

GUIA DE LABORATORIOS

ESTRUCTURA DE DATOS

TEMA: LISTAS ENLAZADAS

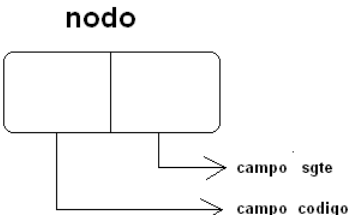
1. Aplicar las listas enlazadas como almacenamiento dinámico y no indexado, utilizado dentro de un programa c/c++ en la resolución de problemas.

EJERCICIO DIRIGIDO**EJERCICIOS # 01**

- a) El nodo tiene un campo información y enlace

```
#include <iostream.h>
#include <malloc.h>

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



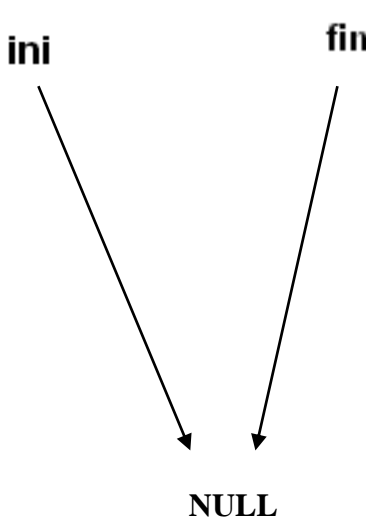
- b) Ahora vamos a crear las siguientes referencias ini y fin del nodo que apuntan al vacío.

```
#include <iostream.h>
#include <malloc.h>

struct nodo
{
    int codigo;
    struct nodo *sgte;
};

struct nodo *ini, *fin ;

void main()
{
    ini = fin = NULL;
}
```



- c) Ahora tenemos que crear un nodo y asignarle un dato, pero para eso tenemos que implementar un procedimiento de nombre ingresos ()

```

void ingresos()
{
    int codi;

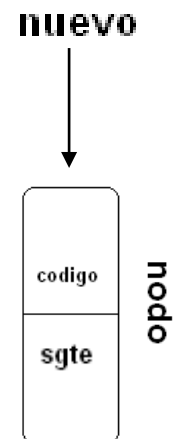
    struct nodo *nuevo;

    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}

```



- d) Luego asignamos un numero entero al campo **nuevo** → código del nodo de la lista enlazada

```

void ingresos()
{
    int codi;

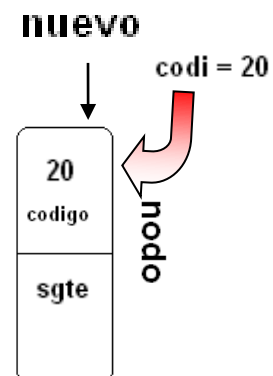
    struct nodo *nuevo;

    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}

```



- e) La condición **inicio==NULL** es verdadero, por lo tanto se ejecuta el primer bloque de la condicional doble, donde se ejecuta una sola línea de código, que representa el apuntalamiento de **ini** al nodo nuevo.

```

void ingresos()
{
    int codi;

    struct nodo *nuevo;

    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}

```

Terminado el primer bloque de la condicional doble, entonces hay dos líneas de código mas en ejecutarse, la primera línea de código representa el apuntalamiento de fin al nuevo nodo, el fin→sgte apunta al valor NULL.

```

void ingresos()
{
    int codi;

    struct nodo *nuevo;

    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

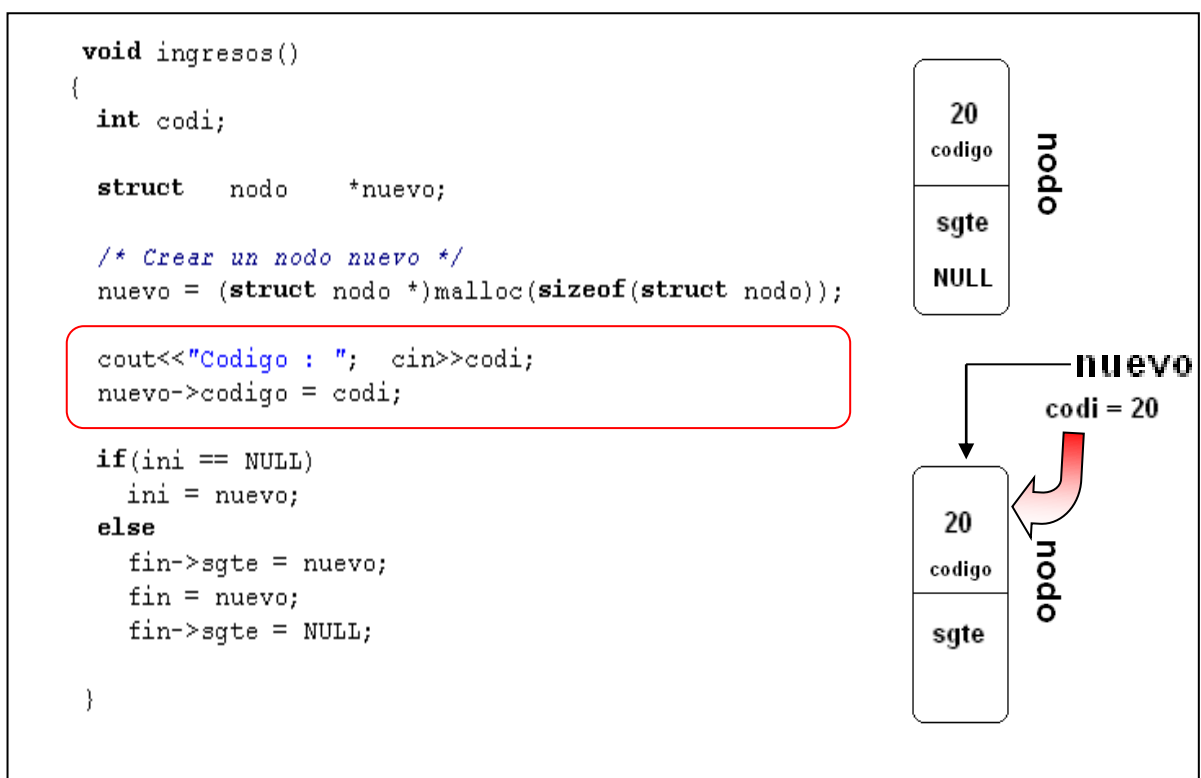
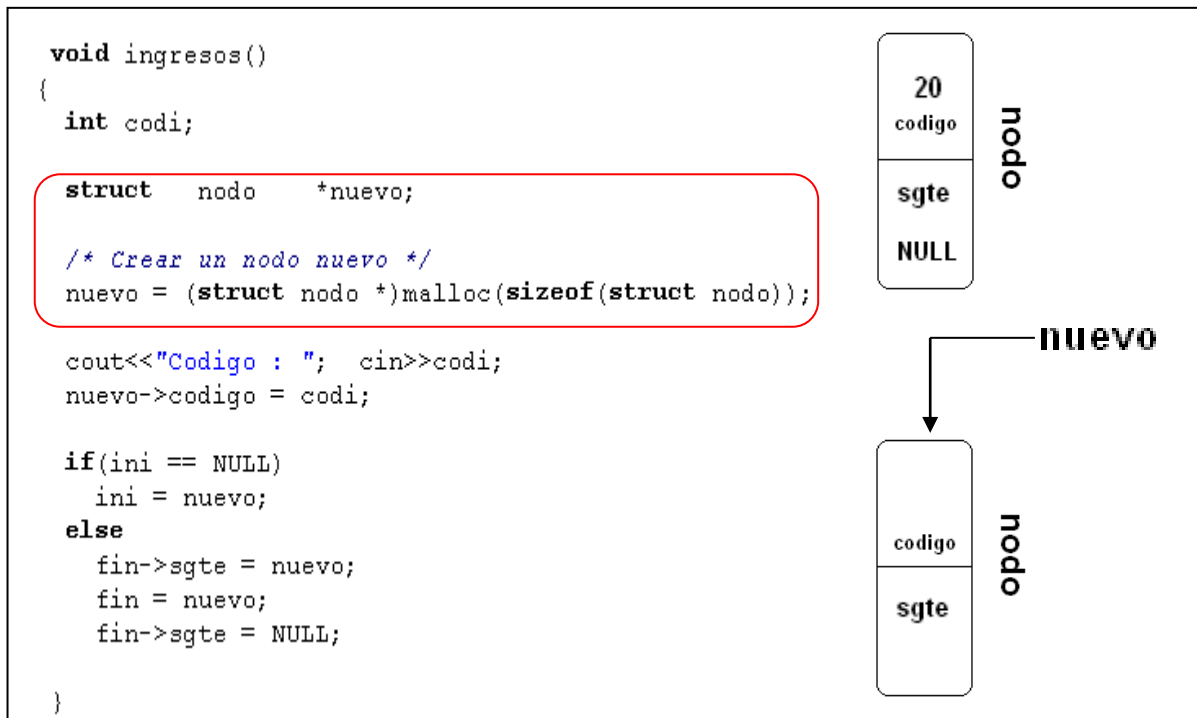
    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}

```

Si observamos bien, el procedimiento se ejecuto una sola vez y por lo tanto hemos logrado crear un solo nodo.

Si ejecutamos por segunda vez el mismo procedimiento, tenemos que crear un segundo nodo.

A continuación vamos a recrear la segunda invocación del procedimiento.



```

void ingresos()
{
    int codi;

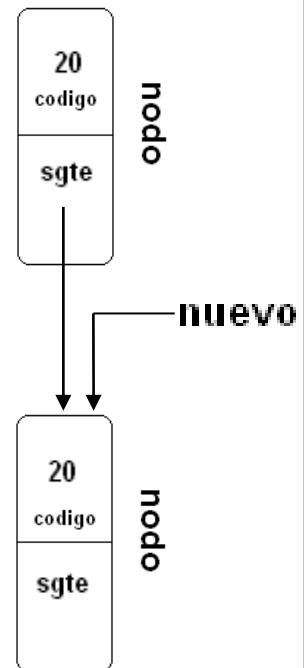
    struct nodo *nuevo;

    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}

```



```

void ingresos()
{
    int codi;

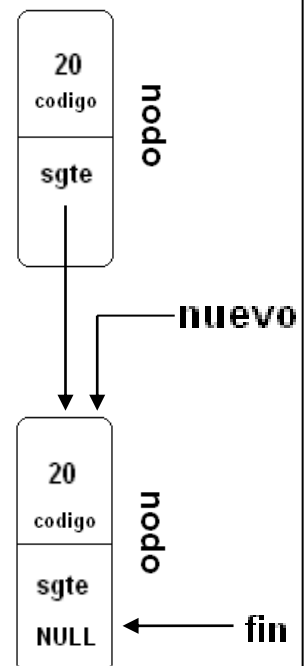
    struct nodo *nuevo;

    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}

```



Si el procedimiento se implementa en el programa en C/C++ obtenemos el siguiente código

```
#include <iostream.h>
#include <malloc.h>

void ingresos();

struct nodo
{
    int codigo;
    struct nodo *sgte;
};

struct nodo *ini, *fin ;

void main()
{
    ini = fin = NULL;

    for(int i=1; i<=2 ; i++)
    {
        ingresos();
    }

}

void ingresos()
{
    int codi;

    struct nodo *nuevo;

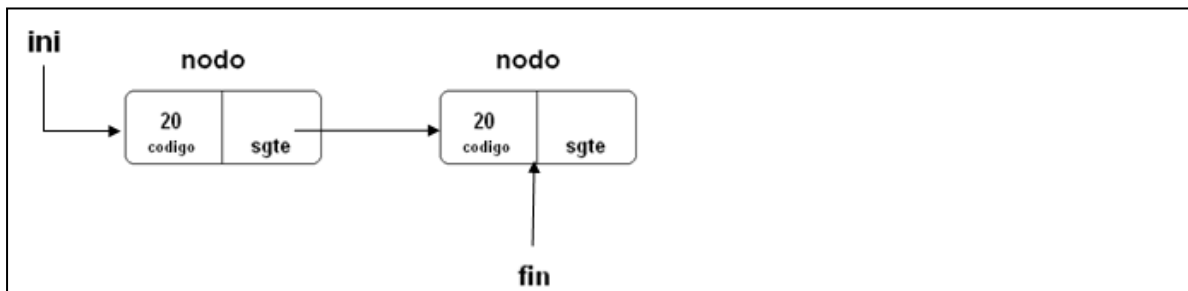
    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;
}
```

Aquí es se ejecuta 2 veces el procedimiento, por lo tanto creamos 2 nodos enlazados.

Aquí mostramos toda la lista enlazada completa:



- f) Ahora tenemos que mostrar todos los elementos que se encuentran en cada nodo, para eso tenemos que crear otro procedimiento.

```
void listado()
{
    struct nodo *aux;
    aux = ini;

    cout<<"LISTADO DE DATOS"<<endl;
    cout<<"Codigos"<<endl;
    cout<<"====="<<endl;

    while(aux!= NULL)
    {
        cout<<aux->codigo<<endl;
        aux = aux->sgte;
    }
}
```

- g) Implementando el procedimiento al programa en C/C++

```
#include <iostream.h>
#include <malloc.h>

void ingresos();

struct nodo
{
    int codigo;
    struct nodo *sgte;
};

struct nodo *ini, *fin ;

void main()
{
    ini = fin = NULL;

    for(int i=1; i<=2 ;i++)
    {
        ingresos();
    }
}
```



```
void listado()
{
    struct    nodo    *aux;
              aux = ini;

    cout<<"LISTADO DE DATOS"<<endl;
    cout<<"Codigos"<<endl;
    cout<<"======"<<endl;

    while(aux!= NULL)
    {
        cout<<aux->codigo<<endl;
        aux = aux->sgte;
    }
}

void ingresos()
{
    int codi;

    struct    nodo    *nuevo;

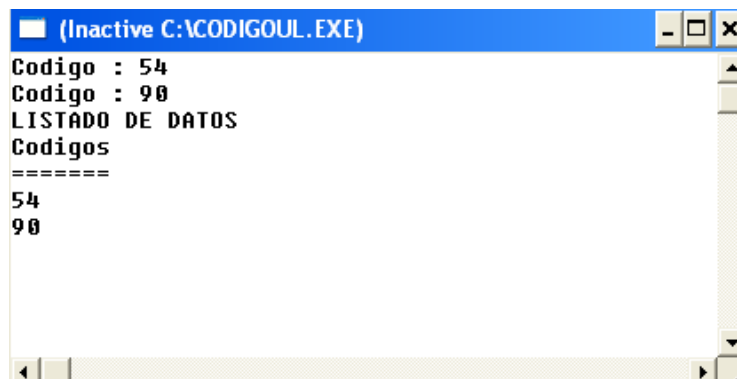
    /* Crear un nodo nuevo */
    nuevo = (struct nodo *)malloc(sizeof(struct nodo));

    cout<<"Codigo : "; cin>>codi;
    nuevo->codigo = codi;

    if(ini == NULL)
        ini = nuevo;
    else
        fin->sgte = nuevo;
        fin = nuevo;
        fin->sgte = NULL;

}
```

h) Ejecutando el programa C/C++



```
(Inactive C:\CODIGOUL.EXE)
Codigo : 54
Codigo : 90
LISTADO DE DATOS
Codigos
=====
54
90
```

EJERCICIOS PROPUESTOS

EJERCICIOS PROPUESTO # 01

La Universidad Privada del Norte, actualmente cuenta con una moderna y equipada infraestructura tecnológica.

La universidad tiene 3 sedes en lima, cuenta con 30 laboratorios de cómputo.

La universidad cada 3 meses realiza una auditoria de la infraestructura de todos los laboratorios de cómputo.

Los auditores tienen un formato para el debido control de los equipos principales de todos los laboratorios de la universidad.

Formato de control de equipos de cómputo

código	Nombre Laboratorio	Numero ordenadores	Numero servidores	Numero Mesas	Numero sillas	observación

Desarrolle un programa en C/C++ que me permita almacenar y visualizar en un reporte los registros (nodos) con el formato presentado anteriormente.

Este es el formato en código C/C++ del registro (nodo)

```
struct nodo
{
    int codigo;
    char nombrelab[20] ;
    int numeroordenadores ;
    int numeroservidores ;
    int numeromesas;
    int numerosillas ;
    char observacion[100] ;

    struct nodo  *sgte;
};
```

Utilizar listas enlazadas para el almacenamiento de todos los registros (nodo) del control de equipos de cómputo

EJERCICIOS PROPUESTO # 02

Una empresa que está en más de 5 continentes aplica una política de seguridad informática en todas las redes informáticas de cada una de sus sedes.

Esta política me permite realizar pruebas de vulnerabilidad en todas las redes y así refinar las políticas de seguridad informática.

La empresa requiere un registro para sus estadísticas del número de vulnerabilidades generales diagnosticadas producto de las pruebas realizadas en todas sus redes

Cada registro (**nodo**) está conformado por los siguientes campos:

- | | |
|------------------------------|--|
| • Código | <pre>struct nodo { int codigo; char nombrecontinente[20] ; char nombrered[20] ; int numerovulnerabilidades ; char mes[20] ; char ano[20] ; struct nodo *sgte; };</pre> |
| • Nombre continente | |
| • Nombre de la red | |
| • Numero de vulnerabilidades | |
| • Mes | |
| • Año | |

Se pide desarrollar un programa en C/C++ que permita utilizar listas enlazadas que realice las siguientes operaciones :

- Adicionar registros
- Mostrar registros
- Buscar registros
- Modificar registros

EJERCICIOS PROPUESTO # 04

Desarrollar un programa en C/C++ que permita la inserción de los valores: 18, 20, 33, 37, 15, 45,65, 17, 15 y 5, luego elimine de la lista los valores 33, 37, 45, 65. Muestre por pantalla el resultado de la lista enlazada de datos ingresados, eliminados y los que quedan.

LISTAS DOBLEMENTE ENLAZADAS

Aplicar las listas doblemente enlazadas para realizar operaciones complejas y de almacenamiento dinámico, utilizado dentro de un programa c/c++ para la resolución de problemas.

EJERCICIO DIRIGIDO

EJERCICIOS # 01

- a) El nodo tiene un campo información y enlace

```
#include<malloc.h>
#include<stdio.h>
#include<iostream.h>
```

```
struct nodo
{
    int num;
    struct nodo *ant ,*sgte ;
};
```

nodo



- b) Ahora vamos a crear las siguientes referencias **ini** y **ult**, **aux** del nodo que apuntan al vacío.

```
#include<malloc.h>
#include<stdio.h>
#include<iostream.h>
```

```
void adicion(int ne) ;
```

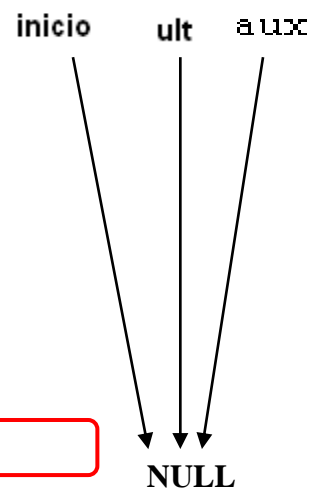
```
struct nodo
{
    int num;
    struct nodo *ant ,*sgte ;
};
```

```
struct nodo *inicio , *ult , *aux ;
```

```
main(){
```

```
    aux=inicio=ult=NULL;
```

```
}
```

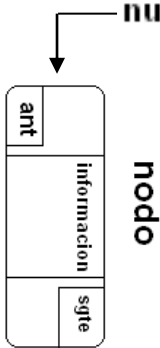


- c) Ahora tenemos que crear un nodo y asignarle un dato, pero para eso tenemos que implementar un procedimiento de nombre `adicion` (`int ne`)

```
void adicion(int ne)
{
    struct nodo *nuevo ;
    nuevo=(struct nodo*)malloc(sizeof(struct nodo));

    nuevo->num=ne;

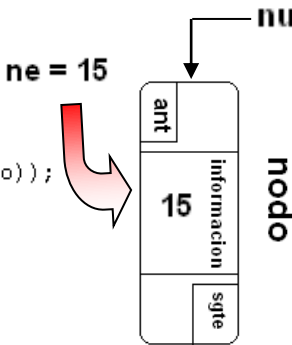
    if(inicio==NULL)
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
    else
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
}
```



- d) Luego asignamos un numero entero al campo `nuevo` → `num` del nodo de la lista doblemente enlazada

```
void adicion(int ne)
{
    struct nodo *nuevo ;
    nuevo=(struct nodo*)malloc(sizeof(struct nodo));
    nuevo->num=ne;

    if(inicio==NULL)
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
    else
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
}
```



- e) La condición **inicio==NULL** es verdadera , por lo tanto se ejecuta el primer bloque de la condicional doble

```

void adicion(int ne)
{
    struct nodo *nuevo ;

    nuevo=(struct nodo*)malloc(sizeof(struct nodo));

    nuevo->num=ne;

    if(inicio==NULL)
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
    else
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
}

```

ne = 15

- f) Dentro del primer bloque de la condicional doble ,se ejecuta tres línea de código que me permite apuntar dos referencias , inicio y ult al nodo nuevo , además en los campos nuevo→sgte y el campo nuevo→ant, ambos apuntan al valor NULL

```

void adicion(int ne)
{
    struct nodo *nuevo ;

    nuevo=(struct nodo*)malloc(sizeof(struct nodo));

    nuevo->num=ne;

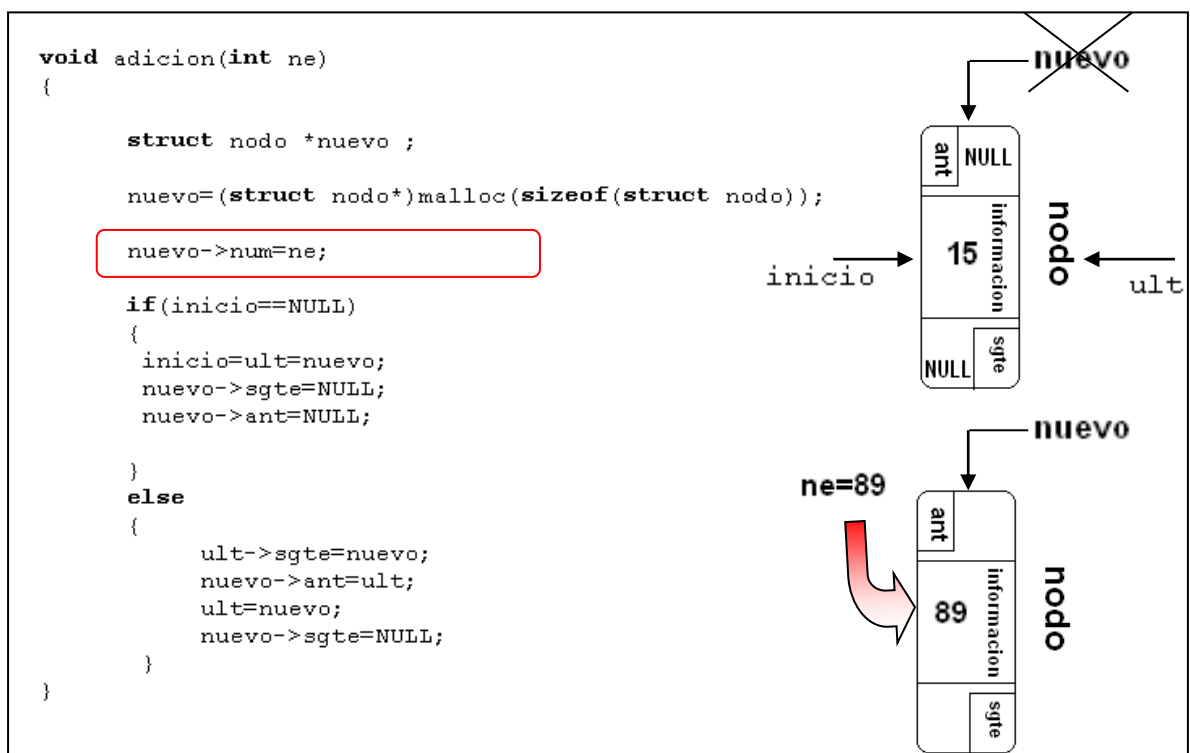
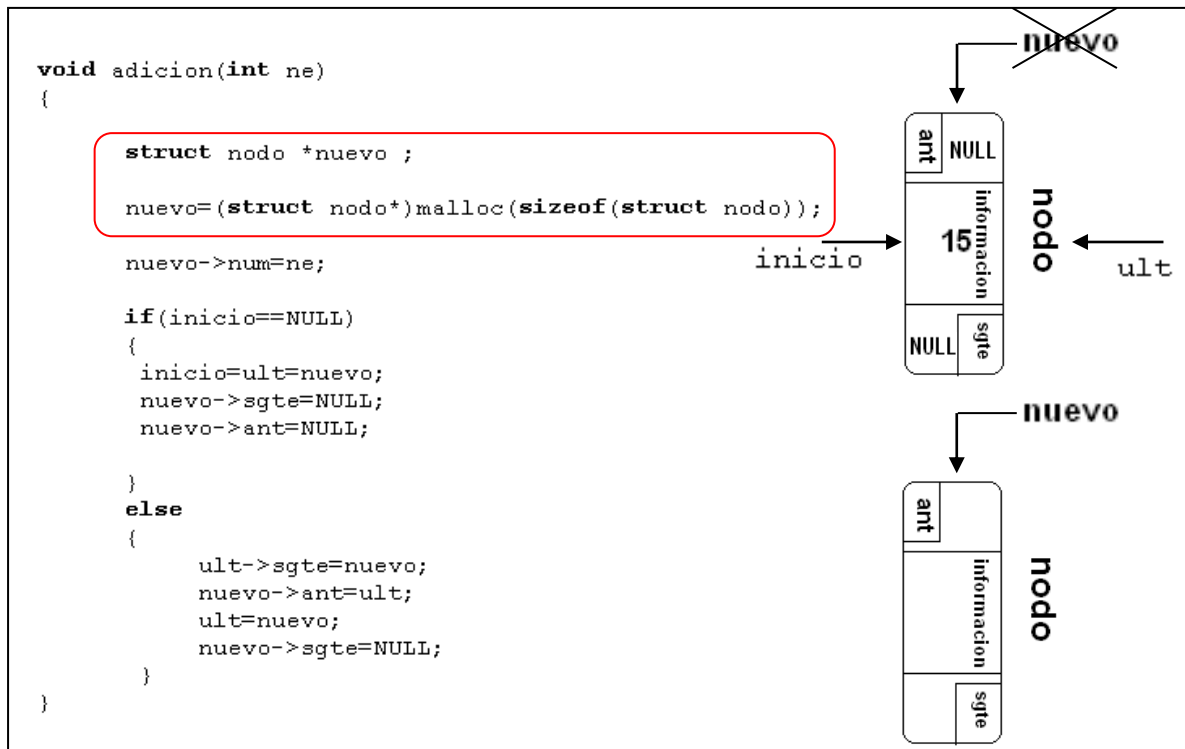
    if(inicio==NULL)
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
    else
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
}

```

Suponiendo que el procedimiento se ejecuto una sola vez y por lo tanto hemos logrado crear un solo nodo.

Si ejecutamos por segunda vez el mismo procedimiento, tenemos que crear un segundo nodo.

A continuación vamos a recrear la segunda ejecución del procedimiento.




```
void adicion(int ne)
{
```

```
    struct nodo *nuevo ;
```

```
    nuevo=(struct nodo*)malloc(sizeof(struct nodo));
```

```
    nuevo->num=ne;
```

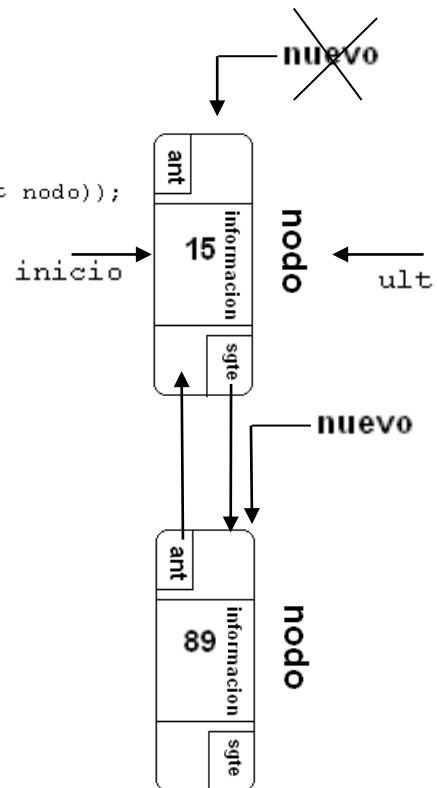
```
    if(inicio==NULL)
```

```
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
```

```
    else
```

```
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
```

```
}
```



```
void adicion(int ne)
{
```

```
    struct nodo *nuevo ;
```

```
    nuevo=(struct nodo*)malloc(sizeof(struct nodo));
```

```
    nuevo->num=ne;
```

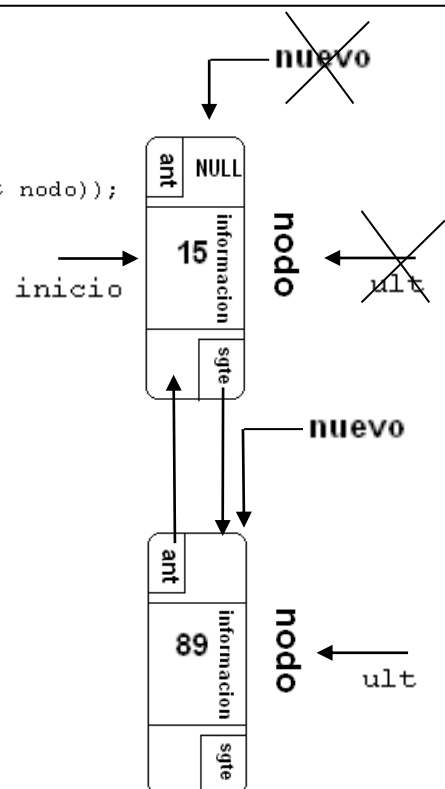
```
    if(inicio==NULL)
```

```
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
```

```
    else
```

```
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
```

```
}
```



```
void adicion(int ne)
{
```

```
    struct nodo *nuevo ;
```

```
    nuevo=(struct nodo*)malloc(sizeof(struct nodo));
```

```
    nuevo->num=ne;
```

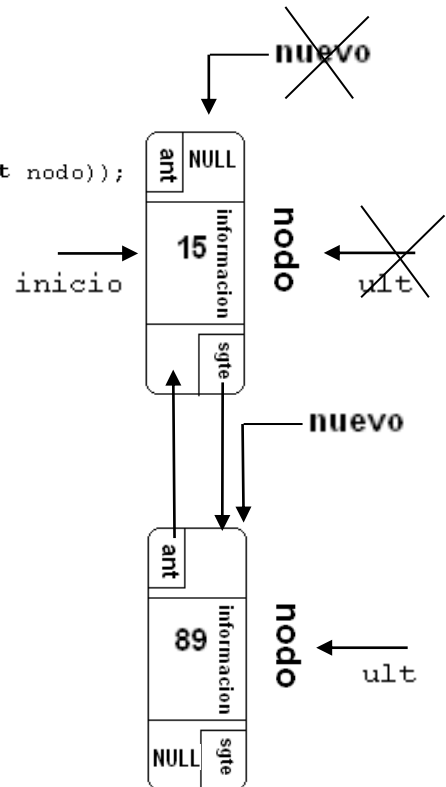
```
    if(inicio==NULL)
```

```
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
```

```
    else
```

```
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
```

```
}
```



g) A continuación vemos el código completo del programa en C/C++

```
#include<stdio.h>
```

```
#include<iostream.h>
```

```
#include<malloc.h>
```

```
void adicion(int ne);
```

```
struct nodo
```

```
{
```

```
int num;
```

```
struct nodo *ant,*sgte;
```

```
};
```

```
struct nodo *inicio,*ult,*aux;
```

```
main()
```

```
{
```

```
    aux=inicio=ult=NULL;
```

```
}
```

```

void adicion(int ne)
{
    struct nodo *nuevo ;

    nuevo=(struct nodo*)malloc(sizeof(struct nodo));

    nuevo->num=ne;

    if(inicio==NULL)
    {
        inicio=ult=nuevo;
        nuevo->sgte=NULL;
        nuevo->ant=NULL;
    }
    else
    {
        ult->sgte=nuevo;
        nuevo->ant=ult;
        ult=nuevo;
        nuevo->sgte=NULL;
    }
}

```

h) La función `adición`, se va a invocar desde la función `main()`

```

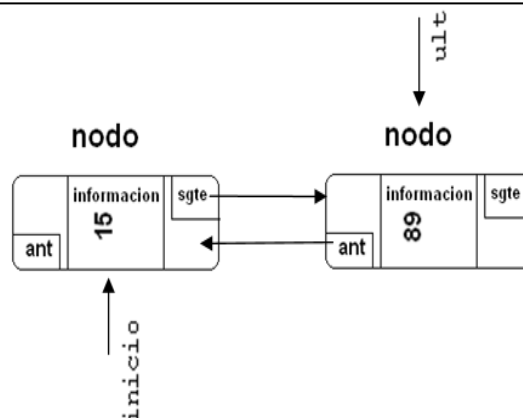
main()
{
    aux=inicio=ult=NULL;

    adicion(15);
    adicion(89);
}

```

Invocando dos veces al procedimiento

i) Si hemos invocado dos veces la función `adición` y hemos enviado el número 15 y el 89, nosotros hemos logrado crear una lista doblemente enlazada, a continuación vamos a ver la estructura de la lista doblemente enlazada que se ha generado en memoria.



- j) Ahora para poder mostrar los elementos por pantalla, vamos a recorrer los nodos de izquierda a derecha y de derecha a izquierda, porque esta lista es doblemente enlazada, a continuación vamos a implementar dos funciones que realizan la visualización de los elementos, tal como se ha especificado anteriormente.

```
#include<stdio.h>
#include<iostream.h>
#include<malloc.h>

void adicion(int ne);
void listaderecha();
void listaizquierda();

struct nodo
{
    int num;
    struct nodo *ant,*sgte;
};

struct nodo *inicio,*ult,*aux;

main()
{
    aux=inicio=ult=NULL;
    adicion(15);
    adicion(89);

    cout<<"\nRecorrer de Izquierda a Derecha"<<endl;
    listaderecha();

    cout<<"\nRecorrer de Derecha a Izquierda"<<endl;
    listaizquierda();
}

void adicion(int ne)
{
    aux=(struct nodo*)malloc(sizeof(struct nodo));
    aux->num=ne;

    if(inicio==NULL){
        inicio=ult=aux;
        aux->sgte=NULL;
        aux->ant=NULL;
    }
    else
    {
        ult->sgte=aux;
        aux->ant=ult;
        ult=aux;
        aux->sgte=NULL;
    }
}
```

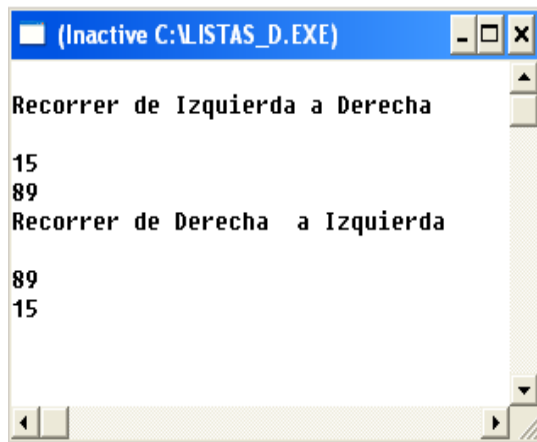
Invocando
dichos
procedimientos

```
void listaderecha()
{
    aux=inicio;
    while(aux!=NULL)
    {
        cout<<"\n"<<aux->num;
        aux=aux->sgte;
    }
}

void listaizquierda()
{
    aux=ult;
    while(aux!=NULL)
    {
        cout<<"\n"<<aux->num;
        aux=aux->ant;
    }
}
```

Implementando
dichos
procedimientos

k) Ejecutando el programa



```
(Inactive C:\LISTAS_D.EXE)

Recorrer de Izquierda a Derecha

15
89
Recorrer de Derecha a Izquierda

89
15
```

EJERCICIOS PROPUESTOS

1. Desarrollar un Programa que devuelva la SUMATORIA de los Números introducidos en los Nodos de la Lista.
2. Escribir un Programa que devuelva los Números Pares e Impares que se encuentran en los Nodos de una Lista.
3. Se tiene una lista de simple enlace, el campo dato es un registro (estructura) con los campos de un producto: nombre, precio, igv, total, n_precio. Escribir una función para calcular el n_precio que es 10% adicional y calcular el promedio de los nuevos precios.
4. Desarrollar un Programa que muestre las notas mayores a 18 dada una Lista Enlazada ordene los Nodos en forma descendente.

.