Programación II
Estructuras Dinámicas

2C 2023 TN – Ing. Elizabeth Barrera

Temas

- *Estructuras dinámicas*
- *Recorridos*
- *Inserción*
- *Eliminación*
- *Resumen*


Estructuras
Dinámicas

Estructuras Dinámicas

Una estructura enlazada o dinámica se basan en el concepto de utilizar ***memoria dinámica****: vamos reservando memoria a medida que se necesita; asimismo, la memoria que ya no se necesita se devuelve al sistema.*

La base es la célula o ***nodo****. Se especifica un nodo que almacena un valor, y luego se van a encadenar tantos nodos como sean necesarios.*

Nodo

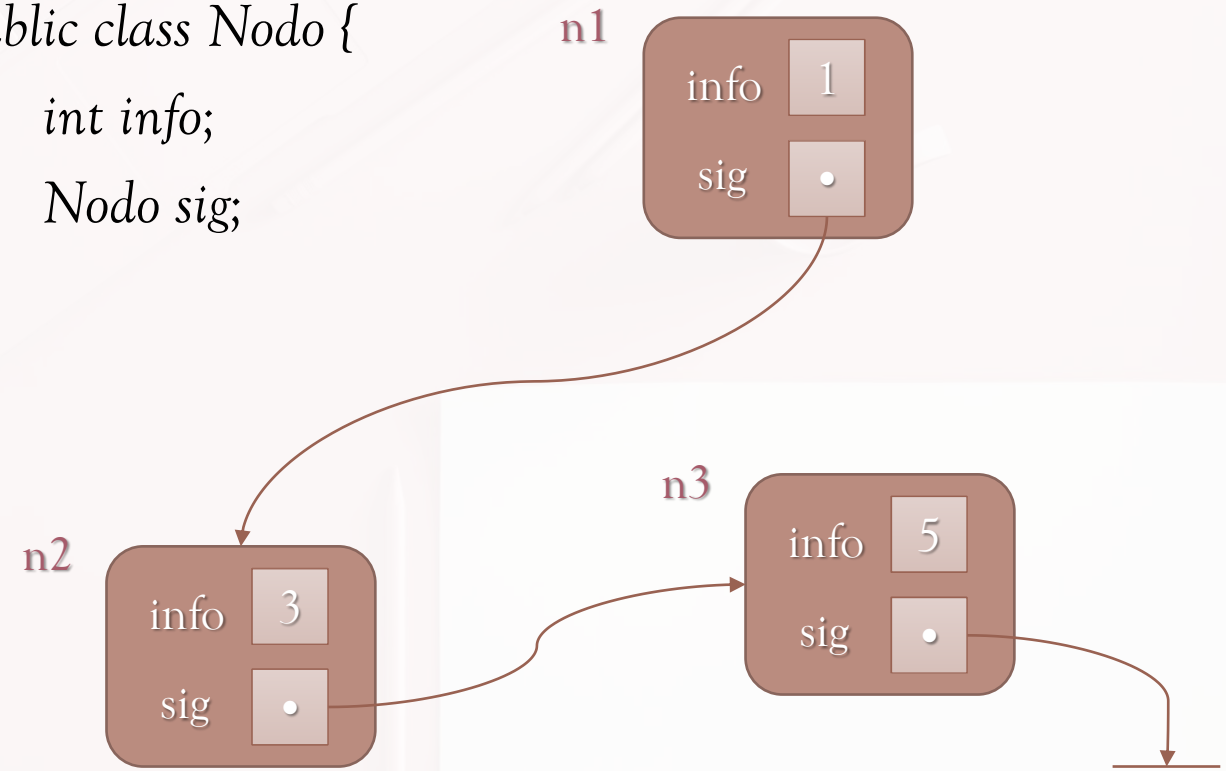
- *En cada nodo se definirá un espacio para almacenar el valor y otro para guardar una referencia a otro nodo. Así con un nodo,* ***tenemos el valor y la referencia a otro nodo****. Esto da lugar a una cadena de nodos potencialmente infinita.*
- *Un nodo que no apunta a ningún otro nodo, tendrá un valor especial en esa posición:* ***null****.*

Nodo

```
public class Nodo {
    int info;
    Nodo sig;
}
```

Ejemplo

```
public class Nodo {
    int info;
    Nodo sig;
}
```




```
Nodo n1 = new Nodo ();
Nodo n2 = new Nodo ();
Nodo n3 = new Nodo ();
n1.info = 1;
n2.info = 3;
n3.info = 5;
n1.sig = n2;
n2.sig = n3;
n3.sig = null;
```


Recorridos

Recorridos

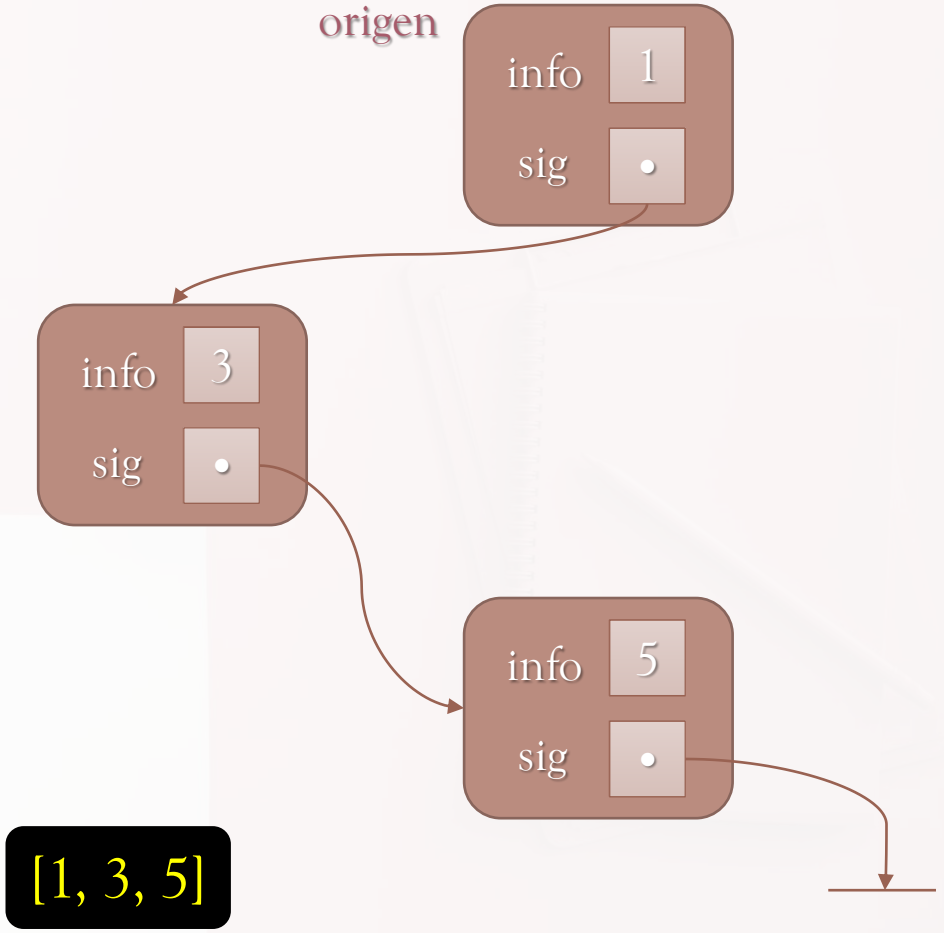
- *Veremos a continuación mecanismos habituales de trabajo con estas estructuras.*
- *Generalmente, tenemos sólo la **referencia al primer nodo** de la estructura. No tenemos acceso directo a cualquier elemento de una lista enlazada, como lo tendríamos en un arreglo.*
- *Veremos una estrategia iterativa y una estrategia recursiva.*

Recorridos

- Para recorrer la lista se utiliza un ***nodo auxiliar*** con el cual nos moveremos a través de la lista.
- Las operaciones, suponiendo que el primer nodo de la lista se llama origen, son:
  - Nodo aux = new Nodo () → para crear el ***nuevo nodo***
  - aux = origen → para ***inicializar*** el nodo auxiliar en el origen
  - aux != null → la condición que nos indica que ***el nodo permanece en la lista***
  - aux = aux.sig → para ***avanzar*** al siguiente nodo de la lista

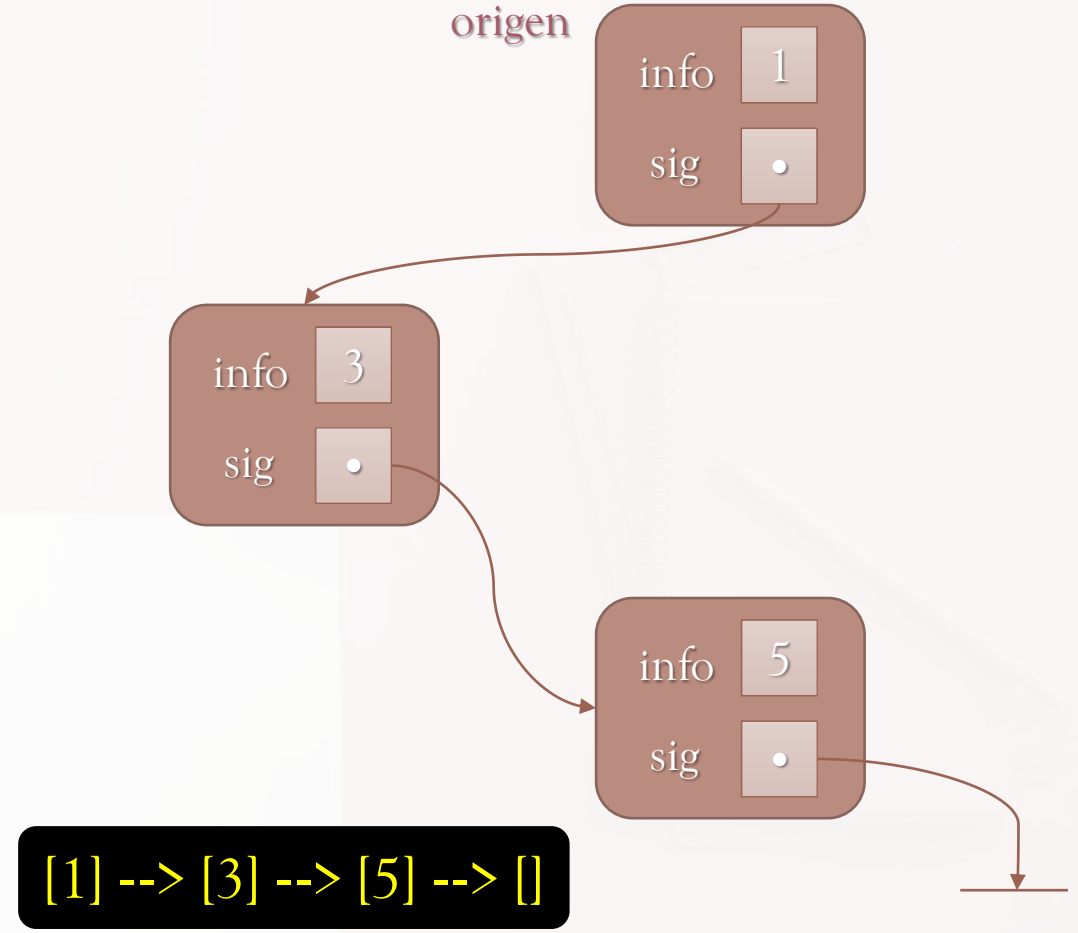
Primera estrategia - iterativa - visualización

```
public static void mostrarLista (Nodo origen) {
    Nodo aux = origen;
    System.out.print("[");
    while (aux != null) {
        System.out.print(aux.info);
        aux = aux.sig;
        if (aux != null)
            System.out.print(", ");
    }
    System.out.println("]");
}
```



Primera estrategia - recursiva - visualización

```
public static void mostrarLista (Nodo origen) {
    System.out.print("[");
    if (origen != null) {
        System.out.print(origen.info + "] --> ");
        mostrarLista(origen.sig);
    } else {
        System.out.println("]");
    }
}
```



Recorridos

- *Para algunos casos podemos necesitar buscar un nodo en particular recorriendo la lista de manera diferente: el nodo viajero aux "mira" al nodo siguiente y no al nodo sobre el que se encuentra.*
- *En este caso, el nodo inicial de la lista debe tratarse como un caso especial.*

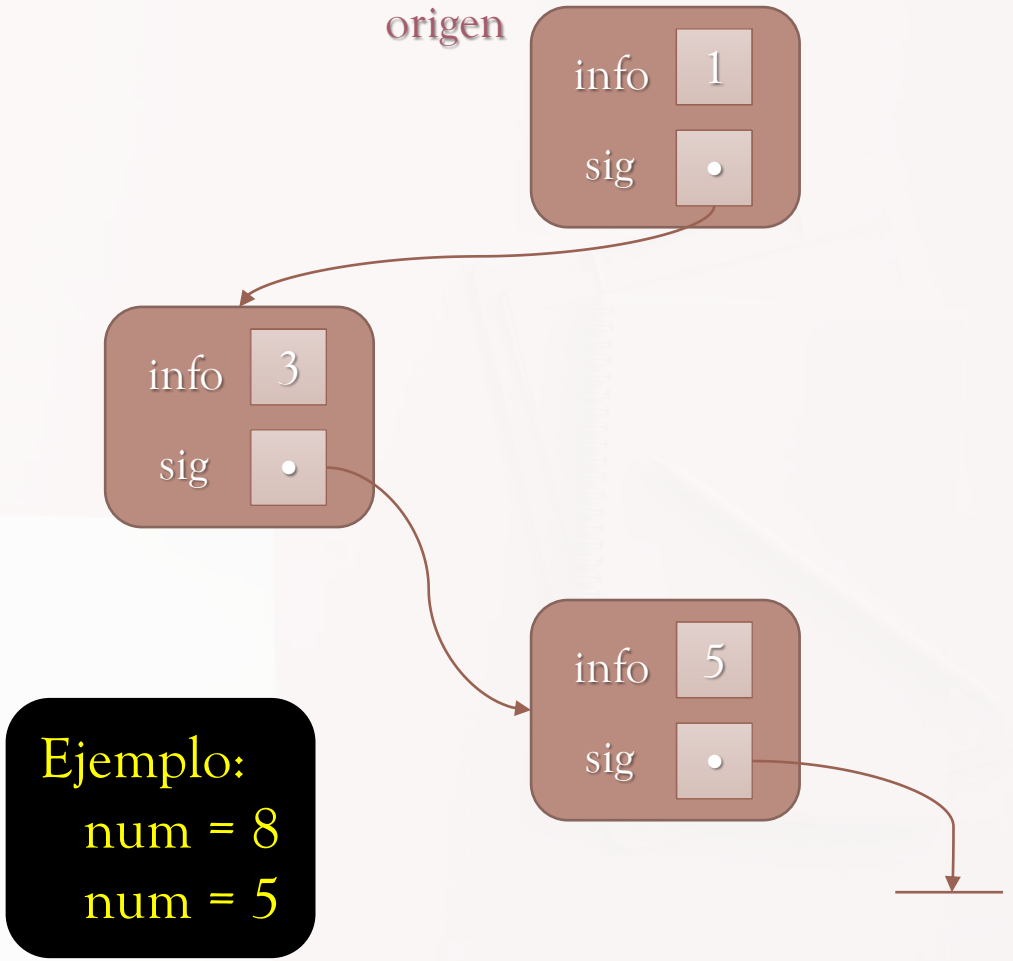
Búsqueda - iterativa

Primera estrategia:

```
Nodo aux = origen;
while (aux != null && aux.info != num)
    aux = aux.sig;
```

Segunda estrategia:

```
Nodo aux = origen;
while (aux.sig != null && aux.sig.info != num)
    aux = aux.sig;
```



Ejemplo:
num = 8
num = 5


Inserción

Inserción

- *Existen varias maneras de insertar elementos en una lista enlazada. Las principales son la inserción al principio, al final y ordenada.*
- *La **inserción al principio** es la más simple de todas: se apunta el nuevo nodo al origen de la lista y el nuevo nodo se convierte a continuación en el nuevo origen de la lista. Utilizaremos la primera estrategia de recorrido vista.*

Inserción

- *La **inserción al final**, así como también la **inserción ordenada**, son ligeramente más complejas porque normalmente no tenemos acceso al último elemento de la lista, hay que encontrarlo. Utilizaremos la segunda estrategia de recorrido vista. En este caso, el nodo inicial de la lista debe tratarse como un caso especial.*
- *Para el caso de la inserción al final, se puede agregar una referencia al último elemento, para evitar el recorrido en cada inserción.*

Inserción al principio

```
Nodo origen;
public static void AgregarI (int num) {
        Nodo nuevo = new Nodo();
        nuevo.info = num;
        nuevo.sig = origen;
        origen = nuevo;
}
```

Ejemplo:
insertamos el 6 a una lista vacía
luego, insertamos el 8
luego, el 3

Inserción al final

```
Nodo origen;
public static void AgregarF (int num) {
        Nodo nuevo = new Nodo();
        nuevo.info = num;
        nuevo.sig = null;
        if (origen == null) {
                origen = nuevo;
        } else {
                Nodo aux = origen;
                while (aux.sig != null)
                        aux = aux.sig;
                aux.sig = nuevo;
        }
}
```

Ejemplo:
insertamos el 6 a una lista vacía
luego, insertamos el 8
luego, el 3

Inserción al final con doble referencia

```
public static void AgregarFF (int num) {
        Nodo nuevo = new Nodo();
        nuevo.info = num;
        nuevo.sig = null;
        if (primero == null)
                primero = nuevo;
        else
                ultimo.sig = nuevo;
        ultimo = nuevo;
}
```

```
Nodo primero, ultimo;
```

Ejemplo:
insertamos el 6 a una lista vacía
luego, insertamos el 8
luego, el 3

Inserción ordenada

```
Nodo origen;
public static void AgregarO (int num)
    Nodo nuevo = new Nodo();
    nuevo.info = num;
    nuevo.sig = null;
    if (origen == null) {
        origen = nuevo;
    } else if (origen.info < num) {
        nuevo.sig = origen;
        origen = nuevo;
    } else {
        Nodo aux = origen;
        while (aux.sig != null && aux.sig.info >= num)
            aux = aux.sig;
        nuevo.sig = aux.sig;
        aux.sig = nuevo;
    }
}
```

Ejemplo:
insertamos el 8 en la lista [9, 7, 5, 1]


Eliminación

Eliminación

- *La eliminación de un elemento de una lista enlazada consiste en **dejarlo inaccesible**, o sea, el elemento sigue existiendo, pero ya no se tiene la referencia de su posición.*
- *Hay que distinguir el caso especial en que se desea **eliminar el origen** de la lista. En este caso, la referencia de la lista se desplaza al segundo elemento (origen.sig se convierte en origen).*

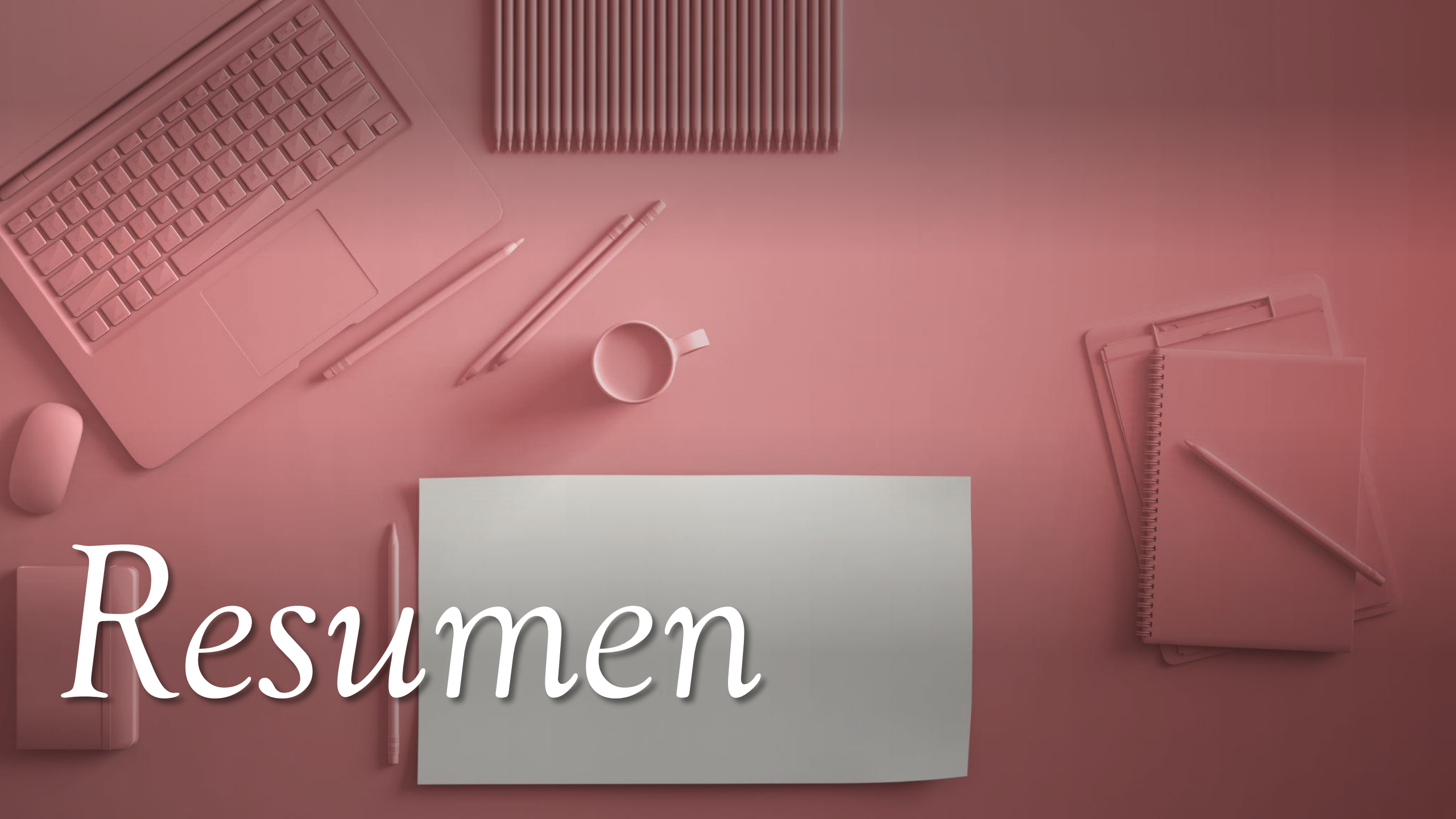
Eliminación

- *Para eliminar cualquier otro elemento, se lo debe **circunvalar**. Para ello, se lo busca según la segunda estrategia de recorridos vista y, cuando se lo encuentra, se lo salta (o sea se requiere que aux.sig pase a ser aux.sig.sig).*

Eliminación

```
Nodo origen;
public static void Eliminar (int num) {
    if (origen != null) {
        if (origen.info == num) {
            //eliminar el origen
            origen = origen.sig;
        } else {
            Nodo aux = origen;
            while (aux.sig != null && aux.sig.info != num)
                aux = aux.sig;
            if (aux.sig != null)
                aux.sig = aux.sig.sig;
        }
    }
}
```

Ejemplo:
eliminamos el 7 en la lista [1, 5, 7, 9]
luego, el 1


Resumen

Resumen

- *Las estructuras enlazadas se forman enlazando nodos. Los **nodos** son las células con las que construimos la estructura.*
- *Los nodos contienen **vínculos** a otros nodos, lo que permite construir la estructura. El atributo sig del último nodo debe ser null.*
- *No se tiene acceso directo a cada nodo de la estructura, por lo que para **recorrerla** se utiliza un nodo auxiliar, que es el que se desplaza.*

Resumen

- *Puede **recorrerse** una estructura enlazada de dos maneras diferentes:*
  - *El nodo auxiliar "mira" el nodo sobre el que está posado (se usa para recuperar o actualizar un valor).*
  - *El nodo auxiliar "mira" al nodo siguiente del nodo sobre el que está posado (se usa para insertar o eliminar un nodo). El nodo inicial debe ser tratado separadamente.*
- *Para **eliminar** un nodo, se lo circunvala, apuntando su anterior a su siguiente.*

Aclaraciones

- *Una estructura enlazada es una estructura que utiliza variables que son referencias a otros objetos. Estas variables almacenan la* ***dirección en memoria de un objeto****.*
- *Un* ***elemento eliminado*** *sigue existiendo en la memoria, sólo se ha vuelto inaccesible.*
- *Si se pierde la referencia a una estructura, se* ***pierde la estructura****, pues ésta se vuelve inaccesible. Por esto se usa un nodo auxiliar que es una* ***copia*** *de la referencia del origen de la lista y no esta referencia directamente.*

Aclaraciones

- *Se las denomina estructuras dinámicas, porque nuevos nodos pueden ser agregados según sean necesarios en tiempo de ejecución; así, se consideran de* ***capacidad ilimitada*** *(en contraposición a las estructuras estáticas, arreglos, cuya capacidad debe definirse de antemano).*
- *El inconveniente de estas estructuras es que* ***no contamos con un acceso directo a un elemento cualquiera*** *de la estructura. Tenemos acceso directo sólo al primero, para buscar cualquier otro elemento, debemos recorrer toda la estructura.*


¡Muchas Gracias!

Bibliografía

- *Programación II – Apuntes de Cátedra – V1.3 – Cuadrado Trutner – UADE*
- *Programación II – Apuntes de Cátedra – Wehbe – UADE*