

Operaciones Sobre Listas Enlazadas Simples

Sesión 3

Operaciones Sobre Listas Simplemente Enlazadas

Inserciones

- Al Inicio de la lista
- Al Final de la Lista
- En una Posición Especifica

Eliminaciones

- Eliminar el primer nodo
- Eliminar el último Nodo
- Eliminar un nodo de una posición específica

Búsqueda y Recorrido

Operaciones básicas con listas

Con las listas tendremos un pequeño repertorio de operaciones básicas que se pueden realizar:

- Añadir o insertar elementos.
- Buscar o localizar elementos.
- Borrar elementos.
- Moverse a través de una lista, anterior, siguiente, primero.

Cada una de estas operaciones tendrá varios casos especiales, por ejemplo, no será lo mismo insertar un nodo en una lista vacía, o al principio de una lista no vacía, o la final, o en una posición intermedia.

Insertar elementos en una lista simplemente enlazada

Insertar un elemento en una lista vacía

- Se parte que se tiene el nodo a insertar y un puntero que apunte a él, además el puntero a la lista valdrá NULL:

Lista vacía

El proceso es muy simple, bastará con que:

1. nodo->siguiente apunte a NULL.
2. Lista apunte a nodo.

Pasos para crear un algoritmo con Listas Enlazadas Simples

1. Definir la estructura del nodo:

```
struct nodo{  
    int dato;  
    struct nodo *siguiente;  
};
```

Se crea una nueva estructura de nombre “nodo” el cual contiene los

miembros:

- dato = tipo entero, donde se almacenará un valor numérico entero
- *siguiente= puntero de tipo **nodo**, donde se almacenará la dirección de memoria del siguiente nodo de la lista o NULL si es el último

1. Crear una estructura de NODO

- Se debe crear la estructura de los elementos de la lista (Nodo)
- En C++ se define como global de la siguiente forma

```
struct nodo{  
    int nro;  
    struct nodo *sgte;  
};
```

2. Crear un Tipo de Dato de la estructura del NODO

- Se debe designar un tipo de dato (*Tlista) según la estructura del nodo creado (Se puede crear tipo puntero o no)
- Con este tipo de dato se pueden posteriormente declarar nuevas variables de este tipo de dato

```
typedef struct nodo *Tlista;
```

OPERACIÓN DE ALTA (Inserción de Elementos)

3. Elaborar una función para Insertar un elemento tipo NODO.

- Insertar al Inicio de la lista
- Insertar al Final de la lista
- Insertar en una Posición de la lista

Pasos para crear un algoritmo con Listas Enlazadas Simples

2. Declaraciones de tipos de datos para manejar listas en C

```
typedef struct  nodo {  
    int dato;  
    struct  nodo *siguiente;  
} tipoNodo;  
  
typedef tipoNodo *pNodo;  
typedef tipoNodo *Lista;
```

Se define un nuevo tipo de dato llamado **tipoNodo**, el cual tiene la estructura de **nodo**. Con ese nuevo tipo de dato (tipoNodo) se declaran las variables puntero **pNodo** y **Lista**. **pNodo** es el tipo para declarar punteros a un nodo. **Lista** se utilizará para declarar listas que contendrá los nodos.

Es muy importante que nuestro programa nunca pierda el valor del puntero al primer elemento, ya que si no existe ninguna copia de ese valor, y se pierde, será imposible acceder al nodo y no podremos liberar el espacio de memoria que ocupa.

Insertar elementos

- **Insertar un elemento en una lista vacía**

Partiremos de que ya tenemos el nodo a insertar y, por supuesto un puntero que apunte a él, además el puntero a la lista valdrá NULL:

El proceso es muy simple, bastará con que:

- **nodo->siguiente** apunte a **NULL**.
- **Lista** apunte a **nodo**

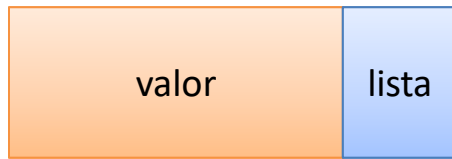
OPERACIÓN DE ALTA (Inserción de Elementos)

3. Elaborar una función para Insertar un elemento tipo NODO.

- Insertar al Inicio de la lista
- Insertar al Final de la lista
- Insertar en una Posición de la lista

Insertar al Inicio de la lista

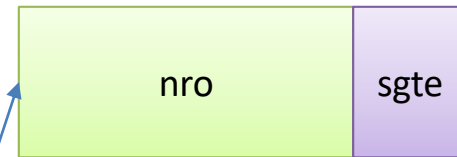
Nuevo Elemento



q

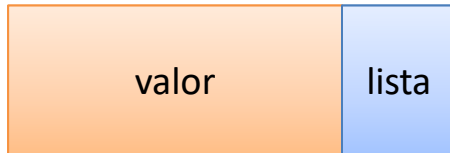
A blue arrow points from the label q to the 'valor' section of the new element node.

Lista Actual



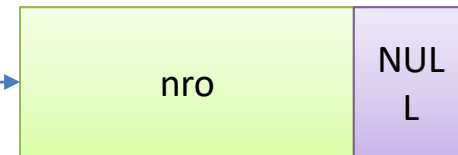
lista

A blue arrow points from the label 'lista' to the 'nro' section of the current list node.



lista = q

A blue arrow points from the label 'lista = q ' to the 'valor' section of the new element node.



OPERACIÓN DE ALTA (Inserción al Inicio de la Lista)

Función main()

```
int main()
{
    Tlista lista = NULL;
    int _dato; // elemento a ingresar

    cout<< "\n NUMERO A INSERTAR: "; cin>>
    _dato;

    insertarInicio(lista, _dato);

    system("pause");
    return 0;
}
```

OPERACIÓN DE ALTA (Inserción al Inicio de la Lista)

Se envía el parámetro *lista* por referencia y el *valor* del nodo

```
void insertarInicio(Tlista &lista, int valor)
{
    Tlista q;
    q = new(struct nodo);
    q->nro = valor;
    q->sgte = lista;
    lista = q;
}
```

Insertar elementos

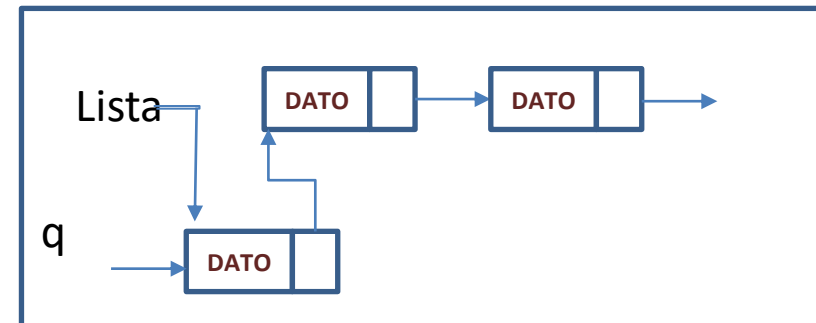
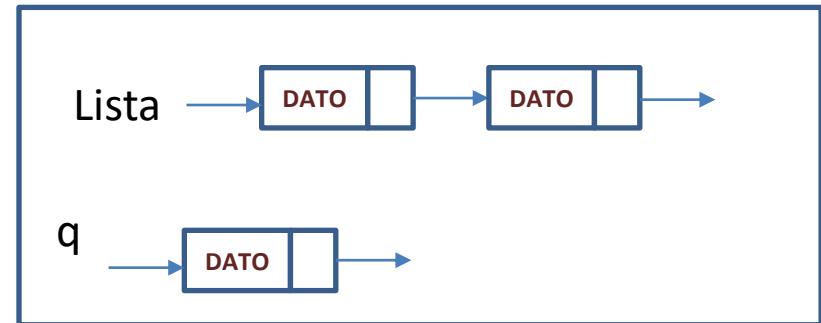
Insertar un elemento en la primera posición de una lista

Partiremos de un nodo a insertar, con un puntero que apunte a él, y de una lista, en este caso no vacía:

El proceso:

- Hacemos que **nodo->siguiente** apunte a **Lista**.
- Hacemos que **Lista** apunte a **nodo**.

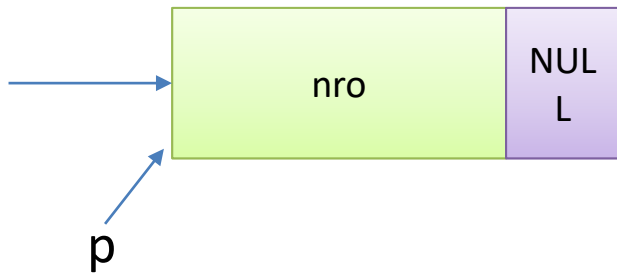
```
Void insertalInicio(Lista &lista, int valor)
{
    Lista q;
    q = new(struct nodo);
    q->dato = valor;
    q->sgte = lista;
    lista = q;
}
```



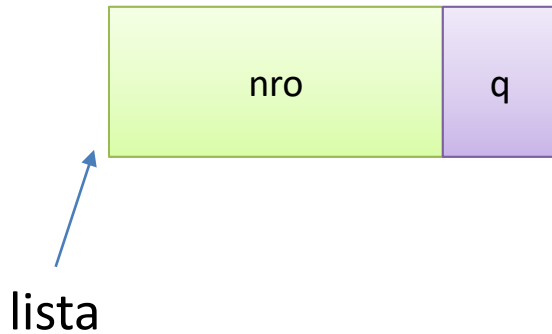
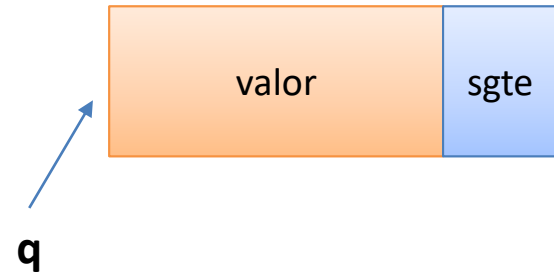
Insertar al Final de la lista

Recorrer los nodos de la lista hasta llegar al último nodo

Lista Actual



Nuevo Elemento



OPERACIÓN DE ALTA (Inserción al Final de la Lista)

Función main()

```
int main()
{
    Tlista lista = NULL;
    int _dato; // elemento a ingresar

    cout<< "\n NUMERO A INSERTAR: "; cin>> _dato;
    insertarFinal(lista, _dato );

    system("pause");
    return 0;
}
```

OPERACIÓN DE ALTA (Inserción al Final de la Lista)

```
void insertarFinal(Tlista &lista, int valor)
{
    Tlista t, q = new(struct nodo);

    q->nro = valor;
    q->sgte = NULL;

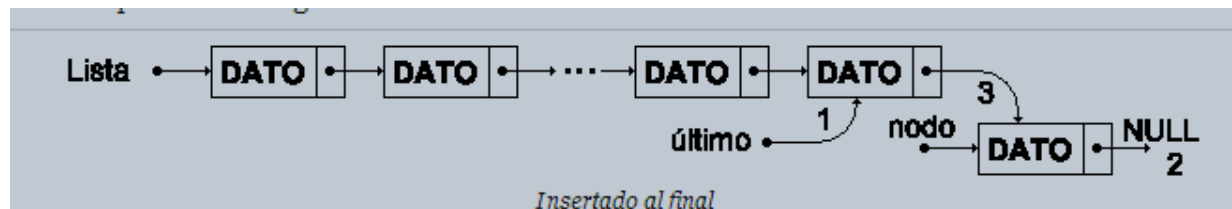
    if(lista==NULL)
    {
        lista = q;
    }
    else
    {
        t = lista;
        while(t->sgte!=NULL)
        {
            t = t->sgte;
        }
        t->sgte = q;
    }
}
```

Insertar un elemento en la última posición

- Se parte de considerar una lista no vacía:

Insertar al final

- Necesitamos un puntero que señale al último elemento de la lista. La manera de conseguirlo es empezar por el primero y avanzar hasta que el nodo que tenga como siguiente el valor NULL.
1. Hacer que nodo->siguiente sea NULL.
 2. Hacer que ultimo->siguiente sea nodo.



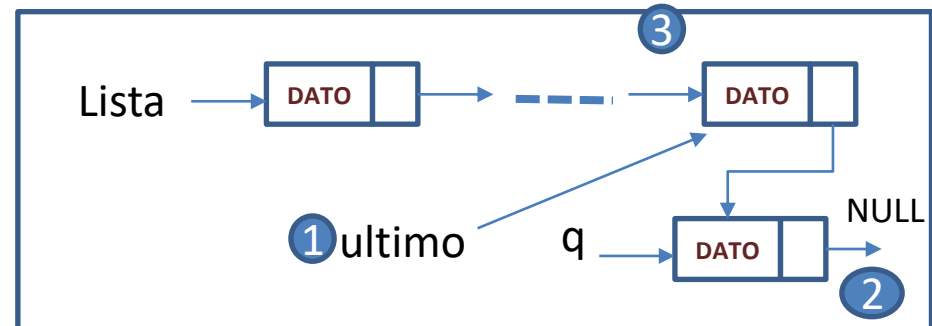
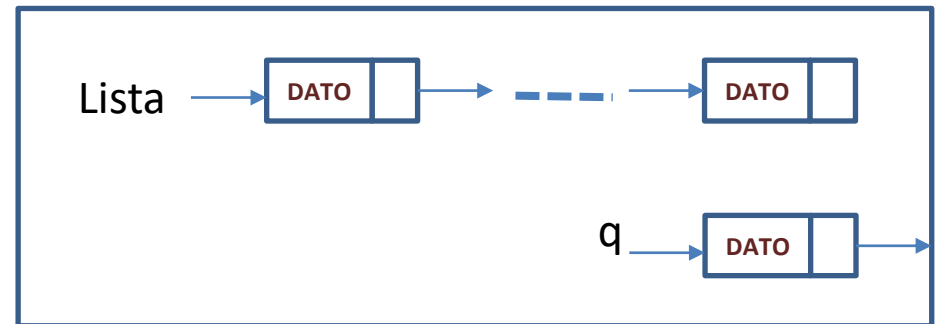
Insertar elementos

Insertar un elemento en la ultima posición de una lista

Partiremos de una lista no vacía

```
void insertarFinal(Lista &lista, int valor)
{
    Lista ultimo, q = new(struct nodo);
    q->nro = valor;
    q->sgte = NULL;
    if(lista==NULL)
    {
        lista = q;
    }
    else
    {
        ultimo = lista;
        while(ultimo ->sgte!=NULL)
        {
            ultimo = ultimo ->sgte;
        }
        ultimo ->sgte = q;
    }
}
```

1. Necesitamos un puntero que señale al último elemento de la lista. La manera de conseguirlo es empezar por el primero y avanzar hasta llegar al nodo que tenga como siguiente el valor NULL.
2. Hacer que nodo->siguiente del nuevo nodo sea NULL.
3. Hacer que ultimo->siguiente tenga la dirección del nuevo nodo (q).



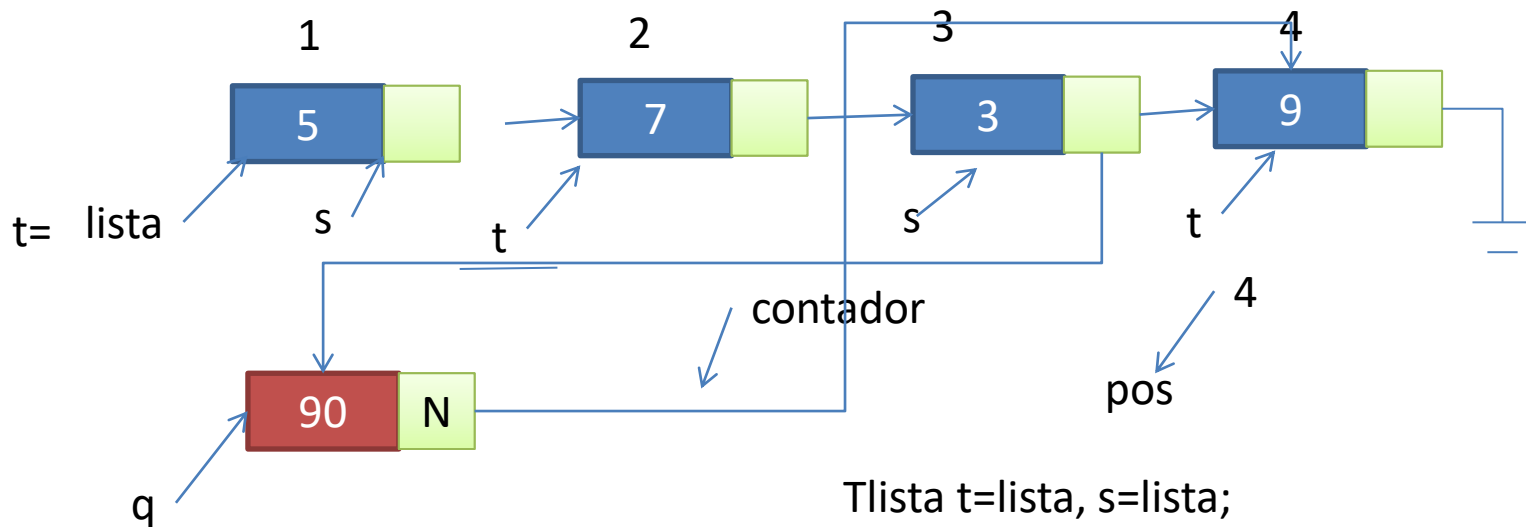
Insertar elemento en Posición

Funcion Main()

```
int main()
{
    Tlista lista = NULL;
    int _dato; // elemento a ingresar

    cout<< "\n NUMERO A INSERTAR: ";cin>> _dato;
        cout<< " Posicion : ";    cin>> pos;
        insertarElemento(lista, _dato, pos);

    system("pause");
    return 0;
}
```



int n=1

```
Tlista t=lista, s=lista;
If (pos==1)
    insertar_inicio (lista, valor);
Else
```

```
while (( t->sgte != NULL) &&(n !=pos)){
    s=t;
    t=t->sgte;
    n++;
}
s->sgte=q;
q->sgte = t;
```

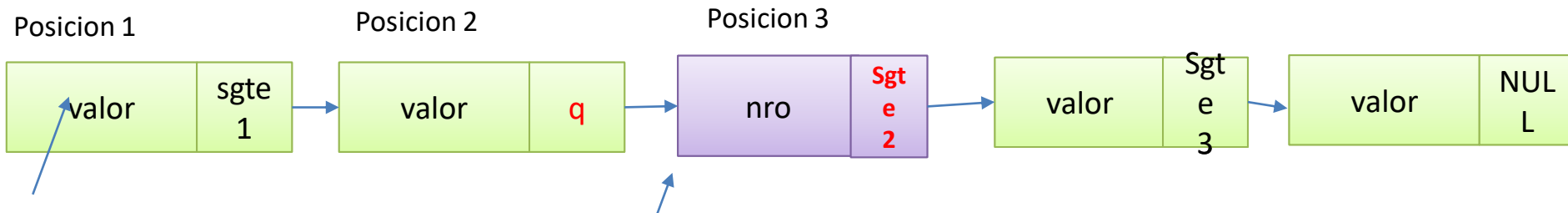
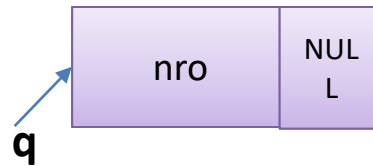
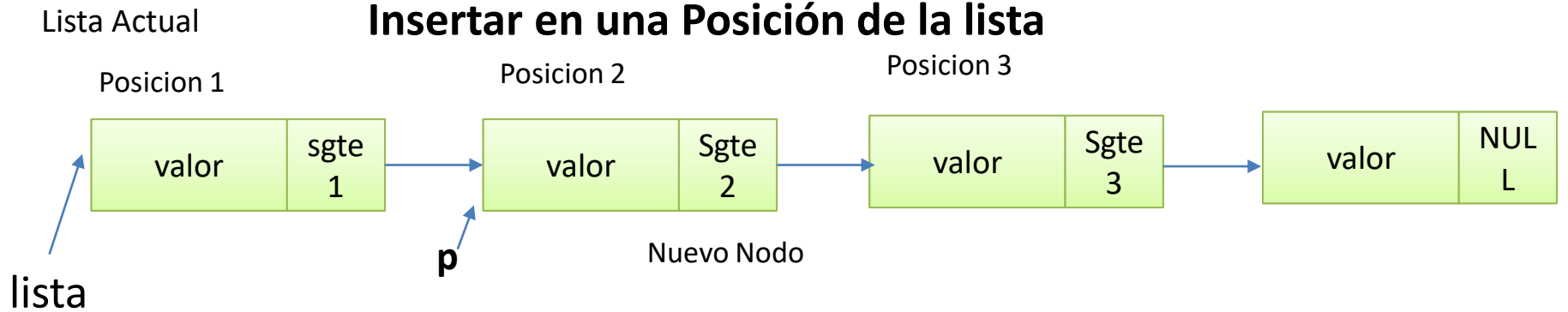
Insertar un nodo en una posición específica

```
void insertarElemento(Tlista &lista, int valor, int pos)
{
    Tlista q, t;
    int i;
    q = new(struct nodo);
    q->nro = valor;

    if(pos==1)
    {
        q->sgte = lista;
        lista = q;
    }
    else
    {
        int x = insertarAntesDespues();
        t = lista;

        for(i=1; t!=NULL; i++)
        {
            if(i==pos+x)
            {
                q->sgte = t->sgte;
                t->sgte = q;
                return;
            }
            t = t->sgte;
        }
    }
    cout<<" Error...Posicion no encontrada..!"<<endl;
}
```

Insertar en una Posición de la lista



Insertar un elemento a continuación de un nodo cualquiera de una lista

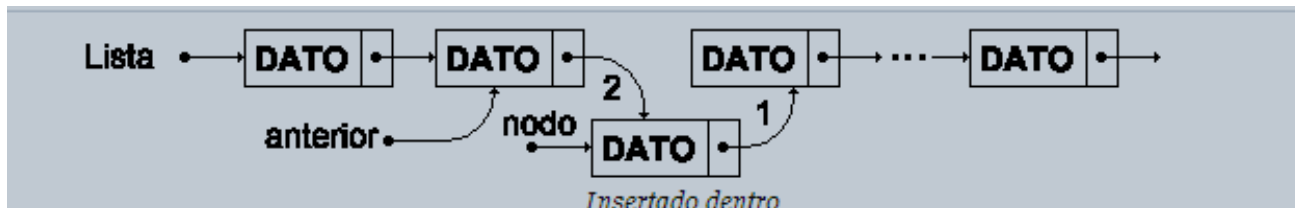
- Ahora el nodo "anterior" será aquel a continuación del cual insertaremos el nuevo nodo:

Insertar dentro

- Suponemos que ya disponemos del nuevo nodo a insertar, apuntado por nodo, y un puntero al nodo a continuación del que lo insertaremos.

El proceso a seguir será:

- Hacer que nodo->siguiente señale a anterior->siguiente.
- Hacer que anterior->siguiente señale a nodo.



Insertar elementos

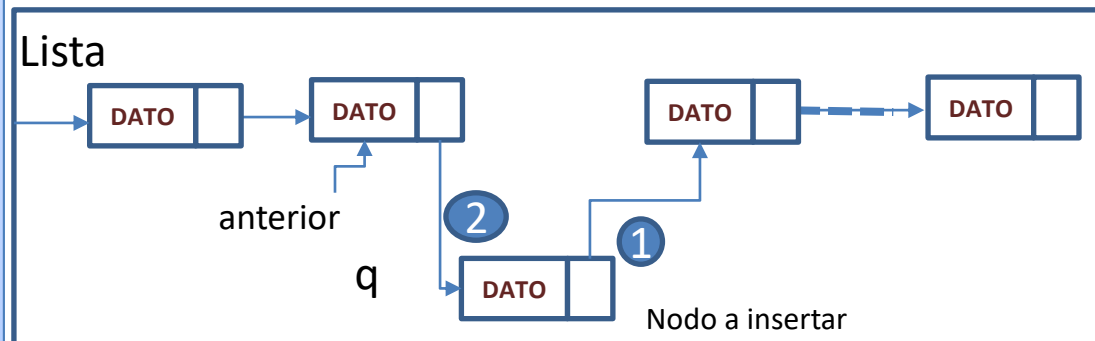
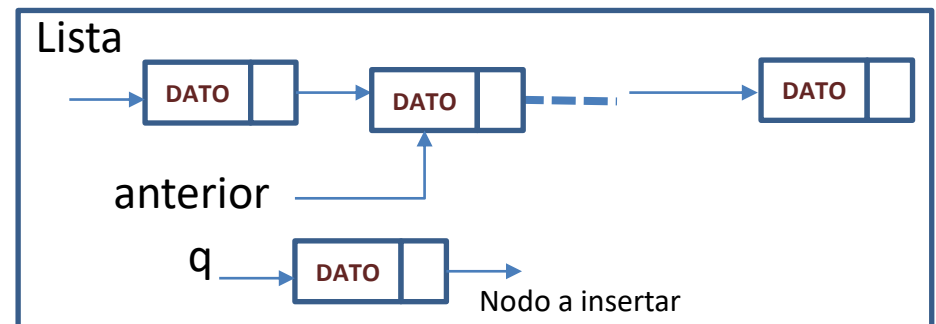
Insertar un elemento a continuación de un nodo cualquiera de una lista

```
void insertarElemento(Lista &lista, int
valor, int pos)
{
    Lista q, t;
    int i;
    q = new(struct nodo);
    q->nro = valor;
    /* Si se elige insertar en la posicion 1 */
    if(pos==1)
    {
        q->sgte = lista;
        lista = q;
    }
    else
    {
        t = lista;
        for(i=1; t!=NULL; i++)
        {
            if(i==pos)
            {
                q->sgte = t->sgte;
                t->sgte = q;
                return;
            }
        }
    }
}
```

Suponemos que ya disponemos del nuevo nodo a insertar, apuntado por **nodo**, y un puntero al nodo a continuación del que lo insertaremos.

El proceso a seguir será:

1. Hacer que **nodo->siguiente** señale a **anterior->siguiente**.
2. Hacer que **anterior->siguiente** señale a **nodo**.



Insertar Nodo al Final de la Lista

Funcion: **insertarFinal**

1.- Crear un nuevo elemento tipo nodo (**q**) y se le asigna espacio de memoria

2.- Se cargan a los miembros del nuevo elemento $q \rightarrow \text{nro} = \text{valor}$
 $q \rightarrow \text{sgte} = \text{NULL}$



```
void insertarFinal(Tlista &lista, int valor)
{
    Tlista t, q = new(struct nodo);

    q->nro = valor;
    q->sgte = NULL;
```

Insertar Nodo al Final de la Lista

Funcion: **insertarFinal**

3.- Se verifica si la Lista inicial (lista) esta vacía, es decir si es NULL

lista: Contenido de la nueva variable tipo nodo creado en el proceso main() con valor NULL

&lista: Dirección de memoria del nuevo elemento **lista**

Si lista=NULL
entonces

se carga a la lista con el nuevo elemento q

Sino

Se recorre la lista hasta llegar al ultimo elemento

Se crea un nuevo elemento auxiliar **t**, con valor igual a la lista original el cual servirá para hacer el recorrido

Se busca el elemento de la lista t que tenga el valor NULL


(indica el ultimo elemento) y al miembro sgte se le asigna la dirección de memoria del nuevo elemento **q**

Insertar Nodo al Final de la Lista

Funcion: **insertarFinal**

```
if(lista==NULL)
{
    lista = q;
}
else
{
    t = lista;
    while(t->sgte!=NULL)
    {
        t = t->sgte;
    }
    t->sgte = q;
}
```



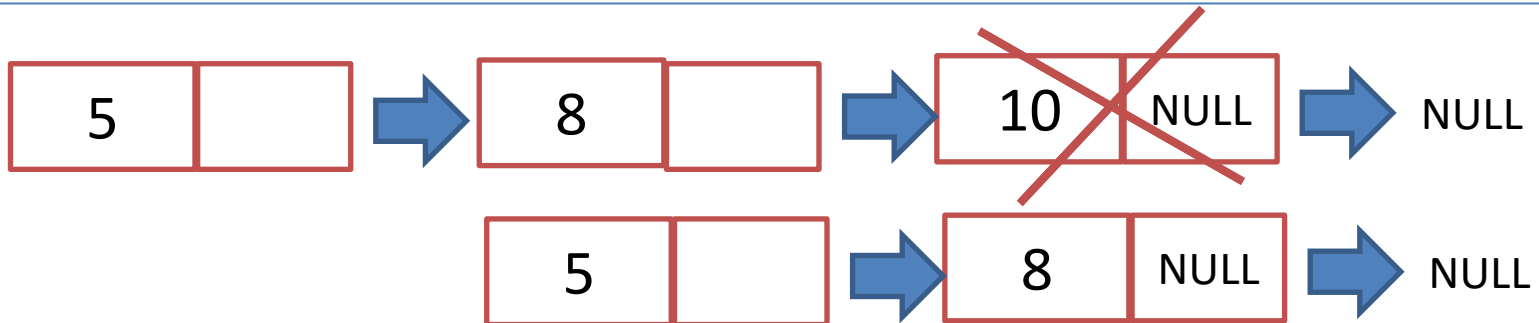
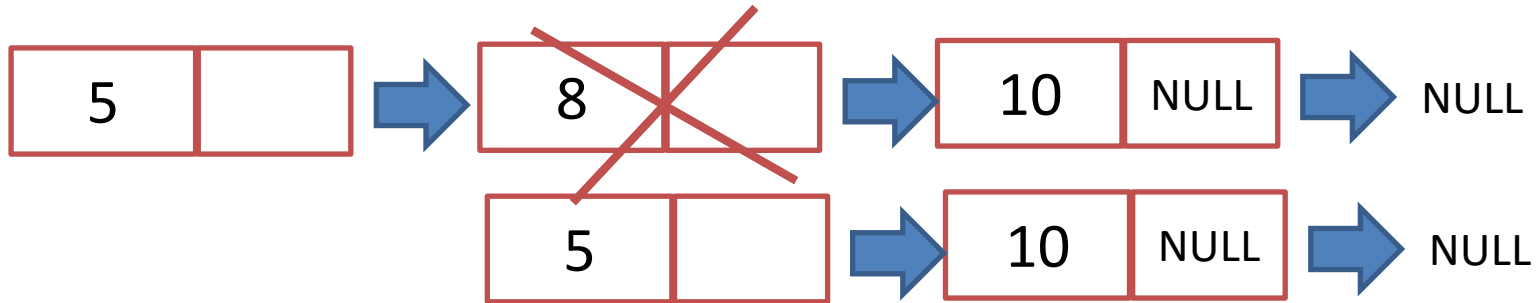
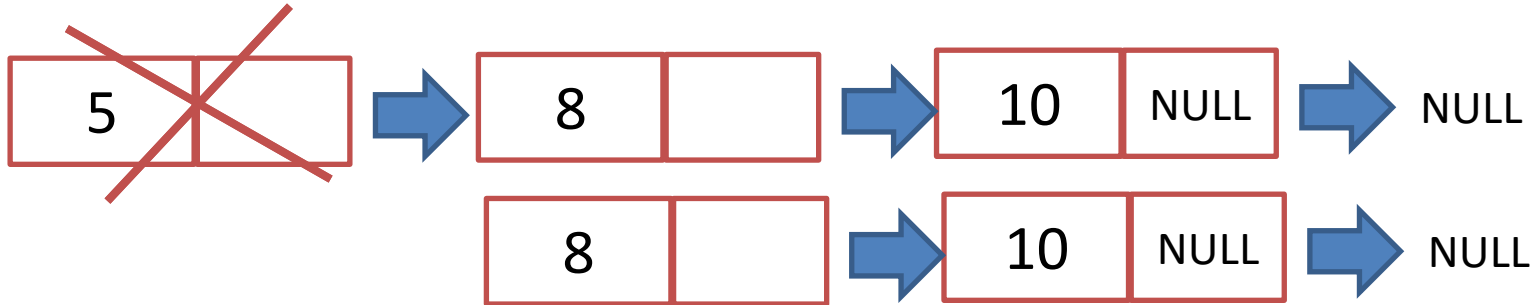


Eliminación de Nodos

- Eliminar el primer Nodo
- Eliminar el último Nodo
- Eliminar un Nodo por posición

Eliminar Nodo de la Lista

Funcion: **eliminarElemento**



Operador DELETE()

- En C + +, el operador **delete** se llama destructor del argumento dado, y libera de la memoria asignada por New. Un llamado a eliminar se debe hacer para cada llamado a NEW para evitar una pérdida de memoria.

```
int *p_var = NULL;    // new pointer declared
p_var = new int;      // memory dynamically allocated

/* .....
other code
.....*/

delete p_var;         // memory freed up
p_var = NULL;         // pointer changed to 0 (null pointer)
```


Eliminar elementos en una lista simplemente enlazada

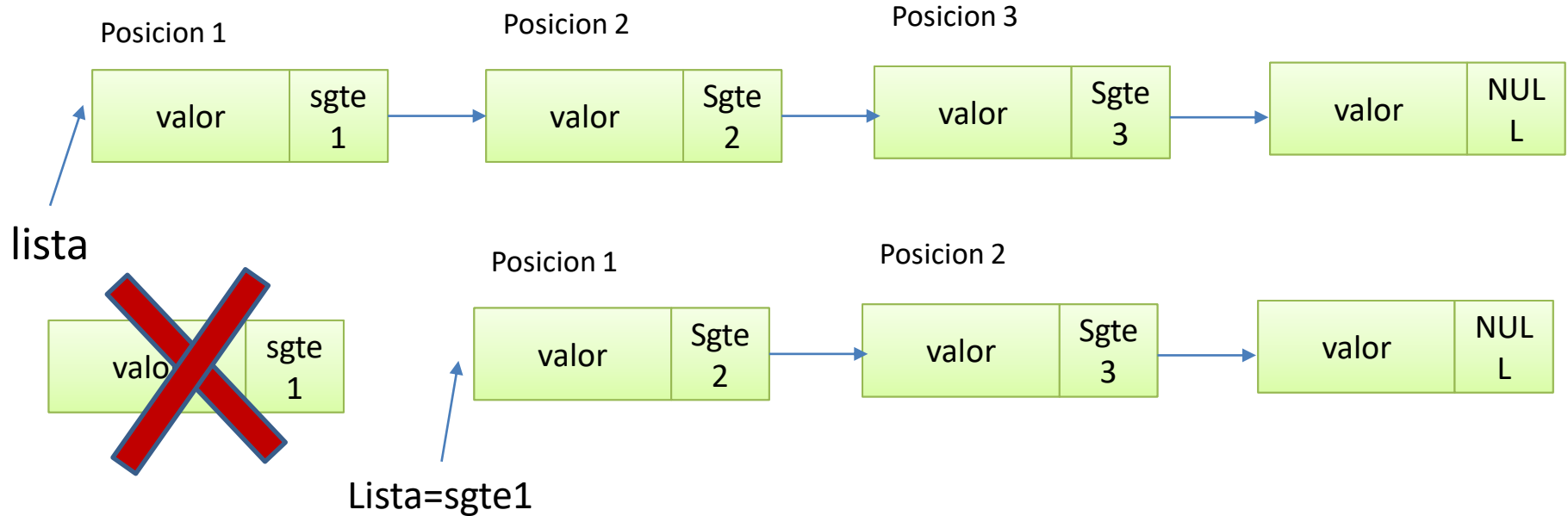
Eliminar un elemento en una lista

- Se deberá verificar que el nodo exista (posición o valor)

La eliminación del nodo puede ser

1. El primer nodo de la lista.
2. El último nodo de la lista.
3. Un nodo en una posición específica
4. Uno o varios nodos por valor

Eliminación del primer nodo de la lista

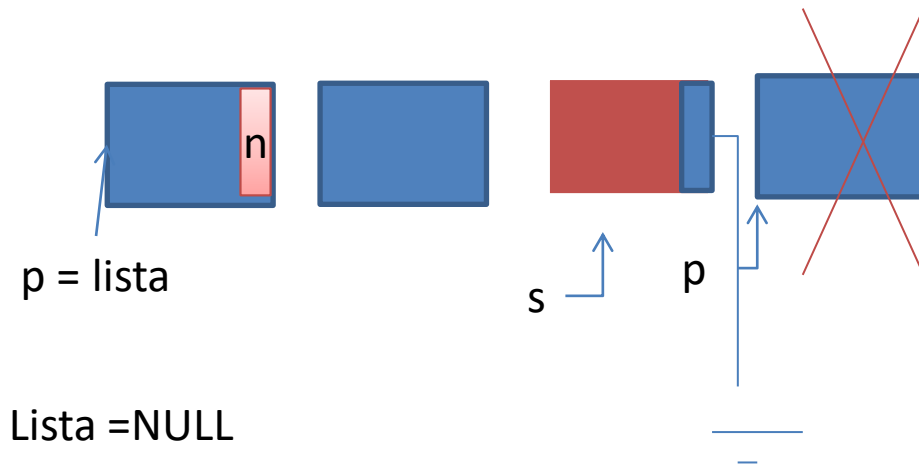


Eliminar nodo Inicio

```
void Eliminar_Inicio(Tlista &lista){  
    tLista t=lista;  
    If(lista !=NULL){  
        lista = lista ->sgte;  
        delete( t);  
    }  
}
```

Eliminar nodo Final

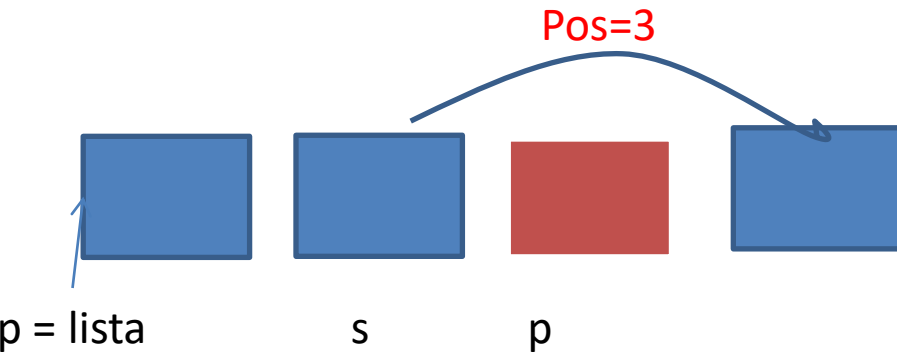
void Eliminar



```
void Eliminar_Final(Tlista &lista){  
    tLista p=lista, s=lista;  
    If(lista->sgte == NULL)  
        lista=NULL;  
    While(p->sgte != NULL){  
        s=p;  
        p=p->sgte;  
    }  
  
    s->sgte = NULL;  
    delete( p);  
}
```

Eliminar nodo por posicion

void Eliminar



```
void Eliminar_posicion(Tlista &lista, int pos, int cont){
```

```
tLista p=lista, s=NULL;
```

```
int n=1;
```

```
While(p->sgte != NULL && n!= pos){
```

```
    s=p;
```

```
    p=p->sgte;
```

```
    n=n+1;
```

```
}
```

```
s->sgte = p->sgte;
```

```
delete(p);
```

```
}
```

Eliminación de Nodos

```
void eliminarElemento(Tlista &lista, int valor)
{
    Tlista p, ant;
    p = lista;

    if(lista!=NULL)
    {
        while(p!=NULL)
        {
            if(p->nro==valor)
            {
                if(p==lista)
                    lista = lista->sgte;
                else
                    ant->sgte = p->sgte;

                delete(p);
                return;
            }
            ant = p;
            p = p->sgte;
        }
    }
    else
        cout<<" Lista vacia..!";
}
```

Eliminar Nodo de la Lista

Funcion: **eliminarElemento**

1.- Se crean dos variables tipo estructura NODO (p y ant)

P: apunta a la actual posición de la lista

Ant: Apunta al elemento anterior de un elemento de la lista

lista: Contenido de la nueva variable tipo nodo creado en el proceso main()
con valor NULL

&lista: Dirección de memoria del nuevo elemento **lista**

```
void eliminarElemento(Tlista &lista, int valor)
{
    Tlista p, ant;
    p = lista;
```

Eliminar Nodo de la Lista

1.- Se verifica si la Lista inicial (lista) esta vacía, es decir si es NULL

Si lista=NULL

entonces

Envia mensaje «Lista Vacía»

Sino

Se recorre la lista hasta llegar al ultimo elemento (WHILE)

2.- dentro del While se pregunta si coincide con el valor

Si valor coincide

entonces

Consulta si la variable p esta posicionado en el mismo lugar a la lista original
Para un único elemento

Sino

Se recorre la lista hasta llegar al ultimo elemento (WHILE)

Eliminar Nodo de la Lista

```
void eliminarElemento(Tlista &lista, int valor)
{
    Tlista p, ant;
    p = lista;
    if(lista!=NULL)
    {
        while(p!=NULL)
        {
            if(p->nro==valor)
            {
                if(p==lista)
                    lista = lista->sgte;
                else
                    ant->sgte = p->sgte;

                delete(p);
                return;
            }
            ant = p;
            p = p->sgte;
        }
    }
}
```

Buscar un nodo por valor

```
void buscarElemento(Lista lista, int valor)
{
    Lista q = lista;
    int i = 1, flag= 0;

    while(q!=NULL)
    {
        if(q->nro==valor)
        {
            cout<<endl<<" Posicion del Nodo: "<< i <<endl;
            flag = 1;
        }
        q = q->sgte;
        i++;
    }

    if(flag ==0)
        cout<<"\n\n Nodo no encontrado..!"<< endl;
}
```