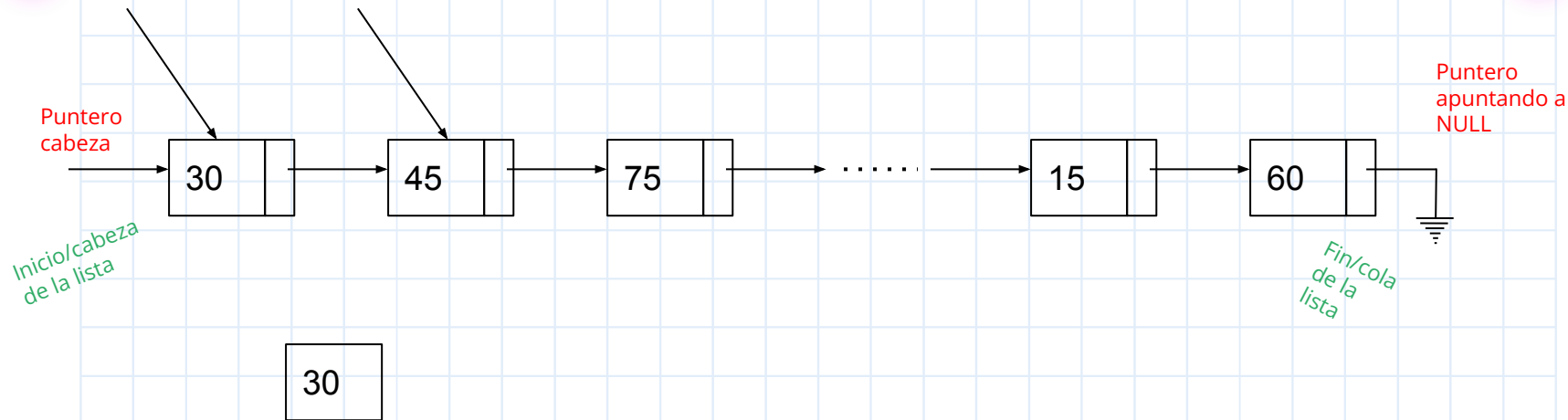


Lista lineal simplemente enlazada



La apariencia de una lista de este tipo es la siguiente:



Donde...

Cada nodo tiene esta estructura

```
struct nodo{
```

```
// Campos...
```

```
// Enlace al siguiente nodo
```

```
};
```

Los campos pueden ser
los que se necesiten.
Por ejemplo:

```
int dato;
```

El campo del puntero
que apunta al siguiente
nodo:

```
struct nodo *sig;
```

Así, tendremos:

```
struct nodo{
```

```
int dato;
```

```
struct nodo *sig;
```

```
};
```

El programa principal

En la función main siempre
declararemos el objeto lista

```
int main(void)
{
    cout << "MANEJO DE LISTAS SIMPLES" << endl << endl;

    ListaSimple objListaSimple;

    // Invocamos las funciones miembro o
    // invocamos un menú.

    cout << endl;
    return 0;
}
```

Las funciones miembro

```
class ListaSimple{  
private:  
    nodo *pInicio;  
  
public:  
    ListaSimple();  
    ~ListaSimple();  
    void insInicio(int);  
    void mostrarListaSimple(void);  
    void mostrarListaRecursiva(void);  
    void mostrarListaRecursivaAux(nodo *);  
    void mostrarListaInversa(void);  
    void mostrarListaInversaAux(nodo *);  
    void insertarFinalLista(void);  
    void insFinal(int dato);  
    void insFinalRec(int);  
    nodo *irUltimoNodoRec(nodo *);  
    void insertarDespuesDeElemento(int, int);  
    void insertarAntesDeElemento(int, int);  
    bool buscarEnListaSimple(int);  
    bool eliminarElemento(int);  
};
```

Están declaradas dentro de una clase

Su tarea es:
Administrar la estructura de datos
(insertar elementos, buscar
elementos, eliminar elementos, ...)

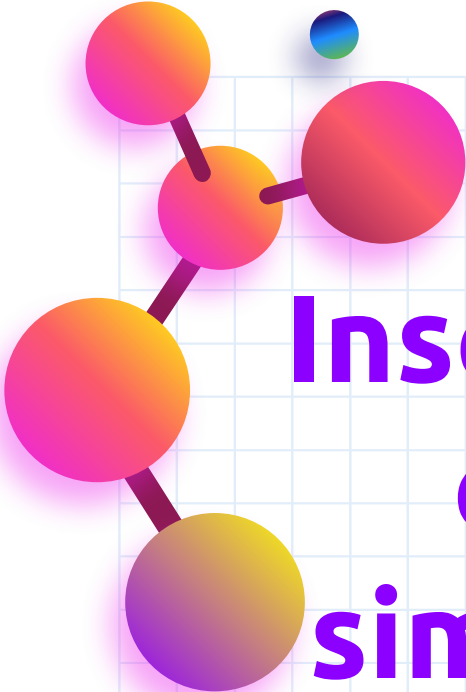
Los cuerpos de las funciones miembro

```
ListaSimple::ListaSimple(void)
{
    ...
}

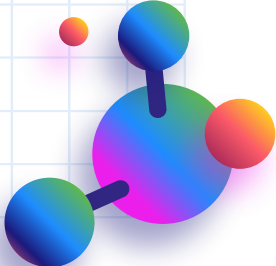
void ListaSimple::insertarAlInicio(void)
{
    ...
}

.
.
.
```

Se ubican entre la definición
de la clase y la función main



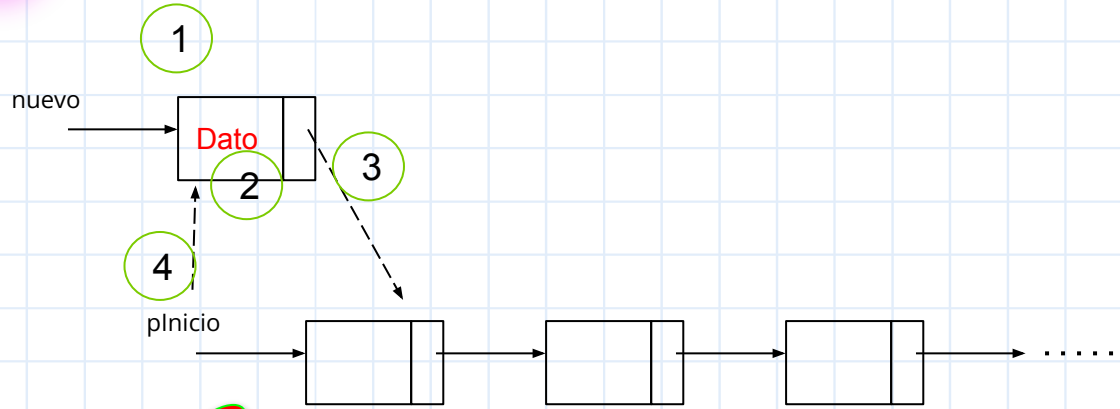
Insertión de elementos en una lista lineal simplemente enlazada



La inserción puede realizarse:

- Al inicio de la lista.
- Al final de la lista.
- En orden ascendente.
- En orden descendente.
- Antes de un elemento determinado.
- Después de un elemento determinado.
- Etc.

La inserción al inicio de la lista



Hay que considerar los siguientes pasos:

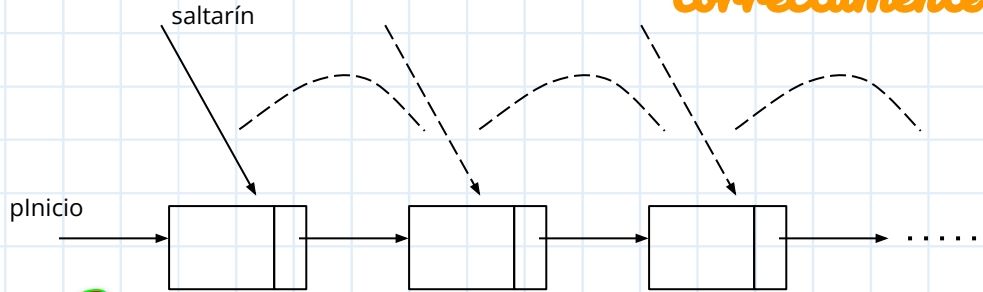
- 1) Crear el nodo.
- 2) Introducir el dato en el nodo.
- 3) Hacer que el puntero del nuevo nodo apunte al primer elemento de la lista.
- 4) Hacer que el puntero cabeza apunte al nuevo nodo creado.

Este mismo esquema funciona si la lista está vacía o hay que hacer otra consideración.

Algoritmo

Mostremos los elementos de la lista

... a ver si es cierto que los insertamos correctamente ...



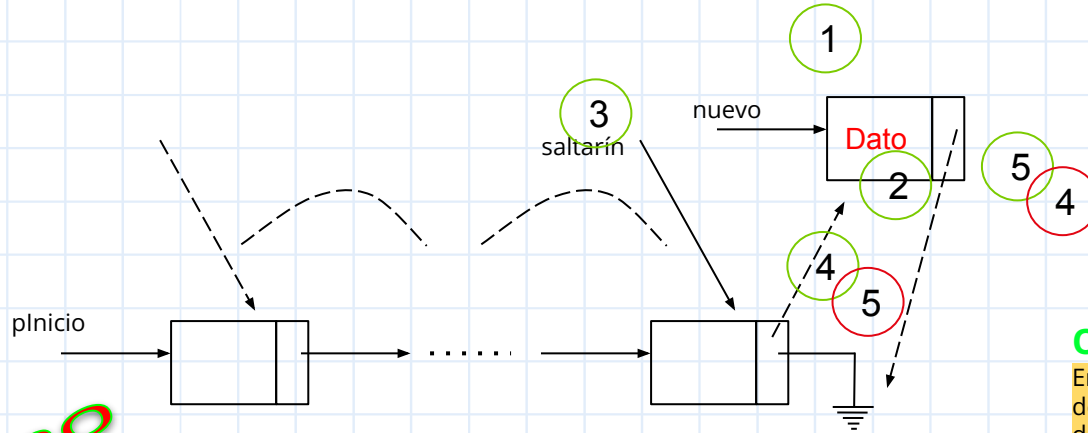
Hay que considerar los siguientes pasos:

- 1) Hacer que el puntero auxiliar apunte al primer nodo de la lista.
- 2) Desplegar el dato del nodo.
- 3) Iterativamente saltar a los consecutivos nodos realizando despliegue de su dato.
- 4) Hasta encontrar el final de la lista.

En qué momento podemos decir que saltarín ha terminado de recorrer toda la lista?

Algoritmo

La inserción al final de la lista



Hay que considerar los siguientes pasos:

- 1) Crear el nodo.
- 2) Introducir el dato en el nodo.
- 3) Llevar un puntero auxiliar hasta el final de la lista, sin salirse de la lista.
- 4) Hacer que el puntero del último nodo de la lista apunte al nuevo nodo.
- 5) Hacer que el puntero del nuevo nodo apunte a NULL.

Este mismo esquema funciona si la lista está vacía o hay que hacer otra consideración

Ojo 1:

En el algoritmo anterior saltarin se salió de la lista. En este algoritmo, saltarin debe permanecer en el último nodo.

Ojo 2:

En este caso, las instrucciones de los pasos 4 y 5 pueden invertirse en el programa.

Ojo 3:

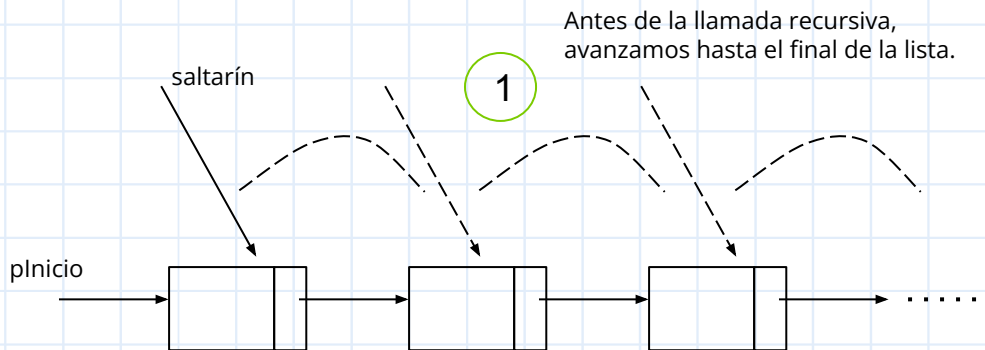
Esto no siempre será posible. Hay que analizar cada problema antes de decidir el orden de las instrucciones.

Mostremos los elementos de la lista en orden inverso

Supongamos que, en una lista lineal simple necesitamos desplegar (recorrer) los nodos al revés.

¿Será que podemos hacerlo aún estando los punteros apuntando en el sentido contrario al que necesitamos?.

La respuesta es que **Sí**. Y debemos utilizar recursión por posposición para lograrlo.

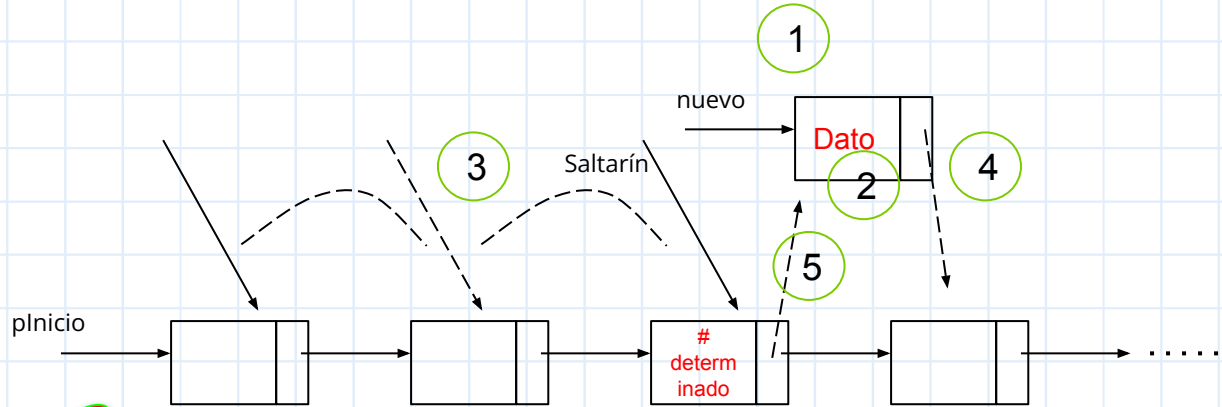


<<====

2

Después de la llamada recursiva, regresamos desplegando.

Inserción después de un elemento determinado



Hay que considerar los siguientes pasos:

- 1) Crear el nodo.
- 2) Introducir el dato en el nodo.
- 3) Saltar hasta ubicarse sobre el nodo que contiene al dato de referencia.
- 4) Hacer que el puntero del nuevo nodo apunte donde apunta el nodo con el dato de referencia.
- 5) Hacer que el puntero del nodo del dato de referencia apunte al nuevo nodo.