INF218
Struktur Data & Algoritma

Linked List

Alim Misbullah, S.Si., M.S.

Review Array...

- Array didefinisikan sebagai kumpulan data yang memiliki tipe data yang sama yang saling berdekatan di dalam memori
- Array diturunkan dari tipe data primitive di dalam bahasa C seperti int, float, char, double, dll
- Elemen dari array dapat diakses menggunakan index dari elemen tersebut
- Setiap element memiliki tipe data dan ukuran yang sama, seperti int = 4 bytes (int a[10] = 10 * 4 = 40 → a[0] untuk elemen 1)
- Array sangat diperlukan untuk data yang berbeda-beda namun memiliki tipe yang sama, seperti nilai mahasiswa

Review Array...

Time Complexity

Algorithm	Average Case	Worst Case
Access	O(1)	O(1)
Search	O(n)	O(n)
Insertion	O(n)	O(n)
Deletion	O(n)	O(n)

Space Complexity

Pada array, space complexity adalah **O(n)**

*Watch this video to understand more about **O notation:** https://www.youtube.com/watch?v=__vX2sjlpXU*

Review Struct dan Pointer...

```c
#include <stdio.h>
struct person
{
   int age;
   float weight;
};

int main()
{
    struct person *personPtr, person1;
    personPtr = &person1;

    printf("Enter age: ");
    scanf("%d", &personPtr->age);

    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);

    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f", personPtr->weight);

    return 0;
}
```

Sebelum ke Linked List

int x = 6; → 4 bytes in memory

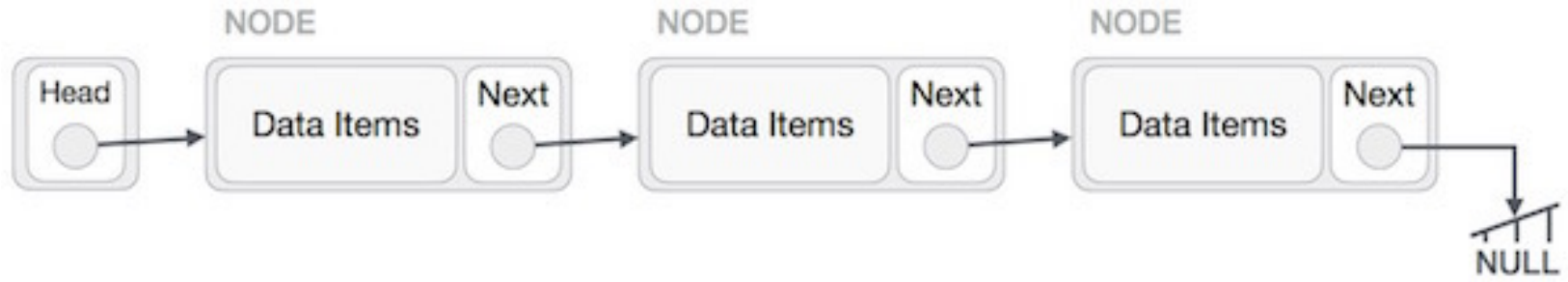
int x[10] = {5, 7, 9}; → ? (4 x 3 = 12)

Kekurangan array:

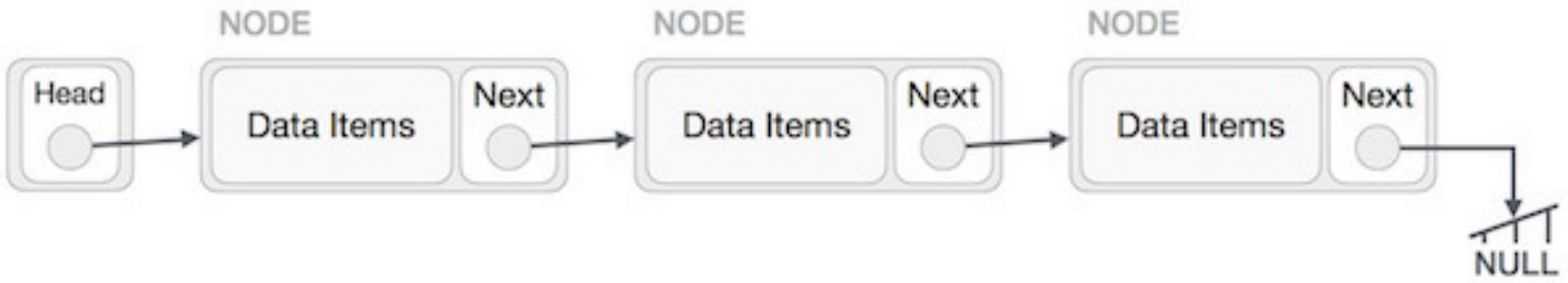
- Tidak dapat menambahkan elemen lebih dari yang sudah di deklarasikan
- Pada array, banyak sekali memory yang akan terbuang dan memory ini tidak dapat digunakan oleh variabel yang lain karena sudah dialokasikan untuk array tersebut

Definisi Linked List

- Linked List merupakan kumpulan **node** yang saling terhubung di dalam memori. Node tersebut terdiri dari 2 bagian yaitu **data** dan **alamat** dari dari berikutnya



Deklarasi Linked List



- **Head** adalah permulaan dari sebuah Linked List
- Setiap node merepresentasikan **data** dan **alamat** dari data berikutnya

```
struct node
{
  int data;
  struct node *next;
};
```

Deklarasi Linked List

```c
/* Initialize nodes */
struct node *head;
struct node *one = NULL;
struct node *two = NULL;
struct node *three = NULL;

/* Allocate memory */
one = malloc(sizeof(struct node));
two = malloc(sizeof(struct node));
three = malloc(sizeof(struct node));

/* Assign data values */
one->data = 1;
two->data = 2;
three->data=3;

/* Connect nodes */
one->next = two;
two->next = three;
three->next = NULL;

/* Save address of first node in head */
head = one;
```

```c
struct node
{
  int data;
  struct node *next;
};
```

Deklarasi Linked List

- Linked memiliki kelebihan untuk memutuskan rantai dan menggabungkan Kembali tanpa merusak struktur element. Misalnya, kita ingin menambahkan nilai 4 diantara nilai 1 dan 2, caranya:
  - Buat struct node yang baru dan alokasi memori ke node tersebut
  - Tambahkan nilai 4
  - Gunakan pointer dari node baru ke nilai 2
  - Ubah pointer dari node yang nilai 1 ke nilai 4 yang baru ditambahkan

Demo Simple Linked List

https://www.programiz.com/dsa/linked-list#c-code

Operasi Pada Linked List

Traverse yaitu menampilkan elemen linked list

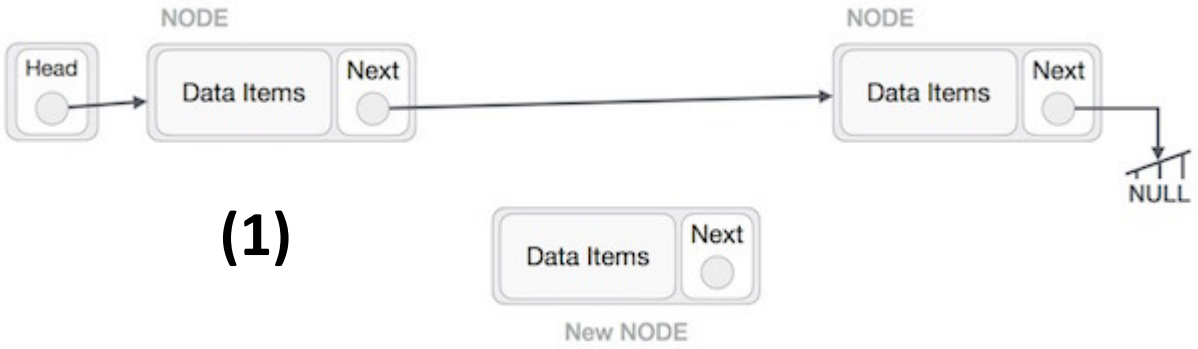
```
struct node *temp = head;
printf("\n\nList elements are - \n");
while(temp != NULL)
{
     printf("%d --->",temp->data);
     temp = temp->next;
}
```

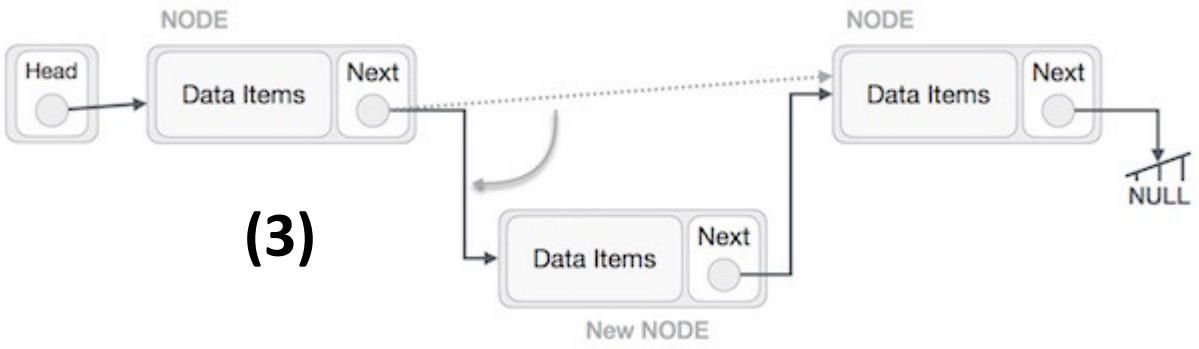
Outputnya adalah:

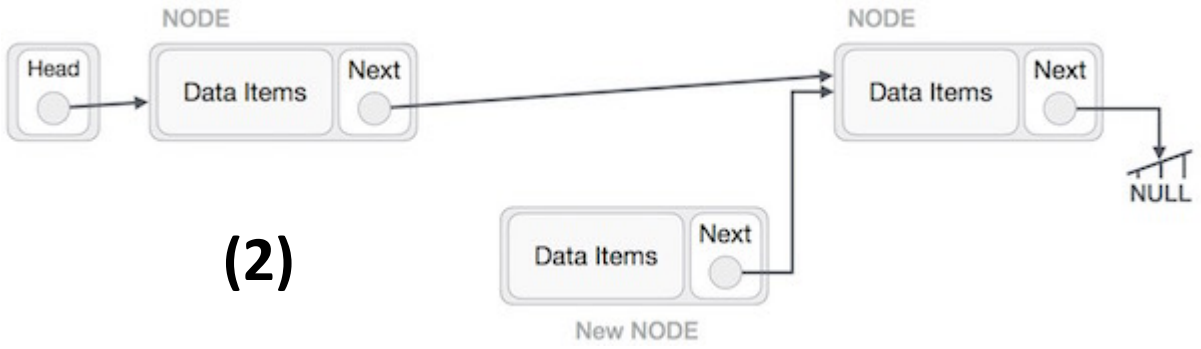
```
List elements are -
1 --->2 --->3 --->
```

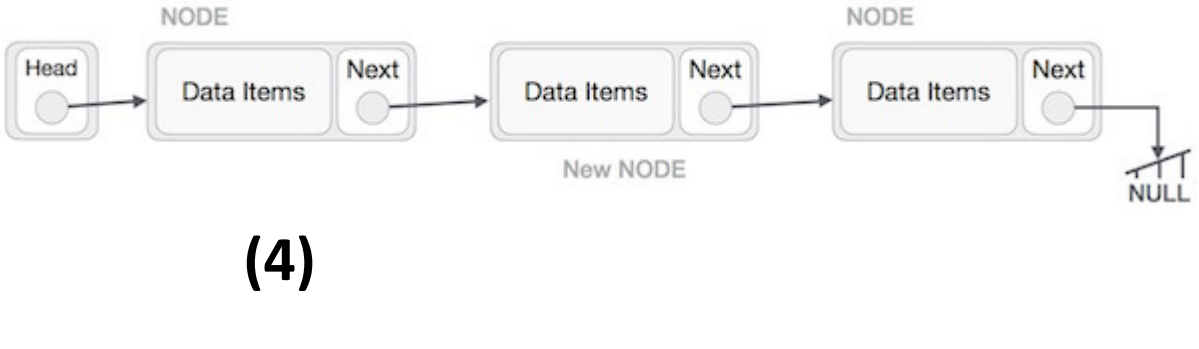
Operasi Pada Linked List

Insertion yaitu menambahkan elemen ke linked list



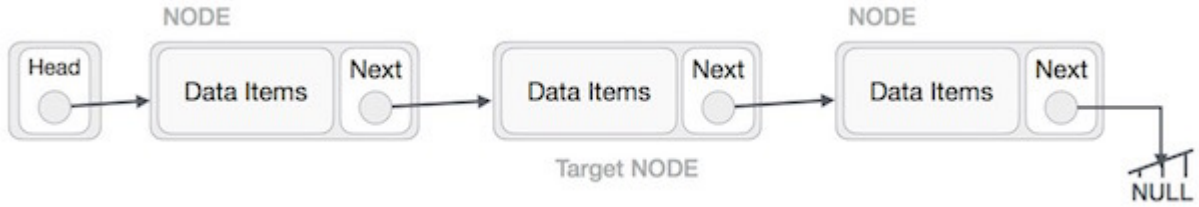




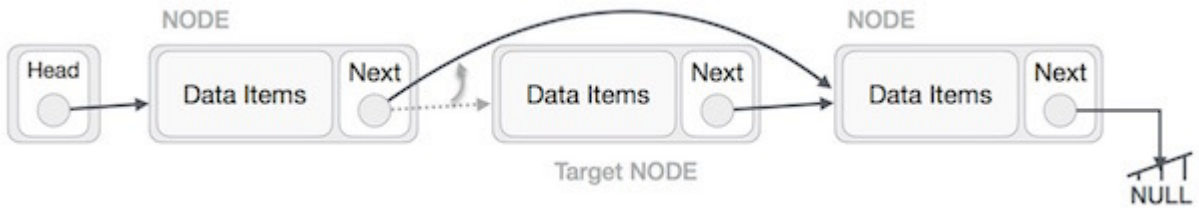


Operasi Pada Linked List

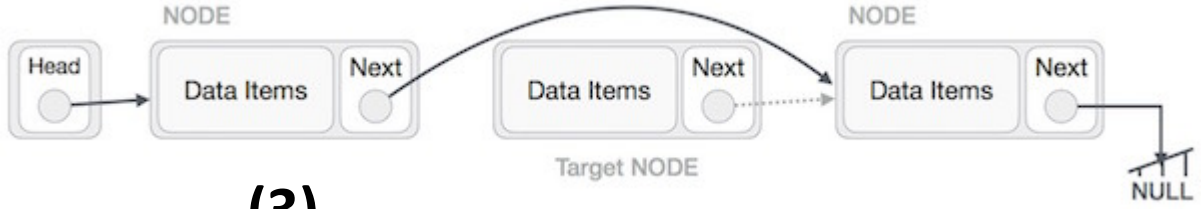
Deletion yaitu menghapus elemen dari linked list



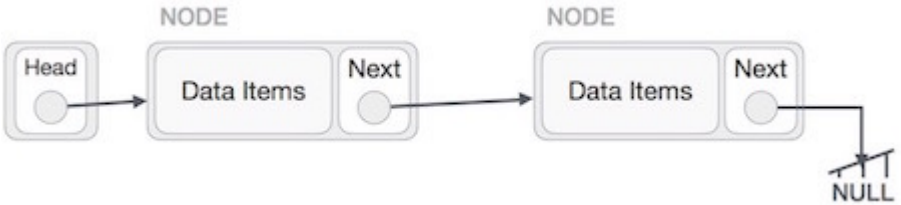
(1)



(2)



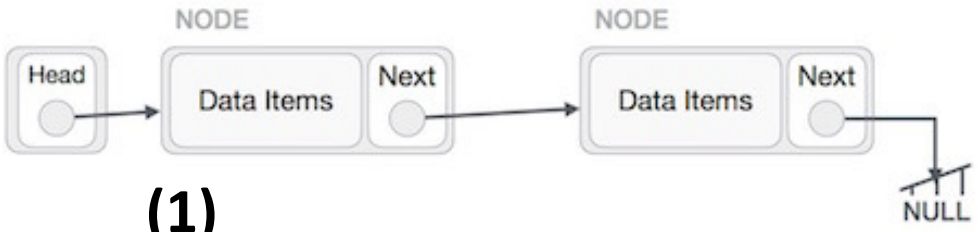
(3)



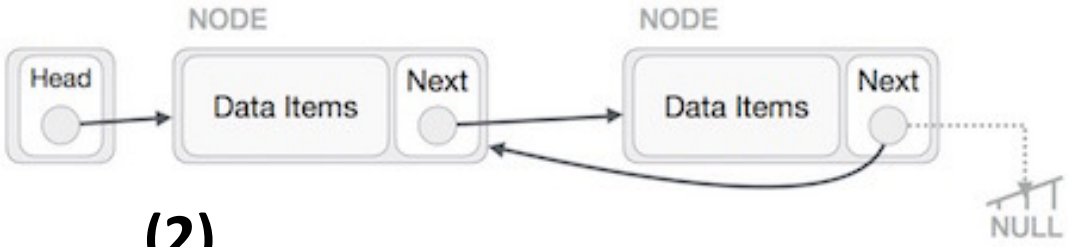
(4)

Operasi Pada Linked List

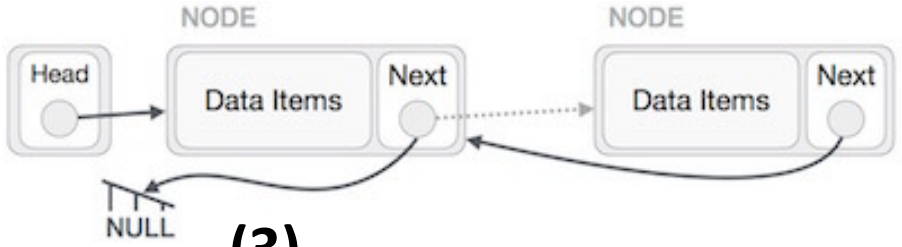
Reverse yaitu membalikkan posisi elemen dari linked list



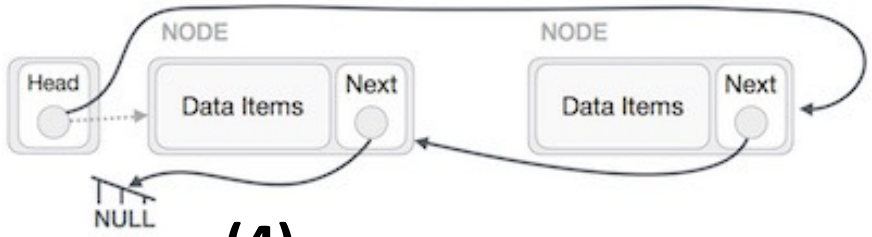
(1)



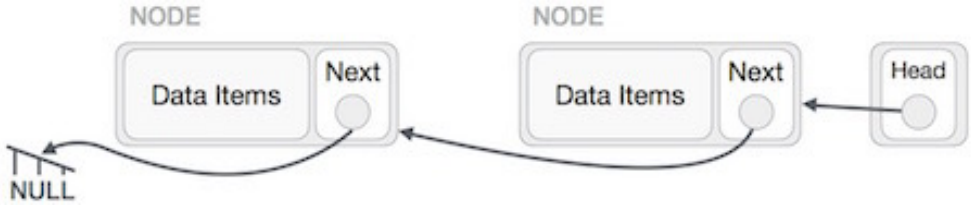
(2)



(3)



(4)



(5)

Demo Operasi Pada Linked List

https://www.tutorialspoint.com/data_structures_algorithms/linked_list_program_in_c.htm

Kompleksitas Linked List

Time Complexity

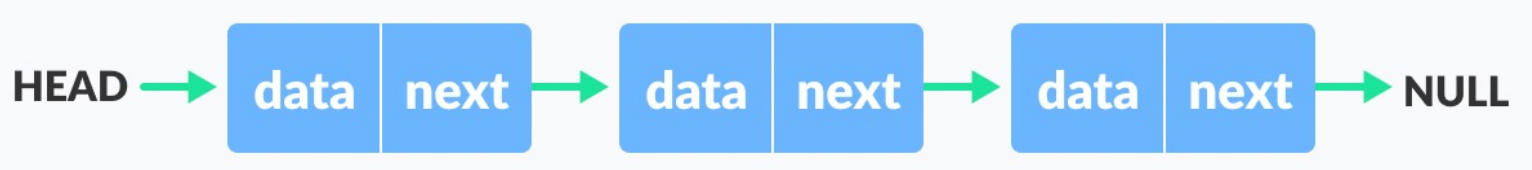
	Worst case	Average Case
Search	O(n)	O(n)
Insert	O(1)	O(1)
Deletion	O(1)	O(1)

Space Complexity: O(n)

Tipe Linked List

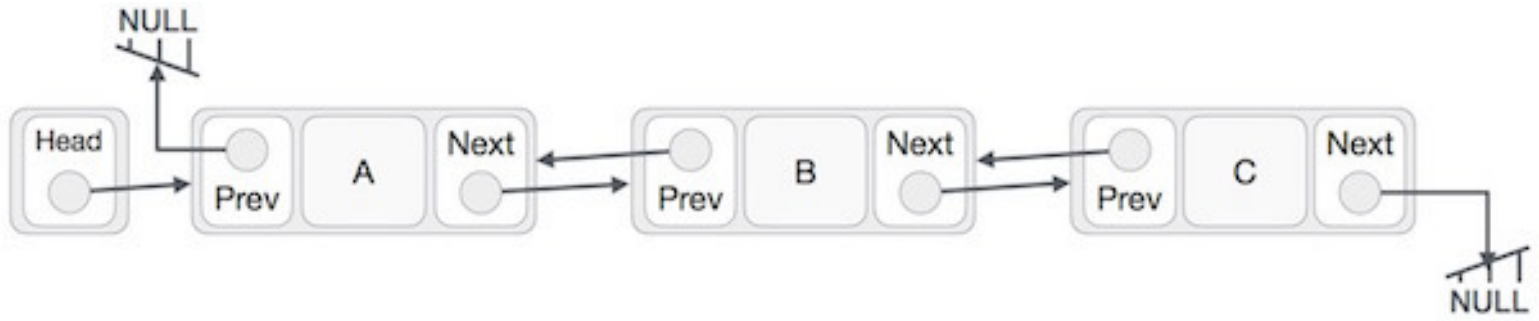
- Singly Linked List
  - Setiap node memiliki data/nilai dan pointer ke node berikutnya
- Doubly Linked List
  - Setiap node memiliki pointer ke node sebelumnya dan berikutnya sehingga linked list memiliki 2 arah (forward dan backward)
- Circular Linked List
  - Akhir dari sebuah node selalu terhubung ke node yang pertama

Tipe Linked List: Singly Linked List




```c
struct node {
    int data;
    struct node *next;
}
```

Tipe Linked List: Doubly Linked List




```c
struct node {
    int data;
    struct node *next;
    struct node *prev;
}
```

Beberapa poin penting dari Doubly Linked List:

- Doubly Linked List mengandung sebuah link elemen yang disebut dengan **first** dan **last**
- Setiap link membawa sebuah data dan dua link yang disebut dengan **next** and **previous**
- Setiap link terhubung dengan link berikutnya menggunakan **next** link
- Setiap link terhubung dengan link sebelumnya menggunakan **previous** link
- Link terakhir ditandai dengan **null** yang menandakan akhir dari sebuah list

Tipe Linked List: Doubly Linked List

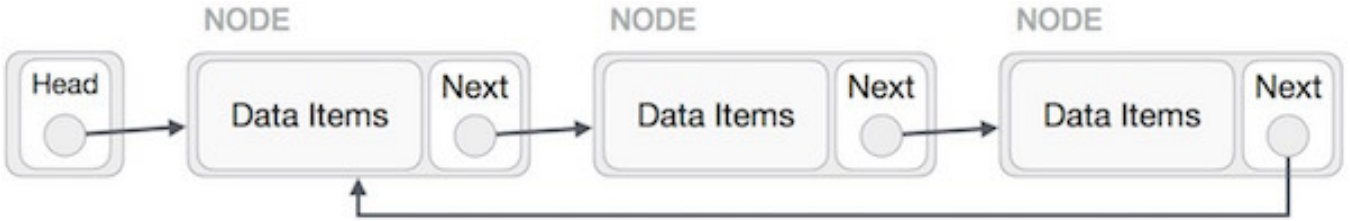
- Operasi dasar pada Doubly Linked List:
  - Insertion: Menambahkan sebuah elemen pada bagian awal list
  - Deletion: Menghapus sebuah element pada bagian awal list
  - Insert Last: Menambahkan sebuah element pada bagian akhir list
  - Delete Last: Menghapus sebuah elemen pada bagian akhir list
  - Delete: Menghapus sebuah elemen dari list menggunakan key
  - Display forward: Menampilkan semua list dari awal ke akhir
  - Display backward: Menampilkan semua list dari akhir ke awal

Demo Doubly Linked List

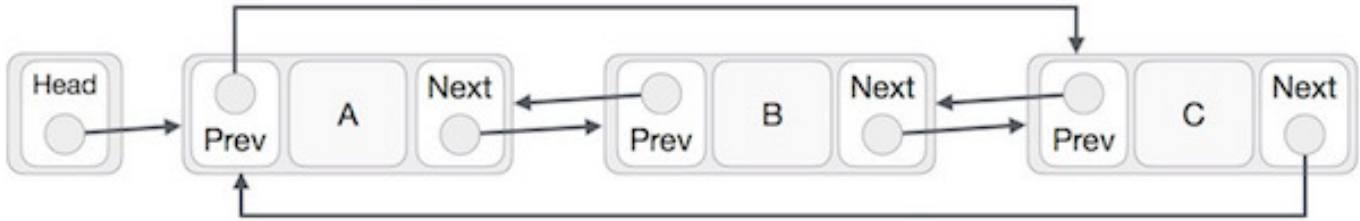
https://www.tutorialspoint.com/data_structures_algorithms/doubly_linked_list_program_in_c.htm

Tipe Linked List: Circular Linked List

Singly Linked List as Circular



Doubly Linked List as Circular



Beberapa poin penting pada Circular Linked List:

- Link yang ada pada list terakhir akan mengarah ke link pertama pada kedua jenis linked list
- Link **prev** pada node pertama akan mengarah ke link **prev** pada node terakhir

```
/* Connect nodes */
one->next = two;
two->next = three;
three->next = one;
```

Tipe Linked List: Circular Linked List

- Operasi dasar pada Circular Linked List:
  - Insert: Menambahkan sebuah elemen pada bagian awal dari list
  - Delete: Menghapus sebuah elemen pada bagian awal dari list
  - Display: Menampilkan list

Demo Circular Linked List

https://www.tutorialspoint.com/data_structures_algorithms/circular_linked_list_program_in_c.htm

Aplikasi Linked List

- Dynamic Memory Allocation
- Diimplementasikan pada Stack dan Queue
- Diaplikasi pada fitur **undo** dari sebuah software
- Hash table dan graphs

Contoh Linked List untuk Praktikum Mandiri

- Java
  - https://www.programiz.com/java-programming/examples/get-middle-element-of-linkedlist
  - https://www.programiz.com/java-programming/examples/linkedlist-array-conversion
  - https://www.programiz.com/java-programming/examples/detect-loop-in-linkedlist
- C
  - https://www.tutorialspoint.com/data_structures_algorithms/linked_list_program_in_c.htm
  - https://www.tutorialspoint.com/data_structures_algorithms/doubly_linked_list_program_in_c.htm
  - https://www.tutorialspoint.com/data_structures_algorithms/circular_linked_list_program_in_c.htm

Referensi

- https://www.programiz.com/dsa/linked-list
- https://www.javatpoint.com/ds-linked-list
- https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm
- https://www.geeksforgeeks.org/data-structures/linked-list/