**PIA**

**Producto Integrador de Aprendizaje de la materia de Estructura de Datos**

Estructura De Datos

**Mónica Ivett Salinas Rodriguez**

**Grupo: 032**

**Edson Luis Rocha Rodríguez**

**1863555**

**Aldo Martinez**

**Roberto Rodríguez**

**Mónica de León**

## **Introducción**

Las estructuras de datos son una parte fundamental de la programación y la informática. Se refieren a la organización lógica y eficiente de los datos en una computadora, diseñada para permitir el acceso y la manipulación de estos de manera eficiente. Algunas de las estructuras de datos más comunes son las listas encadenadas, las listas dobles y circulares, las pilas, la recursividad, los árboles y los grafos.

Las listas encadenadas son una estructura de datos que permiten almacenar y acceder a una colección de elementos en orden. Cada elemento en una lista encadenada se compone de dos partes: un valor y un puntero que apunta al siguiente elemento en la lista. A diferencia de las listas estáticas, las listas encadenadas no necesitan una cantidad de memoria fija, lo que las hace más flexibles y dinámicas. Las listas dobles y circulares son variaciones de las listas encadenadas que ofrecen diferentes ventajas dependiendo del problema que se esté abordando.

Las pilas son una estructura de datos que se utilizan para almacenar y recuperar datos de forma ordenada y según el principio de LIFO (Last In, First Out), es decir, el último elemento en ser ingresado es el primero en ser eliminado. Las operaciones con pilas incluyen la inserción de elementos (push), la eliminación de elementos (pop) y la revisión del elemento superior (peek). Las pilas se utilizan en una amplia variedad de aplicaciones, como la evaluación de expresiones aritméticas y la implementación de algoritmos de búsqueda y clasificación.

La recursividad es una técnica que se utiliza en programación para resolver problemas mediante la repetición de un proceso en sí mismo. Es decir, una función se llama a sí misma dentro de su propia definición, hasta alcanzar un caso base que detenga la recursión. La recursividad se utiliza en muchos problemas, como la implementación de algoritmos de búsqueda y clasificación, y la generación de secuencias y patrones complejos.

Los árboles son una estructura de datos que se utilizan para almacenar y acceder a datos jerárquicos. Cada elemento en un árbol se llama nodo y tiene un padre y uno o varios hijos. Los árboles se utilizan en muchos problemas de búsqueda, como en la organización de directorios y archivos en un sistema operativo, y en la representación de relaciones jerárquicas entre elementos.

Los grafos son una estructura de datos que se utilizan para representar relaciones entre elementos. Un grafo consta de nodos (vértices) y conexiones (aristas) que los unen. Los grafos se utilizan en problemas como la optimización de rutas, la representación de redes de comunicación y la detección de patrones en datos complejos.

En conclusión, las estructuras de datos, las listas encadenadas, las listas dobles y circulares, las pilas y sus operaciones, la recursividad, los árboles y los grafos son herramientas fundamentales para la programación y la informática. Su dominio y comprensión permiten el desarrollo de algoritmos y soluciones eficientes a diversos problemas, y son una parte esencial de la formación de cualquier programador o ingeniero de sistemas.

## **Estructuras de datos**

Las estructuras de datos son formas de organizar y almacenar datos de manera eficiente y efectiva en una computadora. Las estructuras de datos se utilizan en muchos campos de la informática, incluyendo la programación, la ingeniería de software, la inteligencia artificial, la seguridad de la información, la gestión de bases de datos, la visualización de datos y muchos otros.

Las estructuras de datos se pueden clasificar en dos categorías principales: estructuras de datos lineales y estructuras de datos no lineales. Las estructuras de datos lineales son aquellas que organizan los datos en una secuencia o lista, como arrays, listas enlazadas, pilas y colas. Las estructuras de datos no lineales organizan los datos en una estructura jerárquica, como árboles y grafos.

Cada estructura de datos tiene sus propias características y se utiliza para una variedad de propósitos. A continuación, se describen algunas de las estructuras de datos más comunes:

1. Arrays: Un array es una estructura de datos lineal que contiene una colección de elementos del mismo tipo de datos. Cada elemento en un array se identifica por su índice, que es un número entero que indica su posición en el array. Los arrays son útiles para almacenar y acceder a grandes cantidades de datos de manera eficiente.

2. Listas enlazadas: Una lista enlazada es una estructura de datos lineal en la que cada elemento contiene un valor y un puntero que apunta al siguiente elemento de la lista. Las listas enlazadas son útiles cuando se necesita agregar o eliminar elementos con frecuencia, ya que pueden hacerlo de manera eficiente sin tener que mover todos los elementos en la lista.

3. Pilas: Una pila es una estructura de datos lineal que utiliza el principio LIFO (Last In, First Out), lo que significa que el último elemento agregado a la pila es el primero en ser eliminado. Las pilas son útiles para llevar a cabo operaciones en orden inverso al que se recibieron.

4. Colas: Una cola es una estructura de datos lineal que utiliza el principio FIFO (First In, First Out), lo que significa que el primer elemento agregado a la cola es el primero en ser eliminado. Las colas son útiles para mantener un orden específico de los elementos.

5. Árboles: Un árbol es una estructura de datos no lineal que consiste en una colección de nodos interconectados. Cada nodo tiene un valor y cero o más nodos hijos que se conectan a él. Los árboles son útiles para representar jerarquías, como la estructura de un directorio de archivos en una computadora.

6. Grafos: Un grafo es una estructura de datos no lineal que consta de una colección de nodos (también llamados vértices) y un conjunto de aristas que conectan los nodos. Los grafos son útiles para modelar relaciones entre objetos, como redes sociales o mapas de carreteras.

Cada estructura de datos tiene sus propias ventajas y desventajas, y es importante elegir la estructura de datos correcta para la tarea específica que se está realizando. La elección correcta de la estructura de datos puede hacer que un programa sea

## **Listas encadenadas**

Una lista enlazada es una estructura de datos lineal utilizada en programación para almacenar una colección de elementos de manera ordenada. A diferencia de los arrays, las listas enlazadas no tienen una ubicación fija en la memoria. En cambio, cada elemento en la lista enlazada se almacena en un nodo que contiene una referencia al siguiente elemento en la lista.

Cada nodo de una lista enlazada consta de dos partes: el valor del elemento y una referencia al siguiente nodo en la lista. La referencia al siguiente nodo se llama puntero o enlace. El primer nodo de la lista se llama cabeza y el último nodo se llama cola. La lista enlazada puede estar vacía, en cuyo caso la cabeza y la cola son nulas.

La forma en que se conectan los nodos es lo que define la estructura de la lista enlazada. En una lista enlazada simple, cada nodo tiene una referencia al siguiente nodo en la lista. En una lista enlazada doble, cada nodo tiene una referencia tanto al siguiente como al nodo anterior en la lista. También existen listas enlazadas circulares, donde el último nodo de la lista hace referencia al primer nodo, cerrando así la lista.

Las listas enlazadas son útiles para almacenar elementos de forma dinámica, lo que significa que la cantidad de elementos puede variar a lo largo del tiempo. Esto es posible porque los nodos se pueden agregar o eliminar de la lista en cualquier momento, sin tener que mover otros elementos en la lista. Esto hace que las listas enlazadas sean eficientes en términos de espacio y tiempo, ya que solo se necesita agregar o eliminar el nodo relevante y actualizar las referencias.

Las listas enlazadas se utilizan en una variedad de aplicaciones, como en la implementación de pilas y colas, en la gestión de memoria dinámica en sistemas operativos, en la representación de estructuras de datos jerárquicas como árboles y en muchas otras áreas de la programación.

Sin embargo, una desventaja de las listas enlazadas es que el acceso a los elementos de la lista no es tan eficiente como en los arrays, ya que no se pueden acceder directamente a los elementos por su índice. En su lugar, se debe recorrer la lista a través de las referencias de los nodos, lo que puede ser más lento en el peor de los casos.

En resumen, las listas enlazadas son una estructura de datos dinámica y flexible que permite almacenar y organizar una colección de elementos de manera ordenada. Aunque no son tan eficientes en términos de acceso a los elementos como los arrays, son ideales para aplicaciones en las que se necesite agregar o eliminar elementos con frecuencia.

***Operaciones***

Algunas de las operaciones que se pueden usar son muy básicas, tanto en este tipo de estructura como en cualquier otra, las operaciones son insertar, eliminar, buscar, localizar y vaciar la lista, a continuación, daré la explicación de cada una y un ejemplo de cada una de estas operaciones.

**Insertar elementos:**

Inserta un nodo con dato x en la lista, puede realizarse esta inserción al principio o al final de la lista o bien puede ser en el orden

Captura de pantalla de computadora

Descripción generada automáticamente Captura de pantalla de computadora

Descripción generada automáticamente

**Eliminar elementos:**

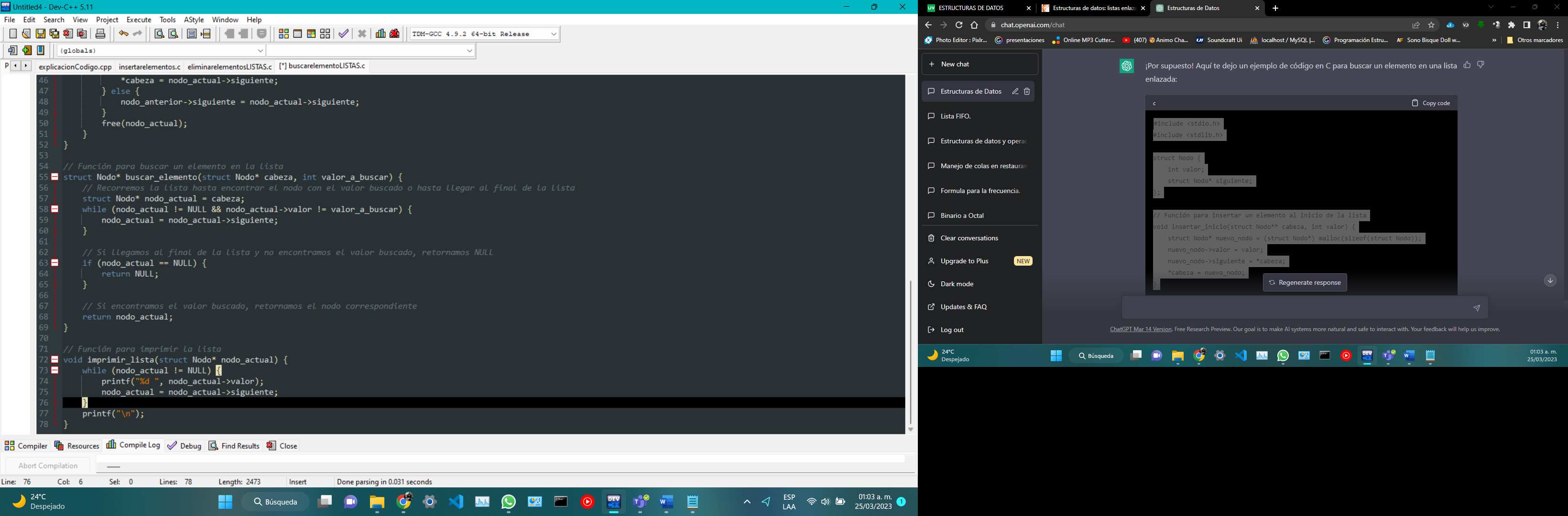
Elimina un nodo de la lista, puede ser según la posición o por el dato.

Captura de pantalla de computadora

Descripción generada automáticamente

**Buscar elementos:**

Busca un elemento cualquiera en la lista



**Vaciar la lista:**

Borra todos los elementos de la lista

Captura de pantalla de computadora

Descripción generada automáticamente

***Listas dobles***

Una lista doblemente enlazada es una lista lineal en la que cada nodo tiene dos enlaces, uno al nodo siguiente, y otro al anterior.

Las listas doblemente enlazadas no necesitan un nodo especial para acceder a ellas, pueden recorrerse en ambos sentidos a partir de cualquier nodo, esto es porque a partir de cualquier nodo, siempre es posible alcanzar cualquier nodo de la lista, hasta que se llega a uno de los extremos.

El nodo típico es el mismo que para construir las listas que hemos visto, salvo que tienen otro puntero al nodo anterior:

Captura de pantalla de computadora

Descripción generada automáticamente

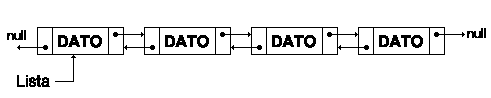
***Declaraciones de tipos para manejas listas dobles:***

Captura de pantalla de computadora

Descripción generada automáticamente

tipoNodo es el tipo para declarar nodos, evidentemente.

pNodo es el tipo para declarar punteros a un nodo.



Lista doblemente enlazada

Lista es el tipo para declarar listas abiertas doblemente enlazadas. También es posible, y potencialmente útil, crear listas doblemente enlazadas y circulares.

El movimiento a través de listas doblemente enlazadas es más sencillo, y como veremos las operaciones de búsqueda, inserción y borrado, también tienen más ventajas.

***Operaciones básicas***

De nuevo tenemos el mismo repertorio de operaciones sobre este tipo listas:

* Añadir o insertar elementos.
* Buscar o localizar elementos.
* Borrar elementos.
* Moverse a través de la lista, siguiente y anterior.

***Listas circulares***

Una lista circular es una lista lineal en la que el último nodo a punta al primero.

Las listas circulares evitan excepciones en las operaciones que se realicen sobre ellas. No existen casos especiales, cada nodo siempre tiene uno anterior y uno siguiente.

En algunas listas circulares se añade un nodo especial de cabecera, de ese modo se evita la única excepción posible, la de que la lista esté vacía.

El nodo típico es el mismo que para construir listas abiertas:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

***Declaraciones de tipos para manejas listas circulares:***

Los tipos que definiremos normalmente para manejar listas cerradas son los mismos que para para manejar listas abiertas:

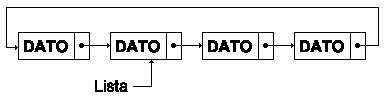
Captura de pantalla de computadora

Descripción generada automáticamente

tipoNodo es el tipo para declarar nodos, evidentemente.

pNodo es el tipo para declarar punteros a un nodo.

Lista es el tipo para declarar listas, tanto abiertas como circulares. En el caso de las circulares, apuntará a un nodo cualquiera de la lista.



Lista circular

A pesar de que las listas circulares simplifiquen las operaciones sobre ellas, también introducen algunas complicaciones. Por ejemplo, en un proceso de búsqueda, no es tan sencillo dar por terminada la búsqueda cuando el elemento buscado no existe.

Por ese motivo se suele resaltar un nodo en particular, que no tiene por qué ser siempre el mismo. Cualquier nodo puede cumplir ese propósito, y puede variar durante la ejecución del programa.

Otra alternativa que se usa a menudo, y que simplifica en cierto modo el uso de listas circulares es crear un nodo especial de hará la función de nodo cabecera.

De este modo, la lista nunca estará vacía, y se eliminan casi todos los casos especiales.

***Operaciones básicas***

A todos los efectos, las listas circulares son como las listas abiertas en cuanto a las operaciones que se pueden realizar sobre ellas:

* Añadir o insertar elementos.
* Buscar o localizar elementos.
* Borrar elementos.
* Moverse a través de la lista, siguiente.

Cada una de estas operaciones podrá tener varios casos especiales, por ejemplo, deberemos tener en cuenta cuando se inserte un nodo en una lista vacía, o cuando se elimina el único nodo de una lista.

## **Pilas**

una pila (también conocida como "stack" en inglés) es una estructura de datos que permite almacenar y recuperar elementos siguiendo una política LIFO (Last-In-First-Out), lo que significa que el último elemento que se insertó en la pila es el primero que se elimina.

La estructura de la pila se puede imaginar como una pila de platos, donde se pueden añadir nuevos platos en la parte superior y retirarlos desde esa misma posición. En una pila, solo se pueden acceder a los elementos que están en la cima de la pila, y para acceder a un elemento que está en una posición inferior, es necesario retirar los elementos superiores.

Las pilas se utilizan comúnmente en muchos lenguajes de programación para realizar tareas como la gestión de llamadas a funciones, el procesamiento de expresiones matemáticas, la reversión de cadenas y la implementación de algoritmos de búsqueda y ordenamiento

***Operaciones***

Las operaciones que se pueden realizar con las pilas son:

Insertar (push)

En primer lugar, hay que decir que esta operación es muy comúnmente denominada push.

La inserción en una pila se realiza en su cima, considerando la cima como el último elemento insertado. Esta es una de las particularidades de las pilas, mientras el resto de las estructuras de datos lineales se representan gráficamente en horizontal, las pilas se representan verticalmente. Por esta razón es por lo que se habla de cima de la pila y no de cola de la cima. Aunque en el fondo sea lo mismo, el último elemento de la estructura de datos.

Las operaciones para realizar para realizar la inserción en la pila son muy simples, hacer que el nuevo nodo apunte a la cima anterior, y definir el nuevo nodo como cima de la pila

|  |
| --- |
| 3 |
| 76 |
| -6 |

Vamos a ver un ejemplo de una inserción, primero de una manera visual con números aleatorios y luego un código:

cima

Captura de pantalla de computadora

Descripción generada automáticamente

Borrar (pop)

Esta operación es normalmente conocida como pop.

Cuando se elimina un elemento de la pila, el elemento que se borra es el elemento situado en la cima de la pila, el que menos tiempo lleva en la estructura.

Las operaciones para realizar son muy simples, avanzar el puntero que apunta a la cima y extraer la cima anterior.

Si aplicamos la operación pop a la pila de 4 elementos representada arriba el resultado sería:

cima

|  |
| --- |
| 0 |
| 3 |
| 76 |
| -6 |

Captura de pantalla de computadora

Descripción generada automáticamente

## **Operaciones con pilas**

***Aplicaciones (notación infija-posfija)***

**Infija:**

La expresión o notación infija es la forma más común que utilizamos para escribir expresiones matemáticas, estas notaciones se refieran a que el operados esta entre los operandos. La notación infija que puede estar completamente patentizada o puede basarse en un esquema de precedencia de operadores así como el uso de paréntesis para invalidad los arreglos al expresar el orden de evaluación de una expresión:

3\*4 =12

3\*4+2 =14

3\*(4+2) =18

Notación infija: la notación habitual, el orden es primer operando, operador, segundo operando.

**Posfija:**

Este, como su nombre lo indica, se refiere a que el operador ocupa la posición después de los operandos de sus características principales son:

* El orden de los operandos se conserva igual que la expresión infija equivalente no utiliza paréntesis ya que no es una operación ambigua
* La operación posfija no es exactamente lo inverso a la operación prefija equivalente

(A+B)\*C = AB + C\*

El orden es primero operando, segundo operando, operador

Si deseamos representar las expresiones (2+(3\*4)) = x y ((2+3)\*4)= x en las tres notaciones mencionadas, el resultado sería:

|  |  |  |
| --- | --- | --- |
|  | **(2+(3\*4)) = *x*** | **((2+3)\*4) = *x*** |
| **Notación infija** | 2+3\*4 = *x* | (2+3)\*4 = *x* |
| **Notación postfija** | 2 3 4 \* + *x* = | 2 3 + 4 \* *x* = |

## **Colas**

***Operaciones***

## **Recursividad**

## **Ordenamiento y búsqueda**

## **Arboles**

## **Grafos**

## **Tablas flash**

# Bibliografía

Aguilar, Luis Joyanes. *monografias.* 2007. https://www.monografias.com/trabajos44/pilas-listas-expresiones/pilas-listas-expresiones2.

Cumplido, Dr. René. *inaoep.* 05 de 04 de 2015. https://ccc.inaoep.mx/ingreso/programacion/corto2015/Curso-PROPE-PyED-5-Pilas-Colas.pdf (último acceso: 04 de 04 de 2023).

Pozo, Salvador. *Con Clase.* 01 de 08 de 2001. https://conclase.net/c/edd/cap4#gsc.tab=0.

POZO, SALVADOR. *CON CLASE.* 1 de 9 de 2001. https://conclase.net/c/edd/cap5#gsc.tab=0.

Villanueva, Sergio Sama. *DSTool.* 5 de enero de 2001. http://www.hci.uniovi.es/Products/DSTool/pilas/pilas-operaciones.html.