# TADs_Listas: Pila y Cola

Ana Lilia Laureano-Cruces
Universidad Autónoma Metropolitana-Azcapotzalco

# Listas …

- Las lista es un conjunto de elementos del mismo tipo donde pueden considerarse extracciones e inserciones.
- Cuando las listas cuentan con acceso restringido se convierten en especializaciones. Tal es el caso de las listas tipo *Cola y Pila*

# Especificación del TAD Lista con base en sus operaciones …

- *Incicializadoras:*
  - InicLista: crea e inicializa una estructura Lista vacía.
- *Constructoras:*
  - InsLista: inserta un elemento de tipo *elem* en una posición *pos* de la estructura Lista.
  - AnxLista: inserta un elemento de tipo *elem* al final de la estructura Lista.

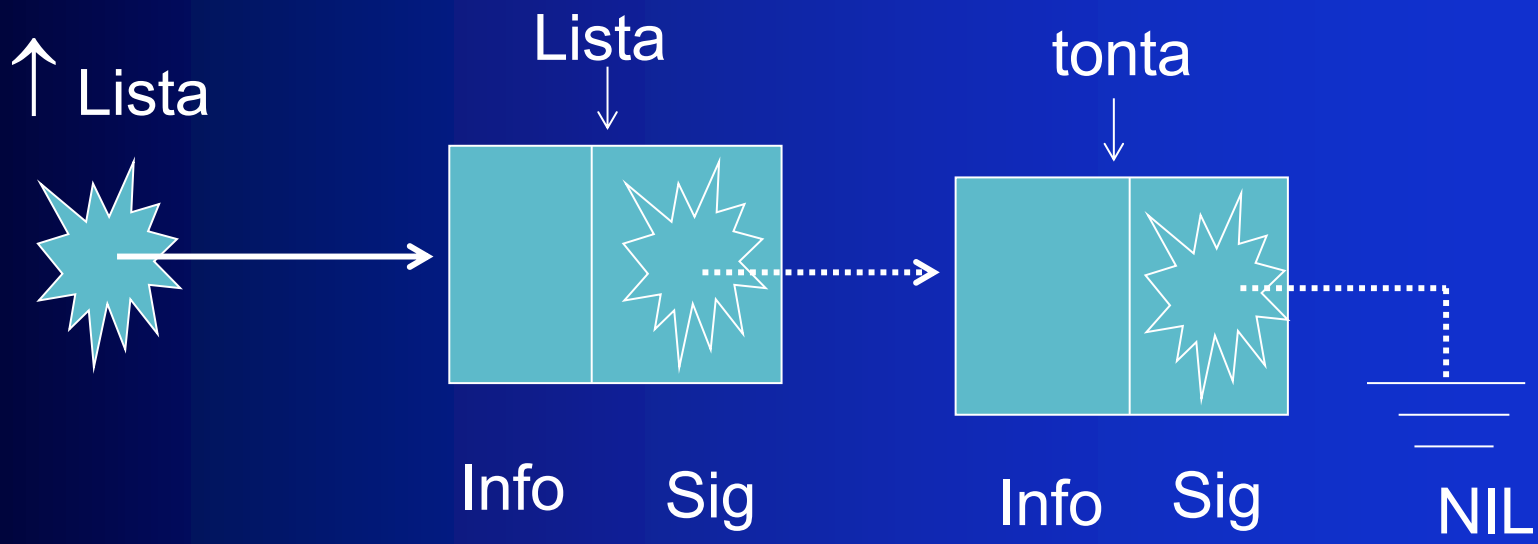
- *Simplificadoras:*
  - ElimLista: elimina un elemento (elem), dada su posición (pos).
  - AnulLista: Limpia la estructura Lista, regresando un lista vacía.

# TADs …

- *Analizadoras:*
  - InfoLista: devuelve el el valor de elemento (elem); dada su posición (pos)
  - LongLista: nos proporciona la longitud de la Lista
  - LlenaLista: valor verdadero si esta llena, falso caso contrario (memoria Estática)

- Las estructuras de datos de los TADs pueden ser diseñadas considerando memoria *estática* o memoria *dinámica*.

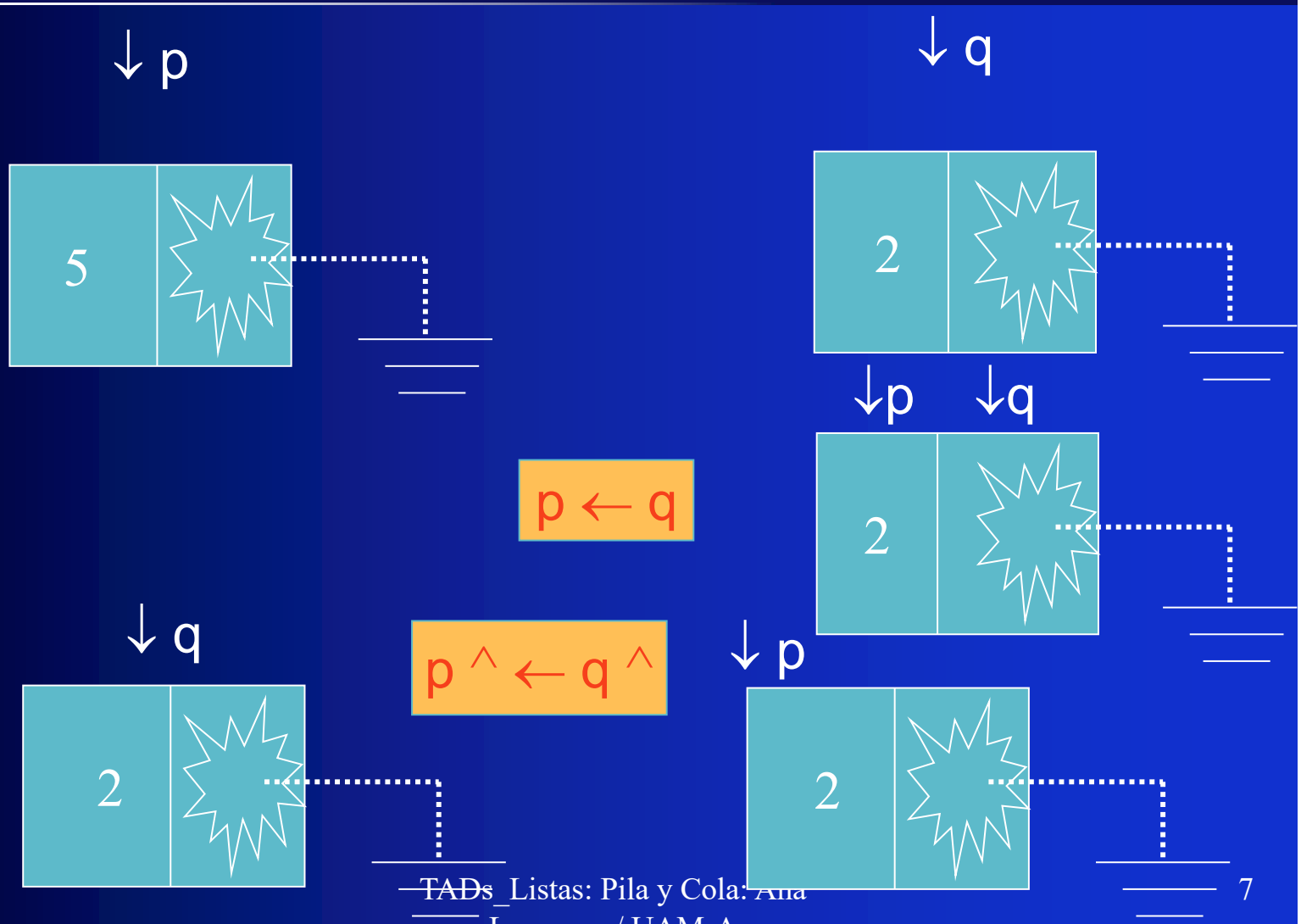
# Memoria dinámica …




Lista↑← tonta↑

Lista↑. Sig

# Memoria dinámica …



# Funcionalidad del TAD Lista considerando memoria estática …

- DOMINIO - CODOMINIO
  - InicLista ( ) → Lista
  - InsLista (Lista, Pos, Elemento) → Lista
  - AnxLista (Lista, Elemento) → Lista
  - ElimLista (Lista, Pos) → Lista
  - AnulLista (Lista) → Lista
  - InfoLista (Lista, Pos) → Elemento
  - LongLista (Lista) → Entero
  - LlenaLista (Lista) → Lógico

# Funcionalidad del TAD Lista considerando memoria dinámica …

- DOMINO - CODOMINIO
  - InicLista ( ) → ^ Lista
  - InsLista (^ Lista, Elemento) → ^ Lista
  - AnxLista (^ Lista, Elemento) → ^ Lista
  - ElimLista (^ Lista) → ^ Lista
  - AnulLista (^ Lista) → ^ Lista
  - InfoLista (^ Lista) → Elemento
  - LongLista (^ Lista) → Entero
  - ExistenElems (^ Lista) → Lógico

# Con memoria dinámica …

- En este caso se puede considerar que la operación InsLista sea utilizada para insertar a un elemento en la primera posición y que LlenaLista se convierta en ExistenElems dado que se utiliza memoria dinámica. Por otro lado ElimLista e InfoLista debe especificarse si es el primero o último elemento.

# Formalización …

- Oper_InicLista:
  - { PreCond: Verdad }
  - { PostCond: Lista: ARRAY [ 1..Tam] ← 0; N ← 1/ N ← Tam}
  - { PostCond: ^ Lista ← NIL }
- Oper_InsLista:
  - { PreCond: ∃ Lista *AND* 1 ≤ pos ≤ N }
  - { PostCond: ∃ Elemento *AND* N = N - 1}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: ∃ Elemento, *IN* Lista}

# Formalización …

- Oper_AnxLista:
  - { PreCond: ∃ Lista *AND* 1 ≤ pos ≤ Tam }
  - { PostCond: ∃ Elemento *AND* N = N - 1}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: ∃ Elemento *AND* ^ Lista.Sig = *NIL AND* ^ Ultimo ← ^ Lista}
- Oper_ElimLista:
  - { PreCond: ∃ Lista *AND* 1 ≤ pos ≤ Tam }
  - { PostCond: ∉ Elemento *AND* N = N + 1}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: ∉ Elemento / ^ Inicio.Info / ^ Ultimo.Info }

# Formalización …

- Oper_AnulLista:
  - { PreCond: ∃ Lista }
  - { PostCond: pos = 1; Lista [ 1..Tam] ← 0}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: ^ Lista = *NIL*}
- Oper_InfoLista:
  - { PreCond: ∃ Lista *AND* 1 ≥ pos ≤ Tam }
  - { PostCond: Elem}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: Elem ← ^ Inicio.Info / ^ Ultimo.Info }

# Formalización …

- Oper_LongLista:
  - { PreCond: ∃ Lista *AND* 1 ≤ N ≤ Tam }
  - { PostCond: N}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: N }
- Oper_ExistenElems:
  - { PreCond: ∃ Lista *AND* 1 ≤ N ≤ Tam }
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: TRUE }
- LlenaLista:
  - { PreCond: ∃ Lista *AND* 1 ≤ N ≤ Tam }
  - { PreCond: no tiene sentido con memoria dinámica }
  - { PostCond: TRUE }

# La Pila …

- La pila es una lista de acceso restringido (especialización), las inserciones y las salidas son por el mismo extremo.

# Especificación del TAD Pila con base en sus operaciones …

- *Incicializadoras:*
  - InicPila: crea e inicializa una estructura Pila vacía.
- *Constructoras:*
  - InsPila: inserta un elemento de tipo *elem* en la Pila. (Tope de la Pila)

# Especificación del TAD Pila con base en sus operaciones …

- *Simplificadoras:*
  - ElimPila: elimina un elemento (elem) de la Pila (el del Tope).
  - AnulPila: Limpia la estructura Pila, regresando una estructura Pila vacía.
- *Analizadoras:*
  - TopePila: devuelve el el valor de elemento (elem); siempre el del Tope
  - LongPila: nos proporciona la longitud de la Pila
  - LlenaPila: valor verdadero si esta llena, falso caso contrario (memoria Estática)

# Funcionalidad del TAD Pila considerando memoria estática …

- InicPila ( ) → Pila
- InsPila (Pila, Elemento) → Pila
- ElimPila (Pila) → Pila
- AnulPila (Pila) → Pila
- TopePila (Pila) → Pila
- LongPila (Pila) → Entero
- LlenaPila (Pila) → Lógico

# Funcionalidad del TAD Pila considerando memoria dinámica …

- InicPila ( ) → ^ Pila
- InsPila (^ Pila, Elemento) → ^ Pila
- ElimPila (^ Pila) → ^ Pila
- AnulPila (^ Pila) → ^ Pila
- TopePila (^ Pila) → Elemento
- LongPila (^ Pila) → Entero
- LlenaPila (^ Pila) → Lógico

# Formalización …

- InicPila:
  - { PreCond: Verdad }
  - { PostCond: Pila: ARRAY [ 1..Tam] ← 0 }
  - { PostCond: ^ Pila }
- InsPila:
  - { PreCond: ∃ Pila *AND* ~ LLenaPila }
  - { PostCond: ∃ Elemento *AND* Tope = Tope - 1}
  - { PreCond: ^ Tope ≠ NIL }
  - { PostCond: ^ Tope.Info = Elemento}

# Formalización …

- ElimPila:
  - { PreCond: ∃ Pila }
  - { PostCond: ∉ Elemento *AND* Tam = N + 1}
  - { PreCond: ^ Pila ≠ NIL }
  - { PostCond: ∉ Elemento / Inicio = ^ Inicio.sig / Penultimo = ^ Penultimo.Sig }

- AnulPila:
  - { PreCond: ∃ Pila *AND* (1 ≥ Tope ≤ Tam) }
  - { PostCond: Tope = Tam + 1}
  - { PreCond: ^ Pila ≠ NIL }
  - { PostCond: ^ Pila = NIL}

# Formalización …

- TopePila:
  - { PreCond: ∃ Pila}
  - { PostCond: Elemento = Pila [tope]}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: Elem ← ^ Inicio.Info / ^ Ultimo.Info }
- LongPila:
  - { PreCond: ∃ Pila *AND* 1 ≥ N ≤ Tam }
  - { PostCond: N}
  - { PreCond: ^ Lista ≠ NIL }
  - { PostCond: N }

# Formalización …

- LlenaPila:
  - { PreCond: ∃ Lista *AND* 1 ≥ N ≤ Tam }
  - { no tiene sentido con memoria dinámica }
  - { PostCond: TRUE }

# La formalización de las operaciones del TAD Pila se desarrollaran con base en las operaciones de LISTA …

- InicPila:
  - InicLista (Lista)
- InsPila:
  - InsLista ( Elemento, pos = 1, Lista)
- ElimPila:
  - PostCond: ElimLista (Lista, pos =1)
- AnulPila:
  - AnulLista (Lista)

# La formalización de las operaciones del TAD Pila se desarrollaran con base en las operaciones de LISTA …

- TopePila:
  - InfoLista (Lista, pos = 1)
- LongPila:
  - LongLista (Lista)
- LlenaPila:
  - LlenaLista (Lista)

# La Cola …

- La cola es una lista de acceso restringido (especialización) donde todas las inserciones son por un extremo y las extracciones por otro.

# Especificación del TAD Cola con base en sus operaciones …

- *Incicializadoras:*
  - InicCola: crea e inicializa una estructura Cola vacía.
- *Constructoras:*
  - InsCola: inserta un elemento de tipo *elem* en la Cola. (por defecto será en el último lugar)

# Especificación del TAD Cola con base en sus operaciones …

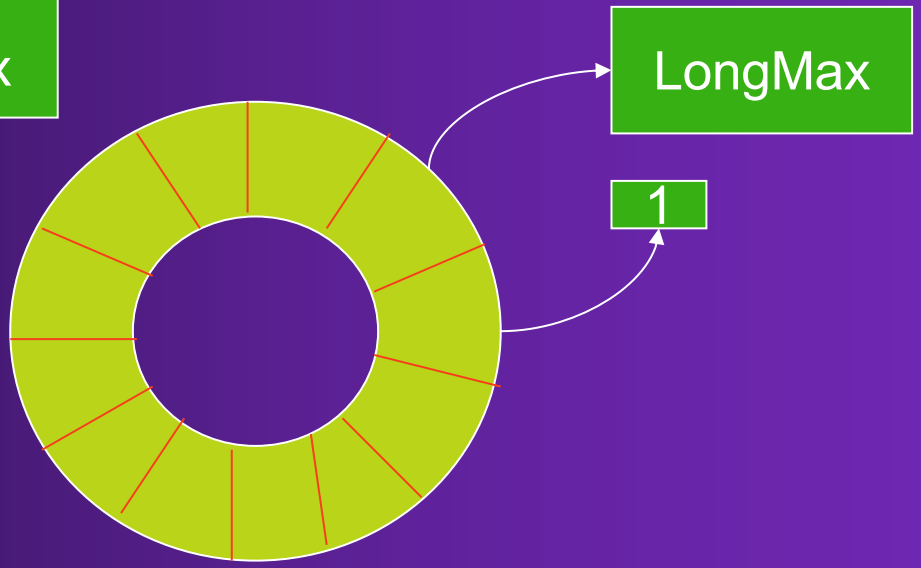
- *Simplificadoras:*
  - ElimCola: elimina un elemento (elem) de la Cola (por defecto será el primero).
  - AnulCola: Limpia la estructura Cola, regresando una estructura Cola vacía.
- *Analizadoras:*
  - InfoCola: devuelve el el valor de elemento (elem); siempre el primero.
  - LongCola: nos proporciona la longitud de la Cola
  - LlenaCola: valor verdadero si esta llena, falso caso contrario (memoria Estática)

# Funcionalidad del TAD Cola considerando memoria estática …

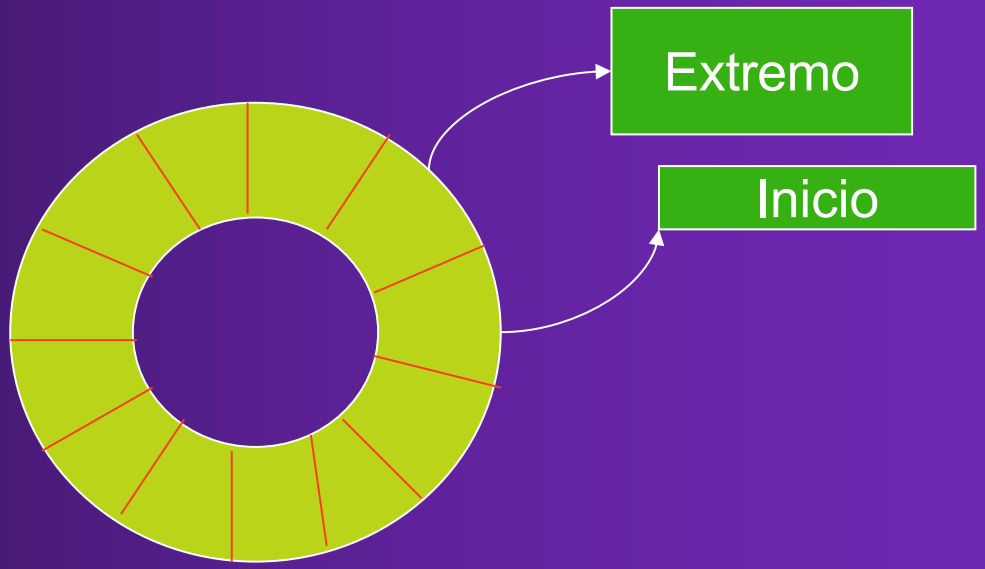
- InicCola ( ) → Cola
- InsCola (Cola, Elemento) → Cola
- ElimCola (Cola) → Cola
- AnulCola (Cola) → Cola
- InfoCola (Cola) → Elemento
- LongCola (Cola) → Entero
- LlenaCola (Cola) → Lógico
- VaciaCola (Cola) → Lógico

# Anatomía de la estructura Cola-Circular …

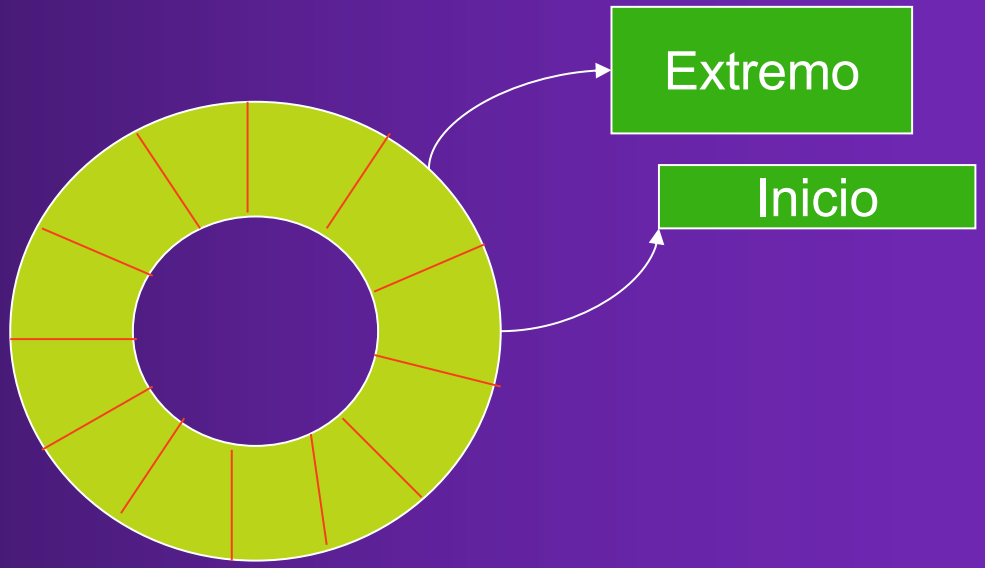
Inicio ← 1,
Extremo ← LongMax



# La estructura Vacía …

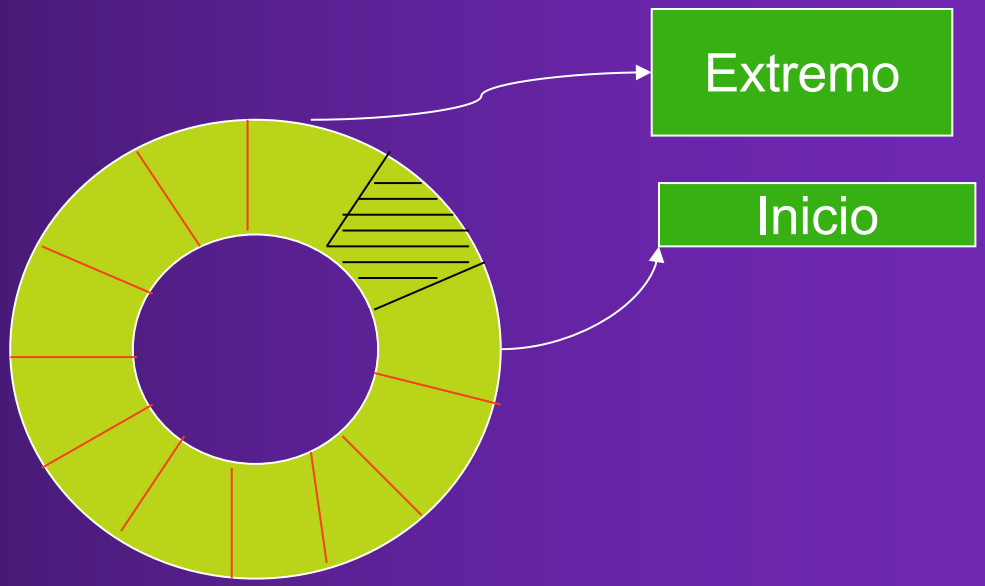


# La estructura Llena …




La posición relativa de los apuntadores es la misma
Cuando esta llena y cuando esta vacía

# La estructura Vacía …



# Formalizción …

- InicCola:
  - { PreCond: Verdad }
  - { PostCond: Cola: ARRAY [ 1..Tam] ← 0, Inicio = 1, Extremo = LongMax}
- InsCola:
  - { PreCond: ∃ Cola ~ LLenaCola }
  - { PostCond: ∃ Elemento *AND* Extremo ← Extremo + 1}

# Formalizción …

- ElimCola:
  - { PreCond: ~ VaciaCola }
  - { PostCond: $\notin$ Elemento *AND* Extremo $\leftarrow$ Extremo - 1}
- AnulCola:
  - { PreCond: $\exists$ Cola}
  - { PostCond: InicCola}

# Formalizción …

- InfoCola:
  - { PreCond: ∃ Cola }
  - { PostCond: InfoCola ← Cola [ Inicio] }

- LongCola (Cola) → Entero
  - { PreCond: ∃ Cola }
  - { PostCond: LongCola ← | Extremo - Inicio | }

- ExisteCola:
  - { PreCond: ~ VaciaCola }
  - { PostCond: Verdadero / Falso }
- LlenaCola:
  - { PreCond:  ssi Inicio = Extremo + 2 }
  - { PostCond: Verdadero }
- VaciaCola:
  - { PreCond:  ssi Inicio = 1 *AND* Extremo = LongMax }
  - { PostCond: Verdadero }