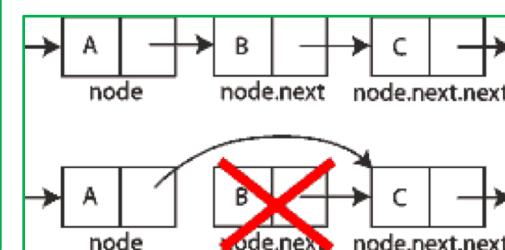
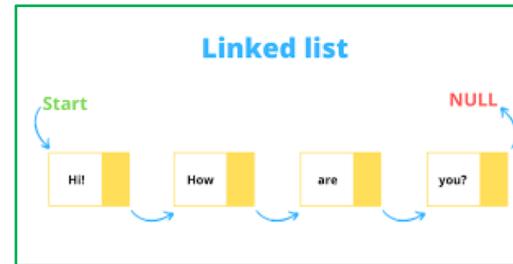
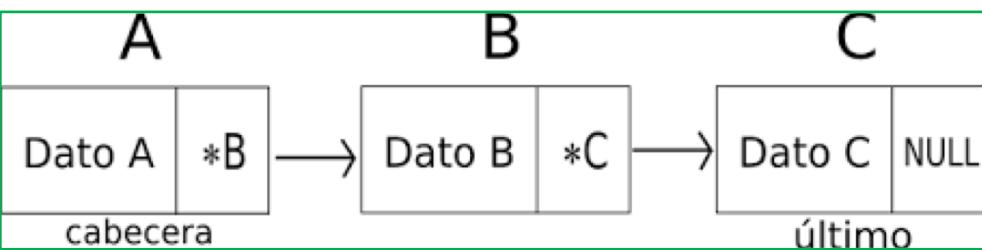
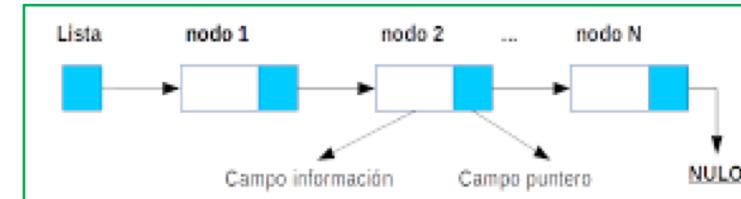
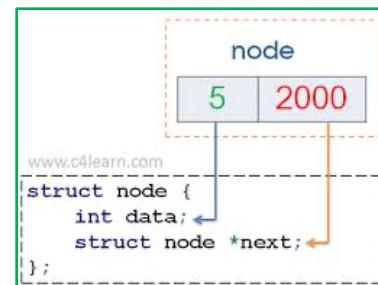
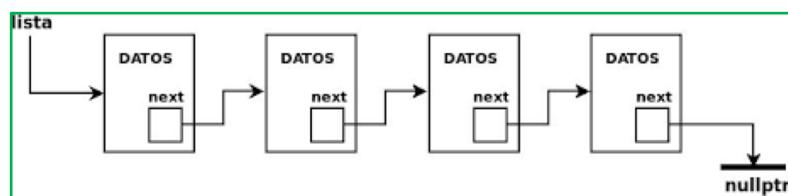
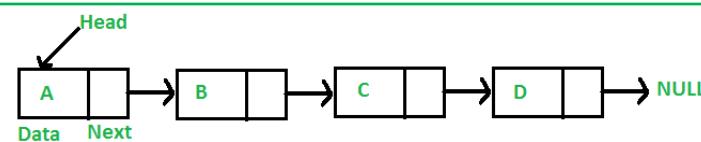
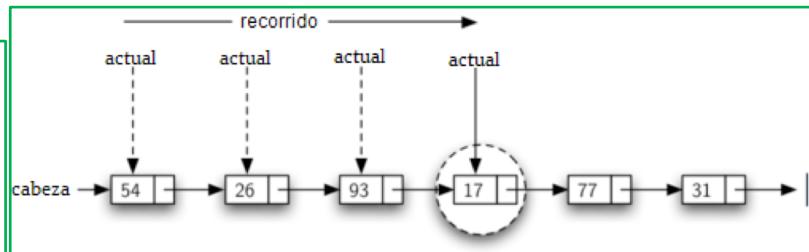
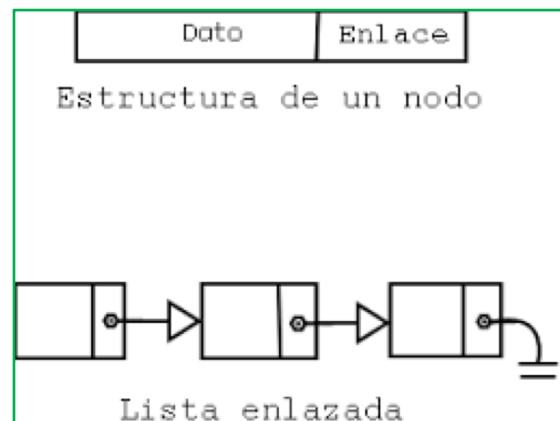


Estructura de Datos Dinámica Lineal

Lista Enlazada Simple



Agenda

- Definición de Lista EnlazadaSimple
- Declaración del nuevo tipo **TArticulo** y **PtrTArticulo**
- Declaración de variables estáticas tipo Puntero **PtrTArticulo**
- Inicializar una lista : **Abarrotes**
- Crear Artículo Individual
- Piloto: Crear y AgregarInicio desde i=100 hasta i=200 (General)
- Piloto: Crear y AgregarInicio desde i=100 hasta i=200 (Pasos)
- AgregarInicio (algoritmo)
- Piloto: Crear y AgregarInicio desde i=100 hasta i=200 (Pasos detallados)
- Lista Enlazada (Listar)
- Listar y AgregarFinal (comparación)
- Piloto: Crear y AgregarFinal desde i=200 hasta i=300 (General)
- AgregarFinal (algoritmo)
- Piloto: Crear y AgregarFinal desde i=200 hasta i=300 (Detallado1)
- Piloto: Crear y AgregarFinal desde i=200 hasta i=300 (Detallado2)
- Lista : Destruir
- Lista : Buscar

Main

```
void main(int argc, char* argv[])
{
    //*****
    PtrTArticulo Llantas;
    PtrTArticulo Abarrotes;
    PtrTArticulo Nuevo;
    PtrTArticulo Prueba;

    //*****
```



```
//*****
IniciarInventario(Abarrotes);

for (int i = 100; i <= 200; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarInicioInventario(Abarrotes, Nuevo);
}
GuardarInventario(Abarrotes);
ListarInventario(Abarrotes);
DestruirInventario(Abarrotes);
system("cls");
//*****PRUEBA DE CARGAR*****
IniciarInventario(Prueba);
CargarInventario(Prueba);
ListarInventario(Prueba);
system("pause");
DestruirInventario(Prueba);
```



```
//*****
IniciarInventario(Abarrotes);
for (i = 200; i <= 300; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarFinalInventario(Abarrotes, Nuevo);
}
ListarInventario(Abarrotes);
DestruirInventario(Abarrotes);
```

Lista Enlazada Simple

Lista
Abarrotes

??

??

Lista
Abarrotes

Null

Lista
Abarrotes

LISTA ENLAZADA : Estructura de datos lineal dinámica en la que solo se conocen :

- A. EL INICIO
- B. EL FINAL

A. EL INICIO: Puntero Lista

B. EL FINAL: Puntero->Siguiente==NULL

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○

Código:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	○



Declaración del nuevo tipo **TArticulo** y **PtrTArticulo**

Nuevo tipo de Dato **TArticulo**

```
typedef struct TArticulo
{
    int Codigo;
    char Nombre[20];
    int Disponible;
    float Precio;
    TArticulo* Siguiente;
}*PtrTArticulo;
```

Nuevo tipo de Dato **PtrTArticulo**

PtrTArticulo Auxiliar;
o
TArticulo * Auxiliar;

Auxiliar

Auxiliar = new(TArticulo);

Variables Anónimas de tipo **TArticulo**

new(TArticulo)

Codigo:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	

new(TArticulo)

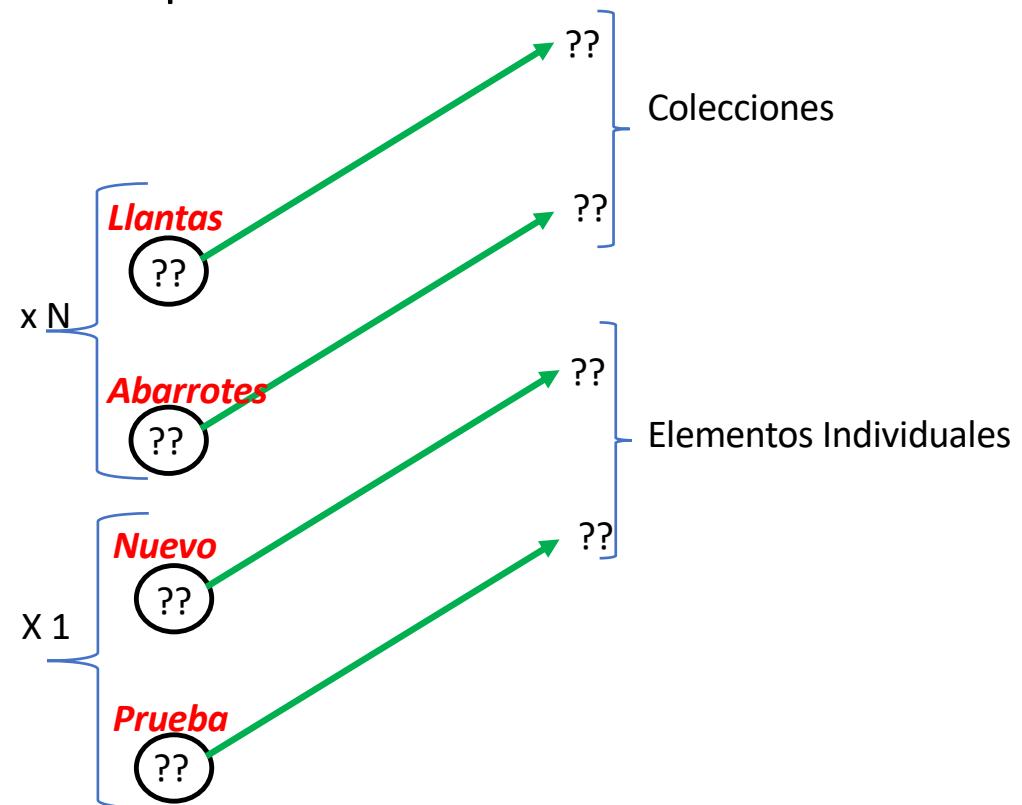
Codigo:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	??

new(TArticulo)

Codigo:	
Nombre:	
Disponible:	
Precio:	
Siguiente:	??

Declaración de variables estáticas tipo Puntero **PtrTArticulo**

```
void main(int argc, char* argv[])
{
    //*****
    PtrTArticulo Llantas;
    PtrTArticulo Abarrotes;
    PtrTArticulo Nuevo;
    PtrTArticulo Prueba;
}
```



Inicializar una lista : Abarrotes

```
...
PtrTArticulo Llantas;
PtrTArticulo Abarrotes;
PtrTArticulo Nuevo;
PtrTArticulo Prueba;

/*****************/
IniciarInventario(Abarrotes);
```

```
void IniciarInventario(PtrTArticulo& Lista)
{
    Lista = NULL;
```

Parámetro por referencia *Lista* es un
ALIAS de un puntero PtrTArticulo

Abarrotes = *Lista*



Lista = NULL);

Crear Artículo Individual

```

for (int i = 100; i <= 200; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarInicioInventario(Abarrotes, Nuevo);
}

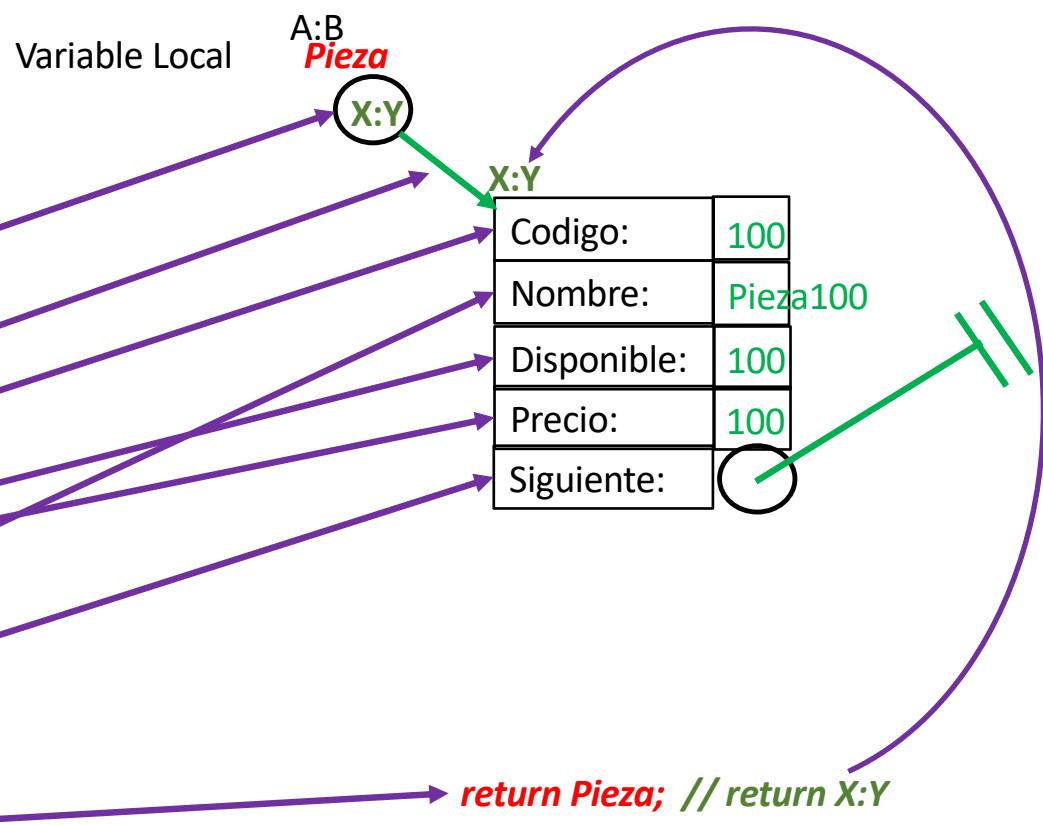
PtrTArticulo CrearArticulo(int NCodigo, int NDisponible, float NPrecio)
{
    PtrTArticulo Pieza = new(TArticulo);
    char buffer[5];

    Pieza->Codigo = NCodigo;
    Pieza->Disponible = NDisponible;
    Pieza->Precio = NPrecio;

    strcpy_s(Pieza->Nombre, "Pieza");
    _itoa_s(NCodigo, buffer, 10);
    strcat_s(Pieza->Nombre, buffer);

    Pieza->Siguiente = NULL;
    return Pieza;
}

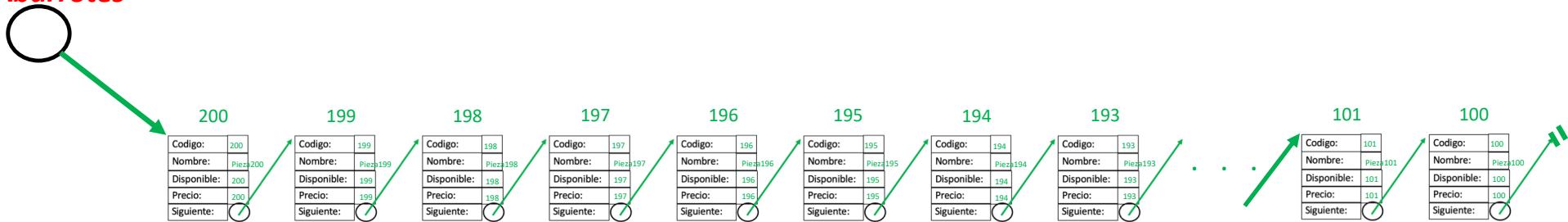
```



Piloto: Crear y AgregarInicio desde i=100 hasta i=200(General)

```
for (int i = 100; i <= 200; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarInicioInventario(Abarrotes, Nuevo);
}
```

Lista=
Abarrotes

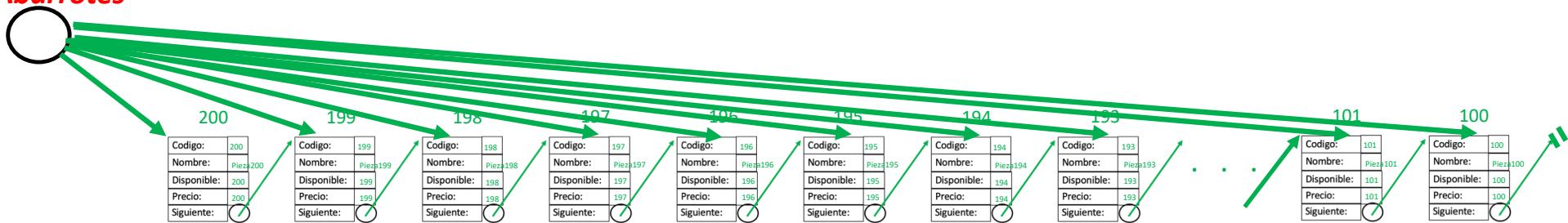


Piloto: Crear y AgregarInicio desde i=100 hasta i=200 (Pasos)

```
for (int i = 100; i <= 200; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarInicioInventario(Abarrotes, Nuevo);
}
```

Inserta al inicio/Inserción invertida : Invierte el orden. Coloca el 1er elemento de último, el segundo de penúltimo..... y el N-ésimo de primero.

Lista=
Abarrotes



Aunque lista desde el inicio a N, N-1, N-2, N-3... hasta llegar a el 1ero.

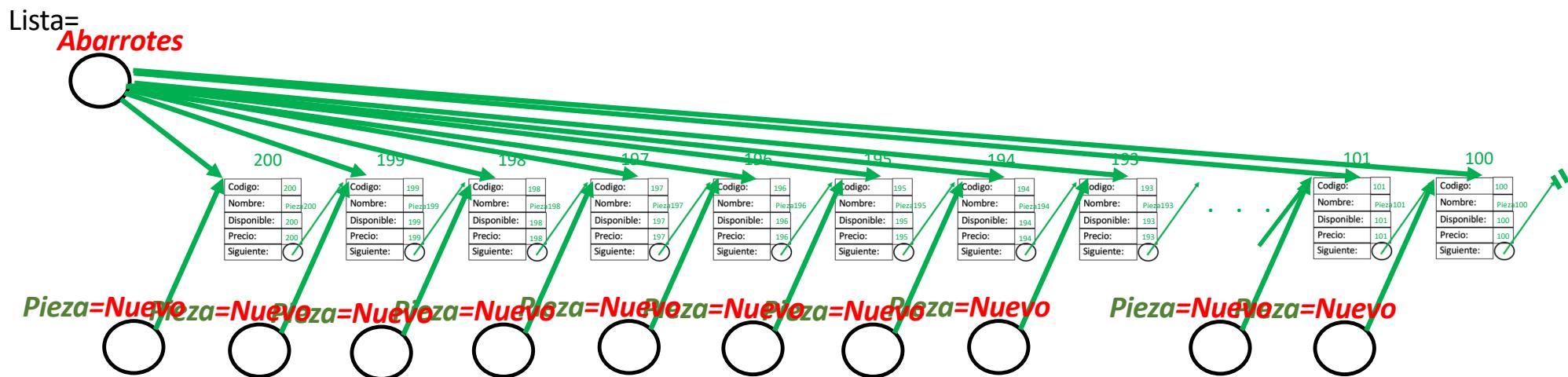
AgregarInicio (algoritmo)

```
for (int i = 100; i <= 200; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarInicioInventario(Abarrotes, Nuevo);
}
```

```
void AgregarInicioInventario(PtrTArticulo& Lista, PtrTArticulo& Nuevo)
{
    Nuevo->Siguiente = Lista;
    Lista = Nuevo;
}
```

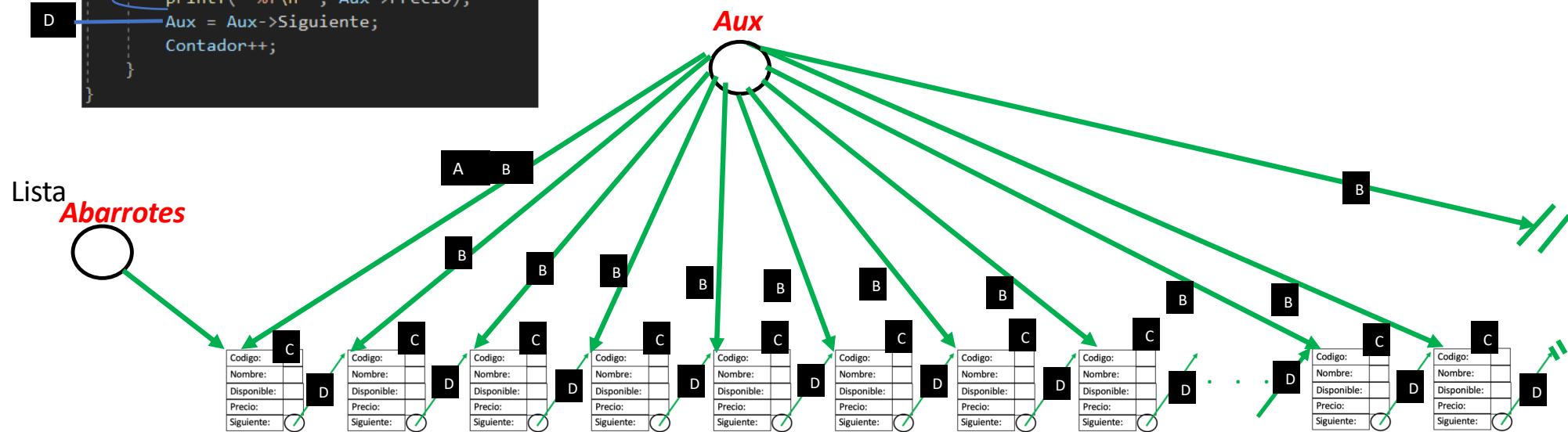
Piloto: Crear y AgregarInicio desde i=100 hasta i=200 (Pasos detallados)

```
for (int i = 100; i <= 200; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarInicioInventario(Abarrotes, Nuevo);
}
```



Lista Enlazada (Listar)

```
ListarInventario(Abarrotes);  
  
void ListarInventario(PtrTArticulo& Lista)  
{  
    int Contador = 1;  
    PtrTArticulo Aux;  
    Aux = Lista;  
    while (Aux != NULL)  
    {  
        printf(" %d ", Contador);  
        printf("%d ", Aux->Codigo);  
        printf("%s ", Aux->Nombre);  
        printf(" %d ", Aux->Disponible);  
        printf(" %f\n ", Aux->Precio);  
        Aux = Aux->Siguiente;  
        Contador++;  
    }  
}
```



Listar y AgregarFinal (comparación)

```
void ListarInventario(PtrTArticulo& Lista)
{
    int Contador = 1;
    PtrTArticulo Aux;
    Aux = Lista;
    while (Aux != NULL)
    {
        Pregunta si se salió de la lista o si está vacía?
        Aux = Aux->Siguiente;
        Contador++;
    }
}
```

```
void AgregarFinalInventario(PtrTArticulo& Lista, PtrTArticulo& Nuevo)
{
    PtrTArticulo Aux;
    Aux = Lista;
    if (Aux != NULL) Pregunta si lista no es vacía ?
    {
        while (Aux->Siguiente != NULL) Pregunta si está en el último nodo?
        {
            Aux = Aux->Siguiente;
        }
        Aux->Siguiente = Nuevo; Aquí agrega Nuevo al final final !!
    }
    else
    {
        Lista = Nuevo; Aquí agrega Nuevo al “final” que es el inicio pues la lista esta vacía
    }
}
```

Piloto: Crear y AgregarFinal desde i=200 hasta i=300 (General)

Lista

Abarrotes



??

Lista

Abarrotes



??

Lista=

Abarrotes



200

Código:	200
Nombre:	Pieza200
Disponible:	200
Precio:	200
Siguiente:	

201

Código:	201
Nombre:	Pieza201
Disponible:	201
Precio:	201
Siguiente:	

202

Código:	202
Nombre:	Pieza202
Disponible:	202
Precio:	202
Siguiente:	

203

Código:	203
Nombre:	Pieza203
Disponible:	203
Precio:	203
Siguiente:	

204

Código:	204
Nombre:	Pieza204
Disponible:	204
Precio:	204
Siguiente:	

205

Código:	205
Nombre:	Pieza205
Disponible:	205
Precio:	205
Siguiente:	

206

Código:	206
Nombre:	Pieza206
Disponible:	206
Precio:	206
Siguiente:	

207

Código:	207
Nombre:	Pieza207
Disponible:	207
Precio:	207
Siguiente:	

...

...

...

299

Código:	299
Nombre:	Pieza299
Disponible:	299
Precio:	299
Siguiente:	

300

Código:	300
Nombre:	Pieza300
Disponible:	300
Precio:	300
Siguiente:	

Si se lista desde el inicio imprime el 1ero, 2do, 3ro, 4to... hasta el N-1 y N.

```
InicializarInventario(Abarrotes);
for (i = 200; i <= 300; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarFinalInventario(Abarrotes, Nuevo);
```

Inserta al Final/Inserción ordenada por orden de llegada : Coloca el nuevo elemento de último cada vez... Para un elemento N .. Deberá recorrer primero los N-1 elementos.

AgregarFinal (algoritmo)

```
InicializarInventario(Abarrotes);
for (i = 200; i <= 300; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarFinalInventario(Abarrotes, Nuevo);
}
```

```
void AgregarFinalInventario(PtrTArticulo& Lista, PtrTArticulo& Nuevo)
{
    PtrTArticulo Aux;
    Aux = Lista;
    if (Aux != NULL)
    {
        while (Aux->Siguiente != NULL)
        {
            Aux = Aux->Siguiente;
        }

        Aux->Siguiente = Nuevo;
    }
    else
    {
        Lista = Nuevo;
    }
}
```

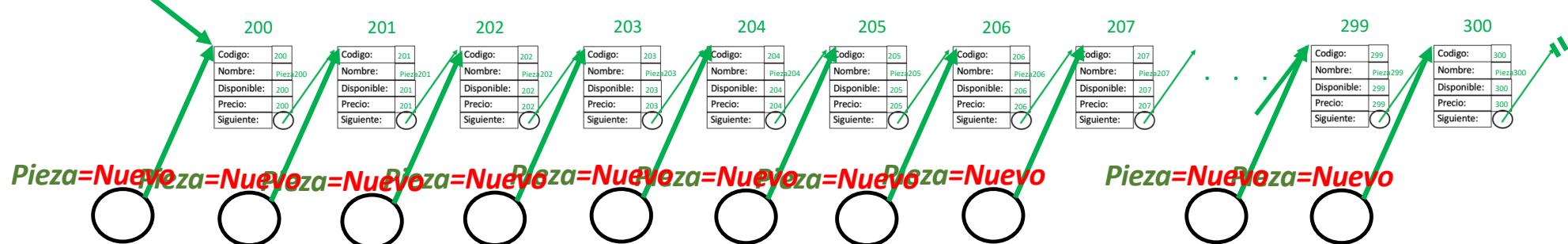
Piloto: Crear y AgregarFinal desde i=200 hasta i=300 (Detallado1)

```

InicializarInventario(Abarrotes);
for (i = 200; i <= 300; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarFinalInventario(Abarrotes, Nuevo);
}

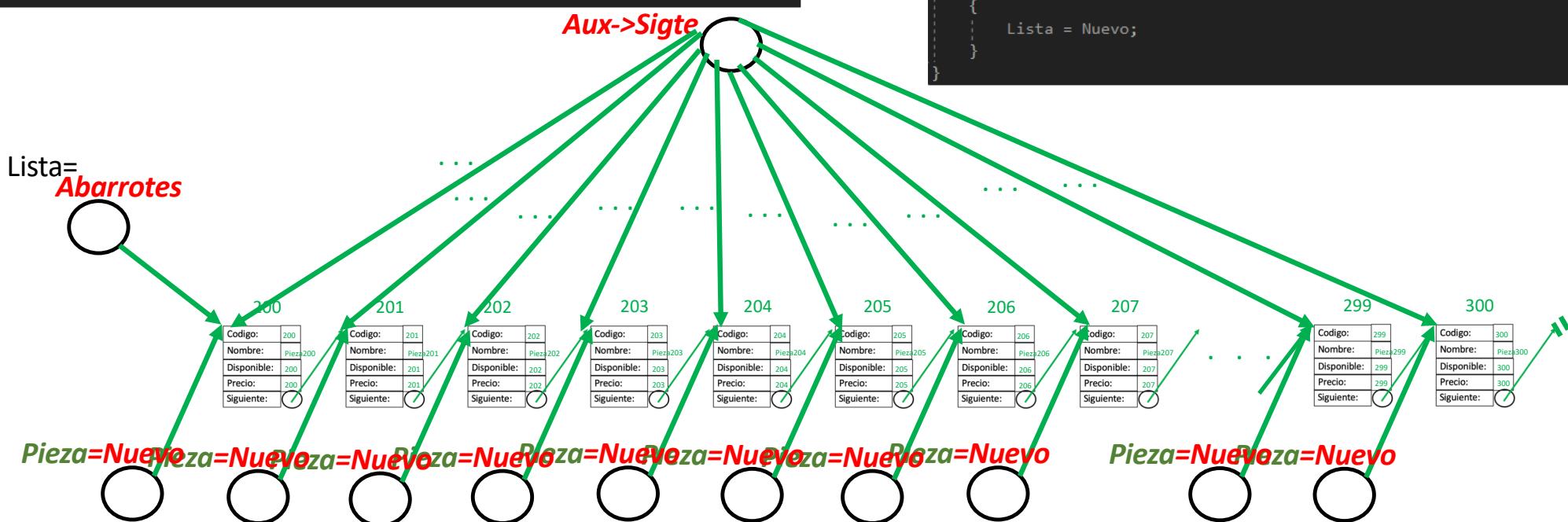
```

Lista=
Abarrotes



Piloto: Crear y AgregarFinal desde i=200 hasta i=300 (Detallado2)

```
InicializarInventario(Abarrotes);
for (i = 200; i <= 300; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarFinalInventario(Abarrote
}
```



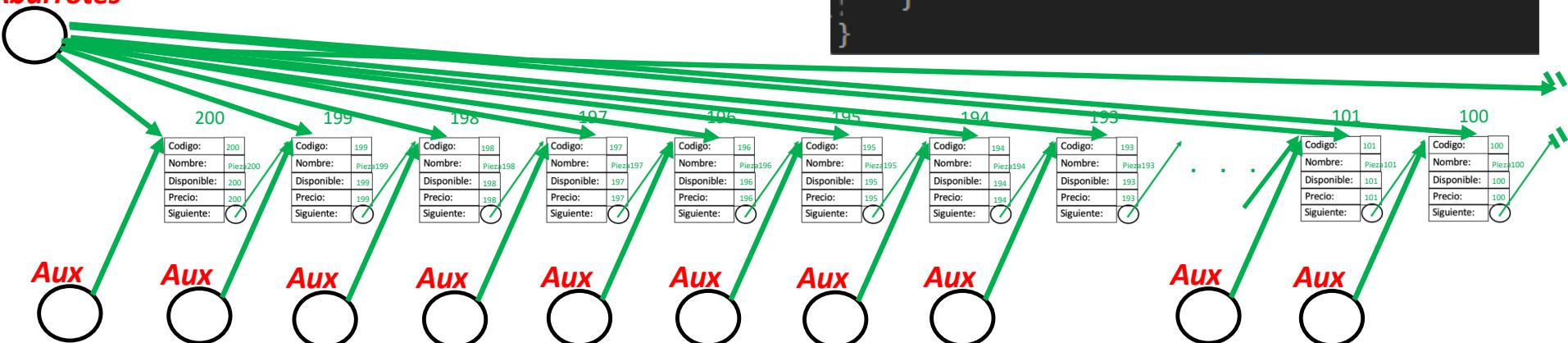
TEC | Tecnológico de Costa Rica

Lista : Destruir

```
InicializarInventario(Abarrotes);
for (i = 200; i <= 300; i++)
{
    Nuevo = CrearArticulo(i, i, i);
    AgregarFinalInventario(Abarrotes, Nuevo);
}
ListarInventario(Abarrotes);
DestruirInventario(Abarrotes);
```

```
void DestruirInventario(PtrTArticulo& Lista)
{
    PtrTArticulo Aux;
    Aux = Lista;
    while (Aux != NULL)
    {
        Lista = Lista->Siguiente;
        delete(Aux);
        Aux = Lista;
    }
}
```

Lista=
Abarrotes



Lista : Buscar

```
PtrTArticulo BuscarArticulo(PtrTArticulo& Lista, int cual)
{
    bool encontro = false;
    PtrTArticulo Aux;
    Aux = Lista;

    while (encontro != true && Aux != NULL)
    {
        if (Aux->Codigo == cual)
            encontro = true;
        else
            Aux = Aux->Siguiente;
    }
    return Aux;
}
```