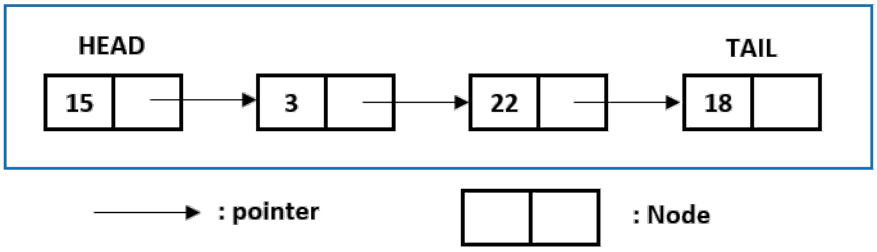
1. **Single, Double, Circular Linked List**

Single:

Sebuah linked list terdiri dari sejumlah Node. Di Single Linked list, sebuah node terdiri atas:

- 1 pointer
- Data (int, double, char, float, string)

Pointer ini menghubungkan antara satu node ke node lainnya. Node paling kiri disebut **HEAD**. Node paling kanan disebut **TAIL**, pointernya mengarah ke **NULL**. Single Linked List hanya bisa traverse maju ke arah kanan.

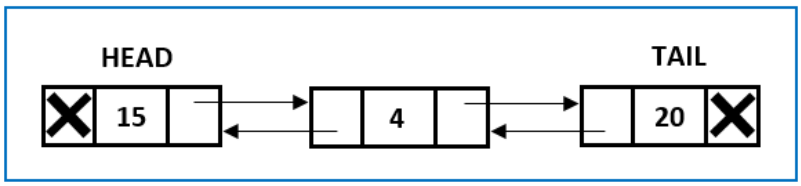


Double:

Di Double Linked List, sebuah node terdiri atas:

- 2 pointer
- Data (int, double, char, float, string)

Pointer kiri menghubungkan sebuah node ke node sebelumnya, sedangkang pointer kanan menghubungkan sebuah node ke node selanjutnya. Node paling kiri disebut **HEAD**, pointer kirinya mengarah ke **NULL**. Node paling kanan disebut **TAIL**, pointer kanannya mengarah ke **NULL**. Untuk Double Linked List, bisa traverse maju ke arah kanan maupun traverse mundur ke arah kiri.



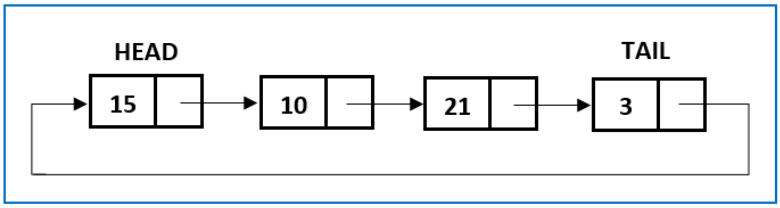
X : NULL

Circular:

Di Circular Linked List, sebuah node terdiri atas:

- 1 pointer
- Data (int, double, char, float, string)

Circular mirip dengan Single Linked List, yaitu memiliki 1 pointer yang menghubungkan sebuah node dengan node selanjutnya dan hanya bisa traverse maju ke arah kanan. Perbedaanya adalah circular bisa mengalami loop berulang, karena pointer pada Node **TAIL** mengarah ke Node **HEAD**. Sehingga ketika sebuah Node telah traverse hingga **TAIL**, Node tsb akan traverse kembali lagi ke Node **HEAD**.



2. **Perbedaan Linked List dan Array**

ARRAY	LINKED LIST
Size dari array tidak dapat diubah pada saat program sedang dijalankan.	Size dari Linked List bisa berubah (bertambah/berkurang) pada saat program dijalankan
Memiliki index order	Memiliki pointer dan node head and tail
Mudah untuk memanipulasi sebuah data dari array, karena telah diketahui posisi tiap data dalam array	Tidak bisa mengetahui posisi sebuah data dan harus membuat sebuah node *curr untuk traverse ke posisi data tsb
Sedikit memakai memory	Memerlukan memory allocation yang lebih karena ada pointer

3. **Floyd's algorithm**

Algoritma ini digunakan untuk menentukan apakah terdapat cycle dalam sebuah linked list.

- Menggunakan 2 pointer yaitu, pointer “fast” dan pointer “slow”
- Pointer “slow” traverse maju sebanyak 1 langkah
- Pointer “fast” traverse maju sebanyak 2 langkah
- Ketika pointer “slow” dan pointer “fast” traverse dan keduanya bertemu di satu node, itu menandakan bahwa terdapat cycle dalam linked list tsb.

Pseudocode:

```
DetectCycle
    IF (NOT head) THEN
        return false

    slow = head;
    fast = head->next;

    DOWHILE (fast NOT = NULL AND slow NOT = NULL)
        IF (fast = slow) THEN
            Return true
        fast = fast->next->next
        slow = slow->next
    ENDDO

    return false
END
```