



TUTORIA #10

LISTAS

CONCEPTO

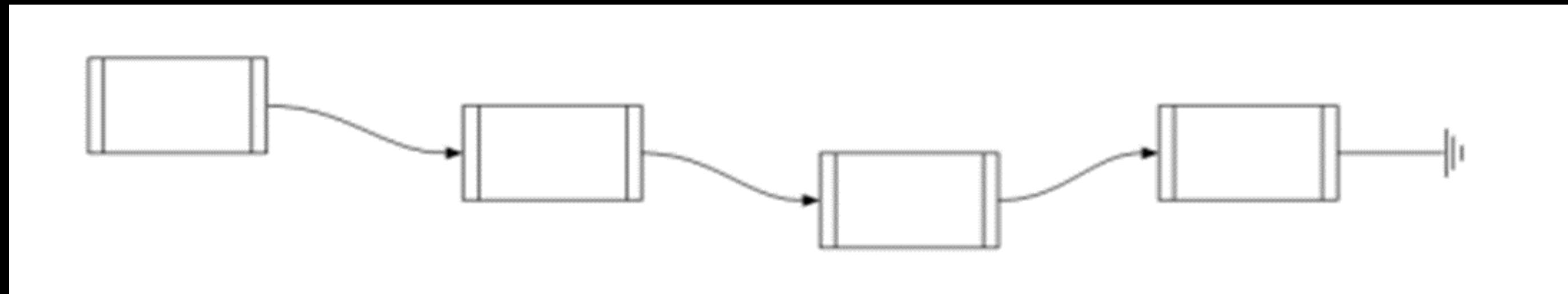


En programación, una lista es un tipo de estructura de datos que permite almacenar y acceder a un conjunto de elementos en un orden específico. Las listas son similares a los arreglos, pero a diferencia de los arreglos, las listas en algunos lenguajes de programación permiten insertar y eliminar elementos en cualquier posición en tiempo de ejecución.

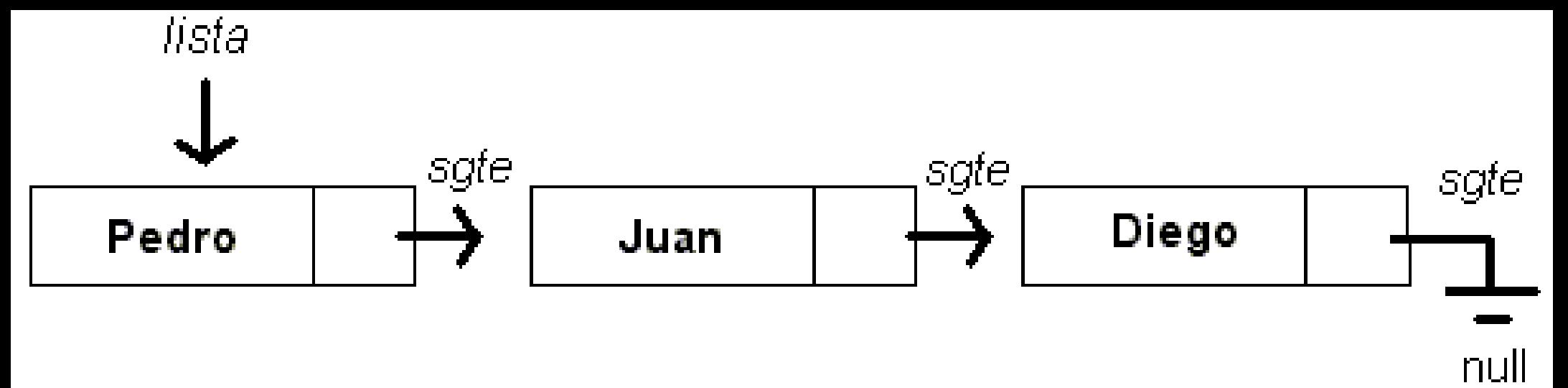
Las listas son una estructura de datos dinámica lo que significa que su tamaño puede cambiar dinámicamente dependiendo de la cantidad de elementos que se agreguen o quiten.



En propias y sencillas palabras una lista es un arreglo dinámico creado a partir de punteros, la cuál va agregando y uniendo datos indefinidamente, hasta que el usuario quiera detenerse o la memoria de la maquina se acabe.



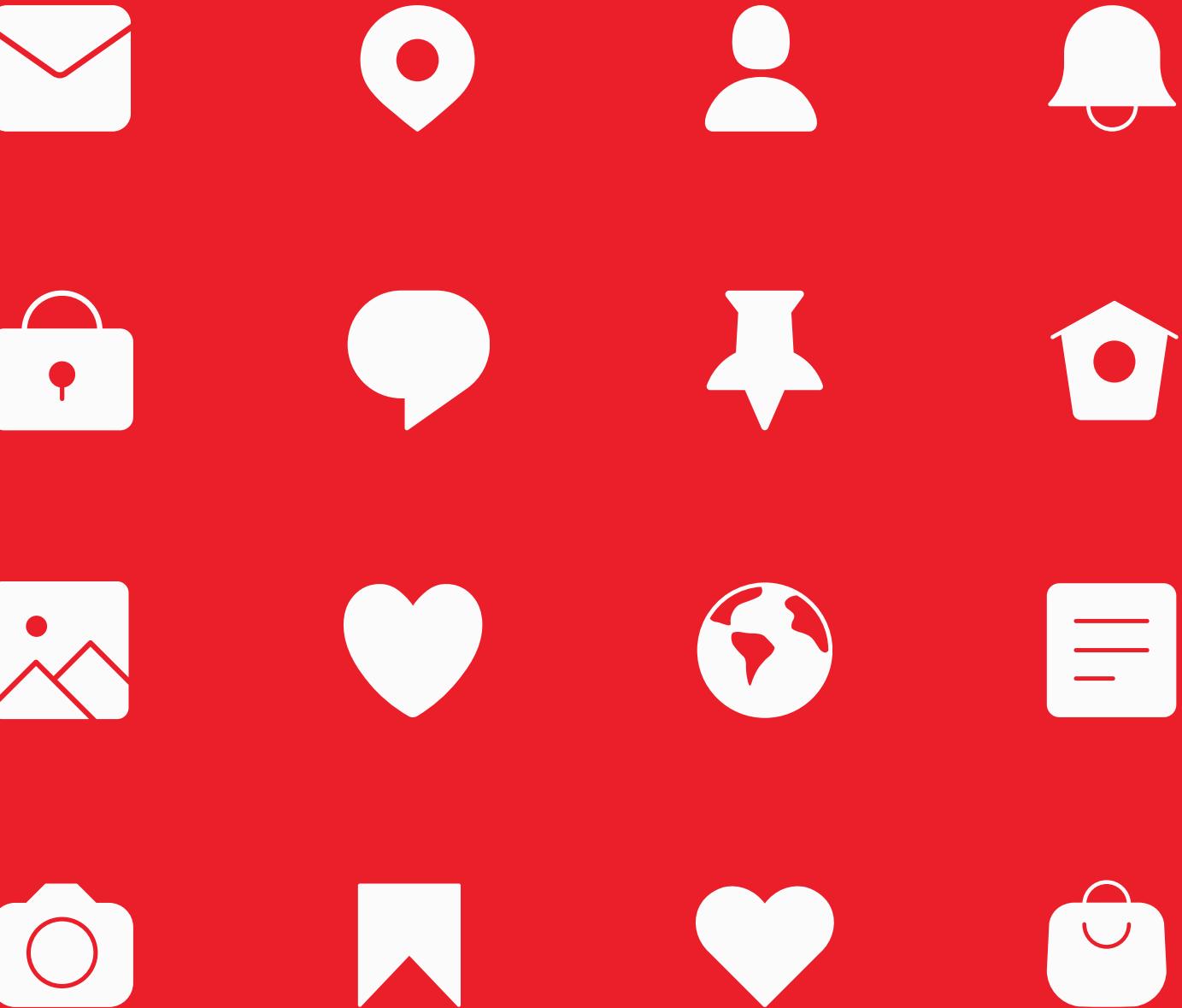
Un nodo en una lista es lo que hace que la lista tenga sentido, es una estructura quien almacena la información que se quiera guardar. Además el nodo también almacena información mediante punteros que le indican quien es su sucesor en el orden de la lista o quien es su antecesor



NODO

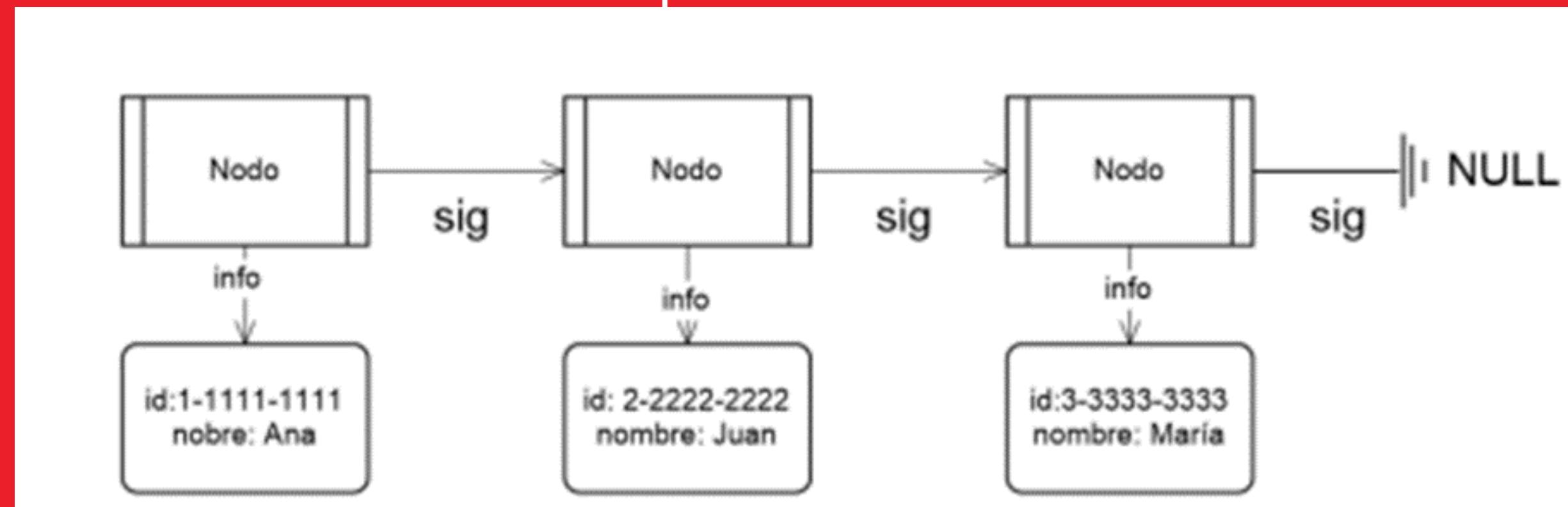
TIPOS DE LISTAS

- Lista línea o simplemente enlazada.
- Lista circular simplemente enlazada.
- Lista doblemente enlazada.
- Lista circular doblemente enlazada.



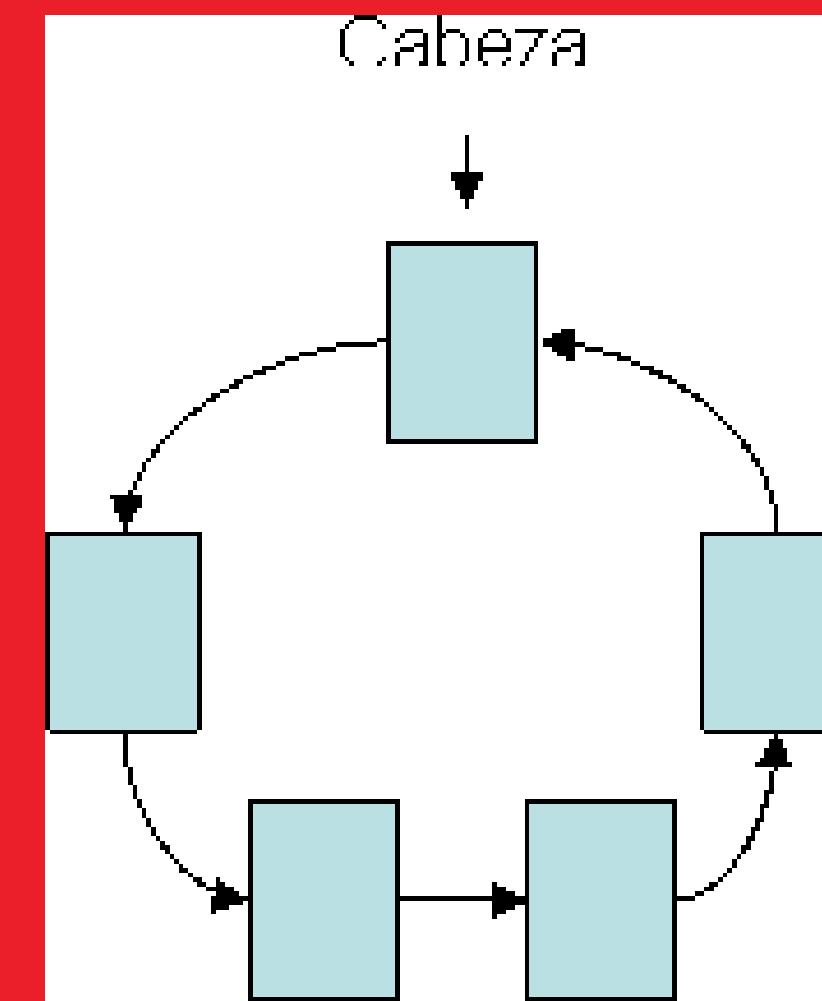
LISTA SIMPLEMENTE ENLAZADA

Estas listas contienen solo un puntero al siguiente nodo.



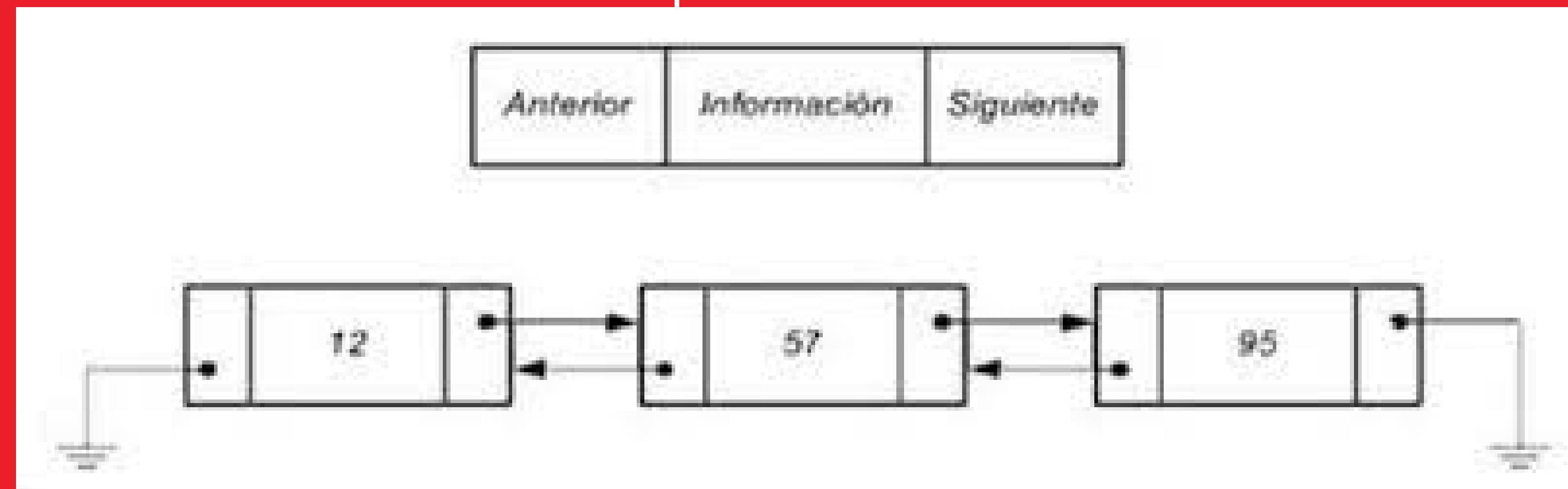
LISTA CIRCULAR SIMPLEMENTE ENLAZADA.

Lista lineal en la que el último nodo apunta al primero



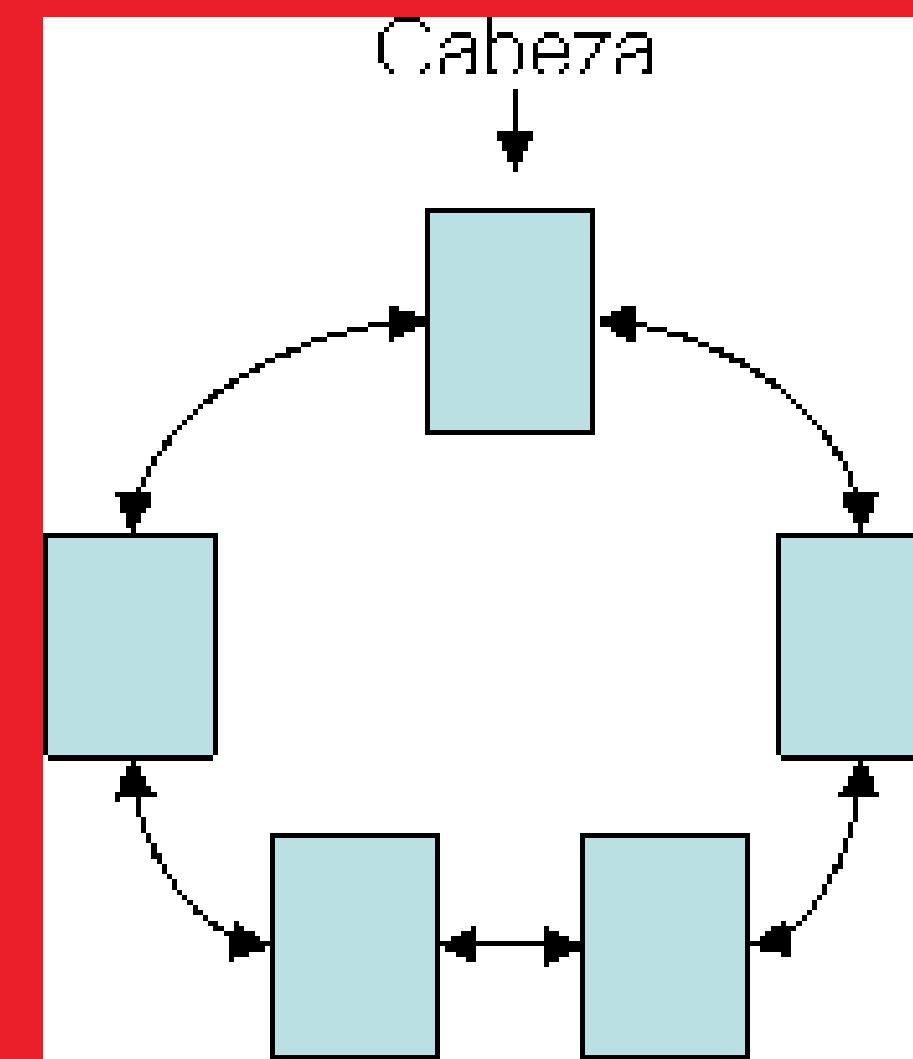
LISTA DOBLEMENTE ENLAZADA..

Estas listas contienen un puntero al siguiente nodo y al anterior.



LISTA CIRCULAR DOBLEMENTE ENLAZADA.

Lista lineal en la que el último nodo apunta al primero. Además de un nodo al siguiente y otro al anterior



- Insertar
- Eliminar
- Impresión
- Busqueda

OPERACIONES



ESTRUCTURA LISTAS

- Plantillas

```
● ● ●  
1 template <typename T>  
2 class List {  
3 private:  
4     struct Nodo {  
5         T number;  
6         Nodo* next;  
7         Nodo(T value) : number(value), next(nullptr) {}  
8     };  
9     Nodo* header;  
10  
11 public:  
12     List() :header(nullptr) {
```

ESTRUCTURA LISTAS

- Clases

```
● ● ●  
1 #include <iostream>  
2 using namespace std;  
3 class Nodo {  
4 private:  
5     Nodo* next;  
6     int value;  
7 public:  
8     Nodo();  
9     int getValue();  
10    Nodo* getNext();  
11    void setValue(int pValue);  
12    void setNext(Nodo* pNext);  
13}  
14};
```

```
● ● ●  
1 #include <iostream>  
2 #include "nodo.h"  
3 class List {  
4 private:  
5     Nodo* header;  
6 public:  
7     List() :header(nullptr) {}  
8     void print();  
9     void insertHeader(int value);  
10    void insertFinal(int value);
```

INSERTAR EN LA CABEZA



```
1 void insertHeader(T value) {  
2     Nodo* newNode = new Nodo(value);  
3     if (!header) {  
4         header = newNode;  
5     }  
6     else {  
7         newNode->next = header;  
8         header = newNode;  
9     }  
10 }
```



```
1 void List::insertHeader(int value) {  
2     Nodo* newNode = new Nodo();  
3     newNode->setValue(value);  
4     if (!header) {  
5         header = newNode;  
6         return;  
7     }  
8     newNode->setNext(header);  
9     header = newNode;  
10 }
```

INSERTAR AL FINAL



```
1 void insertFinal(T value) {  
2     Nodo* newNodo = new Nodo(value);  
3     if (!header) {  
4         header = newNodo;  
5     }  
6     else {  
7         Nodo* aux = header;  
8         while (aux->next) {  
9             aux = aux->next;  
10        }  
11        aux->next = newNodo;  
12    }  
13 }
```



```
1 void List::insertFinal(int value) {  
2     Nodo* newNodo = new Nodo();  
3     newNodo->setValue(value);  
4     if (!header) {  
5         header = newNodo;  
6         return;  
7     }  
8     Nodo* aux = header;  
9     while (aux->getNext()) {  
10        aux = aux->getNext();  
11    }  
12    aux->setNext(newNodo);  
13 }
```

ELIMINAR LISTA



```
1 void deleteList() {  
2     if (header) {  
3         while (header) {  
4             Nodo* aux;  
5             aux = header;  
6             header = aux->next;  
7             aux->next=nullptr;  
8             delete aux;  
9         }  
10    }  
11 }
```



```
1 void List::deleteList() {  
2     if (header) {  
3         Nodo* aux;  
4         while (header) {  
5             aux = new Nodo();  
6             aux = header;  
7             header = aux->getNext();  
8             aux->setNext(nullptr);  
9             delete aux;  
10        }  
11    }  
12 }
```