Estructura de datos

La cantidad de líneas de código no representa que tan eficiente es un algoritmo como tal.

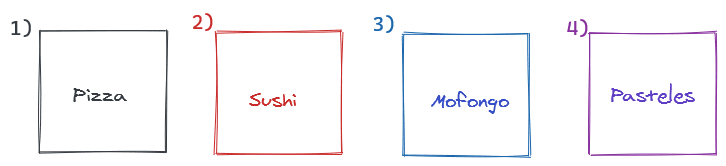
A la vez un algoritmo (como array.Sort) funciona mucho mas rápido o lento, según como se le ingresen los datos. Si se le ingresan aleatoriamente, por ejemplo, tardará mucho mas tiempo en ordenarlos.

Los algoritmos tienen diferentes tasas de crecimientos: quizás con pocos datos uno funcione muy bien, pero con muchos muy mal, y otro algoritmo, con pocos funcione regularmente, pero con muchos, siga funcionando prácticamente igual.

Estructuras de datos nativas: el framework de .NET nos da la opción de utilizar estas estructuras de datos importando ciertos namespaces, algunos lenguajes de programación no tienen la capacidad de utilizar estas estructuras nativamente y el programador tiene que escribirlas el mismo desde cero.

Array

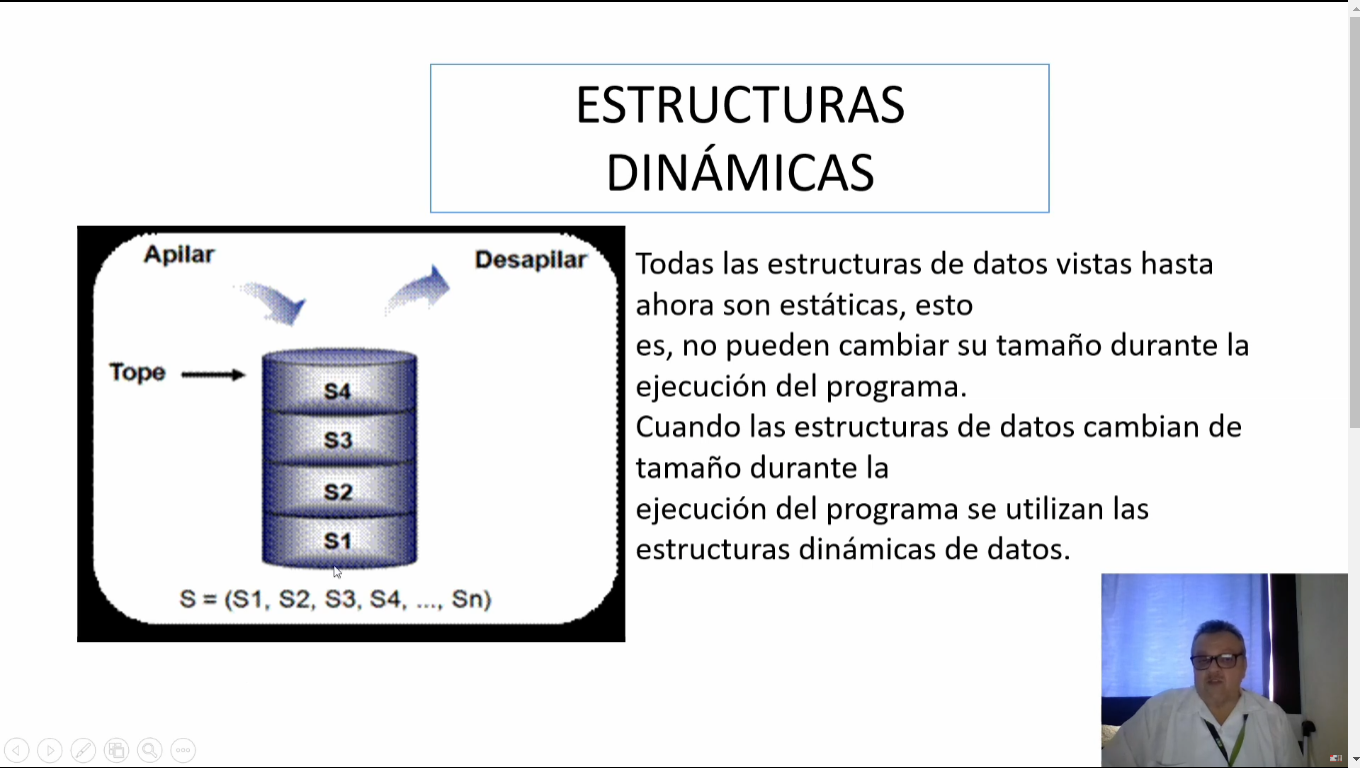
Colección ordenada de elementos los cuales tienen un índice identificador.



Muchas de las estructuras de datos que utilizamos son en realidad Arrays con funcionalidades añadidas, los Arrays son como el papa o mama de las demás estructuras de datos.

Cuando utilizamos Arrays en C# debemos especificar el tamaño al momento de crear una nueva instancia.



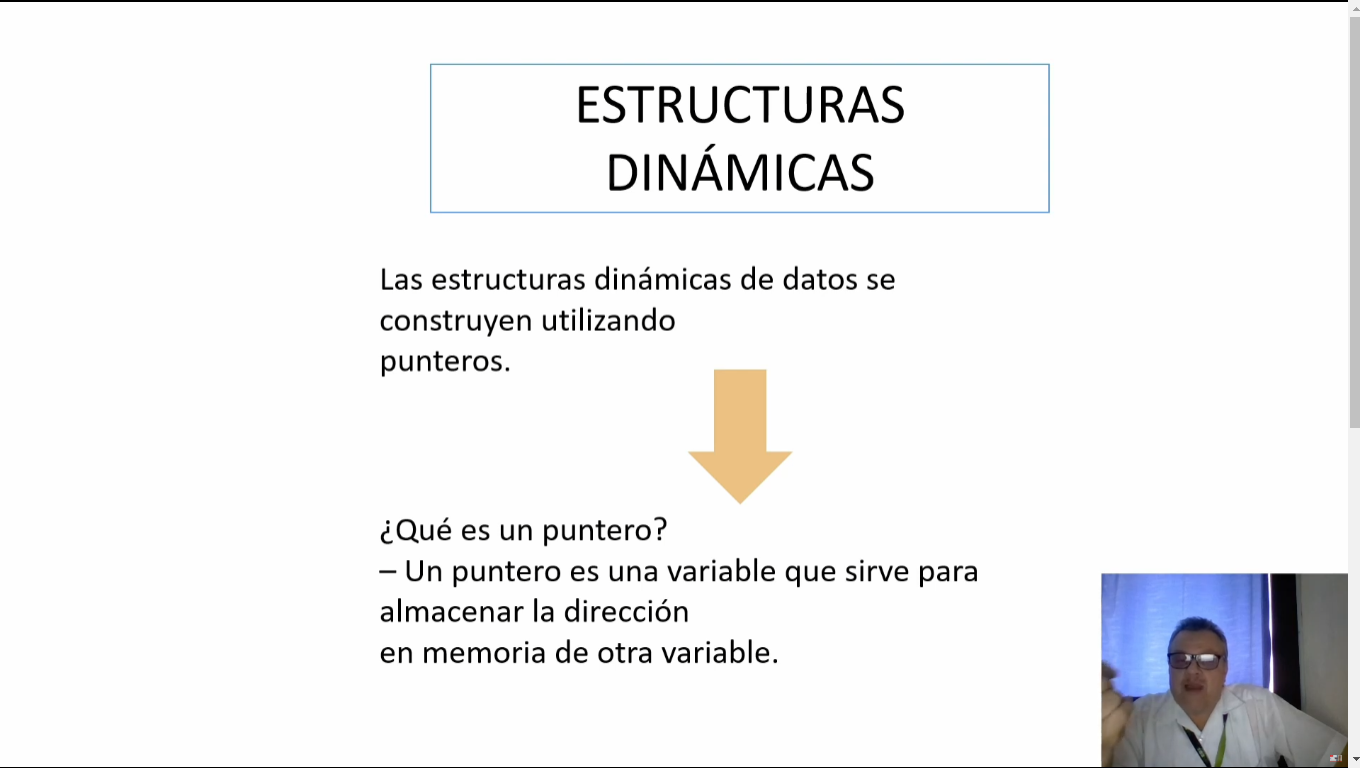


Punteros. Apuntan a direcciones de memoria, para que se pueda localizar alguna información.

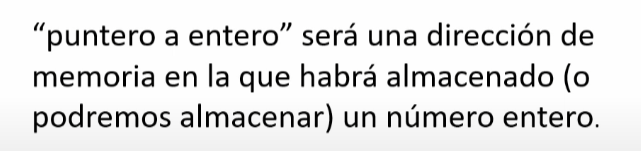
Los estáticos son como datos del propio sistema.

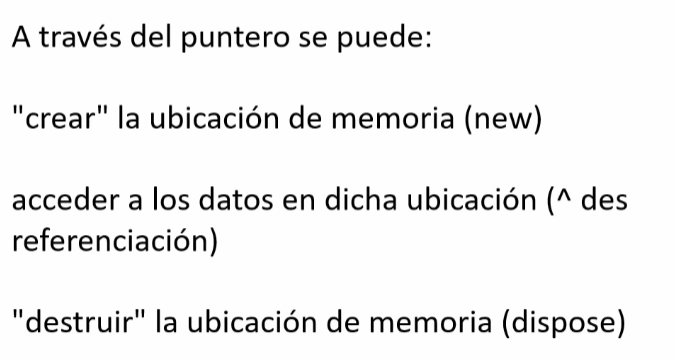
A los dinámicos no se les asigna memoria o recursos desde un inicio, sino que se les da a medida que se los necesita. Pueden cambiar su tamaño cuando se las va utilizando.

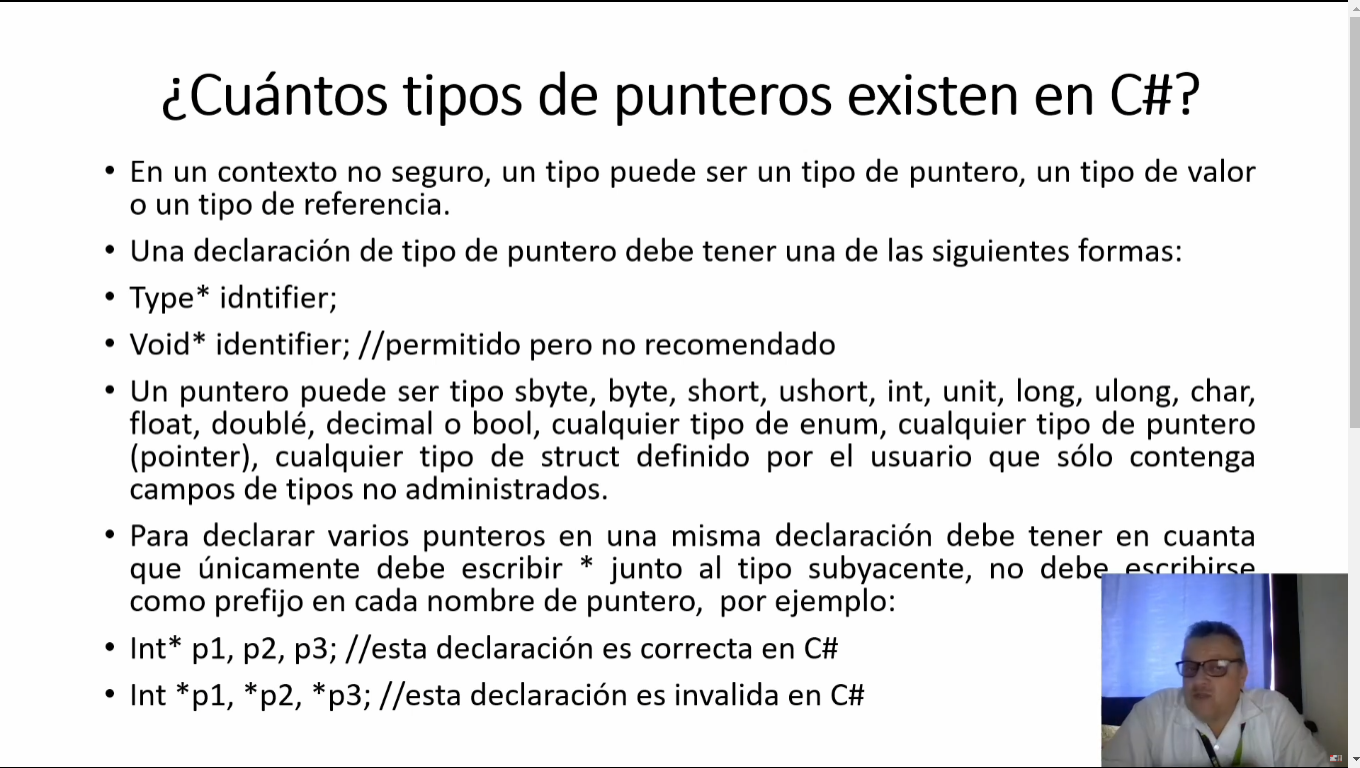
Todas las dinámicas utilizan punteros.

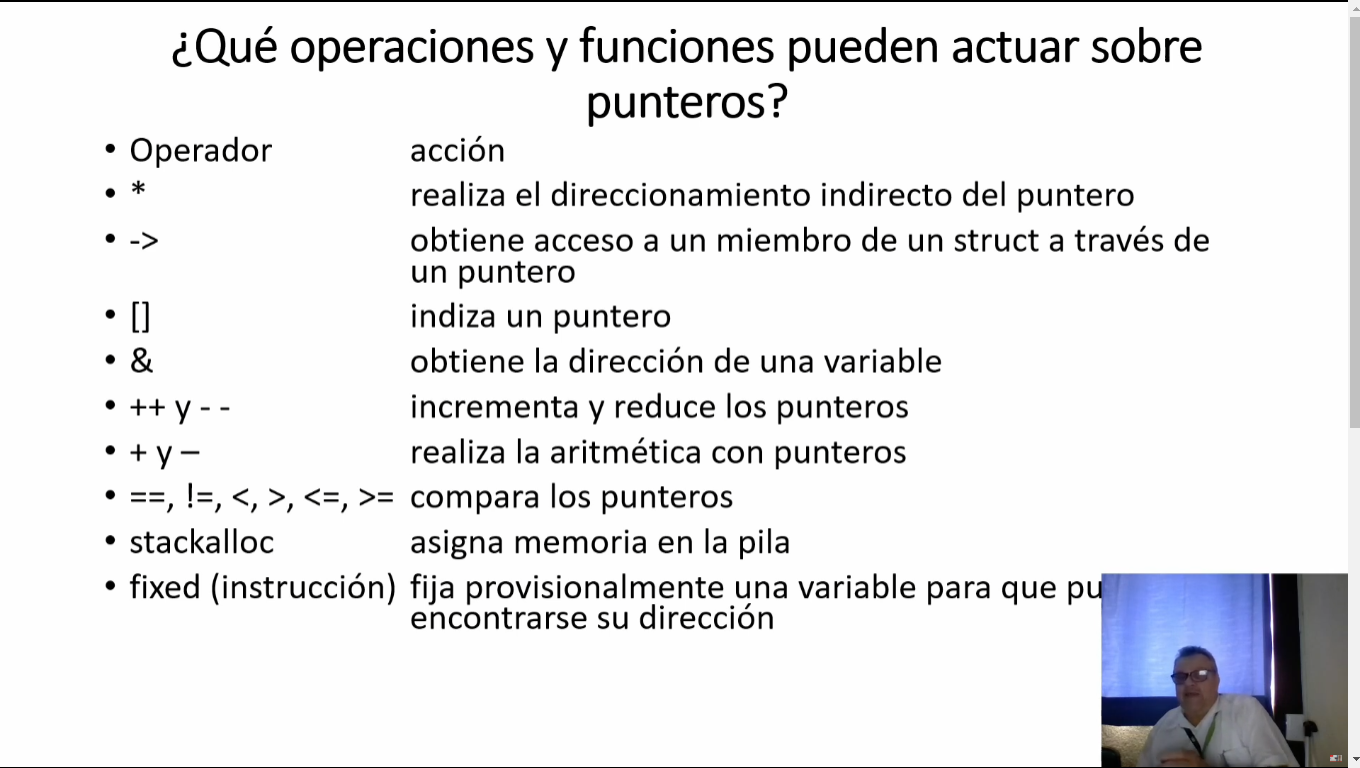


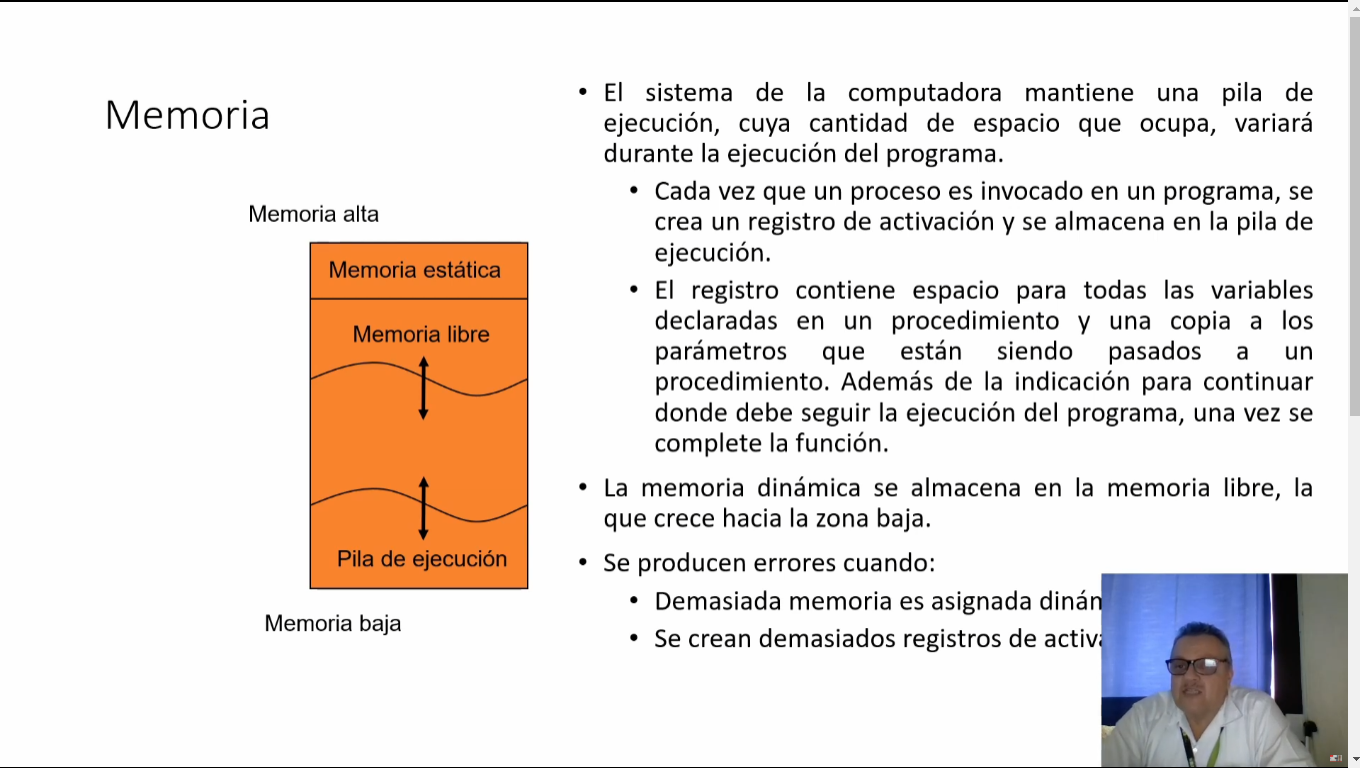
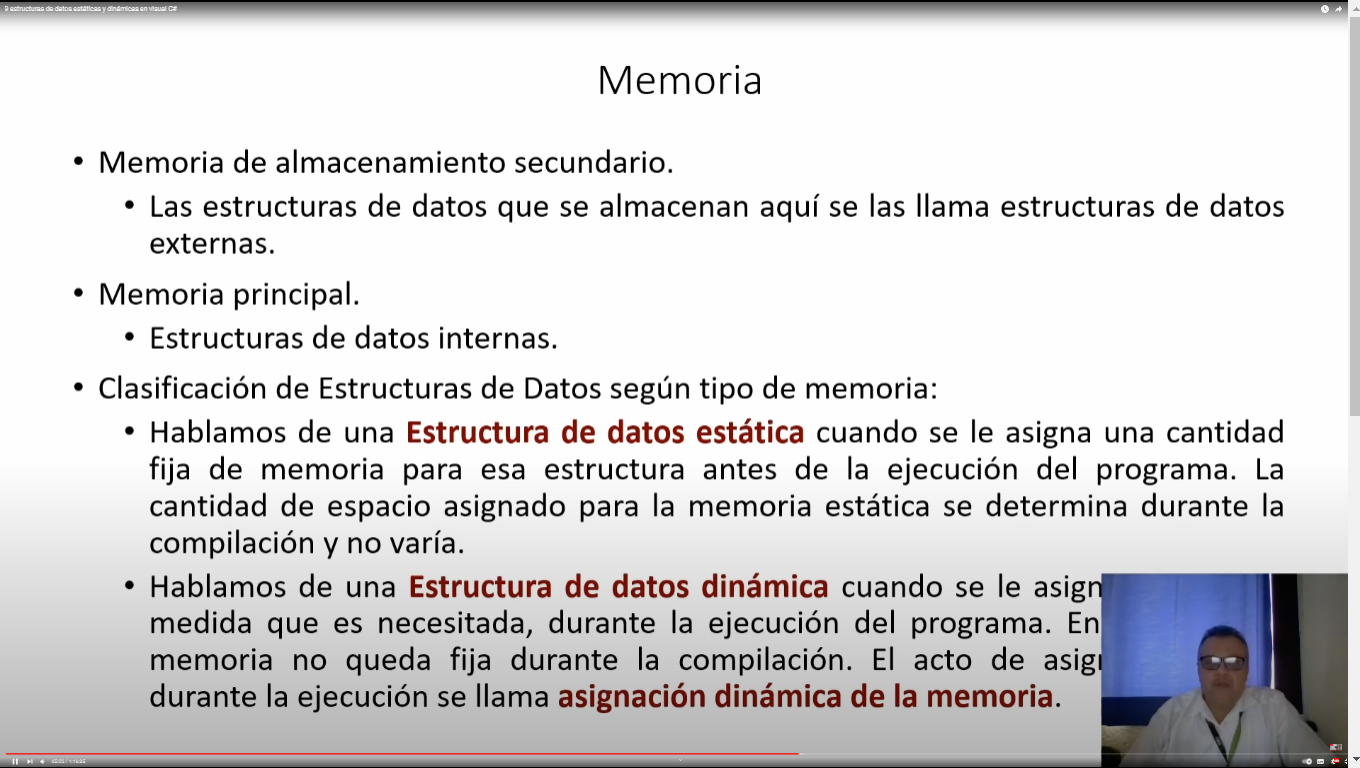
Varios tipos de punteros:

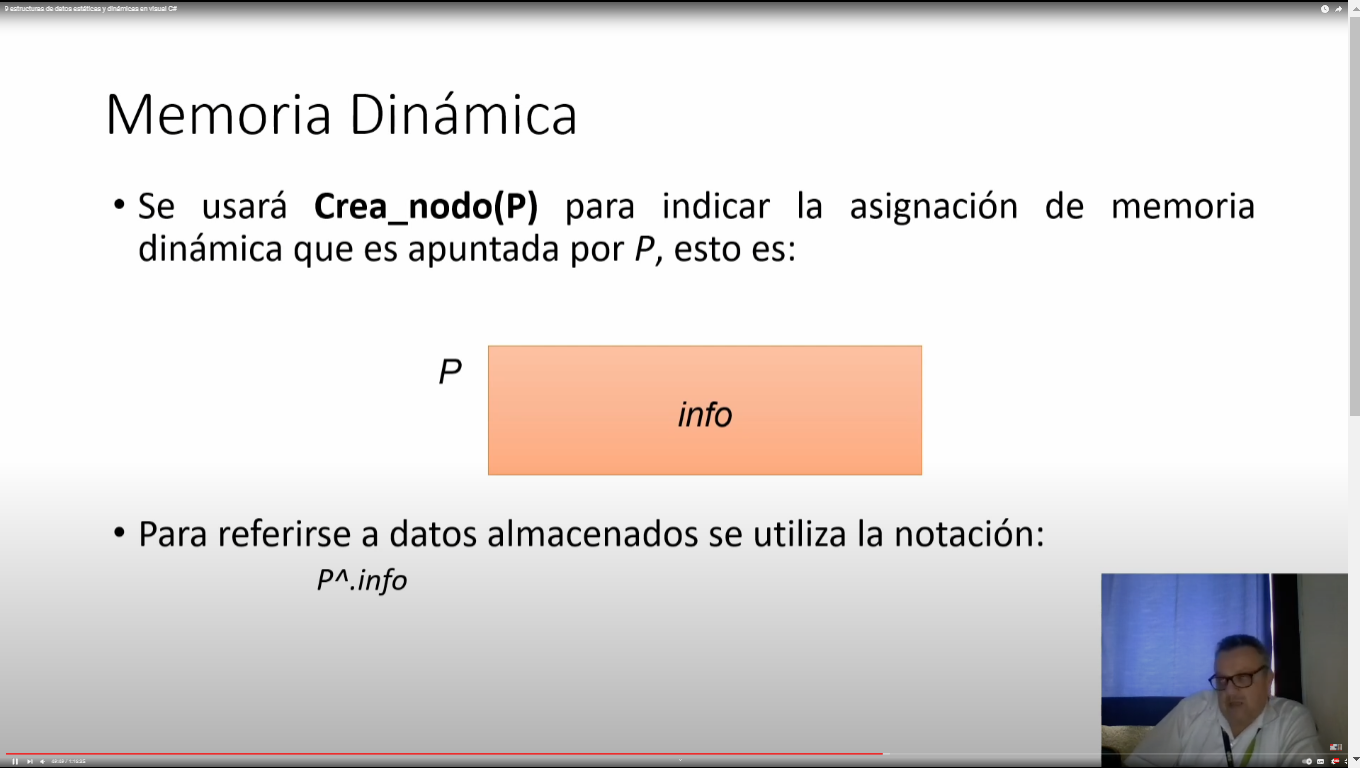
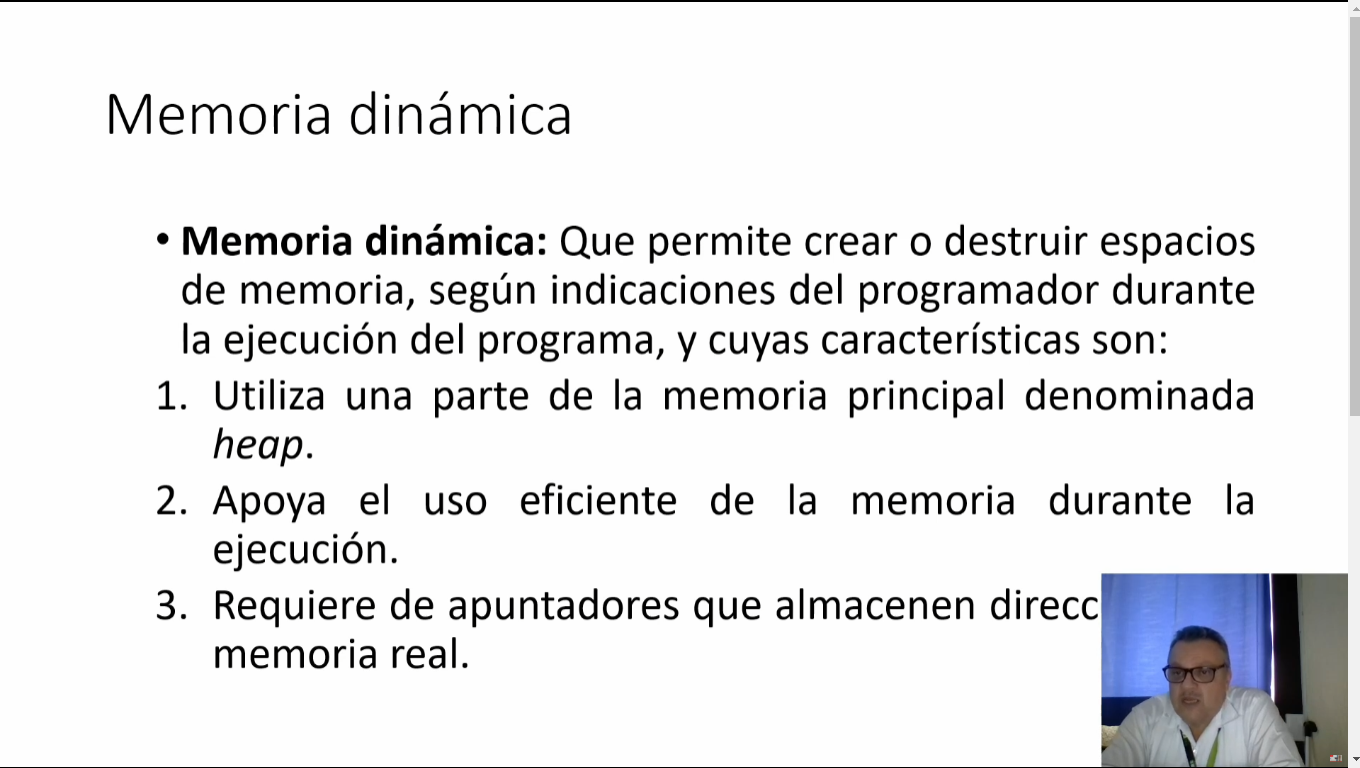


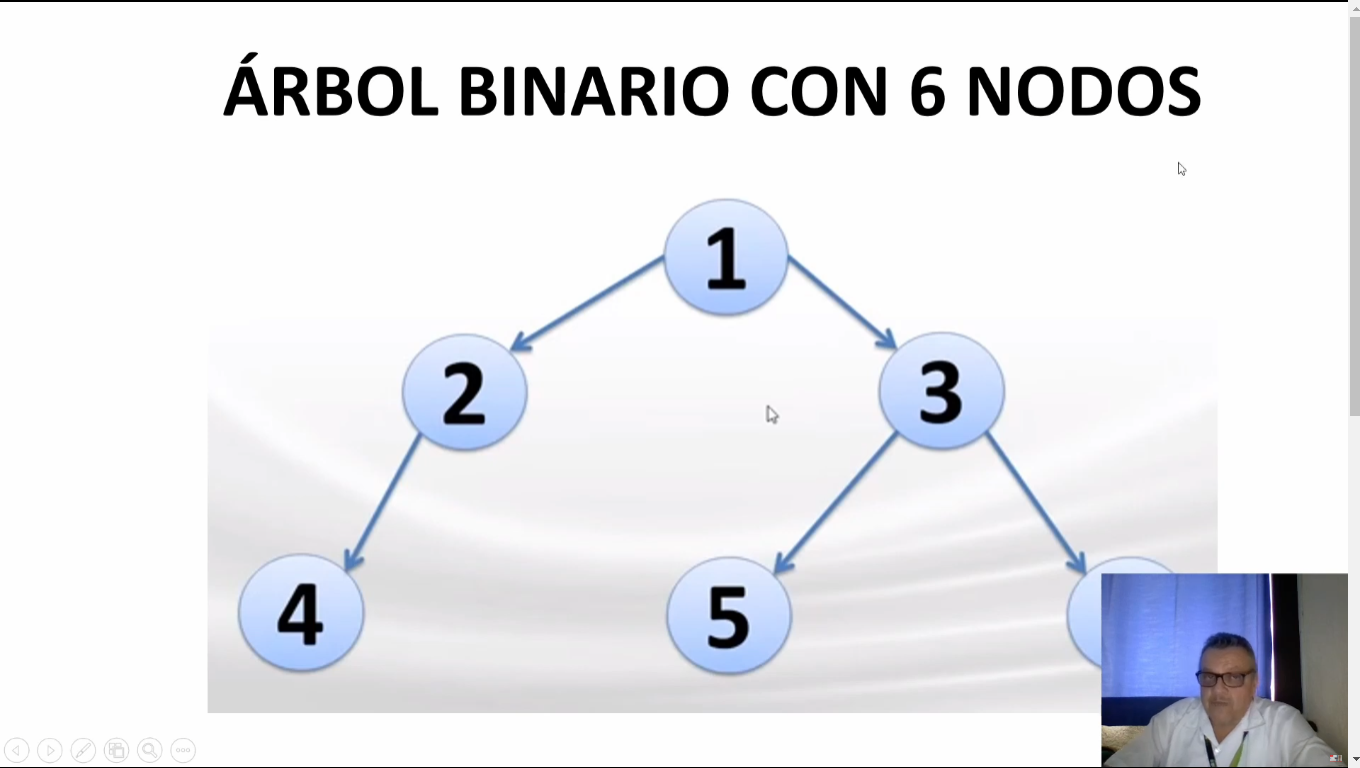
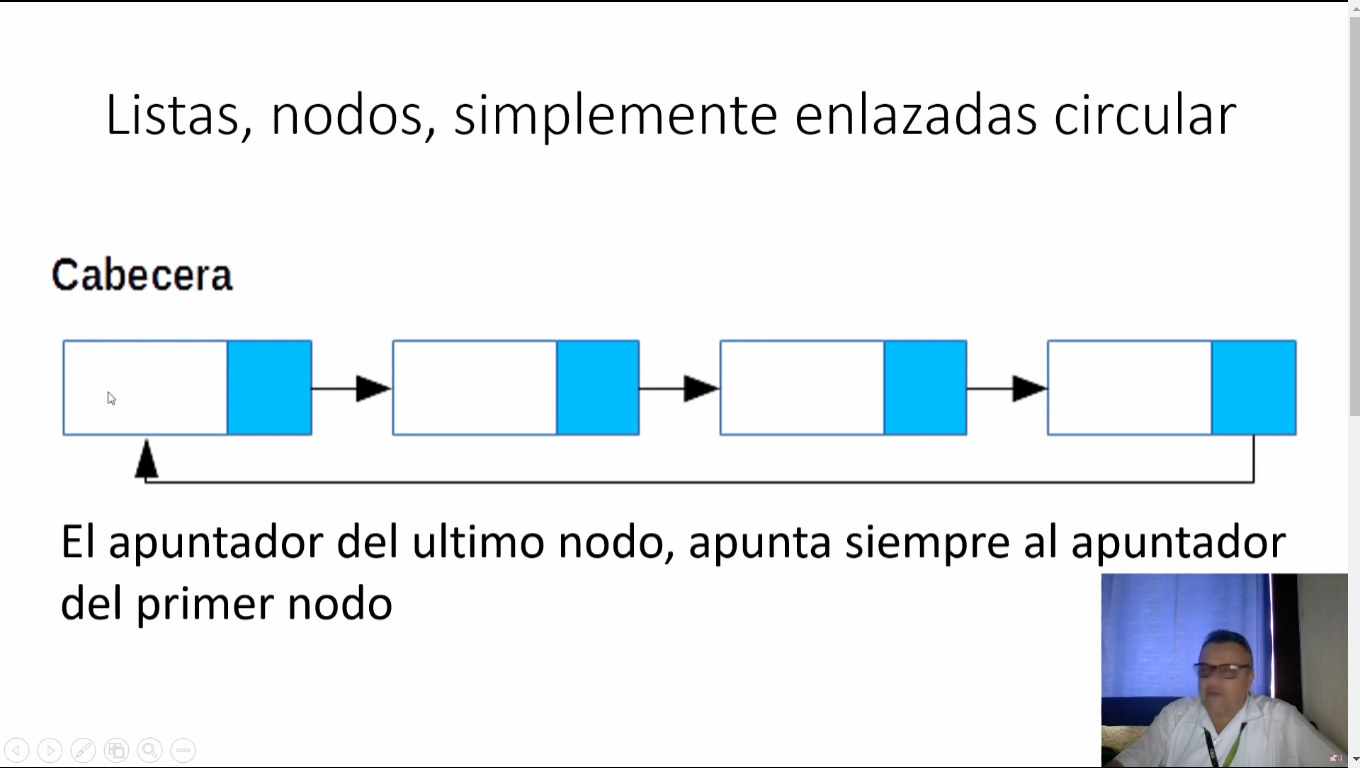
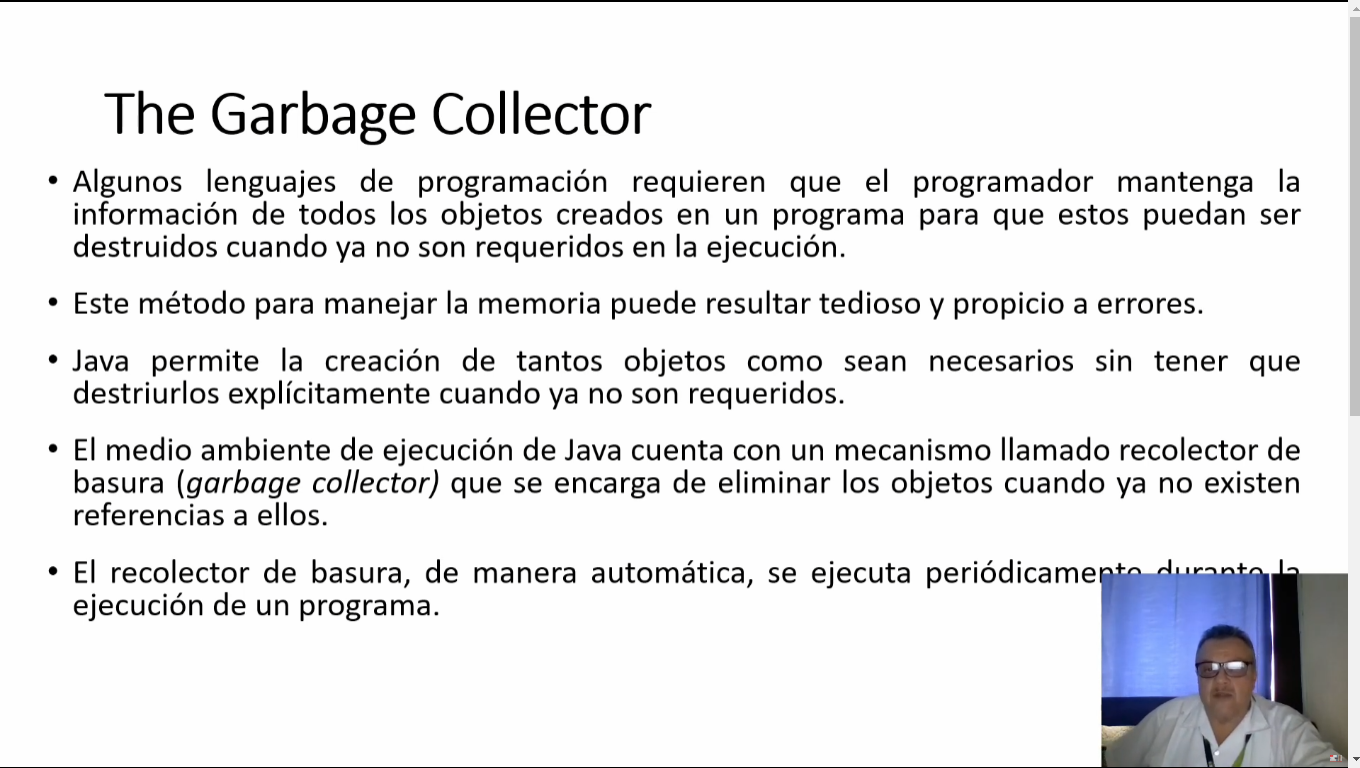
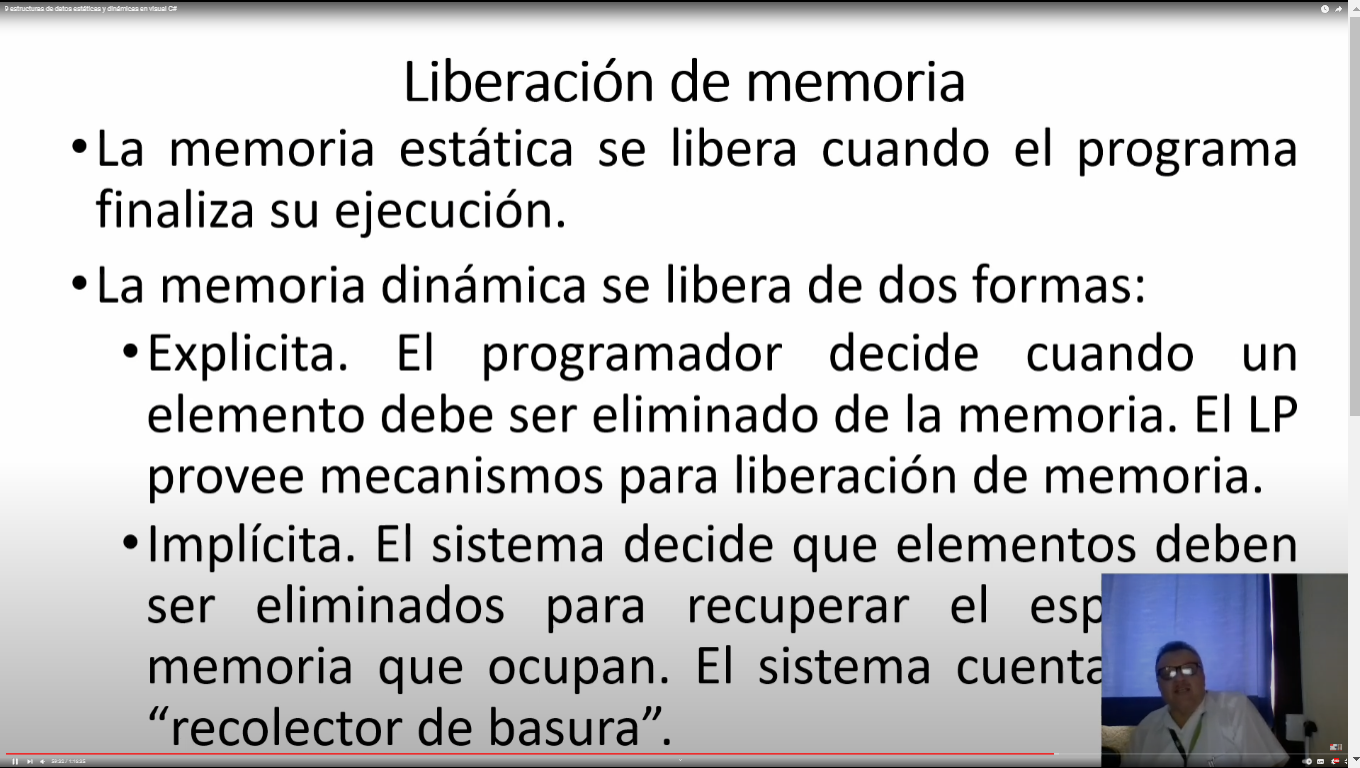
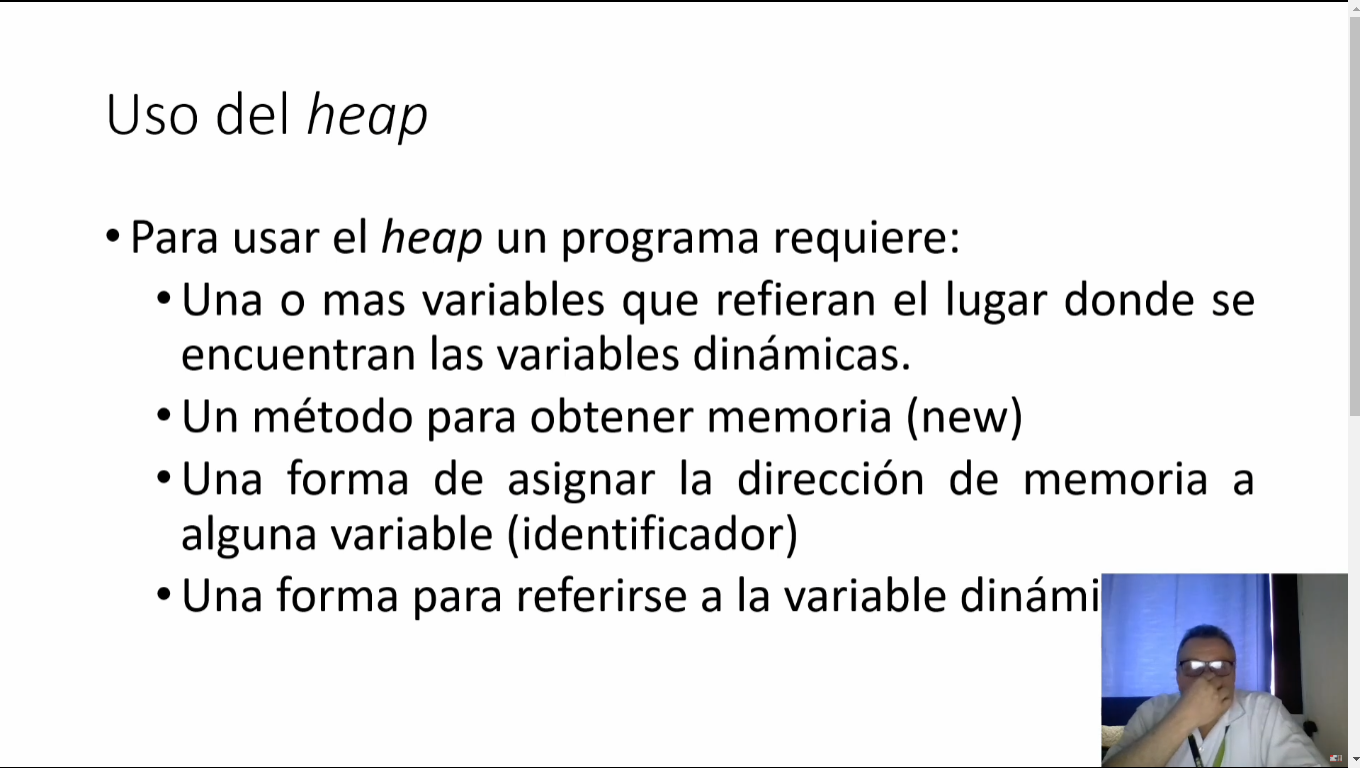
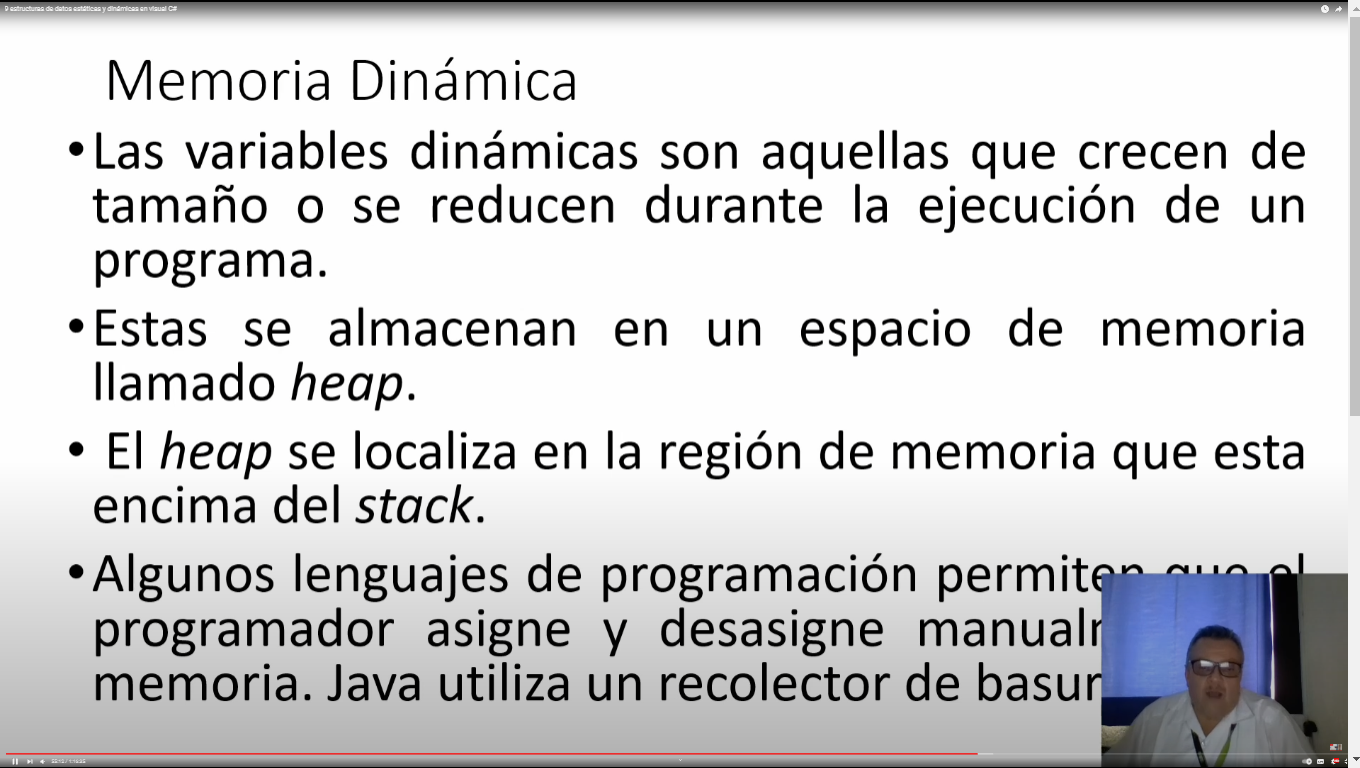
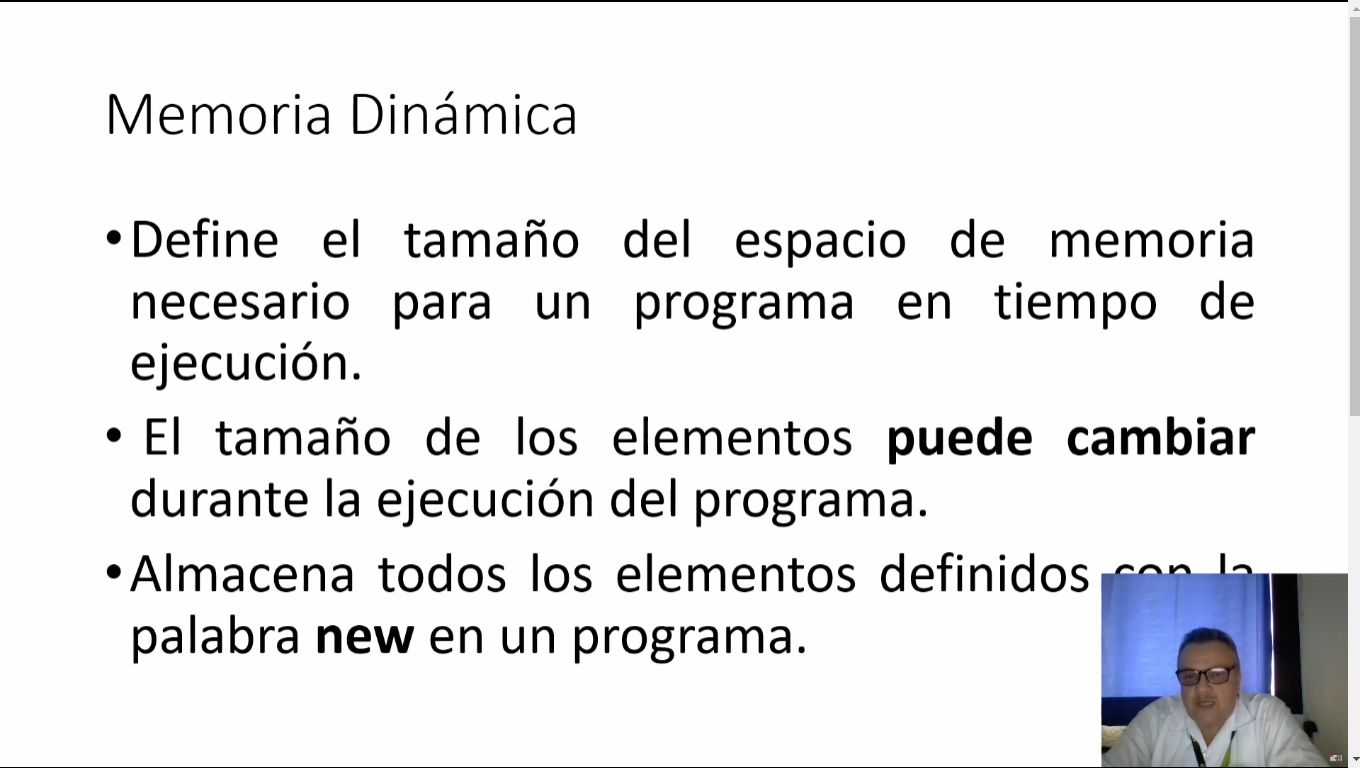
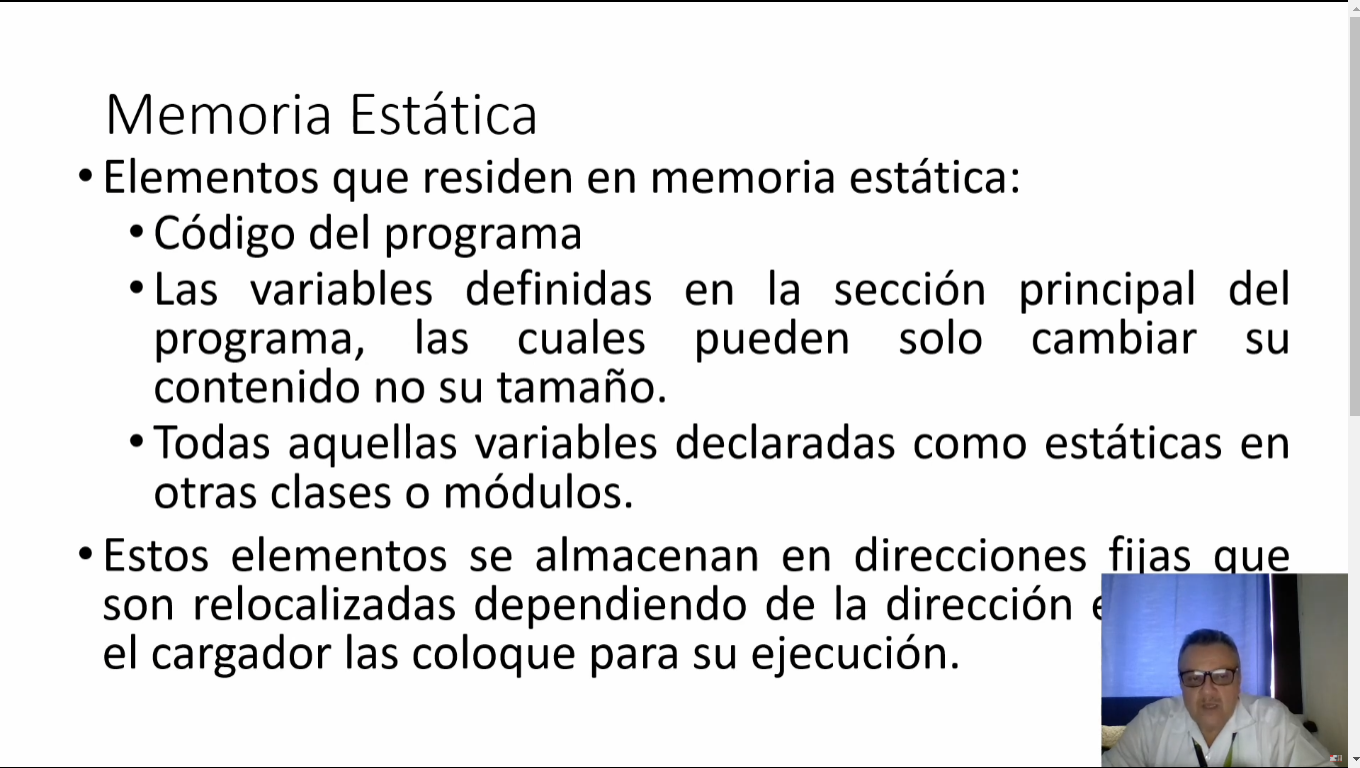
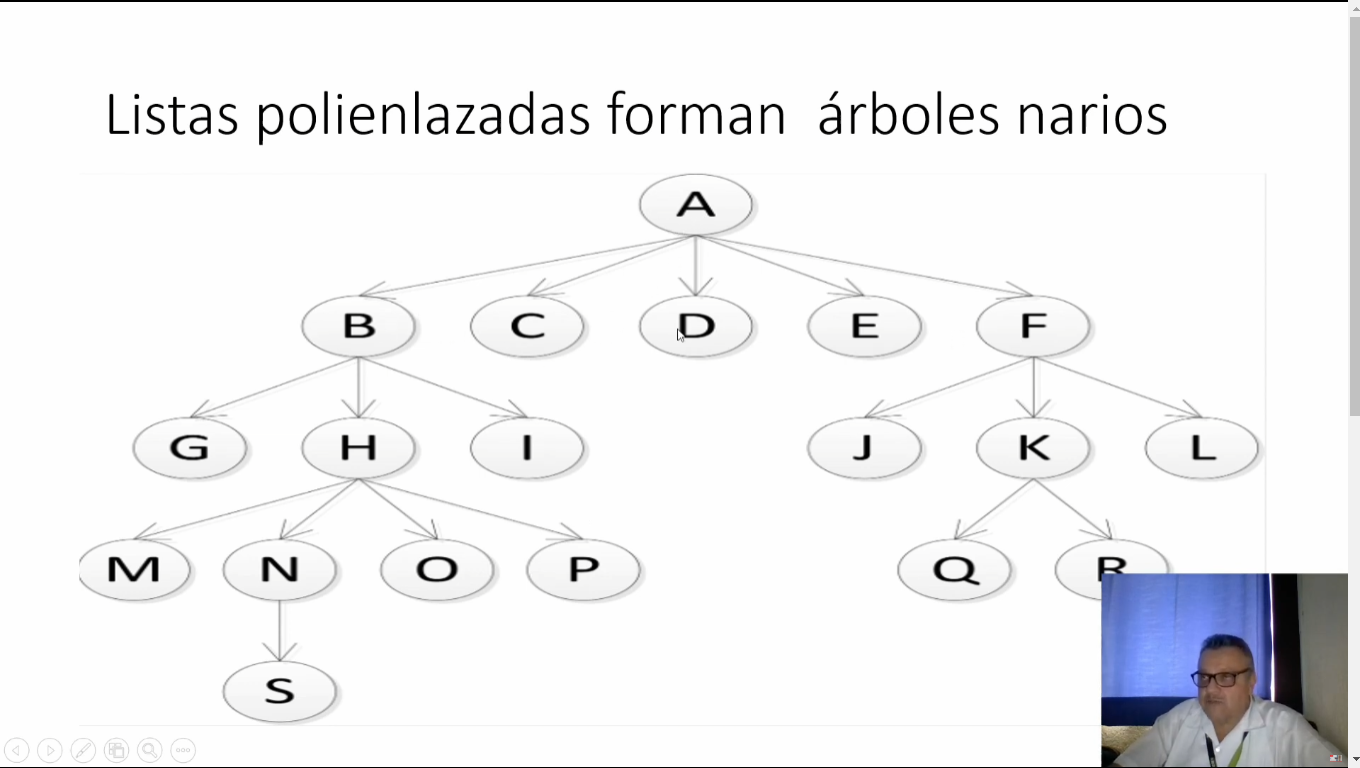
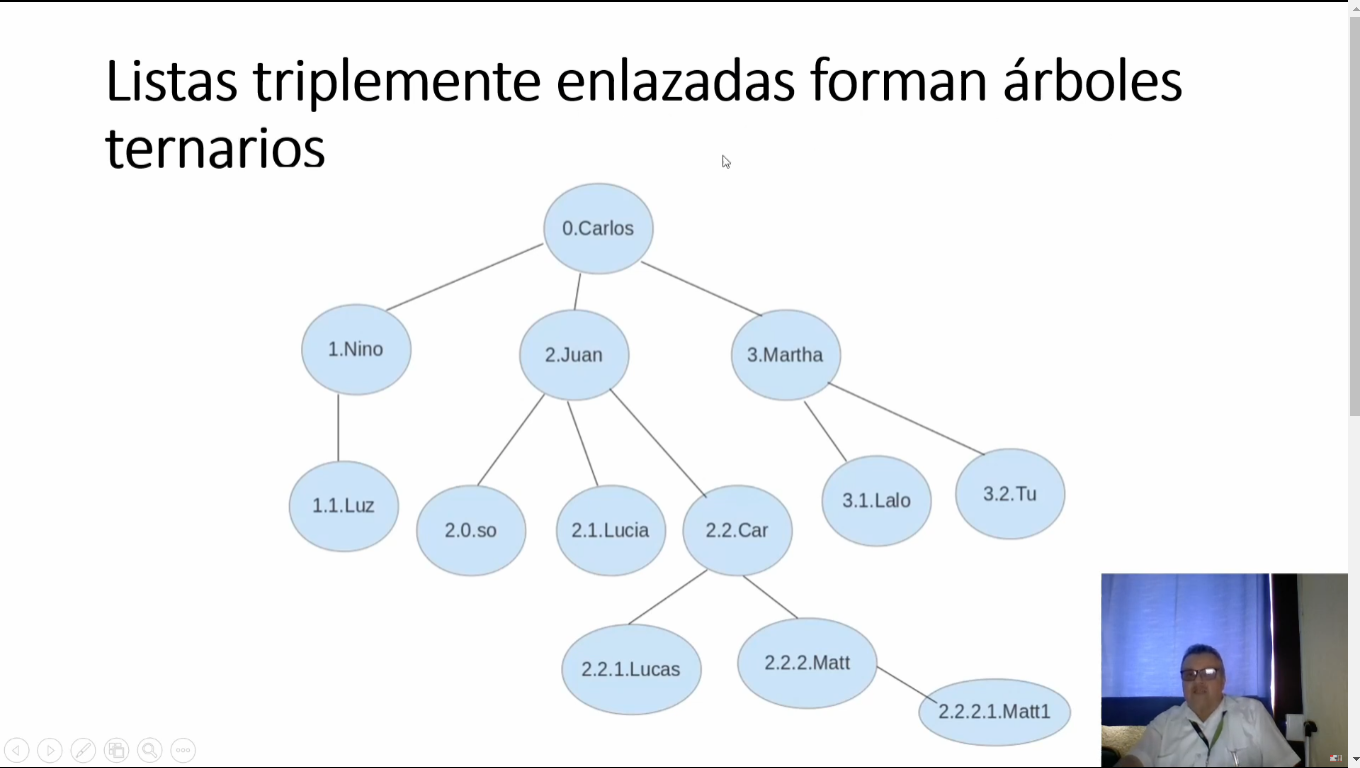
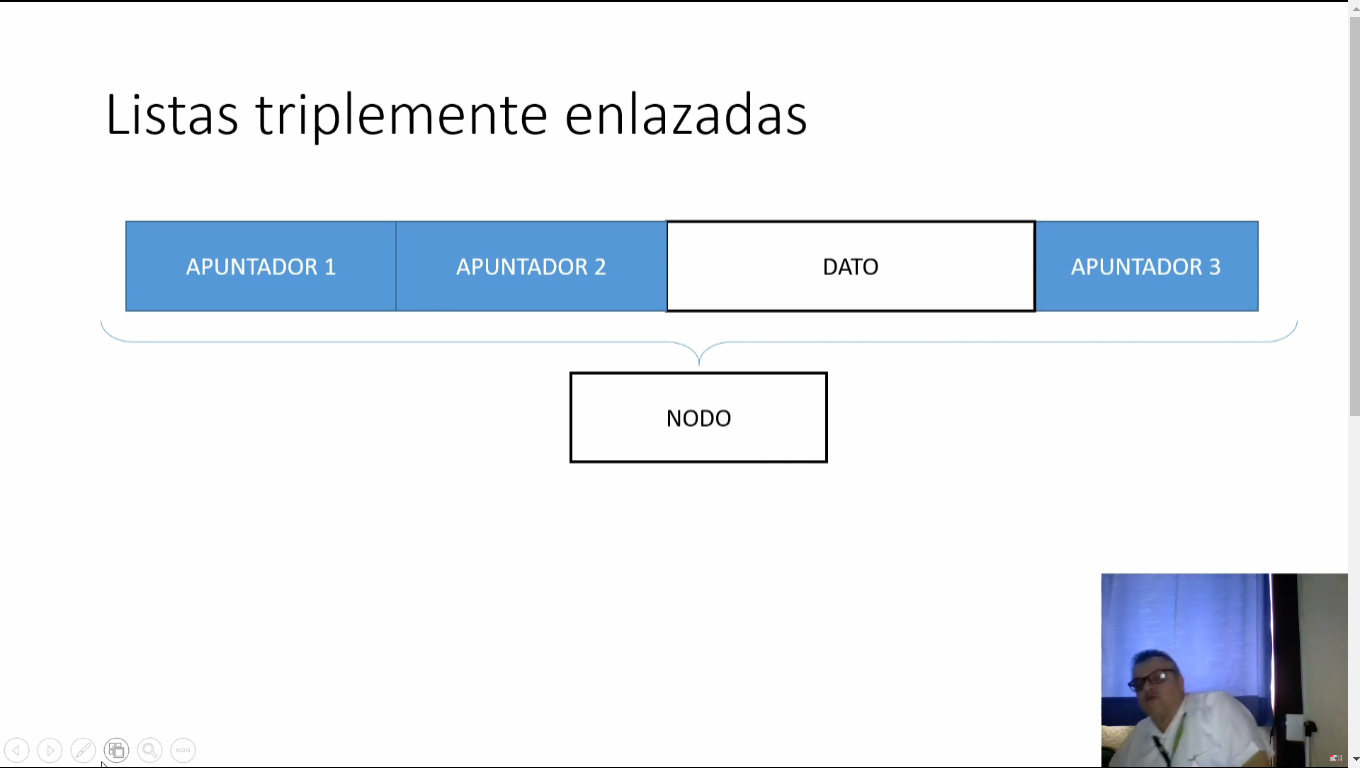




El \* indica que es un puntero y que está buscando una dirección de memoria. 





En estos binarios, cada padres puede tener, máximo 2 hijos. 

2 tipos de memorias

-Stack. Metódico, ordenado, estricto. Estructura LIFO. Ultimo que entra, primero que sale. Se guardan las variables primitivas y las referencias a los objetos declarados.

-Heap. Hermano desordenado, lento. Memoria dinámica. Los objetos se almacenan aquí (en stack solo sus referencias)

C# tiene garbage collector.