Carne: 2025072099

Grupo: 02

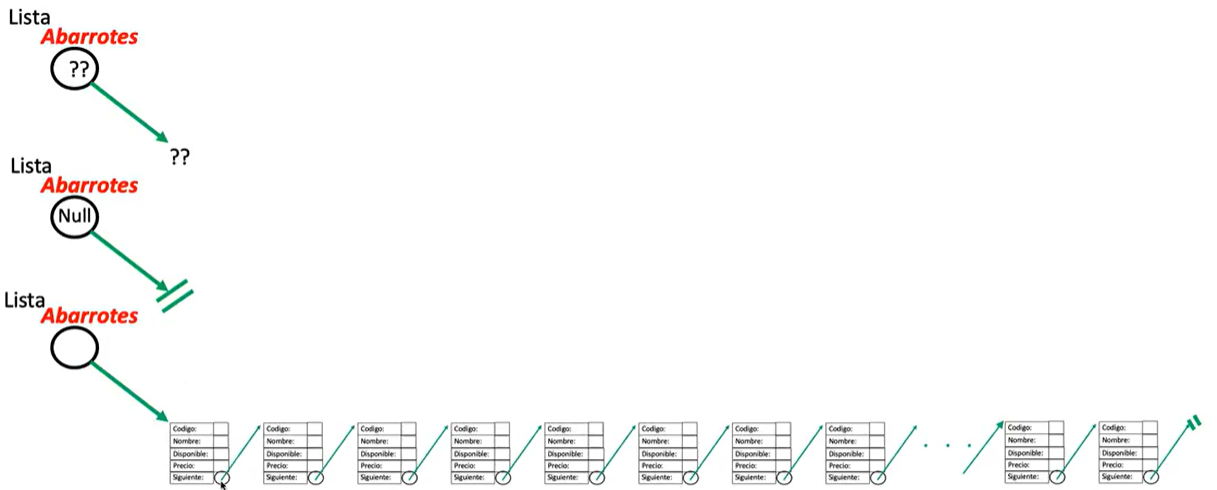
Estudiante: Ronny Espinoza Cordero

Aula invertida #3 Lección Lista Enlazada Simple

---------------------------------------------------------------------------------------------------------------------

**Listas enlazadas**

**Un punto que se debe recalcar aquí son los punteros, aquí se van a ver tres casos, el puntero que de momento no apunta a un valor especifico, el que apunta a Null y el que apunta a una lista enlazada.**

****

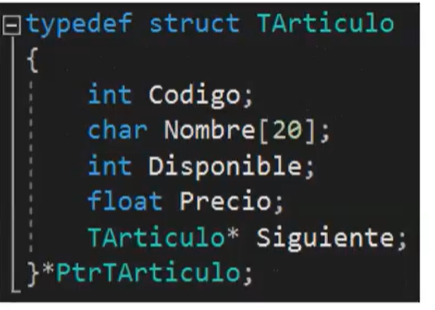
Conceptualmente hablando una lista enlazada es una estructura de datos lineal dinámica en la que solo se conocen el inicio y el fin.

El inicio es el nodo al que apunta el puntero listo (el primer puntero)y el final es un nodo cuyo puntero siguiente apunta a NULL.

Cada nodo es un conjunto de datos más un puntero que apunta al próximo nodo a excepción del ultimo nodo que apunta a NULL. Un nodo se puede entender como algo así:



Para entender mejor el como funciona una lista enlazada se puede ver un ejemplo en programa:



Aquí el proceso se puede dividir en pasos:

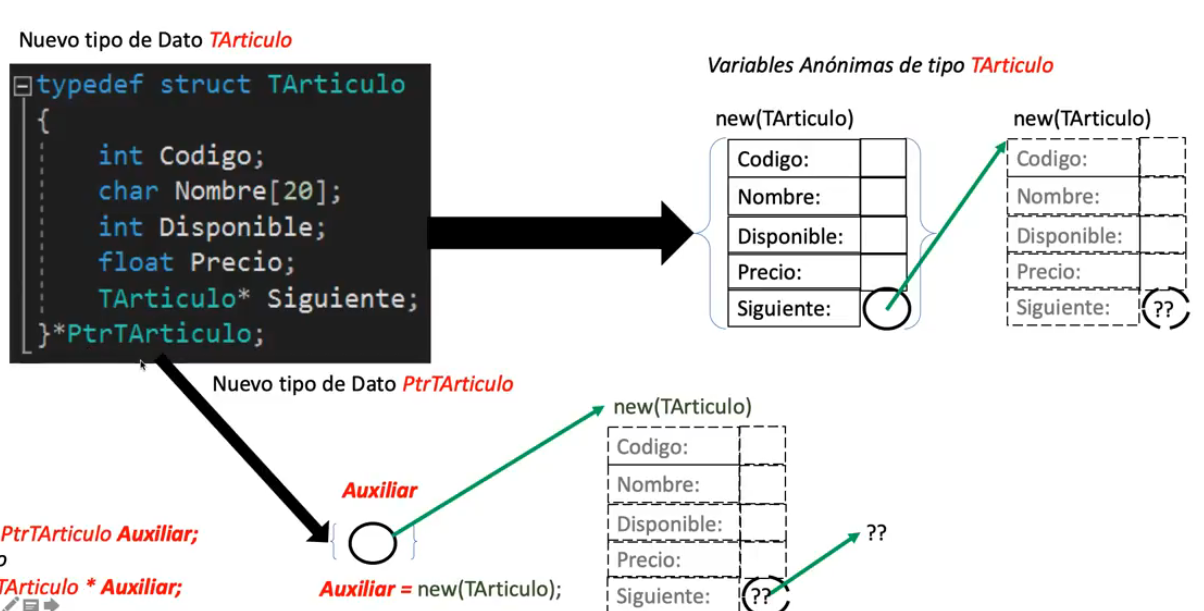
1. Se crea el nuevo tipo de dato llamado TArticulo con el typedef

2. Se definen los datos que va a contener el struct.

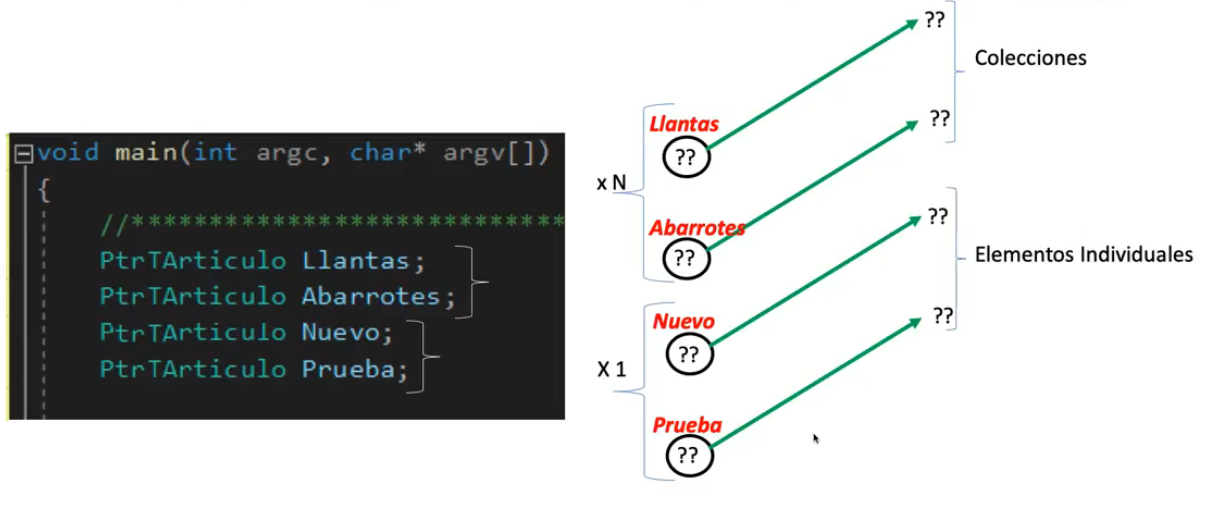
3. Despues de declarar los datos se puede ver la declaración del puntero Siguiente, este puntero apunta a elementos de tipo TArticulo, generando un tipo de declaración recursiva ya que no se ha terminado de declarar el nuevo tipo de dato y ya se esta declarando un subcampo que apunta a un elemento del tipo TArticulo

4. Por ultimo se genera un nuevo tipo de dato llamado PtrTArticulo que solo puede apuntar a elementos de tipo TArticulo. Con este tipo de dato se le pueden asignar punteros que apuntan a elementos de tipo TArticulo con instrucciones como PtrTArticulo Auxiliar o TArticulo \* Auxiliar.

Todo este proceso sirve para tener un tipo de dato que se va a poder usar en variables anónimas y que a la vez estas variables generen un enlace hacia otro tipo de nodos. Ademas el puntero creado se puede usar para este enlace también. Con esta imagen se entiende mejor:

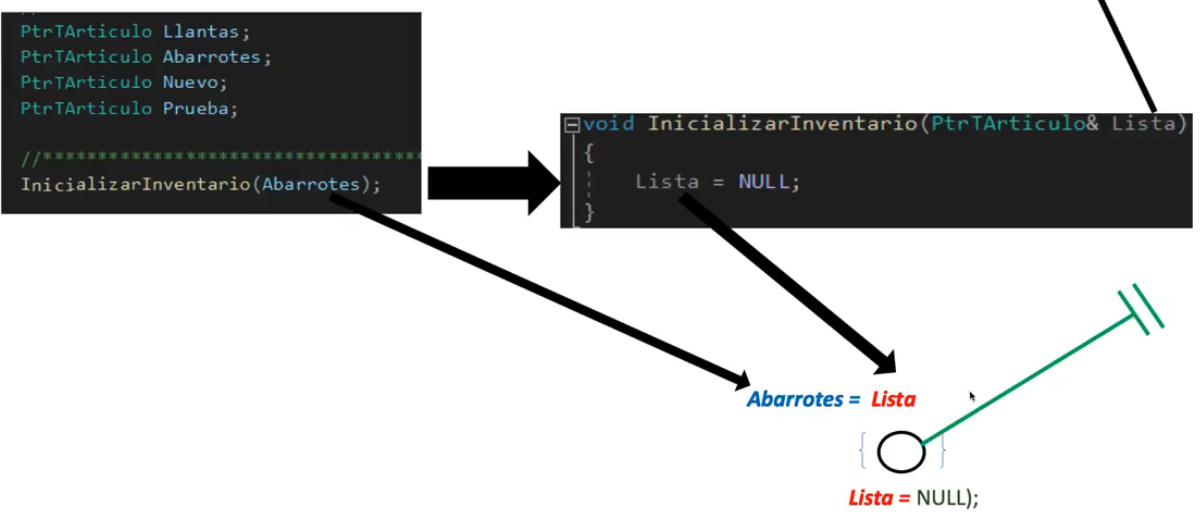


**Crear variables estáticas con punteros.**



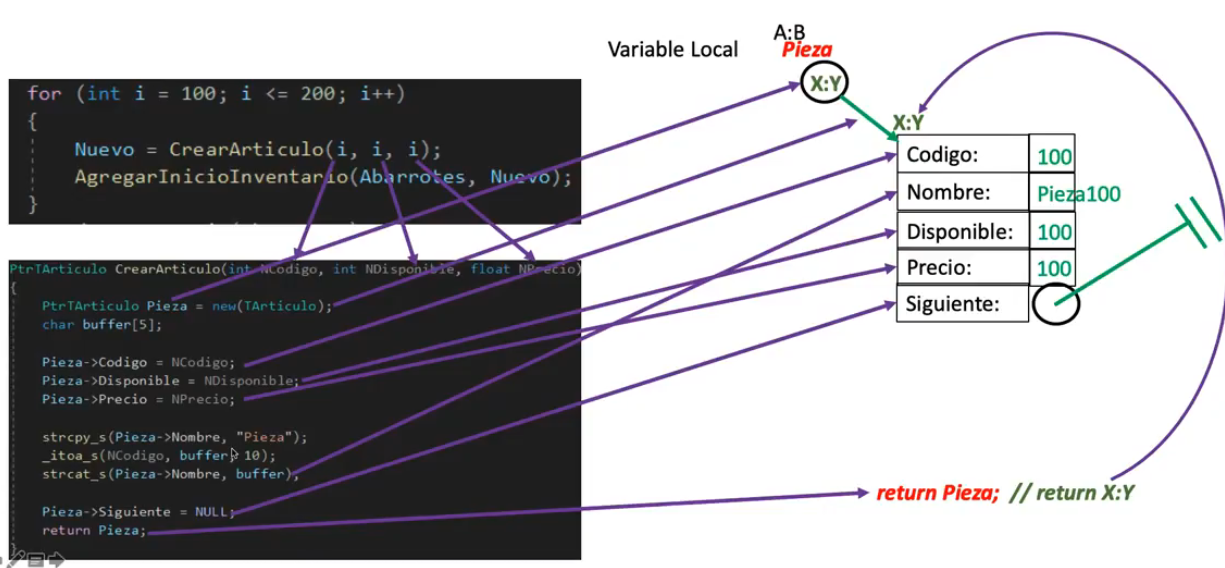
En esta imagen se ve que las variables Nuevo y Prueba están escritos en singular ya que solo contendrán elementos individuales y las variables, llantas y abarrotes están escritos en plural ya que corresponden a colecciones.

**Inicializar lista: Abarrote**



En esta parte se puede ver que en la función principal se llama a la función secundaria InicializarInventario y se le manda el puntero Abarrotes, desde la función secundaria se puede ver que lo se esta mandando es la dirección del puntero ya que usa un parámetro por referencia abarrotes y ahora Abarrotes vale NULL. Con este fragmento se puede ver un ejemplo de los tipos de parámetros y como se pueden usar en este tipo de códigos.

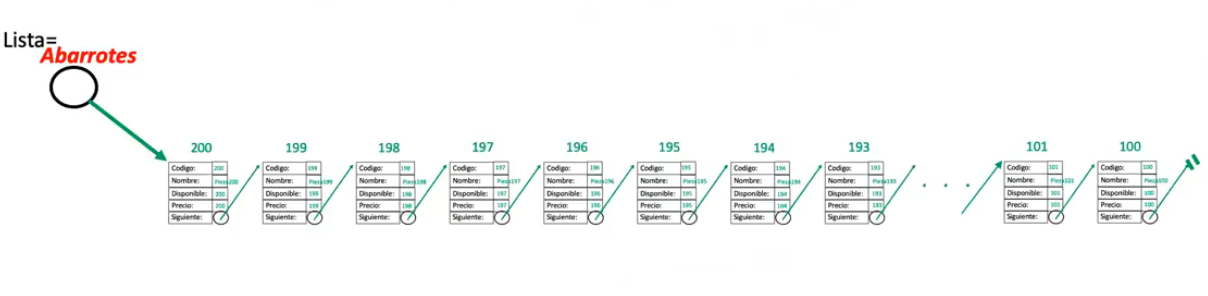
**Creación de articulo individual**

****

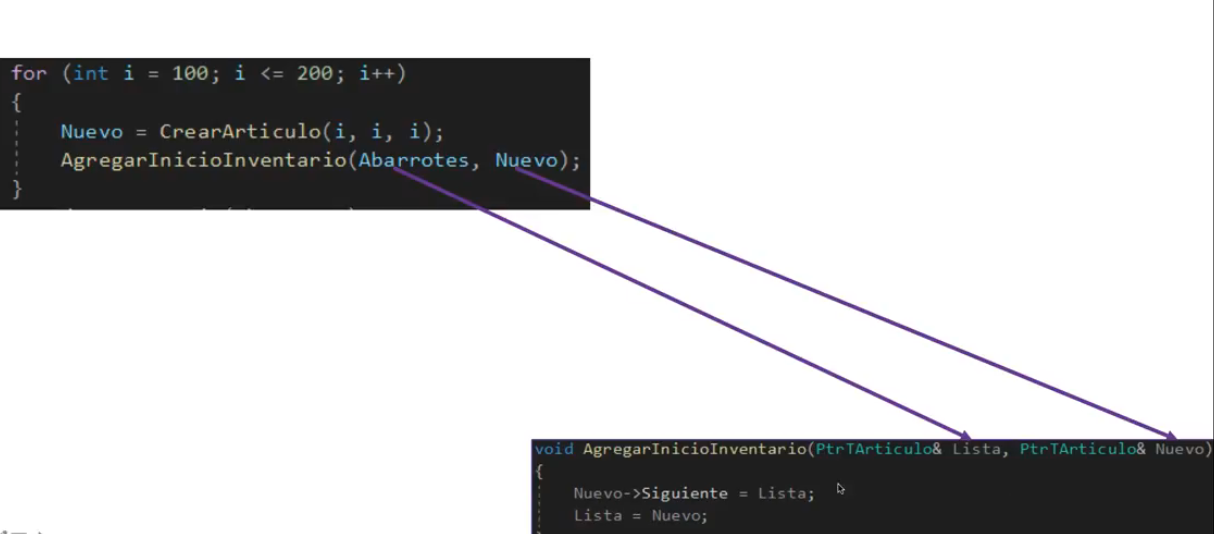
En este nuevo fragmento de código se utilizan las funciones CrearArticulo y AgregarInicioInventario dentro de un ciclo for de forma que se agregan los nodos a la lista, en este caso la función siempre va a agregar la lista al principio, es una lista invertida, no recorre toda la lista hasta llegar al final lo cual lo hace muy eficiente como un algoritmo para agregar elementos en una lista.

En la función secundaria se pasan los valores y se rellenan los datos en la estructura con el nodo apuntando a NULL ya que es la primera estructura que se llena, pero antes se crea una variable local llamada pieza de tipo PtrArticulo que esta relacionada con una variable anónima de tipo TArticulo por lo que almacena la información de esta variable. Finalmente se retorna la variable Pieza que contiene la dirección del nodo que se acaba de completar.

Este código se puede ver así para entenderlo mejor:

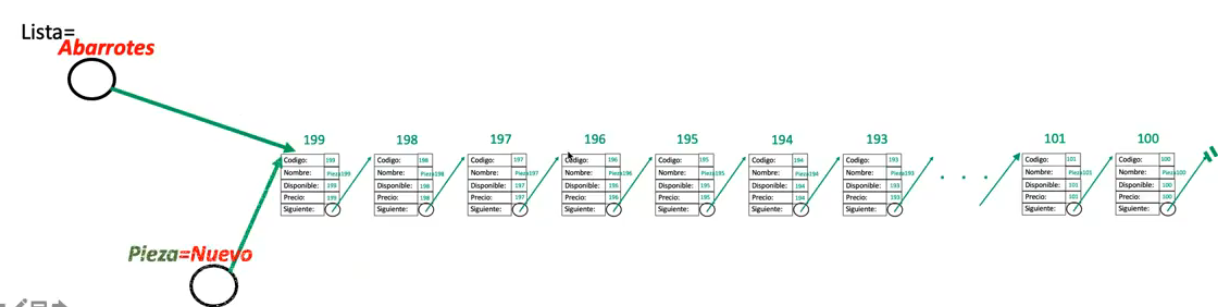


Ahora, anteriormente se vio la mitad del ciclo for, la segunda parte correspondiente a la función AgregarInicioInventario se puede entender así:

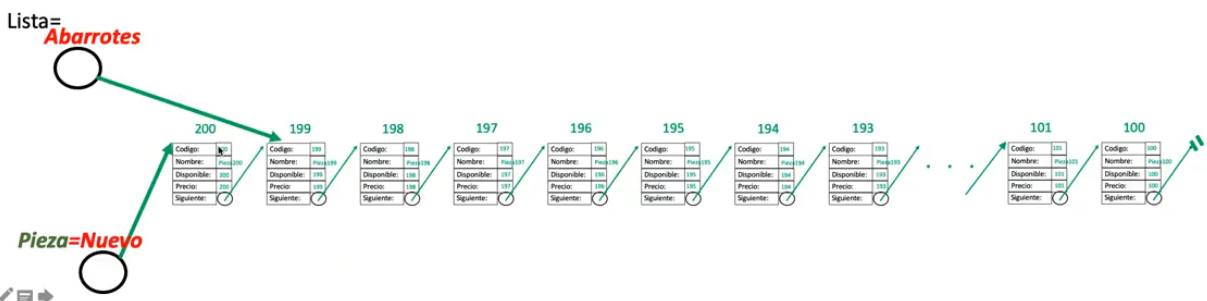


Desde esta función se envían los valores, Abarrotes, Nuevo. Estos dos valores representan la lista que va a ser modificada y la dirección del nodo que se agregó, como se dijo estos valores representan direcciones, son parámetros por referencia. Dentro de la función secundaria se puede ver cómo hay dos líneas de código, en la primera se hace que la Nuevo->Siguiente ahora sea Lista, aquí se debe recordar que Nuevo es la dirección del nodo, y Lista es el último nodo que fue creado. Este proceso puede resultar complicado por lo que se usara un ejemplo más practico con la siguiente secuencia de imágenes:

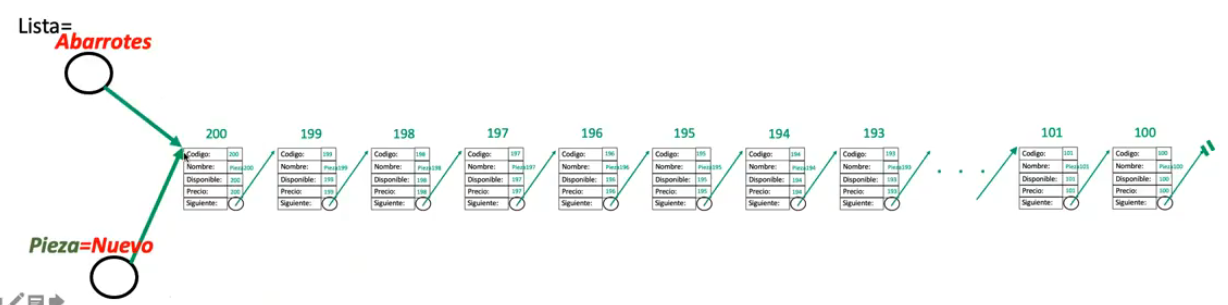
1. Lista apunta al ultimo elemento creado al igual que Nuevo.



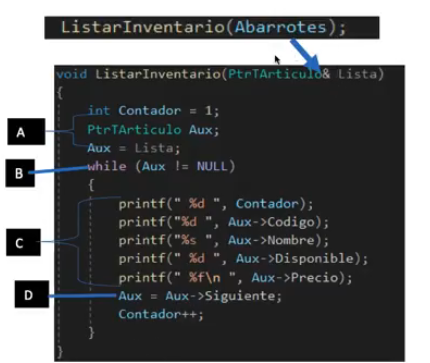
2. Se crea un nuevo elemento y ahora Nuevo tiene una dirección diferente pero a la vez Nuevo->Siguiente apunta al valor que posee Lista en ese momento



3. Y por ultimo la dirección de Lista pasa a ser la dirección del último elemento guardado, es decir la dirección de Nuevo.



**Listar listas enlazadas**

****

Un dato relevante aquí es que aunque en la función el parámetro sea un parámetro por referencia en realidad podría ser un parámetro por valor ya que en si la lista no se va a modificar, solo se va a acceder.

Ahora sí, este código se puede dividir por partes:

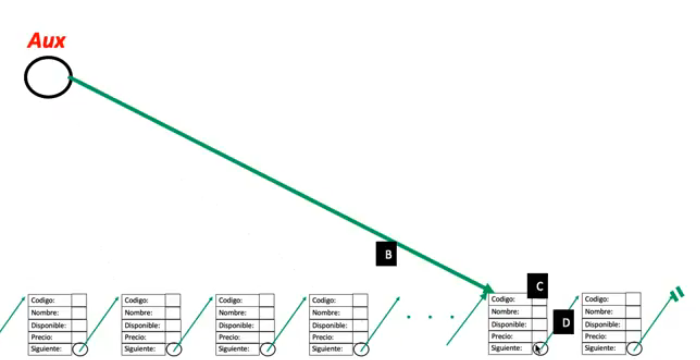
Parte A: Se inicializan dos variables, contador de tipo int y Aux de tipo PtrArticulo, el primero sirve para mostrar el número de nodo en el que se encuentra la información y el segundo es para poder acceder a los diferentes nodos.

Parte B: Se inicializa el ciclo que termina cuando Aux apunte a NULL.

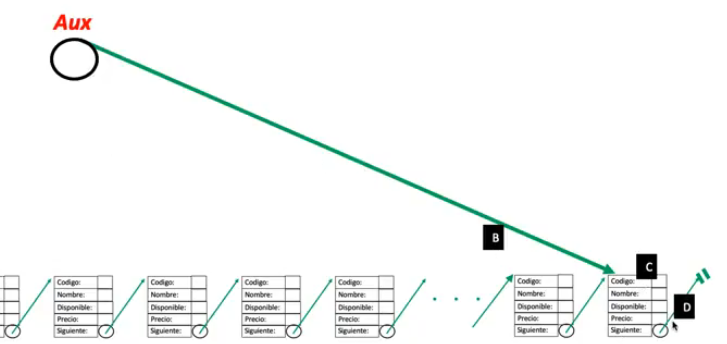
Parte C: Se imprimen los datos de cada campo por el que se pasa.

Parte D: Para poder continuar el ciclo la variable Aux obtiene la dirección del siguiente nodo accediendo a la variable Siguiente del nodo actual, de esta forma cambia su dirección.

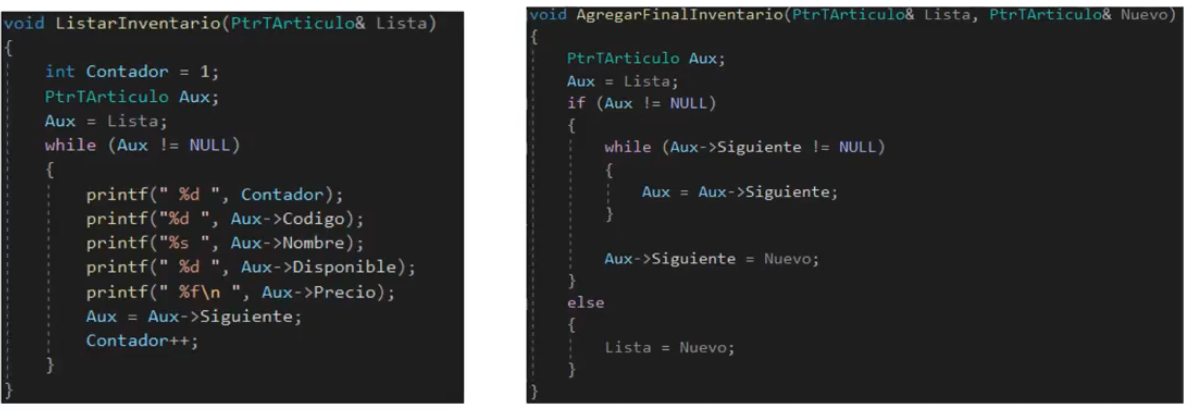
Esto se puede entender gráficamente así:

****

Y luego:

****

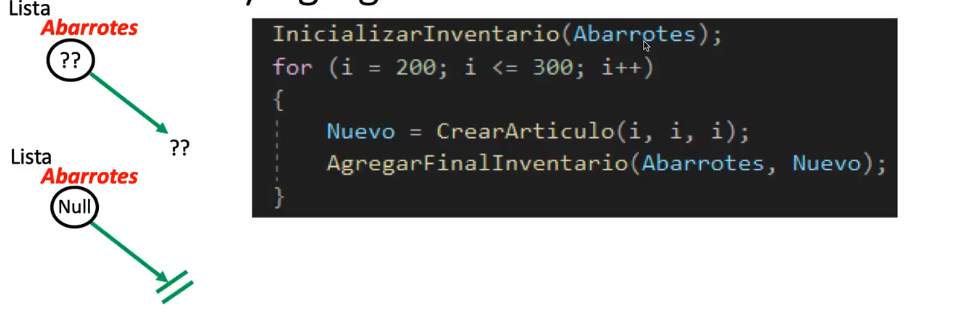
**Comparación entre listar y AgregarFinalInventario**



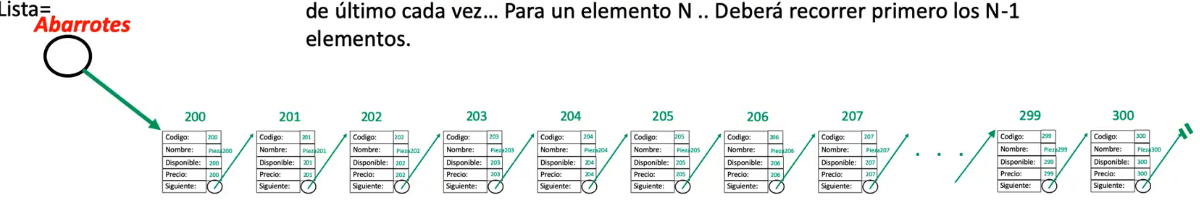
Entre estos códigos se pueden ver ciertas semejanzas, ninguna de las dos devuelve algo, ambas reciben un parámetro por referencia llamado Lista.

Sin embargo hay una diferencia clave entre estos dos conceptos, el ciclo, la función ListarInventario termina cuando se llega al último elemento, cuando se llega a Nulo, en la función AgregarFinalInventario el ciclo termina un paso antes de llegar a Nulo, esto es para poder agregar un elemento al final.

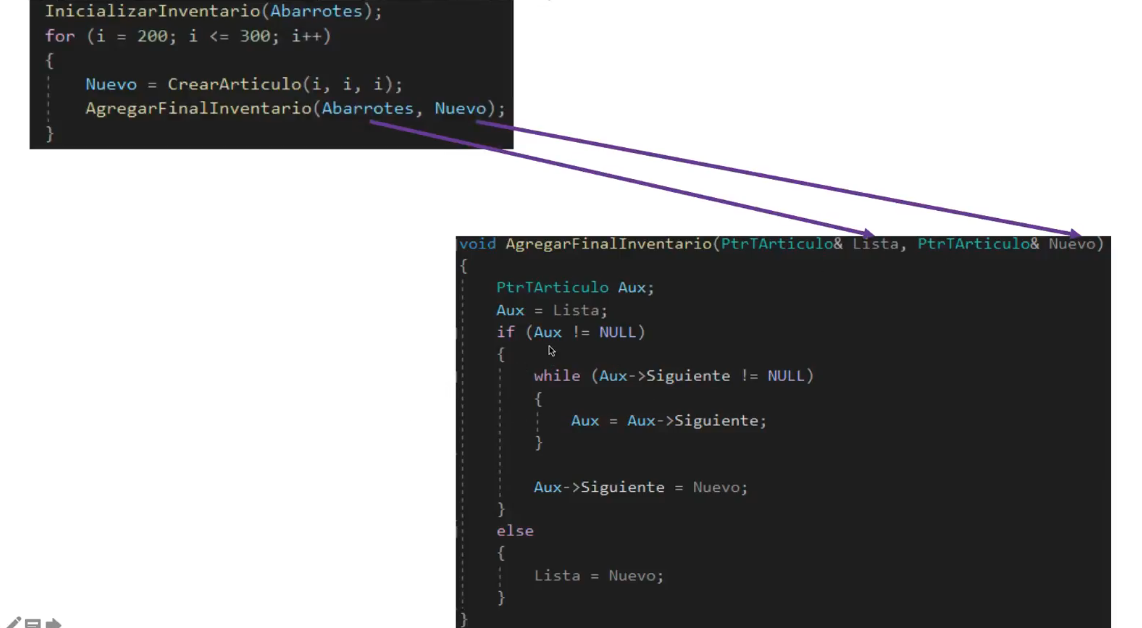
**Crear y agregar final desde i = 200 hasta i = 300**



Este código es muy parecido a la parte inicial de crear articulo a excepción de que esta vez se debe recorrer toda la lista para que así se pueda reconocer el final y de esta forma se pueda saber donde se debe insertar el nuevo valor. En otras palabras se pueden entender como una inserción por orden de llegada. Esto se puede representar así:

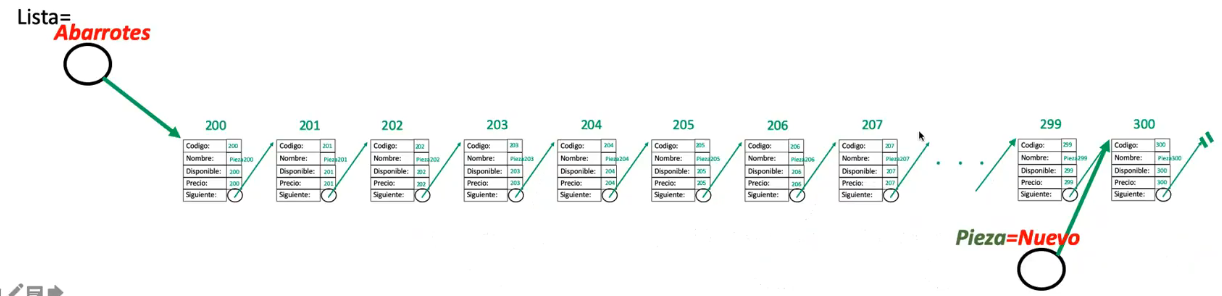


Explicación del proceso: AgregarFinal



Al llamar a la función también se le mandan las direcciones de la lista y la dirección generada en CrearArticulo. Luego en la función secundaria se genera la variable Auxiliar luego si la lista no está vacía empieza el ciclo, en este ciclo se analiza la posición siguiente para poder poner el elemento en última posición.

**Explicación a detalle 1**

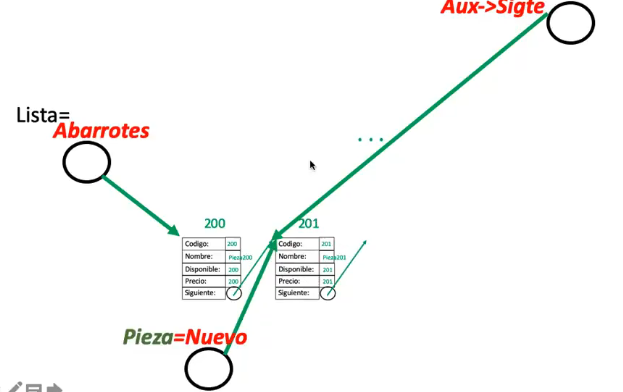


En esta explicación se habla del llamado a la función, aquí la dirección va cambiando a medida que se agregan elementos, mientras que estos se enlazan. Algo muy parecido a lo desarrollado durante el principio del resumen.

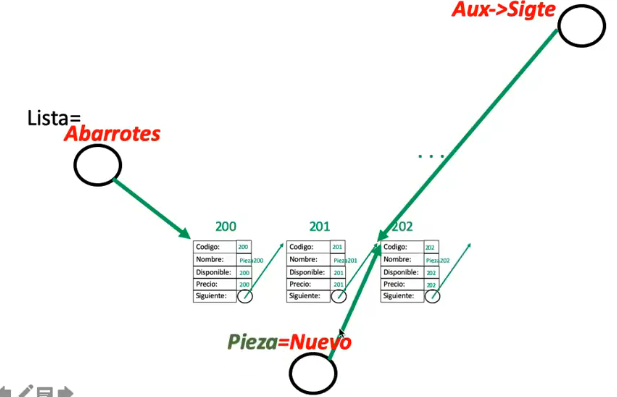
**Explicación a detalle 2**

Este ejemplo explica mejor el como funciona la función secundaria y para entenderlo mejor se debe dar una secuencia.

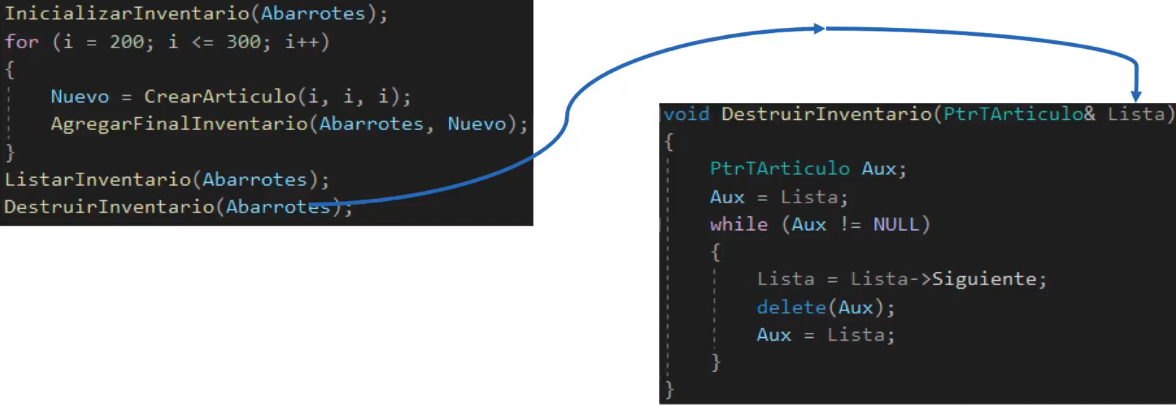
Primero se puede ver como en un principio el aux->Siguiente apunta a un elemento y tras detectar que este nodo apunta a Nulo incluye otro. Además se debe recalcar que hay un ciclo en el que se recorren todos los elementos

****

**Y luego se agrega el próximo elemento repitiendo el proceso una y otra vez.**

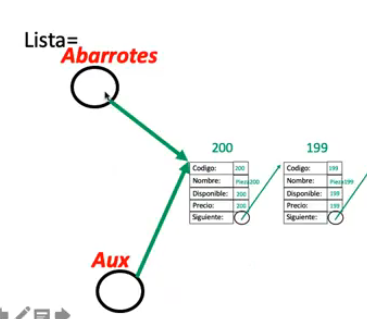


**Lista: Destruir**

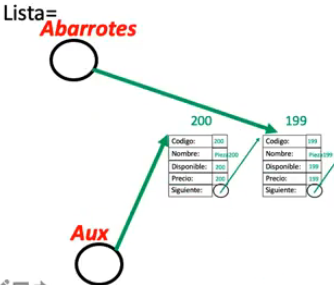


En este código se usa como parámetro la lista de abarrotes y se pasa como parámetro por referencia ya que va a ser modificado. En la función secundaria Aux corresponde a la dirección de la lista y luego empieza el ciclo que se puede resumir en, mientras que Aux apunte a algo, esto se va a eliminar y además el valor de aux va cambiando con un intercambio de valores entre Aux y Lista. Representado de forma gráfica es así:

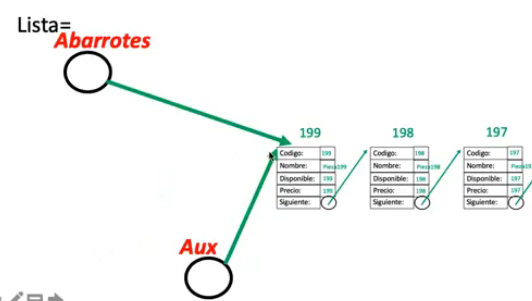
1. Primero aux y lista apunta a la misma direccion ya que Aux es Lista.



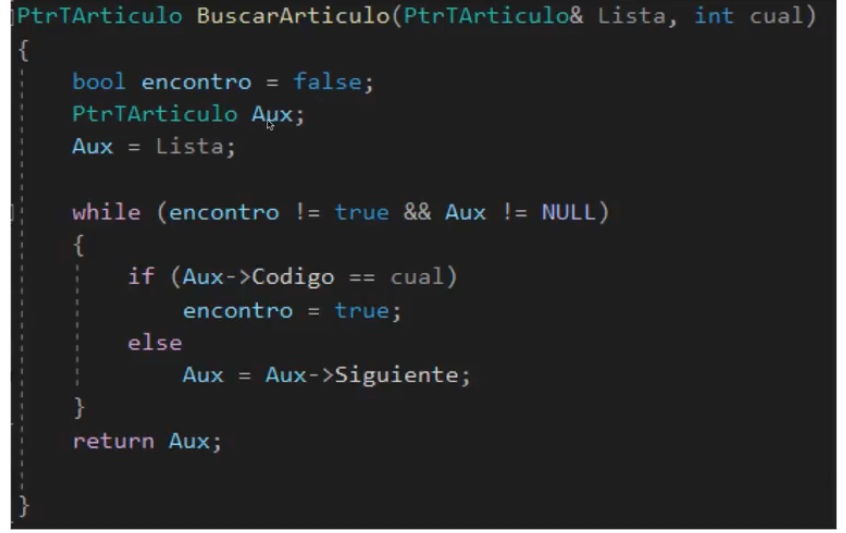
2. Ya que Aux tiene la dirección Lista, esta ultima puede ser cambiada para apuntar al siguiente nodo.



3. Finalmente el elemento que apuntaba Aux se borra y esta variable pasa a apuntar al siguiente nodo para repetir el proceso.



**Lista: Buscar**



En la primera parte de este código se definen las variables, un valor bool para ver si se encontró o no y el auxiliar para recorrer la lista que corresponde al valor contenido en la variable Lista.

Posteriormente se da el ciclo con dos condiciones, primero encontró debe ser false y luego Aux debe ser diferente de Nulo ósea que no se salga del rango.

Si se encuentra un valor igual al que se busca entonces el ciclo termina ya que encontro ahora es True y sino el ciclo continuo. Los resultados son la dirección del auxiliar.