TDA PILA

Ing. Mario Milton López Winnipeg

3 TDA Pila

3 Pilas

3.1 Descripción del TDA Pila.

3.2 Especificación del TDA Pila.

3.3 Ejemplos de uso.

3.4 Implementaciones del TDA Pila.

- 3.4.1 Implementación basada en el TDA Lista.
- 3.4.2 Implementación con vectores.
- 3.4.3 Implementación con apuntadores.

3.1 Descripción del TDA Pila

- Una pila es un caso especial de lista en la cual todas las inserciones y supresiones tienen lugar en un extremo determinado llamado *tope*.

- A las pilas se les llama también listas *LIFO (last-in first-out)* o listas "ultimo en entrar, primero en salir".

- En el TDA Pila no se definen operaciones de posicionamiento en la pila. Esto es debido a que todas las operaciones de acceso se realizan en la misma posición, el tope de la pila.

3.1 Descripción del TDA Pila

Un TDA de la familia pila incluye a menudo las cinco operaciones siguientes:

- CREA. Crea una pila vacía.
- VACIA. Devuelve un valor cierto si la pila está vacía, y falso en caso contrario.
- CIMA. Devuelve el elemento situado el tope de la pila, sin extraerlo.
- METER. Añade un elemento a la pila, quedando éste situado en el tope.
- SACAR. Suprime el elemento situado en el tope de la pila.

3.2 Especificación del TDA Pila

Especificación informal del TDA `Pila`

- **`Pila` = TDA con operaciones `crea`, `vacia`, `cima`, `meter`, `sacar`.**
- **DESCRIPCIÓN:**
  - **Los valores del TDA `Pila` son pilas de elementos del tipo `Elemento`. Las pilas son mutables: `meter` y `sacar` añaden y eliminan elementos en la pila respectivamente.**
- **OPERACIONES:**
  - **`crea(P Pila) devuelve`**
    - **efecto: Devuelve la pila vacía**

3.2 Especificación del TDA Pila

- `vacia( P : Pila ) devuelve (booleano)`
  - efecto: Devuelve cierto si la pila vacía, y falso en caso contrario.
- `cima(P : Pila) devuelve (E:Elemento)`
  - requerimientos: La pila es no vacía.
  - efecto: Devuelve en E el elemento situado en el tope de la pila
- `meter( P: Pila, E:Elemento)`
  - Modifica : Pila
  - efecto: Añade el elemento E a la pila, quedando éste situado en el tope.
- `sacar(P: Pila ,E :Elemento)`
  - Requerimientos: La pila es no vacía.
  - Modifica : Pila
  - Efecto: Suprime el elemento situado en el tope de la pila y lo retorna

3.2 Especificación del TDA Pila

Especificación Formal

- *Tipo:* `Pila (Elemento)`
- *Sintaxis:*
  - `crea` → `Pila`
  - `vacia(Pila)` → `booleano`
  - `cima(Pila)` → `Elemento`
  - `meter(Pila, Elemento)` → `Pila`
  - `Sacar(Pila, ES Elemento)` → `Pila`

3.3 Ejemplos de Uso

```
publico void imprimir(ES Pila pila)
inicio
        Pila pilaAux;
        mientras no (pila.vacia())
         inicio
            pila.sacar(e);
            pilaAux.meter(e);
            mostrar( e );
         fin
        mientras !(pilaaux.vacia())
         inicio
           pilaaux.sacar(e);
           pila.meter(e);
         fin
fin
```

3.4 Implementaciones del TDA Pila

En esta sección mostraremos tres implementaciones alternativas para el TDA Pila:

1. Implementación basada en el TDA Lista; consiste en definir una Pila utilizando una lista
2. Implementación con vectores; utiliza un array para almacenar los elementos de la Pila
3. Implementación con apuntadores con representación con simple enlace

3.4.1 Implementación basada en el TDA Lista

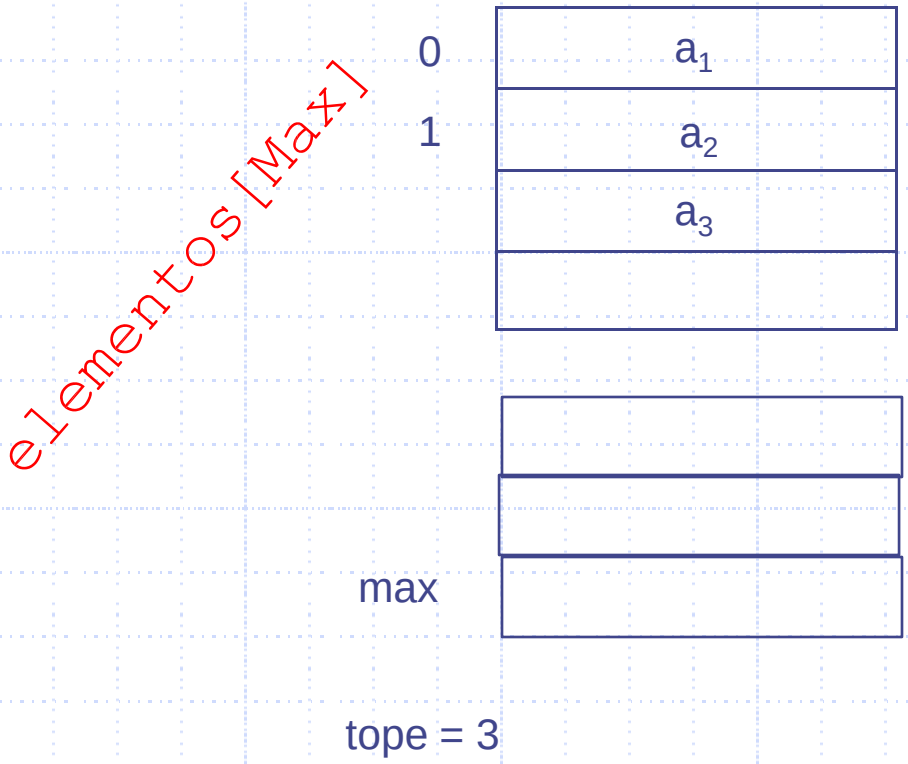
Definición básica de la clase Lista con representación contigua

```
Clase Pila
    Atributos
        L : Lista
    Metodos
       Crear()
       Vacia()
       Meter(E : Elemento)
       Sacar( ES E: Elemento)
       cima()
Fin
Constructor Pila.Crear
 inicio
     // Crear Objeto L
  fin
Pila.meter( E: Elemento)
inicio
    l.inserta(l.primero(),E);

Fin
```

```
Pila.sacar( ES E: Elemento)
Inicio
          l.recupera(l.primero(),E)
          l.suprime(l.primero)
Fin
```

4.4.2 Implementación con vectores

Representación de una pila mediante un vector



3.4.2 Implementación con vectores

Definición básica de la clase Lista con representación contigua:

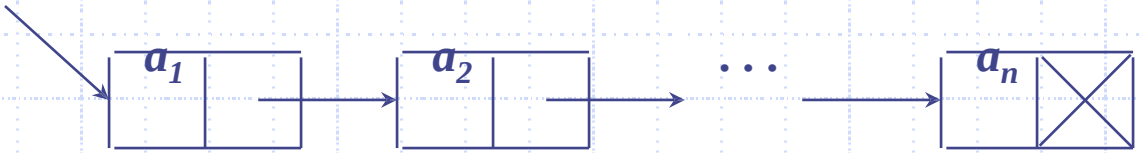
```
Constante max = 100
Tipo de Datos
            Direccion de tipo Entero

Clase Pila
 Atributos
      elementos[Max] vector de tipo TipoElemento
      Tope de tipo Direccion
 Metodos
       …… … … … …
Fin
Pila.Crear
inicio
      tope = 0
fin
```

3.4.2 Implementación con vectores

```
Pila.vacia()
inicio
   retornar (tope = 0)
fin
Pila.meter ( E : Elemento)
inicio
   si tope<MAX entoces
                    tope= tope +1
                    elementos[ tope ] = E
fin
Pila.Sacar ( ES E: Elemento )
inicio
   si no vacia() entoces
                     e= elementos[ tope ]
                     tope = tope - 1
                   caso contrario // Error
Fin
```

3.4.2 Implementación con apuntadores

Tope= DireccionMemoria



Elemento , Siguiente

3.4.2 Implementacion con apuntadores

Definición básica de la clase Pila con representación enlazada con simple enlace:

```
Tipo de dato
 Nodo
     elemento TipoElemento
     sig      Puntero a Nodo
 // fin definicion

 Direccion Puntero a espacio de memoria de tipo Nodo

Clase Pila
  Atributos
      Tope Puntero de tipo Direccion
  Metodos
          ..............
```

3.4.2 Implementación con apuntadores

```
publico Elemento Pila.cima()
      Si (vacia()) Entoces  // Error
                    caso contrario
                      return obtenerdato(Tope,1)
 }
publico  Pila.meter( E : Elemento)
 inicio
      aux = // Pedir Espacio de memoria para Nodo
      si aux <> nulo entoces
                        ponerdato(aux,1,E)
                        ponerdato(aux,2,Tope)
                        Tope = Aux
                      caso contrario // Error
   fin
```

3.4.2 Comparación de las implementaciones

- La elección de una implementación u otra dependerá de los requerimientos de la aplicación en la que se use.
- Es posible obtener representaciones para pilas que permiten operaciones eficientes que se realizan en tiempos de ejecución constantes.