

ELEMINAR REPETIDOS

### LISTA DE NUMEROS

INGRESE UN NUMERO :

REGISTRAR NUMERO

MOSTRAR

lista\_numeros

lst\_repetidos

lst\_actuales

ELEMINAR REPETIDOS

SALIR

ELEMINAR REPETIDOS

### LISTA DE NUMEROS

INGRESE UN NUMERO :

REGISTRAR NUMERO

MOSTRAR

5  
10  
15  
20

ELEMINAR REPETIDOS

SALIR

```
class Lista
{
    Nodo cabeza, nuevo;
    public Lista()
```

```

{
    cabeza = null;
    nuevo = null;
}
public Nodo get_cabeza()
{
    return cabeza; //primero
}
//creamos el nodo
public void crear_nodo( int nro)
{
    nuevo = new Nodo();
    nuevo.Set_num(nro);
    nuevo.Set_enlace(null);
}
public void crear_lista(int num)
{
    Nodo punt;
    crear_nodo(num);
    //preguntamos
    if (cabeza == null) //cuando no hay ningun Nodo
    {
        cabeza = nuevo; //cabeza vaya a la direccion del nuevo
    }
    else
    {
        punt = cabeza;
        while (punt.Get_enlace() != null)
        {
            punt = punt.Get_enlace();
        }
        //hay que elazarlo al puntero
        punt.Set_enlace(nuevo); //direccion de memoria apunta
    }
}
}

```

```

public Nodo Eliminar(int nro)
{
    Nodo punt = cabeza;
    Nodo ant = null;

    while (punt != null)
    {
        if (nro == punt.Get_num())
        {
            cabeza = cabeza.Get_enlace();
            punt.Set_enlace(null);
        }
        else
        {
            if (punt.Get_num() == nro)
            {
                ant.Set_enlace(punt.Get_enlace());
                punt.Set_enlace(null);
                return punt;
            }
        }
    }
}

```

```

    }
    ant = punt;

    punt = punt.Get_enlace();
}
return punt;
}
}
}

```

## EJERCICIO02

ELEMENAR REPETIDOS

LISTA DE NUMEROS

INGRESE UN NUMERO :

REGISTRAR LISTA 1 REGISTRAR LISTA 2

1  
3  
5  
7

2  
4  
6

LIMPIAR

**MOSTRAR**

INTERCALAR

SALIR

ELEMENAR REPETIDOS

LISTA DE NUMEROS

INGRESE UN NUMERO :

REGISTRAR LISTA 1 REGISTRAR LISTA 2

lista\_numeros lst\_numeros2 lstintercalado

LIMPIAR

**MOSTRAR**

INTERCALAR

SALIR

```

class Lista
{
    Nodo cabeza, nuevo;
public Lista()
{
    cabeza = null;
    nuevo = null;
}
public Nodo get_cabeza()
{
    return cabeza;//primero
}
//creamos el nodo
public void crear_nodo( int nro)
{
    nuevo = new Nodo();
    nuevo.Set_num(nro);
    nuevo.Set_enlace(null);
}
public void crear_lista(int num)
{
    Nodo punt;
    crear_nodo(num);
    //preguntamos
    if (cabeza == null)//cuando no hay ningun Nodo
    {
        cabeza = nuevo;    //cabeza vaya a la direccion del nuevo
    }
    else
    {
        punt = cabeza;
        while (punt.Get_enlace() != null)
        {
            punt = punt.Get_enlace();
        }
        //hay que elazarlo al puntero
        punt.Set_enlace(nuevo);//direccion de memoria apunta
    }
}

public int Repetidos(int nro)
{
    Nodo actual = cabeza;
    Nodo anterior = null;
    Nodo aux = null;
    int cont = 0;
    while (actual != null)
    {
        aux = actual;
        if (nro == cabeza.Get_num() && actual == cabeza)
        {
            cabeza = cabeza.Get_enlace();
            anterior = actual;
            actual = actual.Get_enlace();
            aux.Set_enlace(null);
            cont++;
        }
        else
        {
            if (nro == actual.Get_num())
            {

```

```

        anterior.Set_enlace(actual.Get_enlace());
        actual = actual.Get_enlace();
        aux.Set_enlace(null);
        cont++;
    }
    else
    {
        anterior = actual;
        actual = actual.Get_enlace();
    }
}

}
return cont;
}

public Nodo Eliminar(int nro)
{
    Nodo punt = cabeza;
    Nodo ant = null;

    while (punt != null)
    {
        if (nro == punt.Get_num())
        {
            cabeza = cabeza.Get_enlace();
            punt.Set_enlace(null);
        }
        else
        {
            if (punt.Get_num()==nro)
            {
                ant.Set_enlace(punt.Get_enlace());
                punt.Set_enlace(null);
                return punt;
            }
        }
        ant = punt;
        punt = punt.Get_enlace();
    }
    return punt;
}
}
}

```

```

class Lista02
{
    Nodo cabeza, nuevo;
public Lista02()
{
    cabeza = null;
    nuevo = null;

}
public Nodo get_cabeza()
{
    return cabeza; //primero
}
//creamos el nodo
public void crear_nodo(int nro)
{
    nuevo = new Nodo();
    nuevo.Set_num(nro);
    nuevo.Set_enlace(null);
}
public void crear_lista(int num)
{
    Nodo punt;
    crear_nodo(num);
    //preguntamos
    if (cabeza == null) //cuando no hay ningun Nodo
    {
        cabeza = nuevo; //cabeza vaya a la direccion del nuevo
    }
    else
    {
        punt = cabeza;
        while (punt.Get_enlace() != null)
        {
            punt = punt.Get_enlace();
        }
        //hay que elazarlo al puntero
        punt.Set_enlace(nuevo); //direccion de memoria apunta
    }
}

public int Repetidos(int nro)
{
    Nodo actual = cabeza;
    Nodo anterior = null;
    Nodo aux = null;
    int cont = 0;
    while (actual != null)
    {
        aux = actual;
        if (nro == cabeza.Get_num() && actual == cabeza)
        {
            cabeza = cabeza.Get_enlace();
            anterior = actual;
            actual = actual.Get_enlace();
            aux.Set_enlace(null);
            cont++;
        }
        else
        {
            if (nro == actual.Get_num())
            {

```

```

        anterior.Set_enlace(actual.Get_enlace());
        actual = actual.Get_enlace();
        aux.Set_enlace(null);
        cont++;
    }
    else
    {
        anterior = actual;
        actual = actual.Get_enlace();
    }
}

return cont;
}

public Nodo Eliminar(int nro)
{
    Nodo punt = cabeza;
    Nodo ant = null;

    while (punt != null)
    {
        if (nro == punt.Get_num())
        {
            cabeza = cabeza.Get_enlace();
            punt.Set_enlace(null);
        }
        else
        {
            if (punt.Get_num() == nro)
            {
                ant.Set_enlace(punt.Get_enlace());
                punt.Set_enlace(null);
                return punt;
            }
        }
        ant = punt;
        punt = punt.Get_enlace();
    }
    return punt;
}
}
}

```