TEMA:

ESTRUCTURAS DINÁMICAS DE DATOS

ALUMNO:

PEREZ PACHERREZ, Luis Eduardo

ASIGNATURA:

ALGORITMO Y ESTRUCTURA DE DATOS

DOCENTE:

CRISTIAN ROBERTO SANCHEZ FLORES

TURNO:

Noche

2019 – II

Estructuras Dinámicas de datos

Las estructuras dinámicas de datos es una colección de elementos conocidos como nodos, que son normalmente registros. Son estructuras que van creciendo a medida que se ejecuta un programa. Si se compara con un "*array*", éste se amplía y contrae durante la ejecución del programa, basado en los registros de almacenamiento de datos del programa.

Se divide en dos grupos:

- Lineales:
  - Pilas
  - Colas
  - Listas enlazadas.
- No lineales
  - Árboles
  - Grafos

Vamos a tocar a fondo el tema de estructura dinámicas lineales de datos.

Lista Enlazada Simple (Linked List)

Algunos inconvenientes que se presentan en las listas contiguas, se soluciona con las **Listas enlazadas**. Se puede almacenar los elementos de una lista lineal en posiciones de memoria que no sean continuas o adyacentes.

Es un conjunto de elementos en los que cada elemento contiene la posición del siguiente elemento de la lista. Cada elemento debe tener al menos dos campos: un campo que tiene el valor del elemento y un campo (*enlace o link*) que contiene la posición del siguiente elemento. Los elementos de una lista son enlazados por medio de los campos enlaces.

Las listas enlazadas tienen una terminología propia que se suele utilizar normalmente. Primero, los valores se almacenan en un nodo.

Los componentes de un nodo se llaman campos. Un nodo tiene un campo dato y un enlace (indicador o puntero) con el siguiente nodo. El enlace apunta (proporciona dirección o referencia) al siguiente nodo.

Para implementar una lista enlazada depende del lenguaje. C, C++ utilizan simplemente como enlace una variable puntero o puntero. Java no dispone de punteros. Esto lo veremos más adelante.

Aquí mostramos un ejemplo de la creación de una lista enlazada simple en el lenguaje Java.

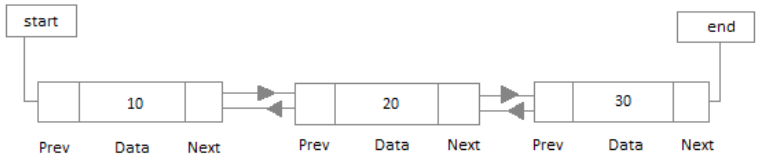
```java
public class SimpleLinkedList {

    public static void main(String[] args) {
        LinkedList <Integer> myList = new LinkedList <Integer>();
        myList.add(1);
        myList.add(2);
        myList.addLast(3);
        myList.addFirst(4);
        myList.add(2, 5);
        myList.add(6);
        myList.add(7);
        myList.add(1,8);
        System.out.println(myList);
        myList.remove(2);
        System.out.println(myList);
    }
}
```

La salida de consola de este código es el siguiente.

```
run:
[4, 8, 1, 5, 2, 3, 6, 7]
[4, 8, 5, 2, 3, 6, 7]
BUILD SUCCESSFUL (total time: 1 second)
```

Lista Enlazada Doble (Doubly Linked List)

Una lista enlazada doble contiene un puntero adicional, normalmente denominado *puntero anterior*, junto con el siguiente puntero y los datos que están allí en lista enlazada.

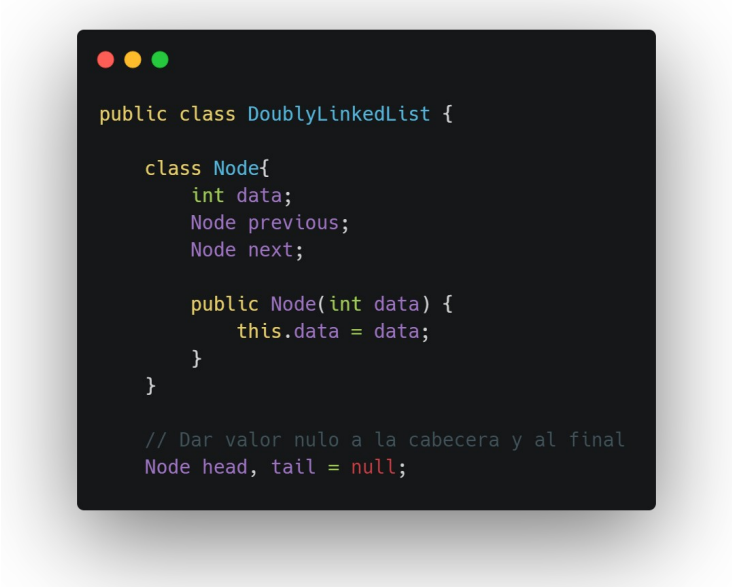


En la lista estudiada anteriormente se podía recorrer a ellas de un solo sentido: de izquierda a derecha. Pero en distintas ocasiones se necesitará recorrer una lista en ambas direcciones.

En estas listas, cada nodo consta del campo INFO de datos y dos campos de enlace o punteros: *Anterior (prev) y Siguiente (next).* Como cada elemento tiene dos punteros, una lista doblemente enlazada ocupa más espacio de memoria que una lista simple para la misma información.

Aquí mostramos un ejemplo de la creación de una lista enlazada simple en el lenguaje Java.

Creamos una clase llamada Node donde vamos a construir un objeto nodo.



```java
public class DoublyLinkedList {

    class Node{
        int data;
        Node previous;
        Node next;

        public Node(int data) {
            this.data = data;
        }
    }

    // Dar valor nulo a la cabecera y al final
    Node head, tail = null;
```

Agregamos el método *AddNode()* que nos servirá para llenar nuestra lista.

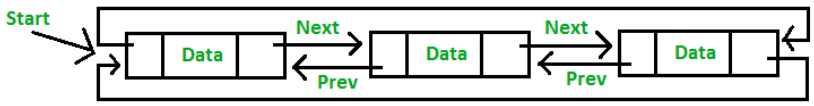
```java
   //Creación de un nodo
    public void addNode(int data) {
    Node newNode = new Node(data);


        if(head == null) {
            // Tanto la cabecera como la "cola" se apunta al nuevo nodo
            head = tail = newNode;
            // La cabecera será null
            head.previous = null;
            // El siguiente de la lista será null por que es el final
            tail.next = null;
        }
        else {
            // Se agregará después de la cola, de modo que la cola siguiente apunte a la instacia del
nodo
            tail.next = newNode;
            // Esto apuntará al final de la lista
            newNode.previous = tail;
            // el último nodo se reemplaza por el nodo creado
            tail = newNode;
            // Al ser el último, el siguiente pasa a nulo.
            tail.next = null;
        }
    }
```

Agregamos el método *getNode()* que nos servirá para mostrar en consola nuestra lista.

```java
// impresión de la lista
public void getNode() {
    // Colocamos un indice
    Node current = head;
    if(head == null) {
        System.out.println("La lista está vacia");
        return;
    }
    System.out.println("Datos de la lista: ");
    while(current != null) {
       System.out.print(current.data + " ");
        current = current.next;
    }
}
}
```

Lista Enlazada Doble Circular (Doubly Circular Linked List)

Es una lista vinculada cuyos nodos están conectados de tal manera que forma un círculo. En la lista circular vinculada, el siguiente puntero del último nodo no está establecido en nulo, pero contiene la dirección del primer nodo, formando así un círculo.



Para implementar la lista enlazada circular, mantenemos un puntero externo "último" que apunta al último nodo en la lista enlazada circular. Por lo tanto, last-> next apuntará al primer nodo en la lista vinculada.

Al hacer esto, nos aseguramos de que cuando insertemos un nuevo nodo al principio o al final de la lista, no necesitemos recorrer toda la lista. Esto se debe a que el último apunta al último nodo mientras que el último-> siguiente apunta al primer nodo.

Esto no hubiera sido posible si hubiéramos apuntado el puntero externo al primer nodo.

Punteros

Un puntero, también llamado apuntador, es una variable cuyo valor es la dirección o posición de otra variable. En las listas enlazadas no es necesario que los datos

Como se indicó, Java no trabaja con punteros. En este caso se resuelve el problema de manera diferente con referencias.

Una referencia es definida como el puntero de un objeto o una referencia nula (de donde proviene el NullPointerException).

Cuando uno declara una variable en Java lo hace de la siguiente manera:

‘a’, apunta hacia una dirección en memoria donde contiene 5. Si no se inicializa en 5, el compilador inicializa a ‘a’ en una dirección con el contenido 0.