

Documento Código Listas Enlazadas

1. Estructura General

El código está organizado en una jerarquía de clases:

Lista Enlazada: Clase base para listas simples

Lista Dblemente Enlazada: Hereda y añade referencia "anterior"

Lista Circular: Hereda y hace el último nodo apunta al primero

Lista Circular Dblemente Enlazada: Combina ambas características

2. Representación de Nodos

Cada nodo es un diccionario con:

Para listas simples: {'valor': X, 'siguiente': Y}

Para listas dobles: {'valor': X, 'anterior': W, 'siguiente': Y}

Donde:

X, Y, W son valores entre 1-98 (datos) o 0/99 (nulos)

Los valores 0 y 99 actúan como punteros nulos

3. Funcionalidades Implementadas

Funciones Obligatorias:

crear_nodo(): Crea la estructura del nodo según el tipo

insertar_inicio(): Añade nodo al principio

insertar_final(): Añade nodo al final

insertar_nodo(): Inserta entre dos nodos existentes

contar_nodos(): Devuelve el número de nodos

eliminar_nodo(): Elimina un nodo específico

`buscar_nodo()`: Busca un nodo por valor

`imprimir_valor_lista()`: Muestra sólo los valores

`imprimir_lista_completa()`: Muestra todos los campos

`imprimir_reves()`: Muestra la lista en orden inverso

`copiar_lista()`: Guarda la lista en un archivo

Función Extra:

`ordenar_lista()`: Ordena los nodos de menor a mayor

4. Manejo de Casos Especiales

Lista vacía: Se verifica en cada operación

Valores inválidos: Se validan rangos (1-98)

Referencias nulas: Se usan 0 y 99 para NULL

Ciclos infinitos: En listas circulares se controlan con conjuntos

5. Interfaz de Usuario

El programa tiene un menú interactivo con:

Menú principal para seleccionar tipo de lista

Submenú para operaciones específicas

Validación de entrada de usuario

6. Archivos Generados

El programa crea archivos con nombres específicos:

`lista_enlazada.txt`

`lista_enlazada_d.txt`

`lista_circular.txt`

lista_circular_d.txt

Cada archivo contiene una representación legible de la lista.

7. Ventajas de esta Implementación

Modularidad: Cada tipo de lista es una clase separada

Reutilización: Herencia para evitar código duplicado

Legibilidad: Código bien comentado y estructurado

Robustez: Manejo de errores y validaciones

Extensibilidad: Fácil añadir nuevas funcionalidades