**Nama : Adrian Aszhar Maulana**

**NPM : 232310008**

**Lab.Algoritma Pemrograman & Struktur Data**

**PENGERTIAN LINKED LIST**

Linked List adalah salah satu struktur data penting dalam pemrograman yang digunakan untuk menyimpan dan mengelola data secara dinamis. Struktur data ini memungkinkan kita untuk dengan mudah membuat tempat baru untuk menyimpan data kapan saja dibutuhkan.

Dalam Linked List, data disimpan dalam bentuk simpul atau node yang saling terhubung satu sama lain dengan menggunakan referensi atau alamat dari simpul selanjutnya dalam urutan. Setiap simpul berisi dua hal penting: data yang ingin kita simpan dan alamat referensi ke simpul berikutnya dalam urutan. Dengan begitu, setiap simpul dapat menyimpan data dan mengetahui alamat simpul selanjutnya.

Setiap Linked List memiliki dua elemen khusus, yaitu “head” dan “tail”:

1. Head; Merupakan simpul pertama dalam Linked List dan berfungsi sebagai titik awal akses ke seluruh data dalam Linked List.
2. Tail: Merupakan simpul terakhir dalam Linked List dan menjadi penanda akhir dari urutan simpul

.

**FUNGSI LINKED LIST**

Linked List memiliki beberapa fungsi penting, antara lain:

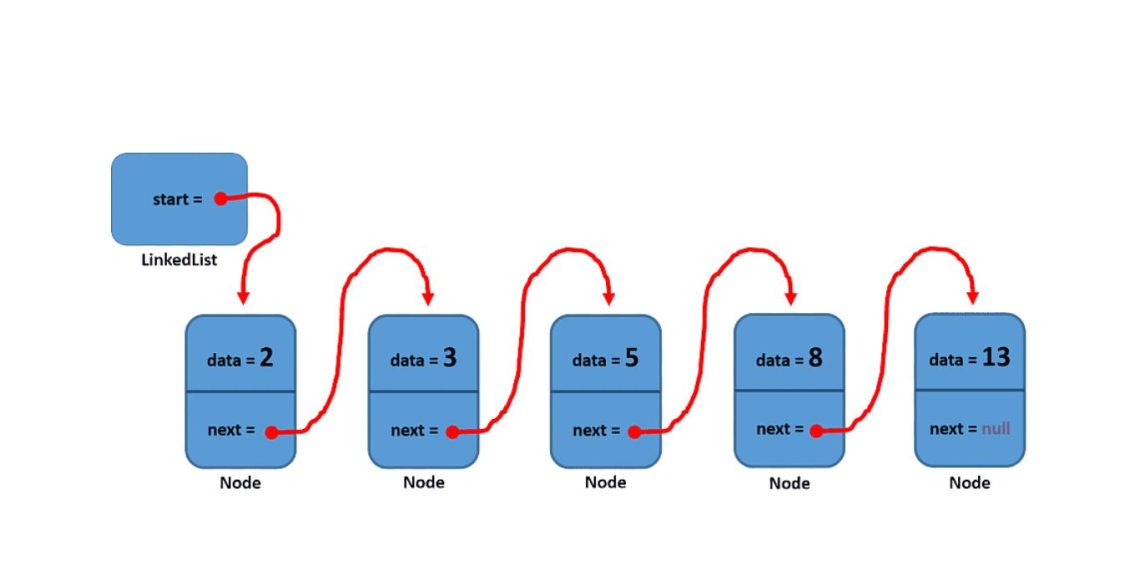
1. Menyimpan dan mengelola data dalam urutan tertentu.
2. Memudahkan penambahan dan penghapusan data secara dinamis tanpa harus menggeser data lain.
3. Digunakan dalam implementasi berbagai algoritma dan struktur data lain seperti stack dan queue.

**JENIS-JENIS LINKED LIST**

Linked List adalah struktur data penting dalam pemrograman yang digunakan untuk menyimpan dan mengatur data secara dinamis.

1. Pada Single Linked List, setiap simpul memiliki dua bagian: data yang ingin disimpan dan referensi ke simpul berikutnya. Simpul terakhir menunjuk ke null sebagai penanda akhir Linked List.
2. Keuntungan Single Linked List adalah struktur datanya yang ringan karena hanya memerlukan satu referensi untuk setiap simpul. Namun, untuk mencari elemen tertentu, kita harus melintasi seluruh Linked List dari awal hingga elemen yang diinginkan.
3. Double Linked List lebih kompleks daripada Single Linked List. Setiap simpul memiliki tiga bagian: data, referensi ke simpul berikutnya, dan referensi ke simpul sebelumnya. Keuntungan Double Linked List adalah kemampuan untuk melintasi Linked List maju (dari depan ke belakang) dan mundur (dari belakang ke depan) dengan mudah. Namun, kelemahannya adalah memerlukan lebih banyak ruang memori karena setiap simpul harus menyimpan dua referensi.
4. Circular Linked List adalah tipe Linked List yang membentuk lingkaran, dengan simpul terakhir terhubung kembali ke simpul pertama. Tidak ada simpul yang menunjuk ke null seperti pada Single Linked List dan Double Linked List.  
   Keuntungan Circular Linked List adalah kemampuan untuk berputar terus menerus tanpa perlu mengubah referensi. Circular Linked List sering digunakan dalam implementasi antrian berputar dan penjadwalan.
5. Multiple Linked List terdiri dari beberapa kumpulan Linked List yang terpisah, tetapi dapat diakses bersamaan. Setiap kumpulan Linked List berisi simpul-simpul yang terkait dalam satu himpunan.  
   Jenis Linked List ini memungkinkan pengaturan data dalam kelompok-kelompok terpisah dan menjaga struktur yang rapi dan terorganisir. Dengan pemahaman tentang perbedaan jenis-jenis Linked List ini, kita dapat membuat pilihan yang tepat dalam menggunakan struktur data yang sesuai dengan kebutuhan aplikasi.

**CONTOH GAMBAR LINKED LIST**



**Perbedaan Antara Array dan Linked List**

Array dan Linked List adalah dua cara berbeda untuk menyimpan dan mengatur data dalam pemrograman.

Pertama, Array memerlukan alokasi memori kontiguitas, yang berarti elemen data harus disusun secara berurutan dalam memori. Ini membuat penambahan atau penghapusan elemen menjadi lebih sulit, karena kita perlu menggeser elemen lain untuk memberi ruang pada posisi yang diinginkan.

Selain itu, ukuran Array tetap dan ditentukan saat deklarasi, sehingga sulit untuk mengubah ukuran Array secara dinamis. Meskipun Array memberikan akses elemen yang cepat melalui indeks, namun ada keterbatasan dalam fleksibilitas dan efisiensi ketika datanya sering berubah.

Kedua, Linked List menggunakan pendekatan yang berbeda. Linked List tidak memerlukan alokasi memori kontiguitas, sehingga setiap elemen data dapat ditempatkan di lokasi memori yang berbeda dan dihubungkan melalui referensi. Ini membuat penambahan dan penghapusan elemen menjadi lebih efisien, karena kita hanya perlu mengatur ulang referensi antar Node.

Ukuran Linked List dapat berubah selama runtime, sehingga memungkinkan penambahan atau penghapusan elemen dengan mudah. Namun, Linked List memerlukan pencarian melalui elemen sebelumnya untuk mengakses data, sehingga akses elemen membutuhkan waktu lebih lama. Meskipun Linked List memiliki keterbatasan, namun ini merupakan pilihan yang fleksibel untuk situasi di mana data sering berubah atau ketika ukuran data tidak diketahui sebelumnya.