

4. Pilas

- Una Pila es una colección de elementos homogéneos dispuestos en orden tal que se recuperan en orden inverso a como se introdujeron.
- La extracción e inserción de elementos en la Pila se realiza por la parte superior
- El único elemento accesible es el último.
- LIFO (Last In, First Out)

4. Pilas

- Una pila viene parametrizada por el tipo de elemento que alberga.
- `TipoElemento`: parámetro genérico del `TipoPila`
- ¿Qué operaciones definimos para el TAD Pila?
- Posibles operaciones serán:
 - `CrearVacia`, `Apilar`, `Desapilar`, `EsPilaVacia`, `Cima`
- Especificación algebraica:

4. Pilas: Especificación

ESPECIFICACION Pilas

PARÁMETROS GENÉRICOS

TipoElemento

FIN PARÁMETROS GENÉRICOS

TIPOS TipoPila

OPERACIONES

(* CONSTRUCTORAS GENERADORAS *)

CrearPilaVacía: \rightarrow TipoPila

Apilar: TipoElemento \times TipoPila \rightarrow TipoPila

(* OBSERVADORAS SELECTORAS *)

PARCIAL Cima : TipoPila \rightarrow TipoElemento

PARCIAL Desapilar : TipoPila \rightarrow TipoPila

(* OBSERVADORAS NO SELECTORAS *)

EsPilaVacía : TipoPila \rightarrow Booleano

4. Pilas: Especificación

(* CONSTRUCTORAS NO GENERADORAS *)

Copiar : TipoPila \rightarrow TipoPila

Destruir: TipoPila \rightarrow TipoPila

VARIABLES

pila: TipoPila;

e: TipoElemento;

ECUACIONES DE DEFINITUD

DEF(Cima(Apilar(e, pila))

DEF(Desapilar(Apilar(e, pila))

4. Pilas: Especificación

ECUACIONES

(* OBSERVADORAS SELECTORAS *)

Cima (Apilar(e, pila)) = e

Desapilar(Apilar(e, pila)) = pila

(* OBSERVADORAS NO SELECTORAS *)

EsPilaVacía (CrearPilaVacía) = CIERTO

EsPilaVacía (Apilar(e, pila)) = FALSO

4. Pilas: Especificación

(* CONSTRUCTORAS NO GENERADORAS *)

Copiar (CrearPilaVacía) = CrearPilaVacía

Copiar (Apilar(e, pila)) = Apilar(e, Copiar(pila))

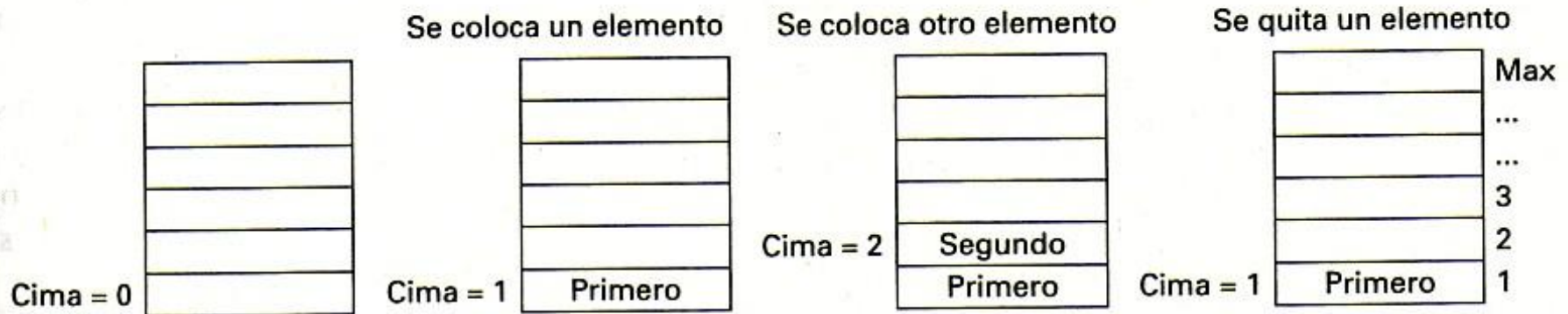
Destruir(CrearPilaVacía) = CrearPilaVacía

Destruir(Apilar(e,pila)) = Destruir(pila)

FIN ESPECIFICACIÓN

4. Pilas: Implementación

Implementación mediante estructuras estáticas.



4. Pilas: Implementación

Implementación mediante estructuras dinámicas.

TYPE

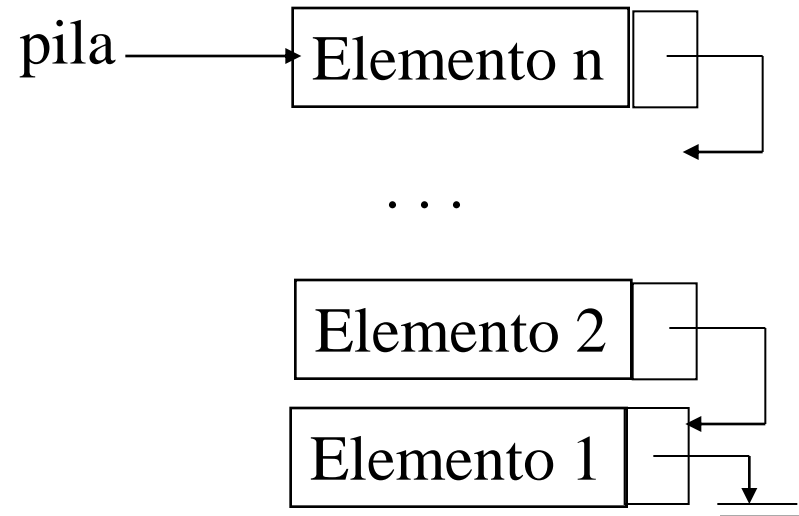
```
TipoPila = ^TipoNode
```

```
TipoNode = RECORD
```

```
    info: TipoElemento;
```

```
    ant: TipoPila;
```

```
END;
```



4. Pilas: Implementación

(* OPERACIONES CONSTRUCTORAS GENERADORAS *)

PROCEDURE CrearPilaVacía(**VAR** pila: TipoPila);

{PRE: Cierto}

{POST: La pila esta vacía o se destruye}

PROCEDURE Apilar(elem: TipoElemento; **VAR** pila: TipoPila);

{PRE: Hay espacio para almacenar elemento en la lista}

{POST: "elemento" queda almacenado en la cima de la "pila"}

{EXCEPCION: Si la pila esta llena no se inserta el elemento}

(* OPERACIONES OBSERVADORAS SELECTORAS *)

PROCEDURE Cima(**VAR** elem: TipoElemento; pila: TipoPila);

{PRE: "lista" no está vacía}

{POST: devuelve el elemento que contiene la cima de la "pila"}

{EXCEPCION: Si la pila esta vacía no devuelve nada}

4. Pilas: Implementación

```
PROCEDURE Desapilar(VAR pila: TipoPila);
```

```
{PRE: La pila no esta vacía}
```

```
{POST: Elimina la cima de la pila}
```

```
{EXCEPCIÓN: "Pila Vacía"}
```

```
(* OPERACIONES OBSERVADORAS NO SELECTORAS *)
```

```
FUNCTION EsPilaVacía(pila: TipoPila): Boolean;
```

```
{PRE: Cierto}
```

```
{POST: devuelve TRUE si "pila" esta vacía}
```

```
(* OPERACIONES CONSTRUCTURAS NO GENERADORAS *)
```

```
PROCEDURE Copiar(pila1: TipoPila; VAR pila2: TipoPila);
```

```
{PRE: Queda memoria para almacenar "pila2"}
```

```
{POST: Devuelve en "pila2" una copia de "pila1"}
```

```
{EXCEPCIÓN: "Memoria Agotada"}
```

4. Pilas: Implementación

```
PROCEDURE Destruir(VAR pila: TipoPila);  
  {PRE: Cierto}  
  {POST: Elimina los nodos de "pila"}
```

IMPLEMENTATION

```
{*****}  
  
  (* OPERACIONES CONSTRUCTORAS GENERADORAS *)  
  
  PROCEDURE CrearPilaVacía(VAR pila: TipoPila); {O(1)}  
  BEGIN  
    pila := NIL  
  END;
```

4. Pilas: Implementación

```
PROCEDURE Apilar(elem: TipoElemento; VAR pila:  
TipoPila); {O(1)}  
  
  VAR  
    auxPNodo: TipoPila;  
  
  BEGIN  
    new(auxPNodo);  
    auxPNodo^.ant := pila;  
    Asignar(auxPNodo^.info, elem);  
    pila := auxPNodo;  
  
  END;
```

4. Pilas: Implementación

```
(* OPERACIONES OBSERVADORAS SELECTORAS *)  
PROCEDURE Cima(VAR elem: TipoElemento; pila: TipoPila); {O(1)}  
BEGIN  
    IF NOT EsPilaVacía(pila) THEN  
        Asignar(elem,pila^.info)  
END;  
  
PROCEDURE Desapilar(VAR pila: TipoPila); {O(1)}  
VAR  
    auxPNodo: TipoPila;  
BEGIN  
    IF NOT EsPilaVacía(pila) THEN BEGIN  
        auxPNodo := pila;  
        pila := pila^.ant;  
        dispose(auxPNodo)  
    END;  
END;
```

4. Pilas: Implementación

```
(* OPERACIONES OBSERVADORAS NO SELECTORAS *)  
  
FUNCTION EspilaVacía(pila: TipoPila): Boolean;  
{O(1)}  
  
BEGIN  
  
    EspilaVacía := (pila = NIL);  
  
END;
```

4. Pilas: Implementación

```
(* OPERACIONES CONSTRUCTORAS NO GENERADORAS *)  
  
PROCEDURE Copiar(pila1: TipoPila; VAR pila2: TipoPila);  
  {O(n)}  
  
VAR  
  auxPNode1, auxPNode2, auxPNodePre2: TipoPila;  
  
BEGIN  
  IF NOT EsPilaVacía(pila1) THEN BEGIN  
    {  
      auxPNode1 := pila1;  
      new(auxPNode2);  
      Asignar(auxPNode2^.info, auxPNode1^.info);  
      auxPNode2^.ant := NIL;  
      pila2 := auxPNode2;  
    } CrearCabecera(...) } CrearNode(...)
```

4. Pilas: Implementación

```
auxPNode1 := auxPNode1^.ant; {Avance en pila1}
```

```
WHILE (auxPNode1 <> NIL) DO BEGIN
```

```
    auxPNodePre2 := auxPNode2;
```

```
    new(auxPNode2);
```

```
    Asignar(auxPNode2^.info, auxPNode1^.info);
```

```
    auxPNode2^.ant := NIL;
```

} CrearNodo(...)

```
    auxPNodePre2^.ant := auxPNode2;
```

```
    auxPNode1 := auxPNode1^.ant
```

```
END {WHILE}
```

```
END {IF}
```

```
END;
```

Hace de extremo de la pila

4. Pilas: Implementación

```
PROCEDURE Destruir(VAR pila: TipoPila); { O(n) }  
BEGIN  
    WHILE NOT EsPilaVacía(pila) DO  
        Desapilar(pila);  
END;  
END.
```