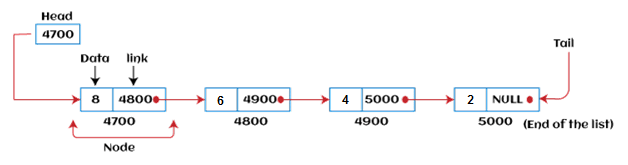
Linked List

Struktur data linier yang mencakup serangkaian simpul yang terhubung. Daftar tertaut dapat didefinisikan sebagai node yang disimpan secara acak dalam memori. Sebuah node dalam linked list berisi dua bagian, yaitu, pertama adalah bagian data dan kedua adalah bagian alamat. Node terakhir dari daftar berisi pointer ke nol. Setelah array, linked list adalah struktur data kedua yang paling banyak digunakan



Mengapa menggunakan linked list daripada array?

Array

* Ukuran array harus diketahui terlebih dahulu sebelum digunakan dalam program.
* Meningkatkan ukuran array adalah proses yang memakan waktu. Hampir tidak mungkin untuk memperluas ukuran array saat dijalankan.
* Semua elemen dalam array harus disimpan secara berurutan dalam memori. Menyisipkan elemen dalam array membutuhkan pergeseran dari semua pendahulunya.

Linkedlist

* Ini mengalokasikan memori secara dinamis. Semua node dari daftar tertaut tidak disimpan secara berurutan dalam memori dan dihubungkan bersama dengan bantuan pointer.
* Dalam linked list, ukuran tidak lagi menjadi masalah karena kita tidak perlu mendefinisikan ukurannya pada saat deklarasi. Daftar bertambah sesuai permintaan program dan terbatas pada ruang memori yang tersedia.

Jenis Linked List

Singly Linked List

didefinisikan sebagai kumpulan elemen yang diurutkan. Sebuah node dalam daftar tertaut tunggal terdiri dari dua bagian: bagian data dan bagian tautan. Bagian data dari node menyimpan informasi aktual yang akan diwakili oleh node, sedangkan bagian link dari node menyimpan alamat penerus langsungnya.

Doubly Linked List

jenis daftar tertaut yang kompleks di mana sebuah simpul berisi penunjuk ke simpul sebelumnya serta simpul berikutnya dalam urutan. Oleh karena itu, dalam daftar tertaut ganda, sebuah simpul terdiri dari tiga bagian: data simpul, penunjuk ke simpul berikutnya secara berurutan (penunjuk berikutnya), dan penunjuk ke simpul sebelumnya (penunjuk sebelumnya).

Circular Linked List

simpul terakhir dari daftar berisi penunjuk ke simpul pertama dari daftar. Kita dapat memiliki daftar tertaut tunggal melingkar serta daftar tertaut ganda melingkar.

Circular Doubly List

jenis struktur data yang lebih kompleks di mana sebuah simpul berisi penunjuk ke simpul sebelumnya serta simpul berikutnya. Daftar tertaut ganda melingkar tidak mengandung NULL di salah satu node. Node terakhir dari daftar berisi alamat node pertama dari daftar. Node pertama dari daftar juga berisi alamat node terakhir di pointer sebelumnya.

### Keuntungan

* **Struktur data dinamis -** Ukuran daftar tertaut dapat bervariasi sesuai dengan persyaratan. Daftar tertaut tidak memiliki ukuran tetap.
* **Penyisipan dan penghapusan -** Tidak seperti array, penyisipan, dan penghapusan dalam daftar tertaut lebih mudah. Elemen array disimpan di lokasi yang berurutan, sedangkan elemen dalam daftar tertaut disimpan di lokasi acak. Untuk menyisipkan atau menghapus elemen dalam array, kita harus menggeser elemen untuk membuat spasi. Sedangkan pada linked list, alih-alih menggeser, kita hanya perlu mengupdate alamat pointer dari node tersebut.
* **Hemat memori -** Ukuran daftar tertaut dapat bertambah atau berkurang sesuai dengan kebutuhan, sehingga konsumsi memori dalam daftar tertaut menjadi efisien.
* **Implementasi -** Kami dapat mengimplementasikan tumpukan dan antrian menggunakan daftar tertaut.

### Kekurangan

* **Penggunaan memori -** Dalam daftar tertaut, node menempati lebih banyak memori daripada array. Setiap node dari linked list menempati dua jenis variabel, yaitu, satu adalah variabel sederhana, dan satu lagi adalah variabel pointer.
* **Traversal -** Traversal tidak mudah dalam daftar tertaut. Jika kita harus mengakses sebuah elemen dalam linked list, kita tidak dapat mengaksesnya secara acak, sedangkan dalam kasus array kita dapat mengaksesnya secara acak dengan indeks. Misalnya, jika kita ingin mengakses node ke-3, maka kita harus melintasi semua node sebelumnya. Jadi, waktu yang dibutuhkan untuk mengakses node tertentu besar.
* **Reverse traversing -** Backtracking atau reverse traversing sulit dilakukan dalam daftar tertaut. Dalam daftar tertaut ganda, lebih mudah tetapi membutuhkan lebih banyak memori untuk menyimpan penunjuk belakang.

### Operasi

* **Penyisipan -** Operasi ini dilakukan untuk menambahkan elemen ke dalam daftar.
* **Penghapusan -** Ini dilakukan untuk menghapus operasi dari daftar.
* **Tampilan -** Ini dilakukan untuk menampilkan elemen daftar.
* **Pencarian -** Ini dilakukan untuk mencari elemen dari daftar menggunakan kunci yang diberikan.

|  |
| --- |
| /\* |
| \* To change this license header, choose License Headers in Project Properties. |
| \* To change this template file, choose Tools | Templates |
| \* and open the template in the editor. |
| \*/ |
| package latlinkedlist; |
|  |
| import java.util.LinkedList; |
|  |
| /\*\* |
| \* |
| \* @author User |
| \*/ |
| public class latihanll1 { |
| public static void main(String[] args) { |
| // TODO code application logic here |
| LinkedList<String> kode = new LinkedList<String>(); |
| kode.add("M001"); |
| kode.add("M002"); |
| kode.add("M003"); |
| kode.add("M004"); |
| System.out.println(kode); |
| // Use addFirst() to add the item to the beginning |
| kode.addFirst("M000"); |
| System.out.println(kode); |
|  |
| // Use addLast() to add the item to the end |
| kode.addLast("M006"); |
| System.out.println(kode); |
|  |
| // Use removeFirst() remove the first item from the list |
| kode.removeFirst(); |
| System.out.println(kode); |
| // Use removeLast() remove the last item from the list |
| kode.removeLast(); |
| System.out.println(kode); |
| // Use getFirst() to display the first item in the list |
| System.out.println(kode.getFirst()); |
| // Use getLast() to display the last item in the list |
| System.out.println(kode.getLast()); |
|  |
|  |
|  |
| } |
| } |