

UNIDAD I. ESTRUCTURA DE DATOS BÁSICAS (LISTAS)

Francisco J. Hernández López

fcoj23@cimat.mx



LISTAS

- Estructura de datos lineal en la que cada elemento solo dispone de un puntero, el cual apuntará al siguiente elemento de la lista o valdrá NULL si es el último elemento
- Existe solo un puntero para acceder a la lista, cuando este puntero es NULL, entonces la lista está vacía.



Nota: lista siempre apunta al nodo inicial

Operaciones:

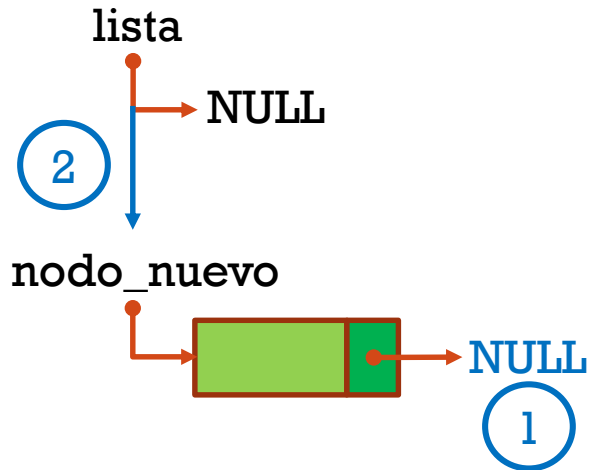
- Agregar un elemento a la lista
- Leer y eliminar un elemento de la lista

IMPLEMENTACIÓN DE LISTAS USANDO MEMORIA DINÁMICA

- Uso eficiente de la memoria sin overflow
- Cada elemento necesita un espacio más en memoria para guardar el apuntador al siguiente elemento
- Underflow → Si la lista está vacía y se intenta eliminar un elemento

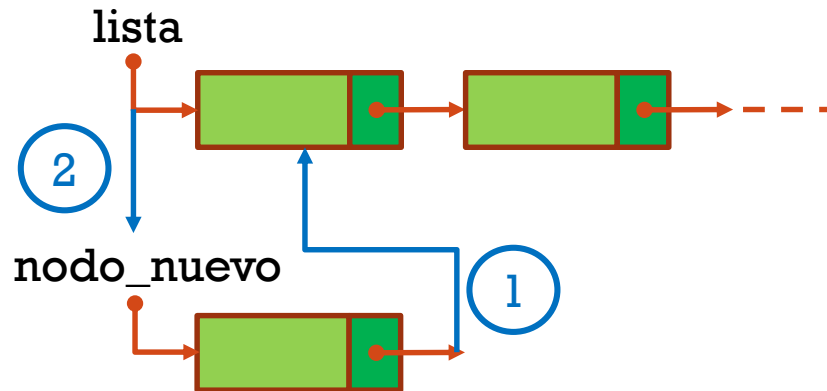
```
typedef struct _nodo{  
    int valor;  
    struct _nodo *siguiente;  
}tipoNodo;
```

INSERTAR UN ELEMENTO EN UNA LISTA VACÍA



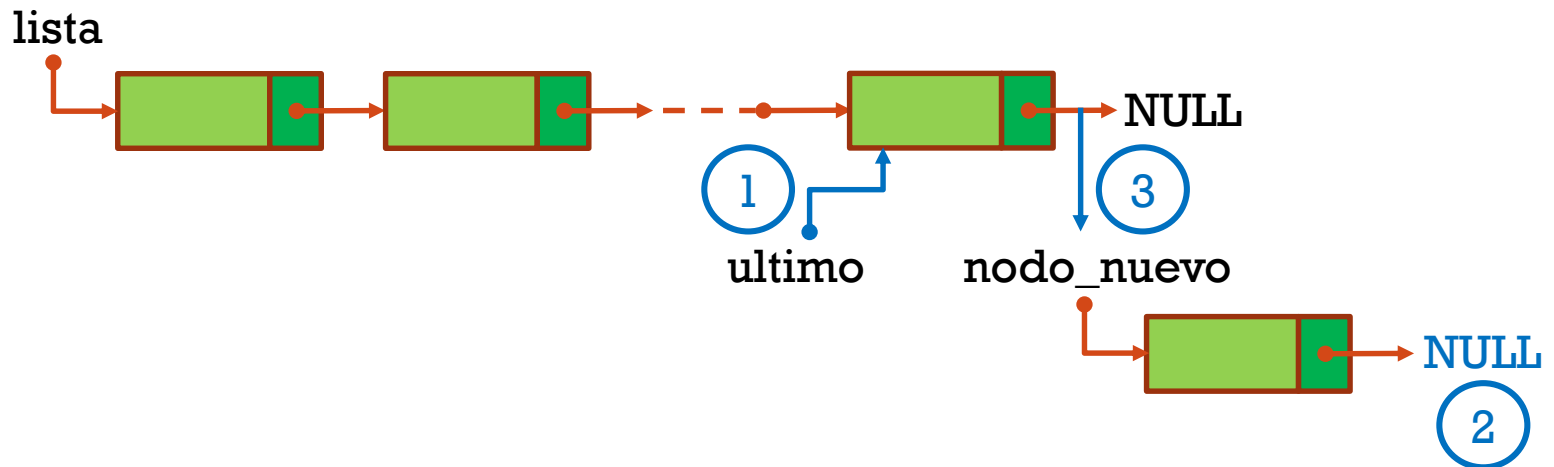
1. `nodo_nuevo` → `siguiente` = `NULL`
2. `lista` = `nodo_nuevo`

INSERTAR UN ELEMENTO EN LA PRIMERA POSICIÓN DE UNA LISTA



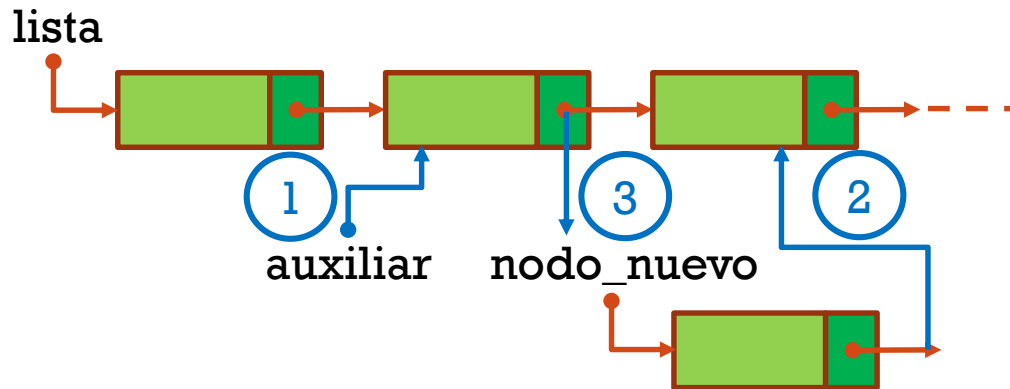
1. `nodo_nuevo`→siguiente = lista
2. lista = `nodo_nuevo`

INSERTAR UN ELEMENTO EN LA ÚLTIMA POSICIÓN DE UNA LISTA NO VACÍA



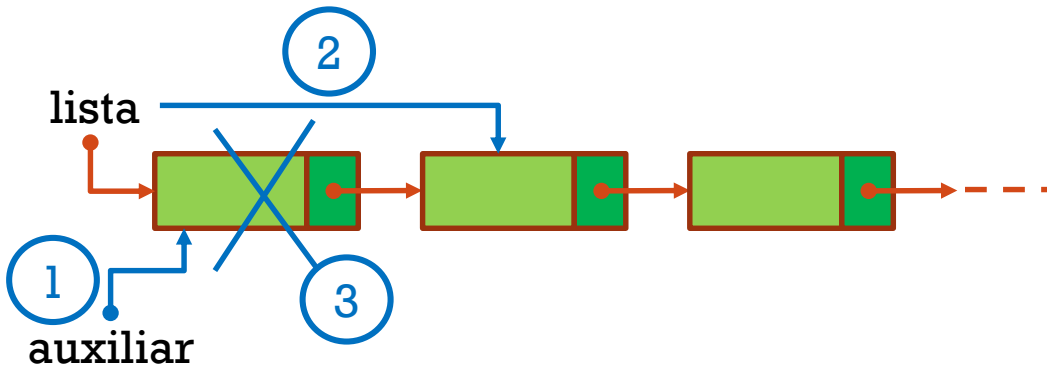
1. Crear un apuntador “ultimo” y avanzarlo desde “lista” hasta que $\text{ultimo} \rightarrow \text{siguiente} = \text{NULL}$
2. $\text{nodo_nuevo} \rightarrow \text{siguiente} = \text{NULL}$
3. $\text{ultimo} \rightarrow \text{siguiente} = \text{nodo_nuevo}$

INSERTAR UN ELEMENTO A CONTINUACIÓN DE UN NODO CUALQUIERA DE UNA LISTA



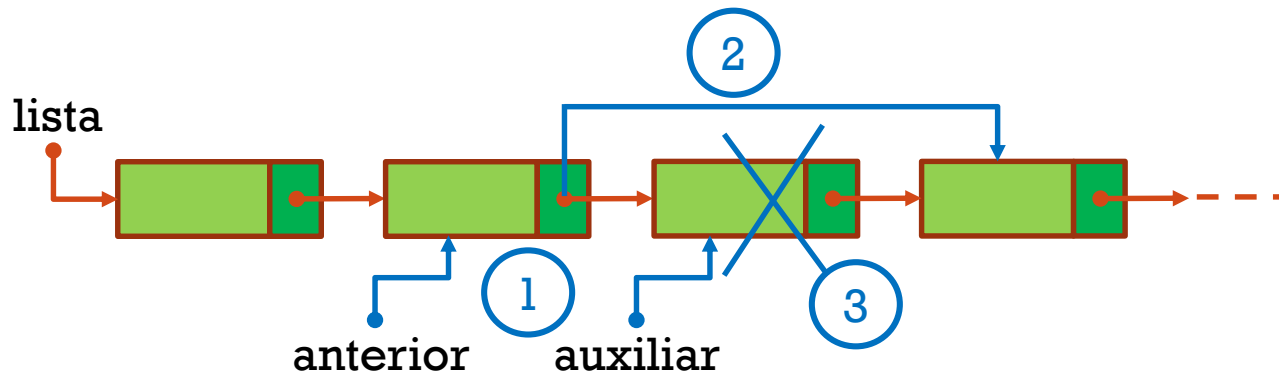
1. Localizar el nodo “auxiliar” donde queremos agregar el nuevo elemento
2. $nodo_nuevo \rightarrow siguiente = auxiliar \rightarrow siguiente$
3. $auxiliar \rightarrow siguiente = nodo_nuevo$

ELIMINAR EL PRIMER NODO DE UNA LISTA



1. auxiliar=lista
2. lista = lista → siguiente
3. Liberar memoria apuntada por “auxiliar”

ELIMINAR UN NODO CUALQUIERA DE UNA LISTA

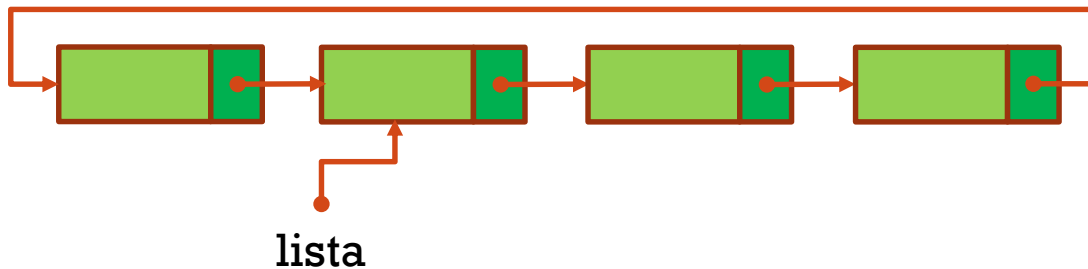


1. Necesitamos 2 apuntadores:
 - a) auxiliar: apunta al nodo que se va a eliminar
 - b) anterior: apunta a un nodo antes del auxiliar
2. anterior → siguiente = auxiliar → siguiente
3. Liberar memoria apuntada por “auxiliar”

PROGRAMAR UNA LISTA

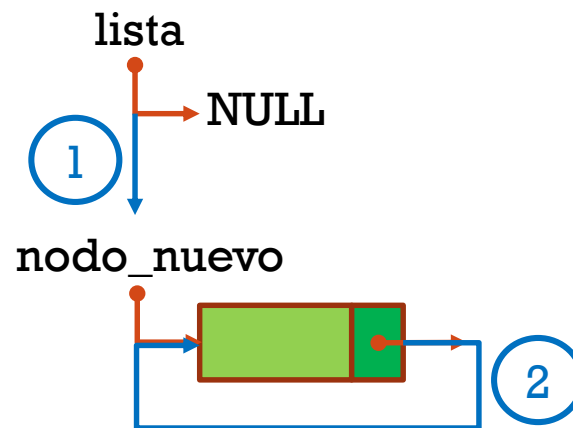
LISTAS CIRCULARES

- Son listas ligadas en las cuales el último nodo apunta al primero



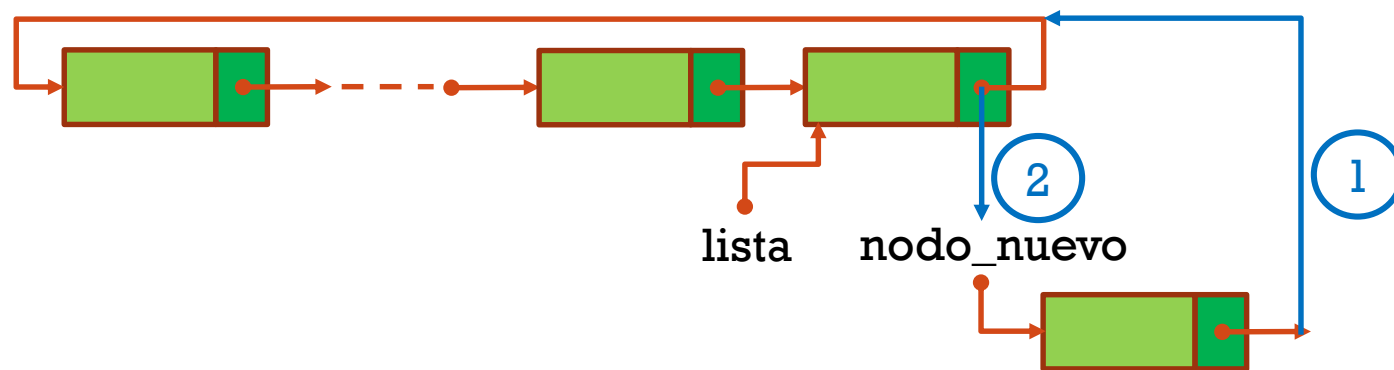
- Son listas sin comienzo, ni final, por lo que “lista” puede apuntar a cualquier nodo
- No existe el apuntador a NULL, solo cuando la lista está vacía

AÑADIR UN ELEMENTO EN UNA LISTA CIRCULAR VACÍA



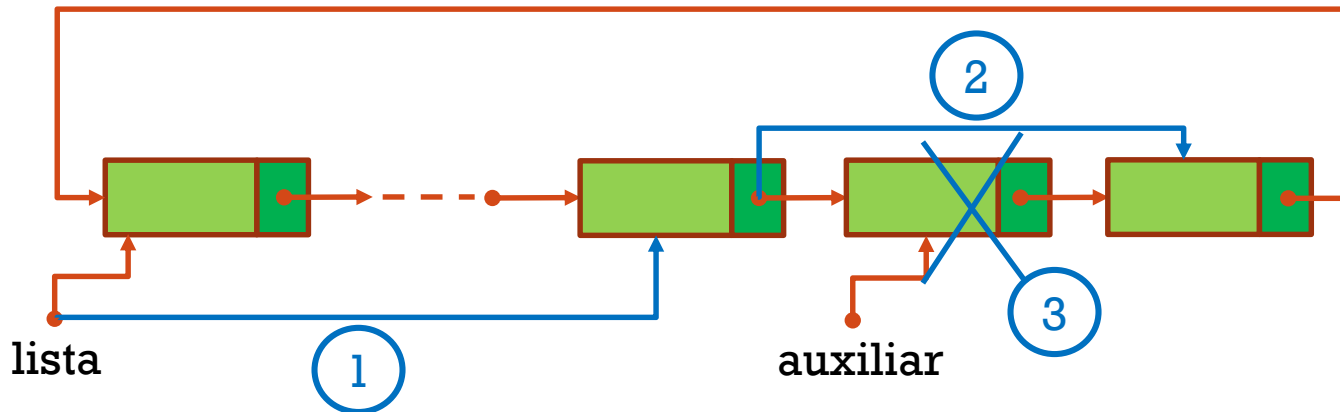
1. `lista = nodo_nuevo`
2. `lista→siguiente=nodo_nuevo`

AÑADIR UN ELEMENTO EN UNA LISTA CIRCULAR NO VACÍA



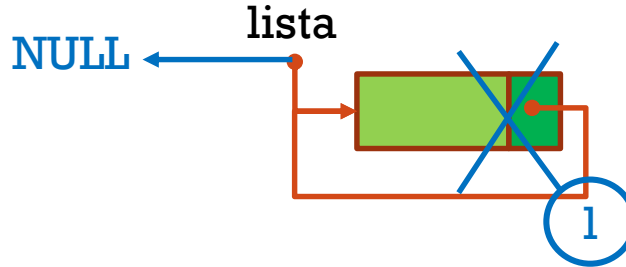
1. `nodo_nuevo` → siguiente = `lista` → siguiente
2. `lista` → siguiente = `nodo_nuevo`

ELIMINAR UN NODO EN UNA LISTA CIRCULAR CON MÁS DE UN ELEMENTO



1. Hacemos que “lista” apunte a un nodo anterior al que queremos eliminar (“auxiliar”)
2. lista → siguiente = auxiliar → siguiente
3. Eliminar la memoria apuntada por auxiliar

ELIMINAR EL ÚNICO NODO DE LA LISTA CIRCULAR



1. Eliminar la memoria apuntada por lista
2. lista = NULL

PROGRAMAR UNA LISTA CIRCULAR

PROGRAMAR EL JUEGO “LA PAPA SE QUEMA...”

- Tenemos N jugadores formando un círculo
- Tenemos otro jugador (extra), fuera del círculo que comienza a decir:
 - La papa se quema, la papa se quema,
 - Cada que menciona la papa se quema, los jugadores del círculo se van pasando una pelota en cierto orden
- Cuando el jugador extra menciona “La papa se quemó”, entonces:
 - El jugador que tenga la pelota, se retira del círculo y le deja la pelota al siguiente jugador
- El juego termina cuando solo queda un jugador en el círculo, el cual es el vencedor