

Listas: Repaso


Modulo Imperativo Programación II - 2025



Listas - Concepto

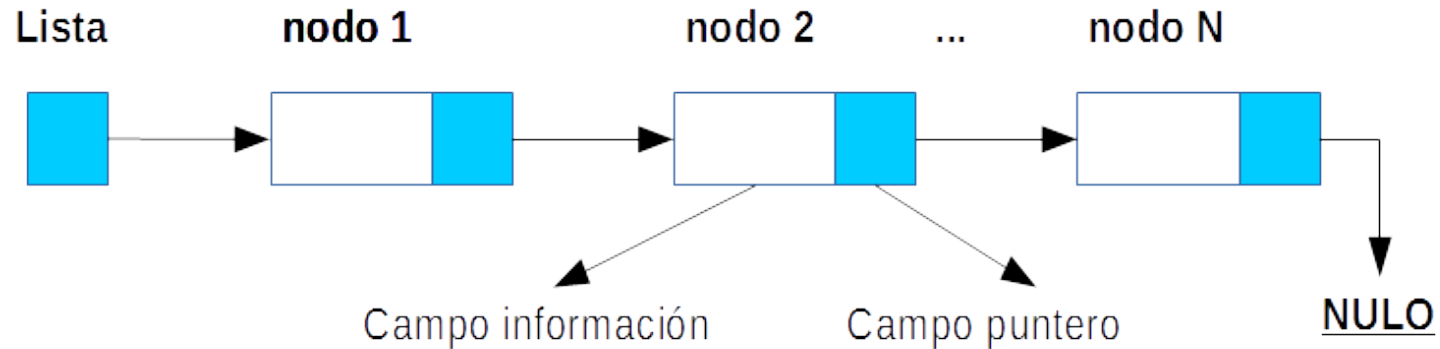
Una lista es Colección de **elementos homogéneos**, con una **relación lineal** que los vincula, es decir que cada elemento tiene un único predecesor (excepto el primero), y un único sucesor (excepto el último).

Los elementos que la componen **no ocupan** posiciones secuenciales o contiguas de memoria. Es decir pueden aparecer dispersos en la memoria, pero mantienen un orden lógico interno.



Listas - Concepto

Lista simplemente enlazada



Listas - Características

- Se crean a partir de **punteros**.
- Son estructuras donde se almacenan datos **sin saber la cantidad de los mismos**.
- Son estructuras **dinámicas**: se reserva/libera memoria para datos según sea conveniente.


Siempre debo **GUARDAR EL PUNTERO INICIAL DE LA LISTA**, es decir el apuntador al primer nodo, PARA LUEGO PODER RECORRERLA, ya que a partir del primer elemento se puede acceder al siguiente y así sucesivamente.



Listas - Operaciones

- **Crear** lista agregando los elementos al inicio
- **Crear** lista agregando los elementos al final
- **Insertar** un nuevo elemento en una lista ordenada
- **Recorrer** una lista
- **Acceder** al k-ésimo elemento de la lista
- **Eliminar** un elemento de la lista
- **Combinar** dos listas ordenadas formando una sola ordenada (**Merge** de Listas)


Repasaremos algunas de estas
operaciones en el ejemplo >>





Listas – Declaración genérica

```
type
  nodo = record
    dato: Integer;
    siguiente: ^nodo;
  end;
type
  listaDinamica = ^nodo;
```



Listas – Insertar al *final*

```
procedure InsertarAlFinal(var lista: listaDinamica; dato: Integer)
var
  nuevoNode, actual: listaDinamica;
begin
  new(nuevoNode);
  nuevoNode^.dato := dato;
  nuevoNode^.siguiente := nil;

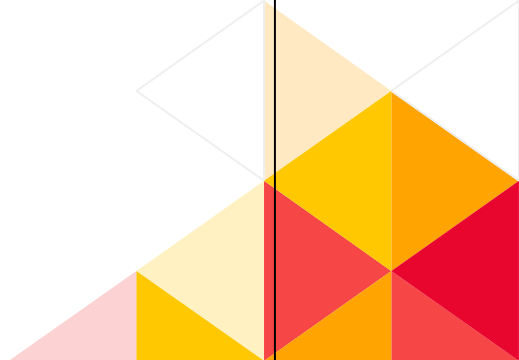
  if lista = nil then
    lista := nuevoNode
  else
    begin
      actual := lista;
      while actual^.siguiente <> nil do
        actual := actual^.siguiente;

      actual^.siguiente := nuevoNode;
    end;
end;
```



Listas – Recorrido de una lista

```
Programa ejemplo;  
var  
  miLista, actual: listaDinamica;  
begin  
  //Falta programa para crea la lista  
  miLista  
  actual := miLista;  
  while actual <> nil do  
    begin  
      writeln(actual^.dato);  
      actual := actual^.siguiente;  
    end;  
  end.
```






Listas – Eliminar un nodo de una lista

```
procedure EliminarNodo(var lista: listaDinamica; dato: Integer);
var
  actual, anterior: listaDinamica;
begin
  actual := lista;
  anterior := nil;

  while (actual <> nil) and (actual^.dato <> dato) do
  begin
    anterior := actual;
    actual := actual^.siguiente;
  end;

  if actual <> nil then
  begin
    if anterior = nil then
      lista := actual^.siguiente
    else
      anterior^.siguiente := actual^.siguiente;

    dispose(actual); // Liberar la memoria del nodo eliminado
  end;
end;
```






Listas - Ejemplo que seguiremos

X dispone de una lista con los tweets realizados durante los últimos 5 segundos.

De cada tweet se conoce: el código y nombre de usuario que lo generó, el contenido del mensaje y si el mismo es o no un retweet.

Esta información no tiene ningún orden y se debe tener en cuenta que podrían existir en la lista varios tweets del mismo usuario.





Listas - Ejemplo que seguiremos

X dispone de una lista con los tweets realizados durante los últimos 5 segundos.

De cada tweet se conoce: el código y nombre de usuario que lo generó, el contenido del mensaje y si el mismo es o no un retweet.

Esta información no tiene ningún orden y se debe tener en cuenta que podrían existir en la lista varios tweets del mismo usuario.

Se desea armar a partir de la lista de tweets disponible, una lista ordenada donde los tweets de cada usuario aparezcan de manera consecutiva.





Listas – Declaración del Ejemplo

Program X;

Type

tweet = record

codigoUsuario :

integer;

nombreUsuario :

string;

mensaje : string;

esRetweet :

boolean;

end;

listaTweets = ^ nodoTweet;

nodoTweet = record

dato : tweet;

sig : listaTweets;

end;





Listas - Ejemplo que seguiremos

¿Cómo se crean los tweets?

¿Cómo se insertan los elementos en la lista que se dispone?



Ver el archivo x.pas



Listas – Agregar al inicio

```
Procedure agregarAdelante ( var l : listaTweets;  
                           t : tweet);
```

```
var
```

```
    aux : listaTweets;
```

```
begin
```

```
    new(aux);  
    aux^.dato := t;
```

→ Creo el nodo y cargo el dato

```
    aux^.sig := l;  
    l := aux;
```

→ Enlazo el siguiente y asigno el nuevo inicio


```
end;
```

Y luego,
¿Cómo podemos generar la nueva estructura?



Listas – Agregar ordenado

Para insertar ordenado en una lista debemos considerar:

- Pedir espacio para el nuevo nodo
 - Guardar el nuevo dato
 - Buscar posición donde se debe insertar (secuencialmente)
 - Reacomodar punteros. Considerando tres casos:
 - El nuevo elemento va en el inicio de la lista.
 - El nuevo elemento va en el medio de dos existentes.
 - El nuevo elemento va al final de la lista.
- 

Listas – Agregar ordenado

```
Procedure agregarOrdenado ( var pri : listaTweets; t : tweet);  
var    nuevo, anterior, actual : listaTweets;  
begin
```

```
  new (nuevo); nuevo^.dato:= n; nuevo^.sig :=
```

Creo el nodo y
cargo el dato

```
  if (pri = nil) then
```

```
    pri := n
```

Primer

elemento

```
  else
```

```
  begin
```

```
    actual := pri;  anterior := pri;
```

```
    while (actual<>nil)and(actual^.dato.nombreUsuario <
```

```
    nuevo^.dato .nombreUsuario)do
```

```
      begin  anterior := actual;
```

```
        actual:= actual^.sig;
```

```
      end;
```

```
      if (anterior = actual)
```

Inserta

```
        pri := nuevo
```

adelante

```
      else
```

Inserta al medio o al final

```
        anterior^.sig := nuevo;
```

```
        nuevo^.sig
```

Actualiza la ref. al siguiente

```
  end;
```

Si la lista no está vacía recorro
hasta encontrar la posición dónde
insertar
(entre anterior y actual)

Listas – Recorrido

Una vez creada la lista, se imprimen sus elementos.

```
Procedure imprimirLista(l: listaTweets);  
begin  
    while (l <> nil) do begin  
        imprimir(l^.dato); ¿Cómo es este  
módulo?  
        l:= l^.sig;  
    end;  
end;
```

¿Por qué paso la lista por valor?
¿Por qué en la condición del while no
escribo (l^.sig <> nil)?
¿Dónde y cómo se llama a este módulo?



Listas – ejercicio de modificación

Modifique la solución anterior para generar una nueva estructura donde se puedan “agrupar” los tweets de manera tal que los datos del usuario no se encuentren repetidos.

¿Qué estructura de datos se podría utilizar para almacenar los tweets de un mismo usuario?

¿Cómo se puede generar la estructura de manera eficiente?

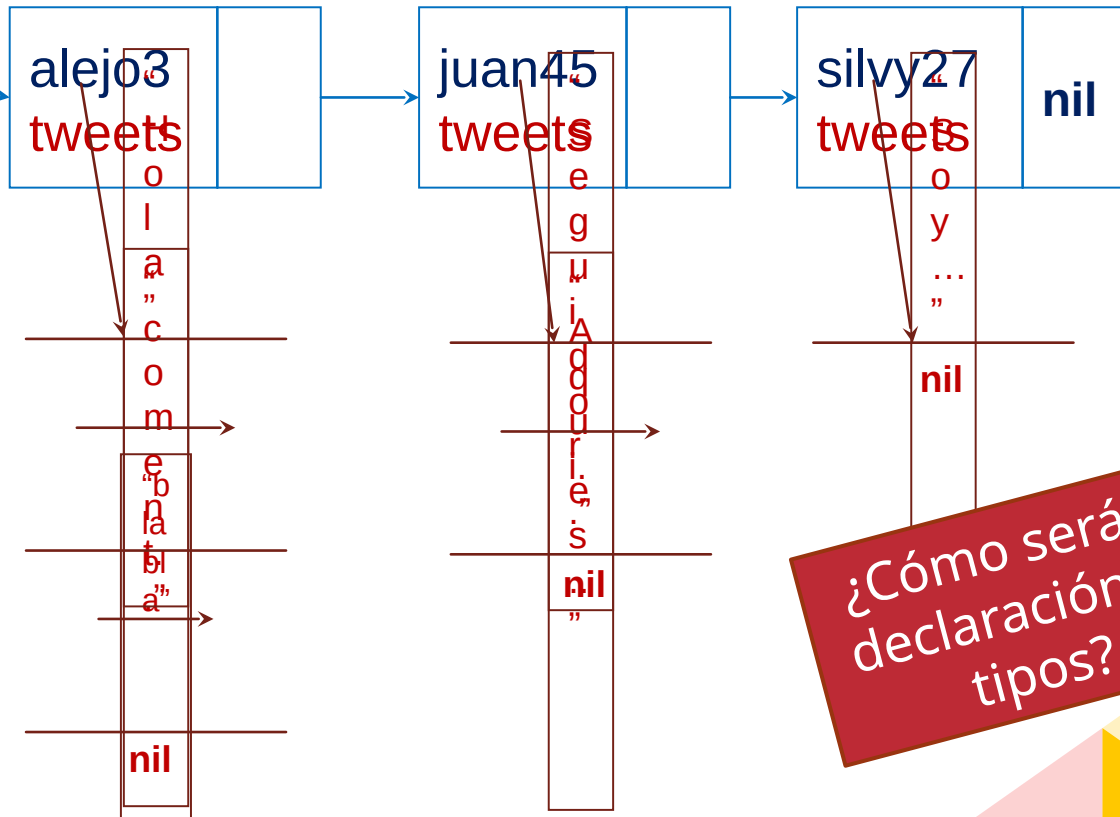


Lista de Listas

En una lista simple donde cada dato almacenado posee la referencia a un nuevo tipo de lista simple interno.

ListaUsuario

Listas de
Tweets de
cada
usuario



¿Cómo será la
declaración de
tipos?



Lista de Listas

¿Cómo se puede generar la estructura de manera eficiente?


Se puede utilizar la lista ordenada para recorrer una única vez todos los tweets.

Tomar los tweets contiguos de un mismo usuario e ir generando la lista interna. Para ello hay que:

Verificar que no se pierdan datos en el proceso de generación de

- Inicializar cada lista interna en nil.
- Detectar cuándo se cambia de usuario en el recorrido de la lista ordenada.
- Agregar cada tweet a la nueva lista.

Una vez armada la lista, completar los datos del usuario y agregarlo a la lista de usuarios.





Merge de Listas

Les dejamos un enlace donde pueden repasar el concepto de Merge de listas

<http://163.10.22.82/OAS/MergeListas/index.htm>
!

