Ser general.
- Ser legible
- No ambigua.

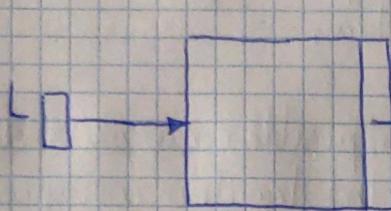
- Tema 3. TADs lineales.

④ TAD Lista.

- Def: Es una secuencia ordenada de elementos de un tipo de dato dado.

- Cada elemento de las listas ~~recibe~~ recibe el nombre de nodo.

↳ (lista vacía)



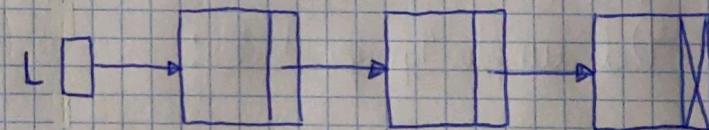
→ Denota el último elemento de la lista (puntero nulo)

(lista con dos elementos)

→ Puntero al elemento siguiente.

④ TAD lista: algunos tipos de lista.

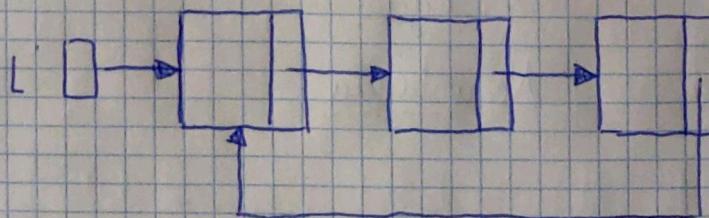
Simplemente enlazada



Doblemente enlazada



Circular



④ TAD Cola

- Estructura FIFO (First-in First-out)

- El primer elemento en entrar es el primero en salir.

- Se corresponde con el concepto natural de cola al que estamos acostumbrados.

④ TAD Pila

- Estructura LIFO (Last-In First Out)

- El último elemento en entrar es el primero elemento en salir.

- Los elementos se sacan en Orden Inverso a como se han introducido.

~~LIFO~~

④ TAD Cola: especificación

Definición: cola de elementos de un TipoBase dado.

Sintaxis:

initializarCola (e/s cola c);

encolar (e/s cola c, e/. elemento e);

desencolar (e/s cola c);

primero (e/. cola c);

vaciaCola (e/. cola c) devuelve {V, F}

- Las operaciones características que definirán el TAD cola:

- inicializarCola(c): crea c como una cola vacía.
- encolar(c, e): inserta en la última posición disponible de la cola c un elemento e.
- desencolar(c): elimina el primer elemento de la cola c.
- primero(c): devuelve el primer elemento de la cola c, pero sin desencolar.
- vaciaCola(c): devuelve verdadero si hay elementos en cola y falso en caso contrario.

④ TAD Pila: especificación.

Definición: pila de elementos de un TipoBase dado.

Sintaxis:

inicializar(e/s pila p);
apilar(e/s pila p, el. elemento e);
desapilar(e/s pila p) devuelve elemento e;
top(e/s pila p) devuelve elemento e;
vaciaPila(e/s pila p) devuelve {V, F}

Semántica:

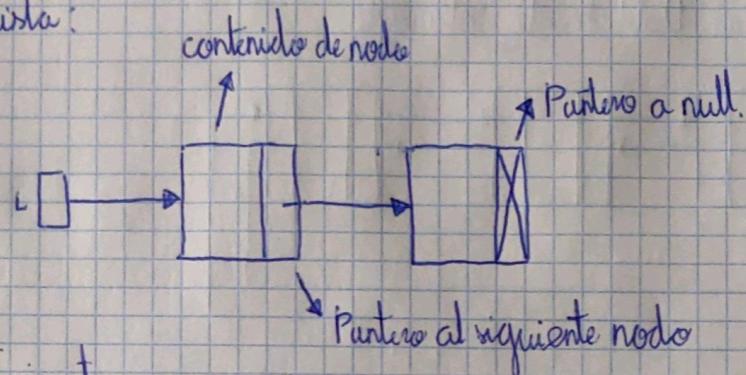
Examen 16-17

- Explica en que condiciones es mejor implementar las listas con arrays y cuando es mejor usar memoria dinámica.

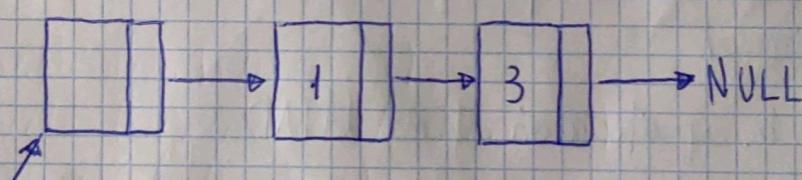
Es mejor utilizar una memoria dinámica cuando no se sabe el número de elementos a tratar.

- ¿En qué se diferencian fundamentalmente el TAD Lista del TAD conjunto? Pon un ejemplo de cada uno.

- Lista:



- Conjunto:



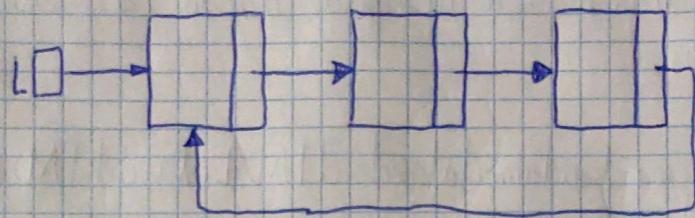
Primeros

Cardinalidad = 2

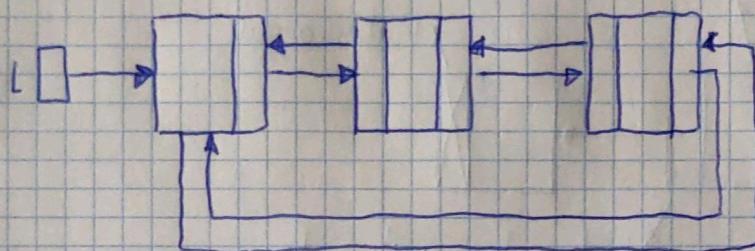
Tiene un puntero, que apunta al primer elemento de la lista de nodos, ese nodo, estaría vacío, tendremos un segundo atributo llamado cardinalidad, que nos dice el número de elementos que forman parte del conjunto.

- Pon un ejemplo real de uso de lista Circular y de lista Circular Dblemente Enlazada.

- Lista Circular.



1) Lista Circular, Dblemente enlazada.



- Sea el TAD Cola, acepta las funciones: CrearCola, Vaciar, Encolar, BuscarElem, ImprimirOrdenContrario (que imprime la cola desde el último elemento hasta el primero).

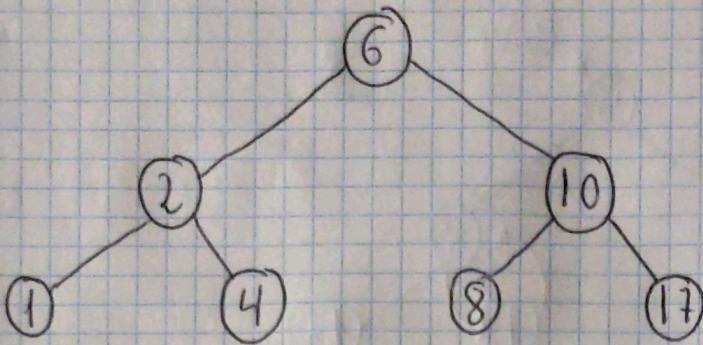
a) Escribe la especificación informal del TAD.

b/ b.1/ el pseudocódigo de la función ImprimirOrdenContrario.

b.2/ el pseudocódigo de como se llamaría dicha función desde el interfaz del TAD.

c) Escribe la semántica de la especificación FORMAL (constructiva), SOLO para dicha función ImprimirOrdenContrario.

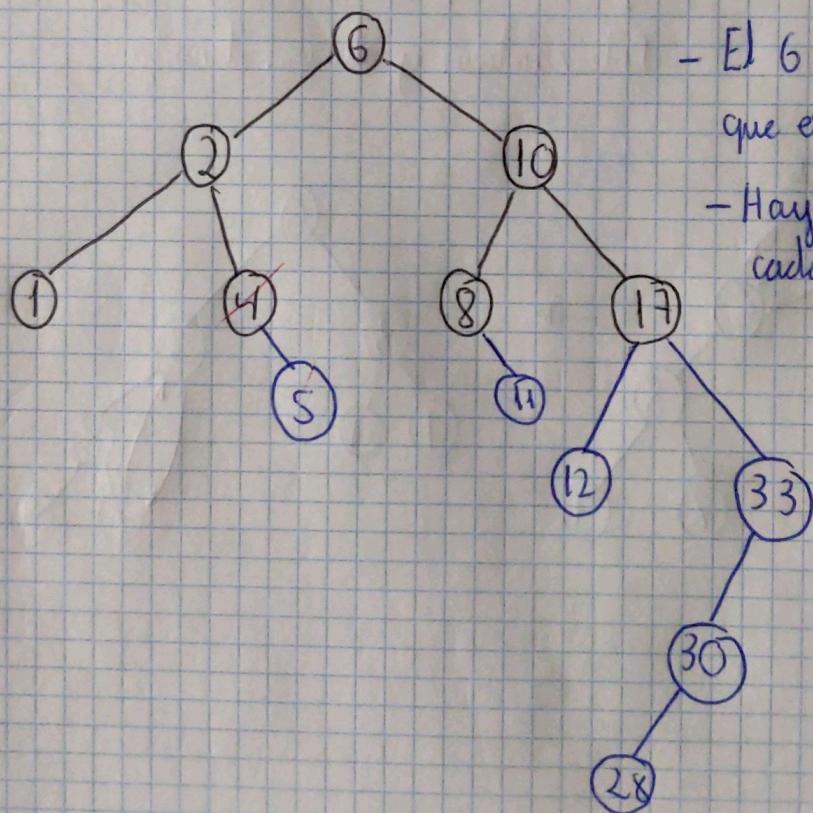
- Supongamos que tenemos el siguiente ABB.



a) Dibujar paso a paso cómo quedaría el árbol tras insertar en el siguiente orden los valores clave: 12, 6, 11, 5, 33, 28.

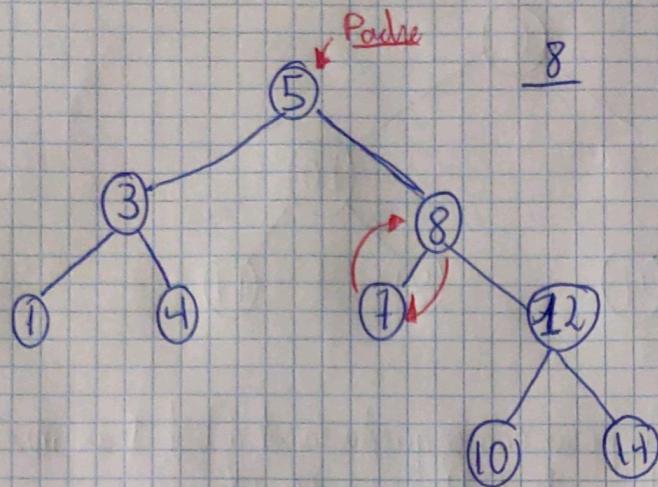
c) Dibujar paso a paso cómo quedaría el árbol tras eliminar en el siguiente orden los valores: 4, 17, 8, 2, 33, 28.

a)

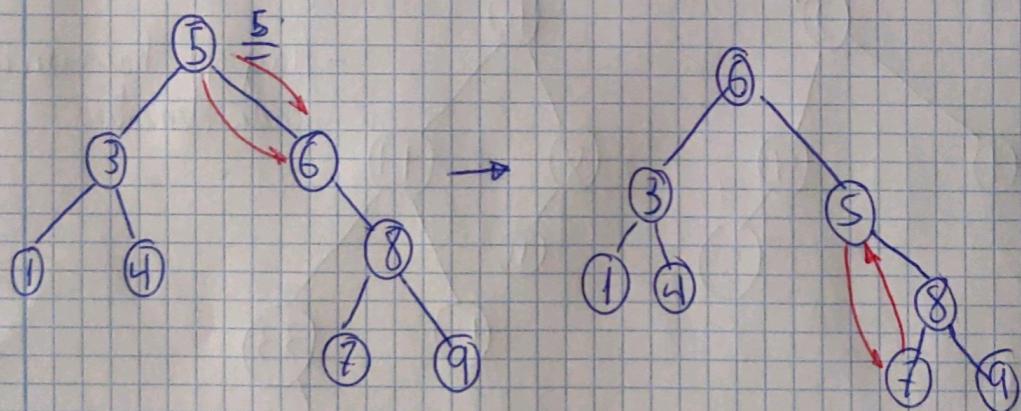


- El 6 no se puede insertar, ya que está en el árbol.
- Hay que hacer un árbol por cada elemento.

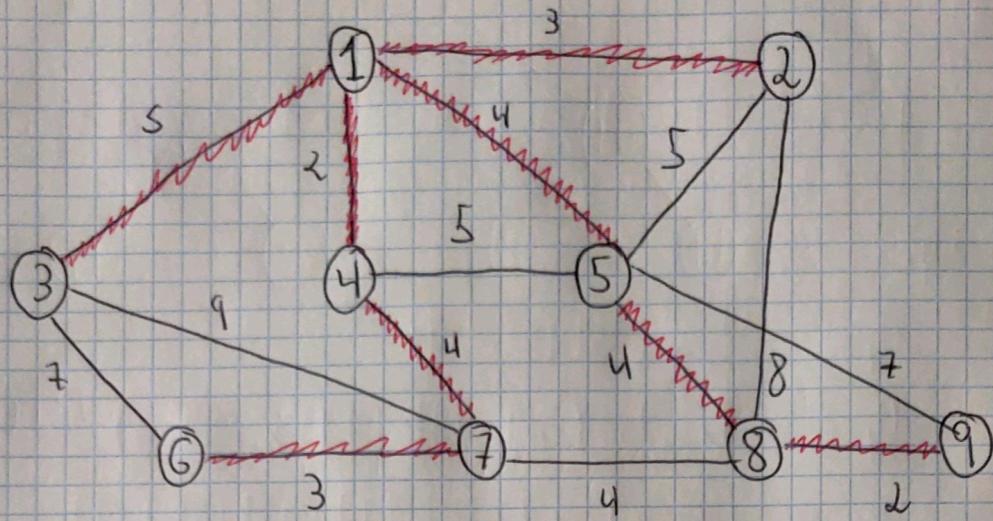
c) Si el nodo es hoja, lo eliminamos directamente.



- Comprobar si la rama subizq y subder tienen elementos.
- Substituir el 8 por el nodo mas grande del subárbol izq o el mas pequeño del subárbol derecho.
- Intercambiar y eliminar el nodo.



- Encontrar el árbol de expansión mínima del siguiente grafo ponderado.



- Prim o Kruskal. (Explique paso a paso el algoritmo de prim).

- Dado el siguiente grafo, encontrar el camino más corto ~~entre todos~~ desde el nodo 1 hasta el resto de nodos. ¿Cuál es la distancia mínima al nodo 3 y que ruta debe seguirse? (Algoritmo de dijkstra).

⊗ Especificación informal.

Abstracción de datos. Notación.

TAD <nombre_tipo> es <lista_operaciones>

Descripción

Descripción textual del tipo.

Operaciones

Especificación informal de las operaciones de la lista anterior.

Fin <nombre_tipo>

⊗ TAD ListaEnteros es Crear, Insertar, Primero, Último, Cabecera, Cola, EsVacio, Igual.

Descripción.

Las listaEnteros son listas de enteros modificables. las listas se crean con las operaciones Crear e Insertar...

Operaciones.

Operación Crear (sal ListaEnteros)

Calcula: Devuelve una lista de enteros vacía.

Operación Insertar (ent I: ListaEnteros; x: entero)

Modifica: I.

Calcula: Añade x a la lista I en la primera posición.

Fin ListaEnteros.

④ TAD Lista[T] es Crear, Insertar, Primero, Último, Cabecera, Cola, EsVacio, Igual.

Descripción.

Las Lista[T] son listas modificables de valores de tipo T. las listas se crean con las operaciones Crear e Insertar.

Operaciones.

Operación Crear (sal Lista[T]) . . .

Operación Insertar (en I : Lista[T]; x: entero) . . .

Operación Primero (ent I : Lista[T]; sal Lista[T])

Fin Lista:

⑤ Especificación formal.

Método Algebraico o Axiomático.

⑥ TAD Natural de los ~~enteros~~ números naturales.

Nombre

Natural

Conjuntos

N Conjunto de naturales

B Conjunto de Booleanos {true, false}

Sintaxis

Cero : $\rightarrow N$

Sucesor : $N \rightarrow N$

esCero : $N \rightarrow B$

Igual : $N \times N \rightarrow B$

suma : $N \times N \rightarrow N$

Semántica.

$\forall m, n \in \mathbb{N}$

1. $\text{esCero}(\text{cero}) = \text{true}$
2. $\text{esCero}(\text{sucesor}(n)) = \text{false}$
3. $\text{igual}(\text{cero}, n) = \text{esCero}(n)$
4. $\text{igual}(\text{sucesor}(n), \text{cero}) = \text{false}$
5. $\text{igual}(\text{sucesor}(n), \text{sucesor}(m)) = \text{igual}(n, m)$
6. $\text{suma}(\text{cero}, n) = n$.
7. $\text{suma}(\text{sucesor}(m), n) = \text{sucesor}(\text{suma}(m, n))$.



TAD genérico pila.

Nombre

Pila[T]

Conjuntos

S Conjunto de pilas.

T Conjunto de elementos que pueden ser almacenados

B Conjunto de booleanos { true, false }

Sintaxis.

CrearPila : $\rightarrow S$

estVacia : $S \rightarrow B$

pop : $S \rightarrow S$

tope : $S \rightarrow T$

push : $T \times S \rightarrow S$

Semántica.

$\forall t \in T; \forall s \in S$

1. $\text{esVacia}(\text{crearPila}) = \text{true}.$
2. $\text{esVacia}(\text{push}(t, s)) = \text{false}.$
3. $\text{pop}(\text{crearPila}) = \text{crearPila}.$
4. $\text{pop}(\text{push}(t, s)) = s$
5. $\text{top}(\text{crearPila}) = \text{"Error, la pila está vacía".}$
6. $\text{top}(\text{push}(t, s)) = t.$

Método constructivo u operacional.

④ Nombre

Lista [I]

Conjuntas

L	Conjunto de listas.
I	Conjunto de elementos.
B	Conjunto de booleanos { true, false }
N	Conjunto de naturales.
M	Conjunto de mensajes { "La lista está vacía" }

Sintaxis

crearlista:

$\rightarrow L$

formarLista:

$I \rightarrow L$

concatenar:

$L \times L \rightarrow L$

último:

$L \rightarrow I \cup M$

cabecera:

$L \rightarrow L$

primero:

$L \rightarrow I \cup M$

cola:

$L \rightarrow L$

esListaVacia:

$L \rightarrow B$

Semántica.

$\forall i \in I; \forall a, b \in L$

1. $\text{ultimo}(\text{crearLista}) = \text{"La lista está vacía."}$

2. $\text{ultimo}(\text{formarLista}(i)) = i$

3. $\text{ultimo}(\text{concatenar}(a, b)) = \text{SI } \text{esListaVacia}(b) \rightarrow \text{ultimo}(a) \mid \text{ultimo}(b).$

4. $\text{cabecera}(\text{crearLista}) = \text{crearLista}.$

5. $\text{cabecera}(\text{formarLista}(i)) = \text{crearLista}.$

6. $\text{cabecera}(\text{concatenar}(a, b)) = \text{SI } \text{esListaVacia}(b) \rightarrow \text{cabecera}(a)$
 $\mid \text{concatenar}(a, \text{cabecera}(b)).$

7. $\text{primero}(\text{crearLista}) = \text{"La lista está vacía."}$

8. $\text{primero}(\text{formarLista}(i)) = i$

9. $\text{primero}(\text{concatenar}(a, b)) = \text{Si } \text{esListaVacia}(a) \rightarrow \text{primero}(b) \quad \cancel{\mid \text{primero}(a)}$

*

Nombre

Pila [I]

Conjuntos

S Conjunto de pilas.

I Conjunto de elementos.

B Conjunto de valores booleanos { true, false }

Sintaxis.

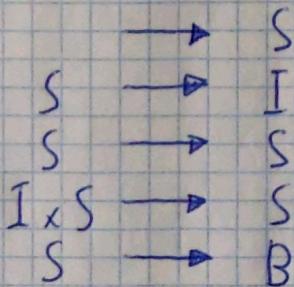
crearPila:

topo :

pop :

push:

esVaciaPila



Semantica.

$\forall i, t \in I; \forall s, n, p \in S; \forall b \in B$

1. $\text{pre-crearPila}() ::= \text{true}$
2. $\text{post-crearPila}(s) ::= s = \text{crearLista}.$
3. $\text{pre-tipe}(s) ::= \text{NOT esListaVacia}(s)$
4. $\text{post-tipe}(s; t) ::= t = \text{primero}(s)$
5. $\text{pre-pop}(s) ::= \text{NOT esListaVacia}(s)$
6. $\text{post-pop}(s; p) ::= p = \cancel{\text{ultimo}} \text{p} = \text{ultimo}(s)$
7. $\text{pre-push}(i, s) ::= \text{true}.$
8. $\text{post-push}(i, s, n) ::= n = \text{concatenar}(\text{formarLista}(i), s)$
9. $\text{pre-esVaciaPila}(s) ::= \text{true}.$
10. $\text{post-esVaciaPila}(s; b) ::= b = \text{esListaVacia}(s)$

- Operaciones características que definirán el TAD pila.
- inicializarPila(p): crea p como una pila vacía.
- apilar(p, e): inserta en la cima (top) de la pila (p) un elemento e (push).
- desapilar(p): elimina el elemento top de la pila p (pop)
- top(p): devuelve el elemento top de la pila p, pero sin desapilar.
- vaciaPila(p): devuelve verdadero si hay elementos apilados y falso en caso contrario.