UNIVERSDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

PROGRAMACIÓN DECLARATIVA

INGENIGERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE

Tema 10.- Listas

Primera parte: Scheme

- Tema 1.- Introducción al Lenguaje Scheme
- Tema 2.- Expresiones y Funciones
- Tema 3.- Predicados y sentencias condicionales
- Tema 4.- Iteración y Recursión
- Tema 5.- Tipos de Datos Compuestos
- Tema 6.- Abstracción de Datos
- Tema 7.- Lectura y Escritura

Segunda parte: Prolog

- Tema 8.- Introducción al Lenguaje Prolog
- Tema 9.- Elementos Básicos de Prolog
- Tema 10.- Listas
- Tema 11.- Reevaluación y el “corte”
- Tema 12.- Entrada y Salida

Segunda parte: Prolog

Tema 8.- Introducción al Lenguaje Prolog

Tema 9.- Elementos Básicos de Prolog

Tema 10.- Listas

Tema 11.- Reevaluación y el "corte"

Tema 12.- Entrada y Salida

Índice

Índice

1. Descripción de lista

- Definición de lista
- Listas y estructuras
- Cabeza y cola de una lista
- Instanciaciones de cabeza y cola
- Cadenas y listas

1. Descripción de lista

- **Definición de lista**
- Listas y estructuras
- Cabeza y cola de una lista
- Instanciaciones de cabeza y cola
- Cadenas y listas

1. Descripción de lista

• Definición de lista

○ Sintaxis

[elemento$_1$, elemento$_2$, …, elemento$_N$]

- Los elementos pueden ser:
  - ❑ átomos, números, cadenas, estructuras, variables u otras listas.

○ Ejemplo

[a,1, “cadena”, persona(‘Juan Lara’, 19),Variable, [b,c]]

1. Descripción de lista

- Definición de lista
- **Listas y estructuras**
- Cabeza y cola de una lista
- Cola o resto de la lista
- Ejemplos de cabeza y cola
- Instanciaciones de cabeza y cola

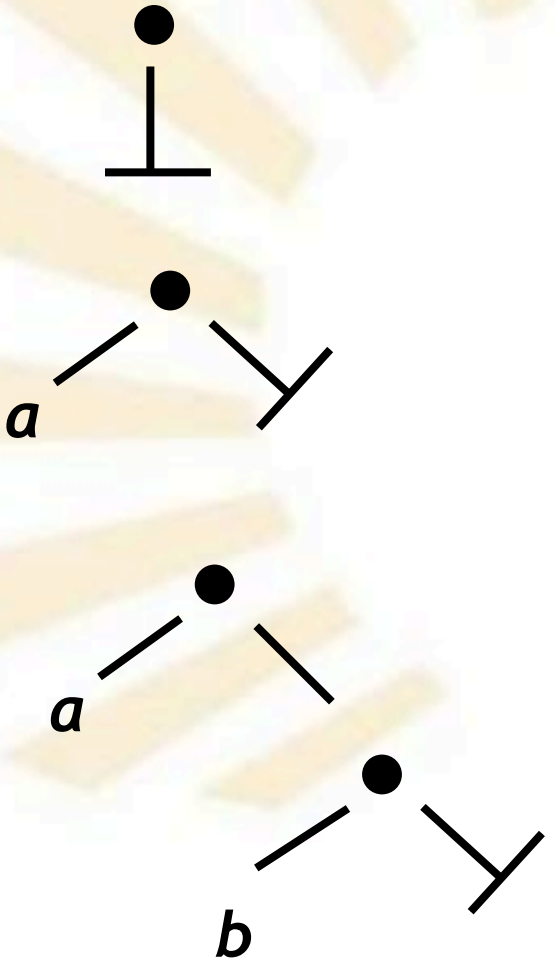
1. Descripción de lista

- **Listas y estructuras**
  - Una lista es equivalente a una estructura cuyo nombre sea el punto “.”

Lista	Estructura
[]	
[a]	*.(a,[])*
[a, b]	*.(a,.(b,[]))*
[a ,b, c]	*.(a,.(b,.(c,[])))*

1. Descripción de lista

- **Listas y estructuras**
  - **Representación gráfica**
    - *[]*
    - *[a] = .(a, [])*
    - *[a,b] = .(a, .(b, []))*



1. Descripción de lista

- Definición de lista
- Listas y estructuras
- Cabeza y cola de una lista
- Instanciaciones de cabeza y cola
- Cadenas y listas

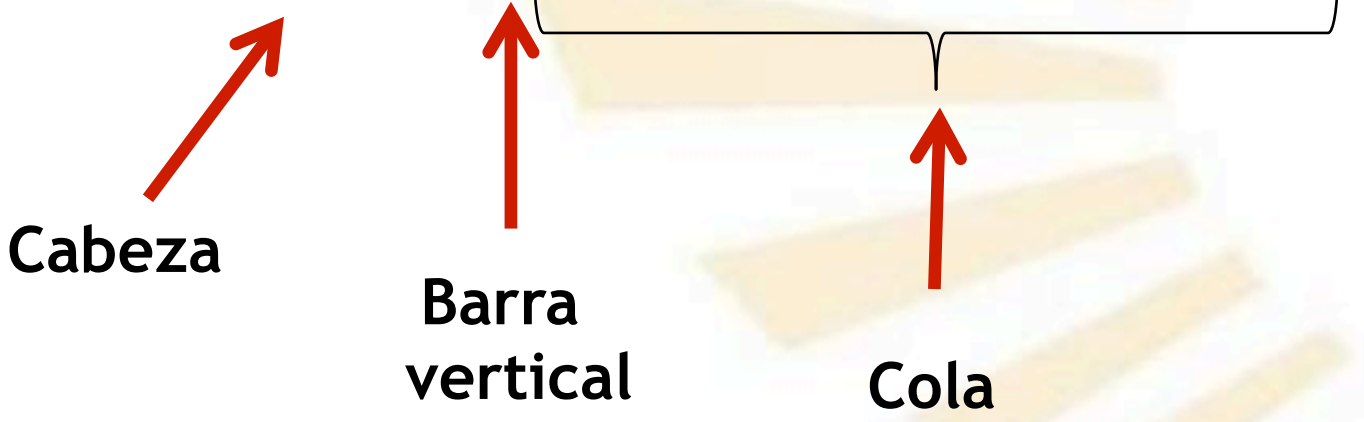
1. Descripción de lista

- **Cabeza y cola de una lista**
  - La lista

    $[elemento_1, elemento_2, \ldots, elemento_N]$

    es equivalente a

    $[elemento_1 \mid [elemento_2, \ldots, elemento_N]\ ]$



1. Descripción de lista

• Cabeza y cola de una lista

○ Ejemplos

Lista		Cabeza	Cola
[]	*[]*		
[a]	*[a | []]*	*a*	*[]*
[a, b]	*[a | [b]]*	*a*	*[b]*
[a ,b, c]	*[a | [b, c]]*	*a*	*[b, c]*

1. Descripción de lista

- **Cabeza y cola de una lista**
  - **Ejemplos**

Lista		Cabeza	Cola
[[a]]	*[[a] | []]*	*[a]*	*[]*
[a, [b]]	*[a | [[b]]]*	*a*	*[[b]]*
[[a ,b], c]	*[[a , b] | c]]*	*[a, b]*	*[c]*

1. Descripción de lista

- ## Cabeza y cola de una lista

  - **Ejemplos**

    [el, hombre]

    - Cabeza: *el*
    - Cola: *[hombre]*

    [[el, hombre], es, [un, mamífero]]

    - Cabeza: *[el, hombre]*
    - Cola: *[es, [un, mamífero]]*

1. Descripción de lista

• Cabeza y cola de una lista

- Otras equivalencias

  [a, b, c]

  es equivalente a

  [a | [b, c]]

  [a, b | [c]]

  [a, b, c | []]

1. Descripción de lista

- ## Cabeza y cola de una lista

  - **Observación**

    [a, b] ≠ [a | b]

    - Lista con dos elementos

      [a, b] = [a | [b]] = .(a, [b]) = .(a, .(b, []))

    - Estructura con dos elementos que no es una lista

      [a | b] = .(a, b)

1. Descripción de lista

- Definición de lista
- Listas y estructuras
- Cabeza y cola de una lista
- Instanciaciones de cabeza y cola
- Cadenas y listas

1. Descripción de lista

• Instanciaciones de cabeza y cola

○ Ejemplos

?- [a, b] = [Cabeza | Cola].
Cabeza = a,
Cola = [b].

?- [a, b] = [Primero, Segundo].
Primero = a,
Segundo = b.

1. Descripción de lista

- ## Instanciaciones de cabeza y cola

  - **Ejemplos**

    ?- [a, b, c, d, e] = [Cabeza | Cola].
    Cabeza = a,
    Cola = [b, c, d, e].

    ?- [a, b, c, d, e] = [Cabeza , Cola].
    false.

1. Descripción de lista

- **Instanciaciones de cabeza y cola**
  - **Ejemplos**

?- [X,Y,Z] = [juan,come,pan].

X = juan,

Y = come,

Z = pan.

?- [X,Y|Z] = [juan,come,pan].

X = juan,

Y = come,

Z = [pan].

1. Descripción de lista

- **Instanciaciones de cabeza y cola**
  - Observación
    - La barra para indicar la cola debe estar al final de la lista.

?- [X | Y , Z] = [juan,come,pan].

Error

1. Descripción de lista

• Instanciaciones de cabeza y cola

○ Ejemplos

?- [gato]=[X|Y].

X = gato,

Y = [].

?- [X,Y|Z]= [ana, bebe, agua].

X = ana,

Y = bebe,

Z = [agua].

?- [[el, Y] | Z]= [[el, mar], [es, grande]].

Y = mar,

Z = [[es, grande]].

1. Descripción de lista

- Instanciaciones de cabeza y cola
  - **Ejemplos**

datos([1, 2, 3, 4, 5,6]).

?- datos([X|Y]).
X = 1,
Y = [2, 3, 4, 5, 6] .

?- datos([_,_,X|Y]).
X = 3,
Y = [4, 5, 6] .

1. Descripción de lista

• Instanciaciones de cabeza y cola

• Ejemplos

frase([[el, hombre], es, [un, mamífero]]).

?- frase([X|Y]).
X = [el, hombre],
Y = [es, [un, mamífero]].

?- frase([_,_,X|Y]).
X = [un, mamífero],
Y = [].

1. Descripción de lista

- ## Instanciaciones de cabeza y cola

  - **Ejemplos**

meses([enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre, diciembre]).

?- meses([Cabeza | Cola]).

Cabeza = enero,

Cola = [febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre|...].

1. Descripción de lista

- ## Instanciaciones de cabeza y cola

  - **Ejemplos**

  ?- meses([Primero, Segundo | Resto]).

  Primero = enero,

  Segundo = febrero,

  Resto = [marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre|...].

1. Descripción de lista

- ## Instanciaciones de cabeza y cola

  - **Ejemplos**

?- meses([X , febrero, _, Y | _]).

X = enero,

Y = abril.

?- meses([_, M2, _, M4, _, M6|_]).

M2 = febrero,

M4 = abril,

M6 = junio

1. Descripción de lista

- ## Instanciaciones de cabeza y cola

  - **Ejemplos**

?- meses([_,_,_,_|[M|Resto]]).

M = mayo,

Resto = [junio, julio, agosto, septiembre, octubre, noviembre, diciembre].

1. Descripción de lista

- **Cadenas y listas**
  - Una cadena es considerada como una lista compuesta por los códigos numéricos de sus caracteres.
  - Ejemplos

?- "Cadena" = L.

L = [67, 97, 100, 101, 110, 97].

?- "Cadena" = [Cabeza | Cola].

Cabeza = 67,

Cola = [97, 100, 101, 110, 97].

Índice

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Es lista
  - Descripción
    - La lista vacía es una lista
    - Si “Cola” es una lista entonces también lo es cualquier lista cuya cola sea “Cola”

      [_ | Cola]

2. Operaciones con listas

- Es lista
  - Código

```
es_lista([]).
es_lista([_|Cola]):-
        es_lista(Cola).
```

2. Operaciones con listas

- Es lista
  - Preguntas

*?- **es_lista**([a,[b],c]).*

true.

*?- **es_lista**(a).*

false.

*?- **es_lista**([a|b]).*

false.

*?- **es_lista**([a|[]]).*

true.

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Longitud
  - Descripción
    - La lista vacía tiene cero elementos
    - Si “Cola” tiene N elementos

      entonces la lista [_ | Cola] tiene N +1

2. Operaciones con listas

- Longitud
  - Código

```
longitud([],0).

longitud([_|Cola],N):-
        longitud(Cola,M),
        N is M + 1.
```

2. Operaciones con listas

- Longitud
  - Preguntas

?- longitud([],R).

R = 0.

?- longitud([a,b,c,d],R).

R = 4.

?- longitud([a,[b,c],d],R).

R = 3.

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Pertenece
  - **Descripción**
    - Un elemento pertenece a una lista si es su cabeza

    - o si pertenece a la cola de la lista

  - **Código**

```
pertenece(X,[X|_]).

pertenece(X,[_|Cola]):- pertenece(X,Cola).
```

2. Operaciones con listas

- Pertenece
  - **Preguntas**

*?- **pertenece**(z, [a, b, c, d]).*

false

*?- **pertenece**(d, [a, b, c, d]).*

true

*?- **pertenece**(d, [[a, b], [c, d]]).*

false

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Concatenar
  - **Descripción**
    - Si se concatena la lista vacía con una lista ***L*** entonces se obtiene la lista ***L***.

[] ***L*** **=** ***L***

    - Si al concatenar una lista ***L1*** con otra ***L2*** se obtiene la lista **L3** entonces la concatenación de ***[X | L1]*** con ***L2*** es ***[X | L3]***.

2. Operaciones con listas

- Concatenar
  - **Código**

```
concatenar([],L,L).

concatenar([X | L1], L2, [X | L3]):-
                        concatenar(L1,L2,L3).
```

2. Operaciones con listas

- Concatenar
  - **Preguntas**

?- concatenar([a, b, c],[d, e], L).

L = [a, b, c, d, e].

?- concatenar([a,[b, c], d],[a, [], b], L).

L = [a,[b, c], d, a, [], b]

2. Operaciones con listas

- Concatenar
  - **Preguntas**
    - Uso de concatenar para descomponer listas

?- concatenar(L1,L2,[a,b,c]).
L1 = [],
L2 = [a, b, c] ;
L1 = [a],
L2 = [b, c] ;
L1 = [a, b],
L2 = [c] ;
L1 = [a, b, c],
L2 = [] ;
false.

Se teclea punto y coma

2. Operaciones con listas

- Concatenar
  - **Preguntas**
    - **Otras aplicaciones de concatenar (1/2)**

*?- concatenar(_, [**Anterior**, mayo, **Posterior** | _] , [enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre, diciembre]).*

***Anterior** = abril,*

***Posterior** = junio .*

2. Operaciones con listas

- Concatenar
  - **Uso**
    - **Otras aplicaciones de concatenar (2/2)**

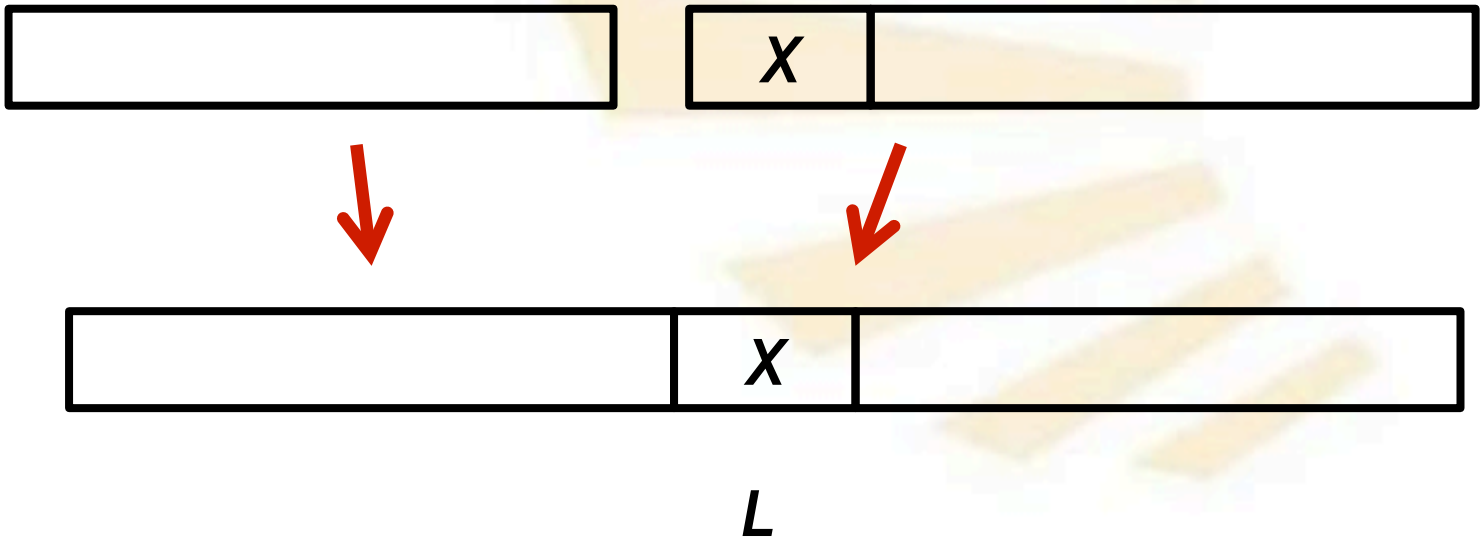
*?-**concatenar**(**Antes**, [mayo | **Después**], [enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre, diciembre]).*

***Antes** = [enero, febrero, marzo, abril],*

***Después** = [junio, julio, agosto, septiembre, octubre, noviembre, diciembre] .*

2. Operaciones con listas

- Concatenar
  - **Redefinición del predicado *pertenece* usando el predicado *concantenar***
    - X es pertenece a L si L se puede obtener concatenando una lista cualquiera con otra lista cuya cabeza sea X.



2. Operaciones con listas

- Concatenar
  - **Redefinición del predicado *pertenece* usando el predicado *concantenar***
    - Código

```
pertenece(X,L):-
        concatenar(_,[X|_], L)
```

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Incluir
  - Incluir al principio
  - Incluir al final

2. Operaciones con listas

- Incluir
  - **Incluir al principio**
    - **Descripción**
      - Para incluir ***X*** al principio de una lista ***L***, se debe crear otra lista cuya cabeza sea ***X*** y su cola sea ***L***.

    - **Código**

```
Incluir_al_principio(X,L, [X|L]).
```

2. Operaciones con listas

- Incluir
  - **Incluir al principio**
    - **Preguntas**

?- incluir_al_principio(a,[1,2,3],L).

L = [a, 1, 2, 3].

?- incluir_al_principio(a,[],L).

L = [a].

2. Operaciones con listas

- Incluir
  - **Incluir al principio**
    - **Preguntas**
      - Otro uso: obtener la cabeza y la cola de una lista

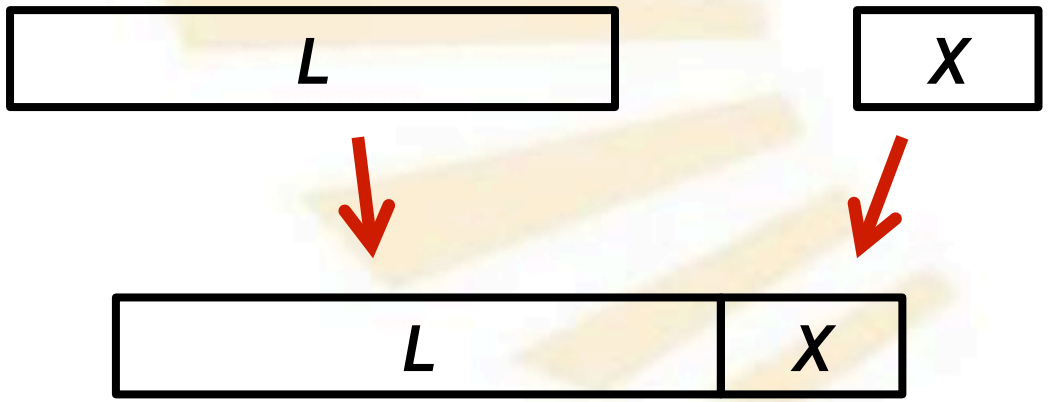
?- incluir_al_principio(Cabeza,Cola,[a,1,2,3]).

Cabeza = a,

Cola = [1, 2, 3].

2. Operaciones con listas

- Incluir
  - **Incluir al final**
    - **Descripción**
      - Para incluir ***X*** al final de una lista ***L***, se debe concatenar ***L*** con una lista compuesta solamente por el elemento ***X***.



2. Operaciones con listas

- Incluir
  - **Incluir al final**
    - **Código**

```
incluir_al_final(X, L,R):-
                    concatenar(L,[X],R).
```

2. Operaciones con listas

- Incluir
  - **Incluir al final**
    - **Preguntas**

?- incluir_al_final(a,[1,2,3],R).

R = [1, 2, 3, a].

?- incluir_al_final(a,[],R).

R = [a].

2. Operaciones con listas

- Incluir
  - **Incluir al final**
    - **Preguntas**
      - Otro uso: extraer el último elemento de una lista.

?- incluir_al_final(Último,L,[1,2,3,a]).

Último = a,

L = [1, 2, 3] .

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

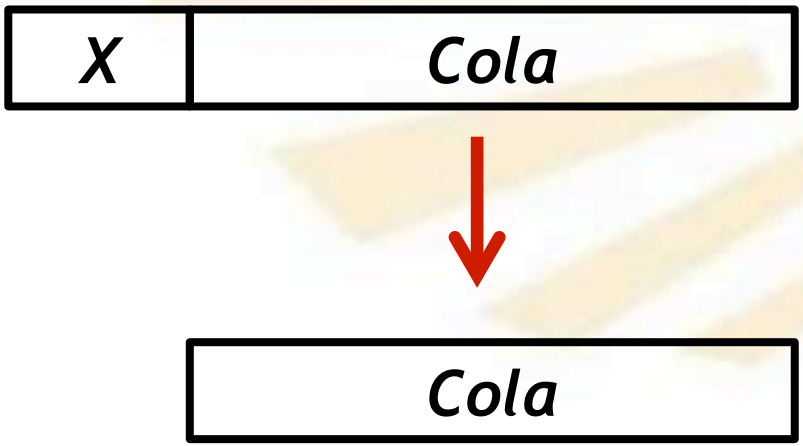
- Borrar
  - Borrar la **primera** aparición de un elemento
  - Borrar **todas** las apariciones de un elemento

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
  - Borrar **todas** las apariciones de un elemento

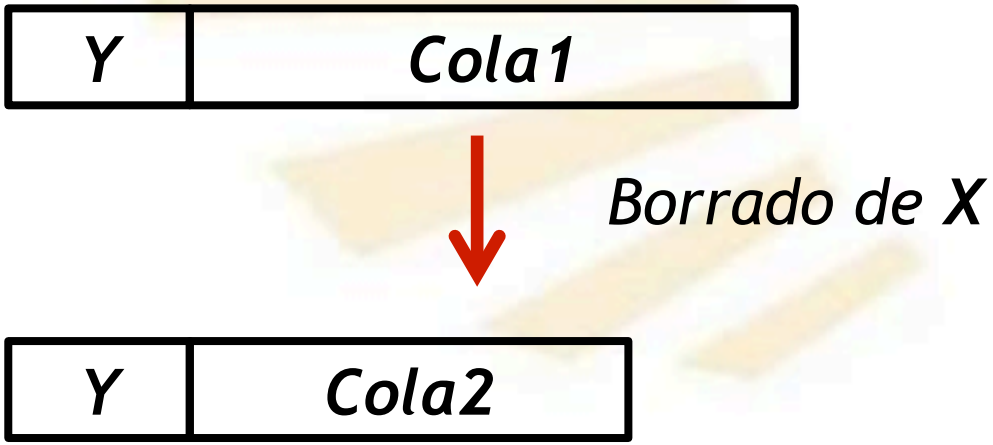
2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - **Descripción**
      - ❑ Caso 1
        - ✓ Si ***X*** es la cabeza de la lista ***L*** entonces, al borrarlo, se obtiene la cola de **L**.



2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - **Descripción**
      - Caso 2:
        - ✓ Si ***X*** no es la cabeza de ***L*** entonces se borra de la Cola.



2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - **Código**

```
borrar(X,[X | Cola],Cola).

borrar(X,[Y|Cola1],[Y|Cola2]):-
                    borrar(X,Cola1,Cola2).
```

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - **Preguntas**

?- borrar(a,[a,b,c],R).

R = [b, c] .

?- borrar(b,[a,b,c],R).

R = [a, c] .

?- borrar(c,[a,b,c],R).

R = [a, b] .

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - Otros usos de ***borrar*** (1/4): redefinir el predicado ***pertenece***
      - ***X*** pertenece a ***L*** si ***X*** se puede borrar de ***L***.

```
pertenece(X,L):-
        borrar(X,L,_).
```

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - Otros usos de ***borrar*** (1/4): redefinir el predicado ***pertenece***
      - ***X*** pertenece a ***L*** si ***X*** se puede borrar de ***L***
        - ✓ ***Preguntas***

          *?- **pertenece**(b,[a,b,c]).*

          true .

          *?- **pertenece**(z,[a,b,c]).*

          false.

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - Otros usos de ***borrar*** (2/4)
      - ❑ Proceso inverso: insertar

*?- **borrar**(a,L,[1,2,3]).*

L = [a, 1, 2, 3] ;

L = [1, a, 2, 3] ;

L = [1, 2, a, 3] ;

L = [1, 2, 3, a] ;

false.

Se teclea punto y coma

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - Otros usos de ***borrar*** (3/4):
      - ❑ Redefinir el predicado ***insertar***
        - ✓ ***X*** se puede insertar en ***L*** si, al borrarlo de otra lista, se obtiene ***L***.

insertar*(X,L,R):-*

borrar*(X,R,L).*

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - Otros usos de ***borrar*** (3/4):
      - ❑ Redefinir el predicado ***insertar***
        - ✓ ***Preguntas***

?- insertar(a,[1,2,3],R).
R = [a, 1, 2, 3] ;
R = [1, a, 2, 3] ;
R = [1, 2, a, 3] ;
R = [1, 2, 3, a] ;
false

Se teclea punto y coma

2. Operaciones con listas

- Borrar
  - **Borrar la primera aparición de un elemento**
    - Otros usos de ***borrar*** (4/4):
      - Borrar las **sucesivas apariciones** de un elemento

*?- **borrar**(a,[a,b,a,c,a],R).*

R = [b, a, c, a] ;

R = [a, b, c, a] ;

R = [a, b, a, c] ;

false.

Se teclea punto y coma

2. Operaciones con listas

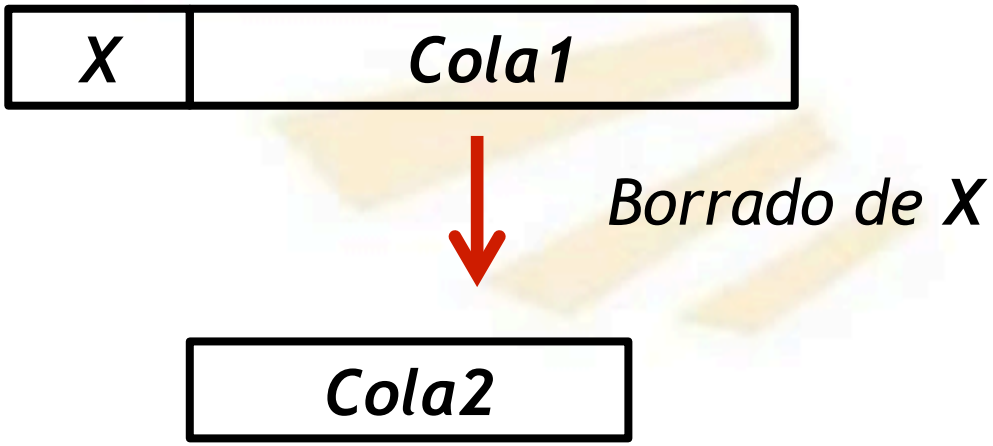
- Borrar
  - Borrar la **primera** aparición de un elemento
  - Borrar **todas** las apariciones de un elemento

2. Operaciones con listas

- Borrar
  - **Borrar todas las pariciones de un elemento**
    - **Descripción**
      - ❑ Caso 1
        - ✓ Si se desea borrar un elemento de la lista vacía entonces se obtiene la lista vacía
      - ❑ Caso 2
        - ✓ Si la lista solamente contiene el elemento buscado entonces se obtiene la lista vacía

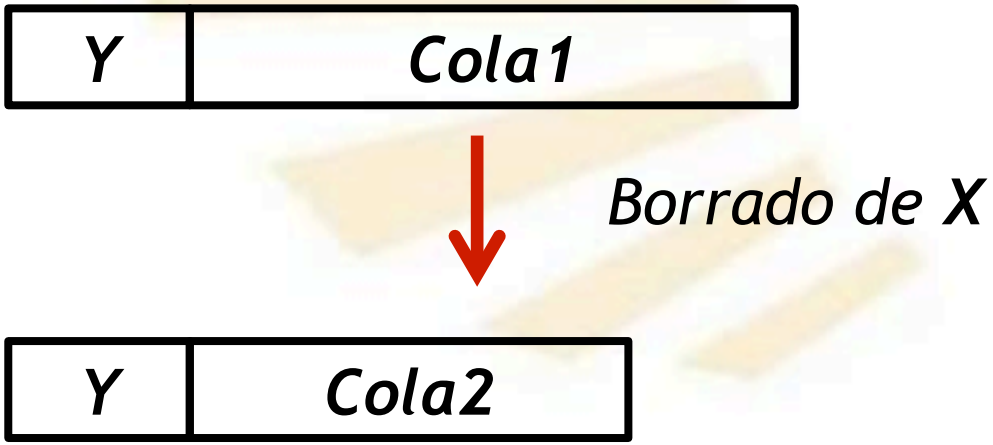
2. Operaciones con listas

- Borrar
  - **Borrar todas las pariciones de un elemento**
    - **Descripción**
      - ❑ Caso 3
        - ✓ Si ***X*** es la cabeza de la lista ***L*** entonces, al borrarlo, se obtiene la cola de **L**, de la cual también hay que borrar a **X**.



2. Operaciones con listas

- Borrar
  - **Borrar todas las pariciones de un elemento**
    - **Descripción**
      - Caso 4
        - ✓ Si ***X*** es no la cabeza de la lista ***L*** entonces hay que borrarlo de la cola de L.



2. Operaciones con listas

- Borrar
  - **Borrar todas las pariciones de un elemento**
    - **Código**

```
borrar(_,[],[]).

borrar(X,[X],[]).

borrar(X,[X|Cola1],Cola2):-
                    borrar(X,Cola1,Cola2).

borrar(X,[Y|Cola1],[Y|Cola2]):-
                    borrar(X,Cola1,Cola2).
```

2. Operaciones con listas

- Borrar
  - **Borrar todas las pariciones de un elemento**
    - **Preguntas**

?- borrar(a,[a,b,a,c,a],R).

R = [b, c] .

?- borrar(b,[a,b,a,c,a],R).

R = [a, a, c, a] .

?- borrar(z,[a,b,a,c,a],R).

R = [a, b, a, c, a] .

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Sublista
  - **Descripción**
    - S es una sublista de ***L*** si está contenida en ***L***.

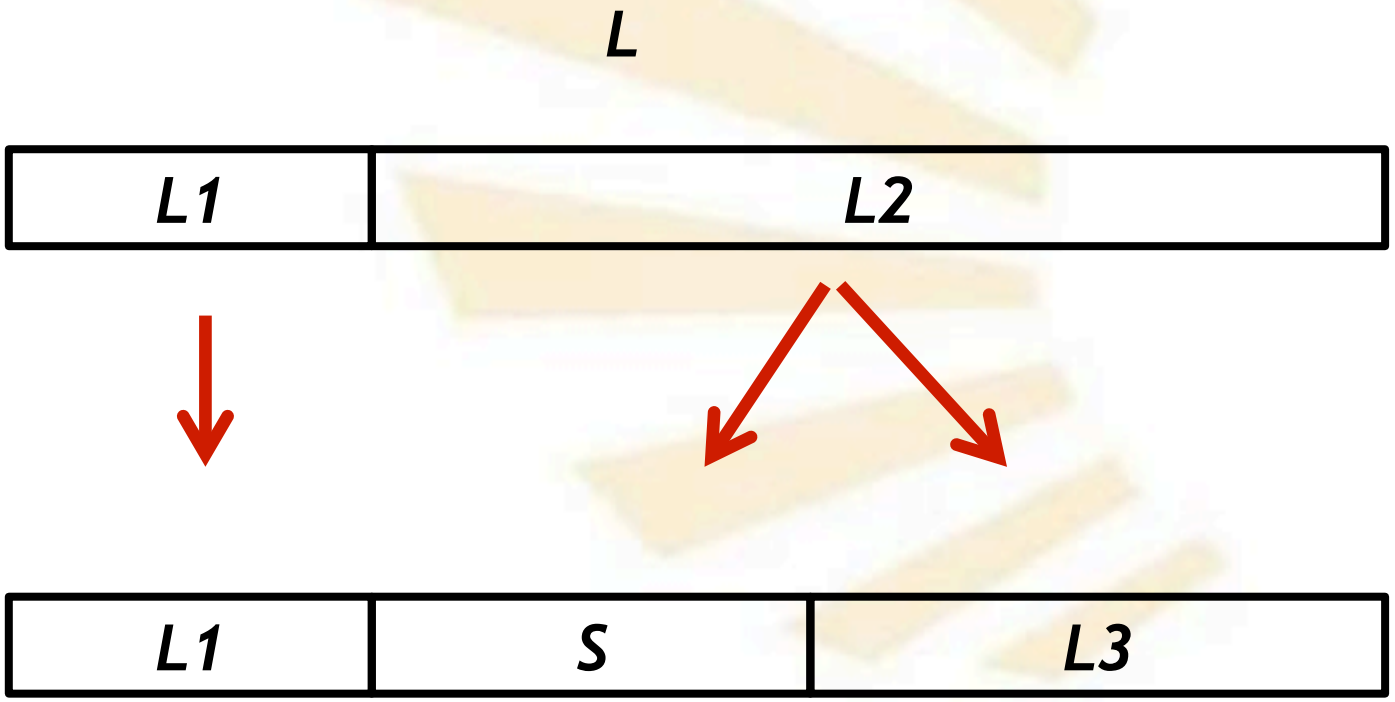
L

2. Operaciones con listas

- Sublista
  - **Descripción**
    - S es una sublista de ***L***
      - ***L*** se puede descomponer en dos sublistas ***L1*** y ***L2***.
      - y ***L2*** se puede descomponer en la sublista S y en otra sublista ***L3***.

2. Operaciones con listas

- Sublista
  - **Descripción**



2. Operaciones con listas

- Sublista
  - **Descripción**
    - ***S*** es una sublista de ***L***
      - ❑ ***L*** se puede descomponer en dos sublistas ***L1*** y ***L2***.
        - ✓ L se obtiene al concatenar ***L1*** y **L2**
      - ❑ y ***L2*** se puede descomponer en la sublista S y en otra sublista ***L3***.
        - ✓ L2 se obtiene al concatenar **S** y ***L3***.

2. Operaciones con listas

- Sublista
  - **Código**

```
sublista(S,L):-
        concatenar(_,L2,L),
        concatenar(S,_,L2).
```

2. Operaciones con listas

- Sublista
  - **Preguntas**

*?- **sublista**([b,c],[a,b,c,d,e]).*

true .

*?- **sublista**([b,a],[a,b,c,d,e]).*

false.

2. Operaciones con listas

- Sublista
  - **Preguntas**
    - Todas las sublistas de una lista

*?- **sublista**(S,[a,b,c]).*
S = [] ;
S = [a] ;
S = [a, b] ;
S = [a, b, c] ;
S = [] ;
S = [b] ;
S = [b, c] ;
S = [] ;
S = [c] ;
S = [] ;
false.

Se teclea punto y coma

2. Operaciones con listas

- Sublista
  - **Otros usos**
    - Redefinir el predicado ***pertenece***
      - Un elemento ***X*** pertenece a una lista ***L*** si la lista compuesta por el elemento ***X*** es una sublista de ***L***.

2. Operaciones con listas

- Sublista
  - **Otros usos**
    - Redefinir el predicado ***pertenece***

      pertenece(*X,L*):-

      sublista([*X*],*L*).

      - Preguntas

        ?- ***pertenece***(z,[a,b,c]).

        false.

        ?- ***pertenece***(b,[a,b,c]).

        true

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

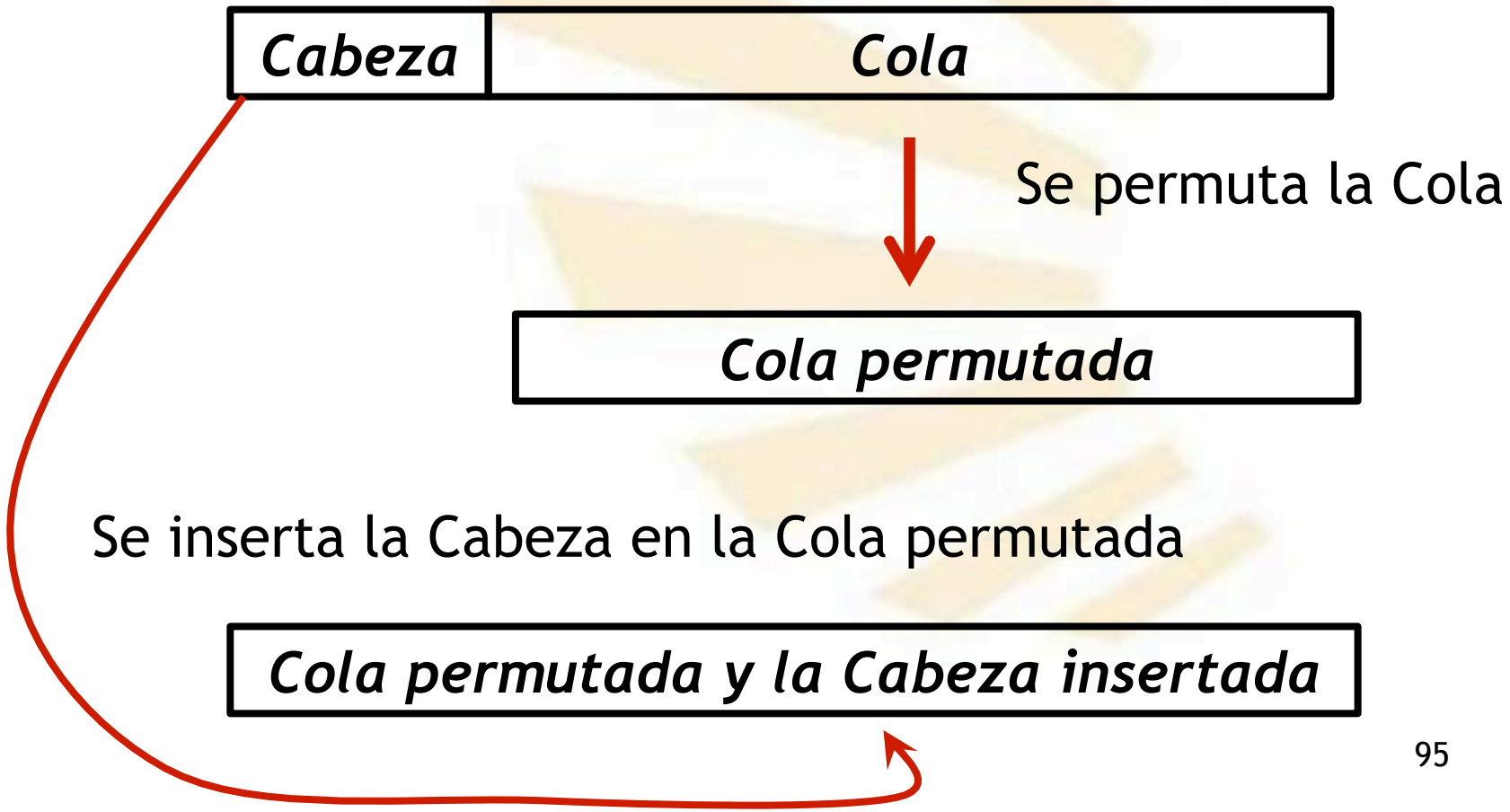
- Permutar
  - **Primera versión**
  - **Segunda versión**

2. Operaciones con listas

- Permutar
  - **Primera versión**
    - **Descripción**
      - ❑ Caso 1
        - ✓ La permutación de la lista vacía es la lista vacía.
      - ❑ Caso 2
        - ✓ La permutación de [Cabeza | Cola] se puede obtener
          - ➢ Permutando la Cola
          - ➢ Insertando la Cabeza en la Cola permutada

2. Operaciones con listas

- Permutar
  - **Primera versión**
    - **Descripción**



2. Operaciones con listas

- Permutar
  - **Primera versión**
    - **Código de la primera versión**

```
permutar([],[]).
permutar([Cabeza | Cola], P):-
        permutar(Cola, Cola_permutada),
        insertar(Cabeza, Cola_permutada, P).
insertar(X,L,R):- borrar(X,R,L).
borrar(X,[X | Cola],Cola).
borrar(X,[Y | Cola1], [Y | Cola2]):-
        borrar(X, Cola1, Cola2).
```

2. Operaciones con listas

- Permutar
  - **Primera versión**
    - **Preguntas**

?- permutar([a,b,c],R).

R = [a, b, c] ;

R = [b, a, c] ;

R = [b, c, a] ;

R = [a, c, b] ;

R = [c, a, b] ;

R = [c, b, a] ;

false

Se teclea punto y coma

2. Operaciones con listas

- Permutar
  - **Primera versión**
    - **Observación**

❑ Si se pregunta al **revés**, muestra los resultados posibles y cae en un bucle infinito.

?- permutar(L,[a,b,c]).
L = [a, b, c] ;
L = [b, a, c] ;
L = [c, a, b] ;
L = [a, c, b] ;
L = [b, c, a] ;
L = [c, b, a] ;
^CAction (h for help) ? abort
% Execution Aborted

Se teclea punto y coma

2. Operaciones con listas

- Permutar
  - **Segunda versión**
    - **Descripción**
      1. Se **borra** un elemento ***X*** de la lista.
      2. Se **permuta** el resto de la lista.
      3. Se **inserta *X*** al **principio** del resto de la lista permutada.

2. Operaciones con listas

- Permutar
  - **Segunda versión**
    - **Descripción**

1. Se **borra** un elemento ***X*** de la lista

2. Se **permuta** el resto de la lista

3. Se **inserta** ***X*** al **principio** del resto de la lista permutada

2. Operaciones con listas

- Permutar
  - **Segunda versión**
    - **Código**

```
permutar([],[]).
permutar(L, [X | P]):-
                    borrar(X,L,Resto),
                    permutar(Resto,P).
borrar(X, [X | Cola], Cola).
borrar(X, [Y|Cola1], [Y | Cola2]):-
                    borrar(X, Cola1, Cola2).
```

2. Operaciones con listas

- Permutar
  - **Segunda versión**
    - **Preguntas**

?- permutar([a,b,c],R).

R = [a, b, c] ;

R = [a, c, b] ;

R = [b, a, c] ;

R = [b, c, a] ;

R = [c, a, b] ;

R = [c, b, a] ;

false.

Se teclea punto y coma

2. Operaciones con listas

- Permutar
  - **Segunda versión**
    - **Observación**
      - Si se pregunta al **revés**, muestra la lista original y cae en un bucle infinito si se pide otra solución.

*?- **permutar**(L,[a,b,c]).*

L = [a, b, c] ;

Se teclea punto y coma

ERROR: Out of global stack

2. Operaciones con listas

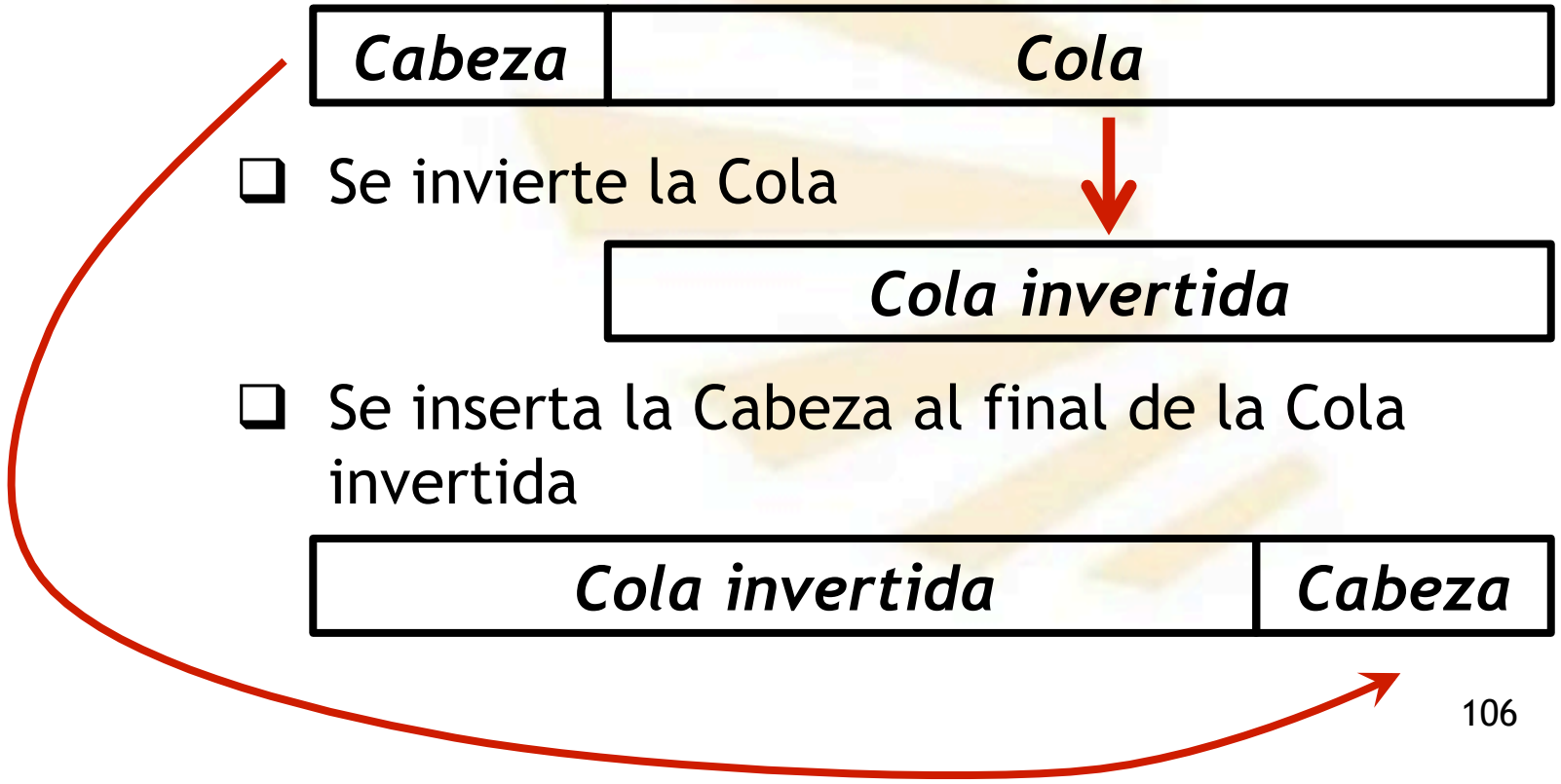
- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Invertir
  - **Descripción**
    - Al invertir la lista vacía, se obtiene la lista vacía
    - Para invertir la lista [Cabeza |Cola]
      - Se invierte la Cola
      - Se inserta la Cabeza al final de la Cola invertida

2. Operaciones con listas

- Invertir
  - **Descripción**
    - Al invertir la lista vacía, se obtiene la lista vacía
    - Para invertir la lista [Cabeza |Cola]



2. Operaciones con listas

- Invertir
  - **Código**

```
invertir([],[]).
invertir([Cabeza|Cola],R):-
                    invertir(Cola,Cola1),
                    incluir_al_final(Cabeza,Cola1,R).
incluir_al_final(X,L,R):-
                     concatenar(L,[X],R).
concatenar([],L,L).
concatenar([Cabeza|L1],L2,[Cabeza|L3]):-
                    concatenar(L1,L2,L3).
```

2. Operaciones con listas

- Invertir
  - **Preguntas**

*?- **invertir**([a,b,c,d],R).*

R = [d, c, b, a].

?- invertir([a,[b,c],d],R).

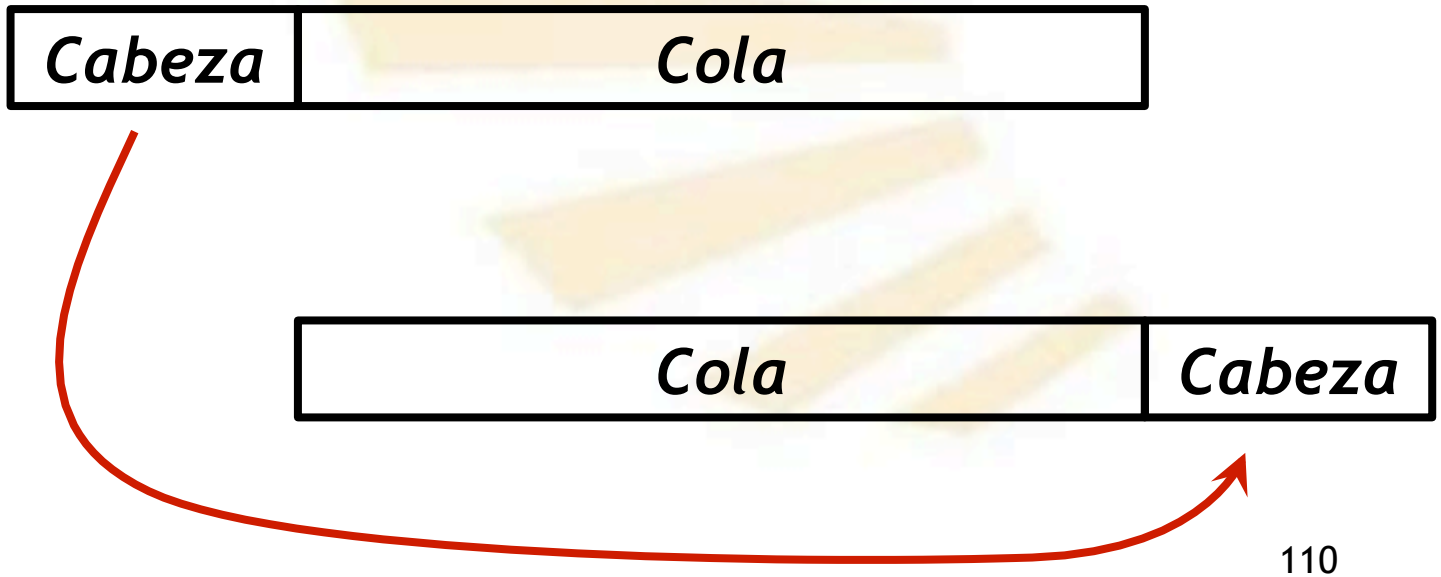
R = [d, [b, c], a].

2. Operaciones con listas

- Es lista
- Longitud
- Pertenece
- Concatenar
- Incluir
- Borrar
- Sublista
- Permutar
- Invertir
- Rotar

2. Operaciones con listas

- Rotar
  - **Descripción**
    - La rotación de la lista vacía es la lista vacía.
    - La rotación de la lista *[Cabeza | Cola]* es la lista que se obtiene al insertar la Cabeza al final de la Cola



2. Operaciones con listas

- Rotar
  - **Código**

```
rotar([],[]).
rotar([X|Cola],R):-
        incluir_al_final(X,Cola,R).
incluir_al_final(X,L,R):-
        concatenar(L,[X],R).
concatenar([],L,L).
concatenar([Cabeza|L1],L2,[Cabeza|L3]):-
        concatenar(L1,L2,L3).
```

2. Operaciones con listas

- Rotar
  - **Preguntas**

?- rotar([a],R).

R = [a].

?- rotar([a,b,c,d],R).

R = [b, c, d, a].

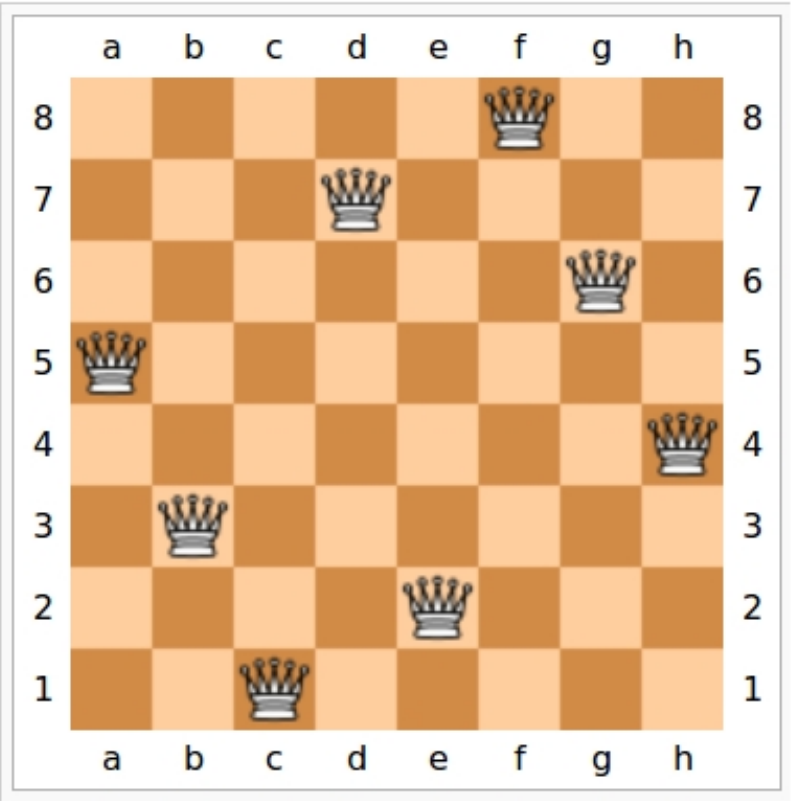
?- rotar([a,b,c,d,e],R).

R = [b, c, d, e, a].

Índice

3. Problema de las ocho reinas

- **Descripción**
  - ❑ Se debe colocar ocho reinas en el tablero de ajedrez sin que se ataquen mutuamente.



Fuente: wikipedia

3. Problema de las ocho reinas

- **Estrategia**
  - Se van a colocar las reinas ordenadas por **filas**
  - Al colocar una nueva reina en un casilla, se comprueba si ataca a las reinas ya colocadas.
    - Si ataca a alguna, se busca otra columna en esa fila.
    - Si no ataca a ninguna reina, el proceso continúa con la colocación de la reina de la siguiente fila.

3. Problema de las ocho reinas

- **Código (1/4)**

```
solución([]).

solución([X, Y | Otras]):-

        solución(Otras),

        pertenece(Y,[1,2,3,4,5,6,7,8]),

        no_ataca([X,Y],Otras).
```

3. Problema de las ocho reinas

- **Código (2/4)**

```
pertenece(X,[X|_]).

pertenece(X,[_|Cola]):-
        pertenece(X,Cola).
```

3. Problema de las ocho reinas

- **Código (3/4)**

```
no_ataca(_,[]).

no_ataca([X,Y],[X1,Y1|Otras]):-

        /* No están en la misma columna */
        Y =\= Y1,

        /* No están en la misma diagonal */
        Y1 - Y =\= X1 - X,
        Y1 - Y =\= X - X1,

        /* No ataca a las demás reinas colocadas */
        no_ataca([X,Y],Otras).
```

3. Problema de las ocho reinas

- **Código (4/4)**
  - Estructura del tablero

```
tablero([1,_,2,_,3,_,4,_,5,_,6,_,7,_,8,_]).
```

  - Escritura de la solución

```
escribir_solucion([]).

escribir_solucion([X, Y | Cola]):-
        write(' Casilla: '), write(X),write(' , '), write(Y),
        nl,
        escribir_solucion(Cola).
```

3. Problema de las ocho reinas

- **Pregunta para obtener las soluciones**

?- tablero(S),solucion(S), escribir_solucion(S).

Casilla: 1 , 4
Casilla: 2 , 2
Casilla: 3 , 7
Casilla: 4 , 3
Casilla: 5 , 6
Casilla: 6 , 8
Casilla: 7 , 5
Casilla: 8 , 1
S = [1, 4, 2, 2, 3, 7, 4, 3, 5|...] ;
…

3. Problema de las ocho reinas

- **Algunas soluciones**

[1, 4, 2, 2, 3, 7, 4, 3, 5, 6, 6, 8, 7, 5, 8, 1] ;

[1, 5, 2, 2, 3, 4, 4, 7, 5, 3, 6, 8, 7, 6, 8, 1] ;

[1, 3, 2, 5, 3, 2, 4, 8, 5, 6, 6, 4, 7, 7, 8, 1] ;

[1, 3, 2, 6, 3, 4, 4, 2, 5, 8, 6, 5, 7, 7, 8, 1] ;

[1, 5, 2, 7, 3, 1, 4, 3, 5, 8, 6, 6, 7, 4, 8, 2] ;

[1, 4, 2, 6, 3, 8, 4, 3, 5, 1, 6, 7, 7, 5, 8, 2] ;

[1, 3, 2, 6, 3, 8, 4, 1, 5, 4, 6, 7, 7, 5, 8, 2] ;

…

Se teclea punto y coma

UNIVERSDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO

PROGRAMACIÓN DECLARATIVA

INGENIGERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE

Tema 10.- Listas