Struktur Data

Linked List

Universitas Muslim Indonesia

Oleh
Lutfi Budi Ilmawan

Pengantar Linked List

- Linked list merupakan struktur data yang terdiri dari sekelompok node yang berantai dengan urutan tertentu.
- ***Node*** (simpul) merupakan elemen yang terdapat pada sebuah linked list yang terdiri dari: ***data*** dan ***link***.
- ***Link*** merupakan *field*/atribut pada linked list yang menyimpan informasi alamat dari node lainnya.
- Data merupakan field pada linked list yang menyimpan data atau item.
- ***Link*** pada elemen terakhir bernilai NULL, yang menunjukkan akhir dari suatu linked list yang disebut ***tail.***
- Elemen awal diakses oleh ***head***.

Keuntungan

- Linked List memiliki struktur data dinamis.
- Operasi penyisipan dan penghapusan ***node*** mudah diimplementasikan.
- Struktur data linier seperti ***stack*** dan ***queue*** yang mudah dijalankan dengan linked list.
- Data dapat berkembang secara *real time* tanpa terjadinya ***memory overhead***.

Kekurangan

- Memiliki kecenderungan untuk menggunakan lebih banyak memori karena pointer yang membutuhkan ruang penyimpanan ekstra. Terutama pada ***double linked list***, dibutuhkan memori tambahan untuk ruang yang dibutuhkan oleh ***next pointer*** dan ***previous pointer***.
- Node dalam ***linked list*** harus dibaca berurutan dari awal secara sekuensial.
- ***Node*** disimpan tidak secara berdekatan, sehingga dapat meningkatkan waktu yang dibutuhkan untuk mengakses elemennya.

Jenis Linked List

- ***Single linked list***, terdiri dari dua *field* yakni *data* dan *next pointer*.
- ***Double linked list***, terdiri dari *tiga field yakni data, next pointer, dan previous pointer*.
- ***Multiple linked list***, terdiri dari 2 atau lebih *link field*.
- ***Circular Linked list***, linked list yang *tail*-nya menunjuk ke *head*.

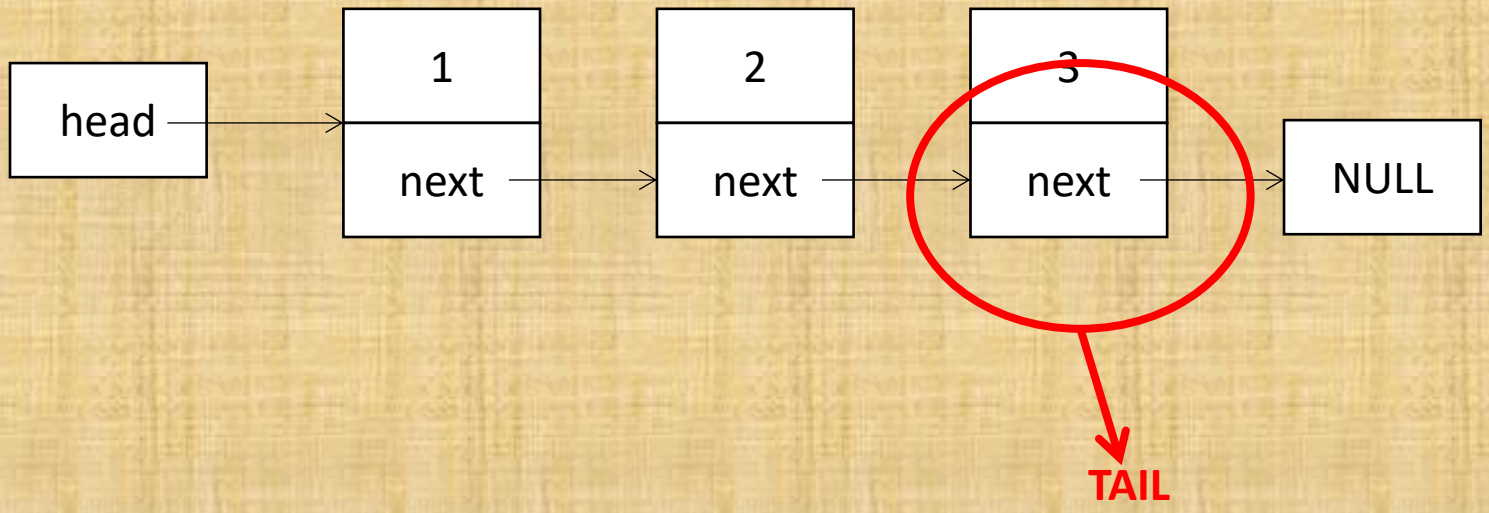
Operator **new** pada c++

- Menyediakan alokasi pada storage cell dalam memori untuk variabel pointer secara dinamis.
- Contoh:

```
int *a;
a = new int;
```

- Variabel pointer a menyimpan alamat memori baru yang telah disediakan untuk menyimpan nilai integer.

Contoh



Deklarasi pada c++

```
struct Node{
        int data;
        Node *next;
};
Node *n, *head, *tail, *x;
```

Penjelasan:

- Node dibuat dengan menggunakan struct/record yang berisi dua buah *field (data* dan **next)*.
- *data* digunakan untuk menyimpan nilai (misal integer).
- *next digunakan untuk menyimpan alamat (*pointer value*) dari node lain.
- **n* digunakan sebagai variabel pointer yang menunjuk pada node baru;
- **head* digunakan sebagai variabel pointer yang menunjuk pada node awal pada linked list.
- **tail* digunakan sebagai variabel pointer yang akan memberikan nilai NULL pada next pointer sebagai node akhir
- *x digunakan sebagai variabel pointer untuk pengaksesan data.

Operasi dalam Single Linked List

- Pembentukan Node Awal
- Penambahan Node
- Penghapusan Node
- Pengaksesan Node
- Pencarian Data

Pembentukan Node Awal

Algoritma:

1. n menunjuk pada node baru
2. n.data diberikan nilai.
3. tail = n
4. head = n
5. tail.next = NULL

Penambahan Node

- Penambahan di awal *linked list*.
- Penambahan di akhir *linked list*.
- Penambahan di tengah *linked list.*

Penambahan di Akhir

Algoritma:

1. n menunjuk **node baru**
2. n.data diberikan nilai.
3. tail.next = n
4. tail = n
5. tail.next = null

Penambahan di Awal

Algoritma:

1. n menunjuk **node baru**
2. n.data diberikan nilai
3. n.next = head
4. head = n

Penambahan di Tengah

Algoritma:

1. x = head
2. while (x.data $\neq$ j) x = x.next
3. n menunjuk pada **node baru**
4. n.data diberikan nilai
5. n.next = x.next
6. x.next = n

Penghapusan Node

- Penghapusan di awal
- Penghapusan di akhir
- Penghapusan di tengah

Penghapusan di Awal

Algoritma:

1. x = head
2. head = head.next
3. hapus node pada lokasi memori yang ditunjuk x.

Penghapusan di Akhir

Algoritma:

1. x = head
2. while (x.next ≠ tail){
   x = x.next;
   }
3. tail = x
4. hapus node pada lokasi memori yang ditunjuk x.next
5. tail.next = null

Penghapusan di Tengah

Algoritma:

1. buat variabel pointer dengan nama *temp=NULL.
2. x = head
3. while (x.data ≠ i){
   temp = x
   x = x.next
   }
4. temp.next = x.next
5. hapus node pada lokasi memori yang ditunjuk x.

Ket:
i adalah data field pada node yg akan dihapus
Algoritma ini dapat digunakan hanya jika nilai yang akan dihapus terdapat pada linked list dan bukan nilai dari node awal atau node akhir.

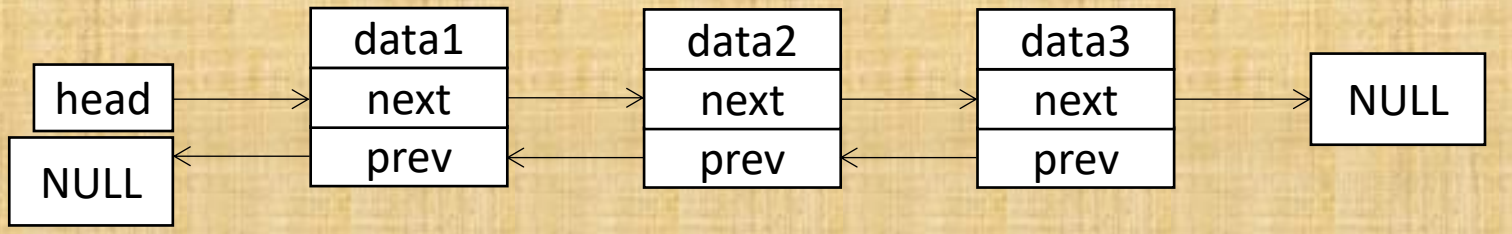
Pengaksesan Data

Algoritma:

1. x = head
2. while (x ≠ NULL){
   print (x.data)
   x = x.next
   }

Double Linked List

- Terdiri dari tiga *field* yakni ***data***, ***next pointer***, dan ***previous pointer***.
- Double linked list dibuat agar data pada sebuah linked list dapat diakses secara mundur.



Operasi dalam Double Linked List

- Pembentukan node awal
- Penambahan node
- Penghapusan node
- Pengaksesan node dari depan
- Pengaksesan node dari belakang

Deklarasi pada c++

```
struct Node{
        int data;
        Node *next;
        Node *prev;
};
node *n, *head, *tail, *x;
```

Penjelasan:

- Node dibuat dengan menggunakan struct/record yang berisi tiga buah *field (data, *next, dan *prev)*.
- *data* digunakan untuk menyimpan nilai (misal integer).
- *next digunakan untuk menyimpan alamat (*pointer value*) dari node setelahnya.
- *prev digunakan untuk menyimpan alamat (pointer value) dari node sebelumnya.
- **n* digunakan sebagai variabel pointer yang menunjuk pada node baru;
- **head* digunakan sebagai variabel pointer yang menunjuk pada node awal pada linked list.
- **tail* digunakan sebagai variabel pointer yang akan memberikan nilai NULL pada next pointer sebagai node akhir
- *x digunakan sebagai variabel pointer untuk pengaksesan data.

Operasi dalam Double Linked List

- Pembentukan node awal
- Penambahan node
- Penghapusan node
- Pengaksesan node dari depan
- Pengaksesan node dari belakang

Pembentukan Node Awal

Algoritma:

1. n menunjuk pada **node baru**
2. n.data diberikan nilai
3. n.prev = null
4. head = n
5. tail = n
6. tail.next = null

Penambahan di Akhir

Algoritma:

1. n menunjuk pada **node baru**
2. n.data diberikan nilai
3. n.prev = tail
4. tail.next = n
5. tail = n
6. tail.next = null

Penambahan di Awal

Algoritma:

1. n menunjuk pada **node baru**
2. n.data diberikan nilai
3. n.next = head
4. head.prev = n
5. n.prev = NULL
6. head = n

Penambahan di Tengah

Algoritma:

1. x = head
2. while(x.data ≠ data_setelah) {
   x = x.next
   }
3. n menunjuk pada **node baru**
4. n.data diberikan nilai
5. n.next = x.next
6. x.next = n
7. n.prev = x
8. x = n.next
9. x.prev = n

Pengaksesan Data dari Depan

Algoritma:

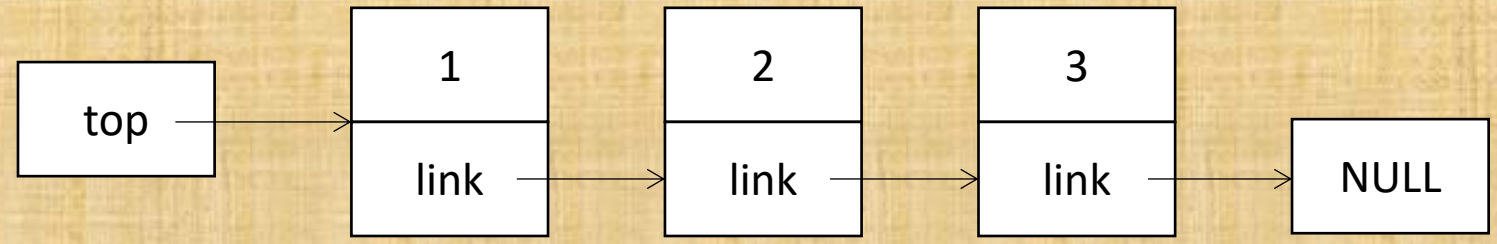
1. x = head
2. while (x ≠ NULL){
   print (x.data)
   x = x.next
   }

Pengaksesan dari Belakang

Algoritma:

1. x = tail
2. while (x ≠ NULL){
   print (x.data)
   x = x.prev
   }

Implementasi Linked List pada Stack



Deklarasi

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

node *top= NULL,*n,*x;
int jml = 0;
```

Penjelasan:

- ***node*** dibuat dengan menggunakan struct/record yang berisi 2 buah *field* ***(data, *link)***.
- ***data*** adalah variabel yg digunakan untuk menyimpan nilai (misal integer).
- ****link*** digunakan untuk menyimpan alamat (*pointer value*) dari node setelahnya.
- ****n*** digunakan sebagai variabel pointer yang menunjuk pada node baru.
- ****top*** digunakan sebagai variabel pointer yang menunjuk pada node paling atas pada stack.
- ****x*** digunakan sebagai variabel pointer untuk pengaksesan data.
- ***jml*** adalah variabel untuk menyimpan nilai jumlah data.

Push

Algoritma:

1. n menunjuk ke node baru;
2. n.data diberikan nilai;
3. n.link = top;
4. top = n;
5. jml = jml + 1;

Pop

Algoritma:

1. x = top;
2. x = x.link;
3. nilai_pop = top.data;
4. hapus node pada lokasi memori yang ditunjuk ***top***
5. top = x
6. jml = jml - 1

Tampilkan isi stack

Algoritma:

1. x = top
2. while (x != NULL){
   print(x.data)
   x = x->link
   }

Tugas Kelas A4

Buat algoritma operasi untuk double linked list:

- Penghapusan node di akhir
- Penghapusan node di awal
- Penghapusan node di tengah

Tugas dikirim ke : lutfibudi.ilmawan@umi.ac.id

Dengan subject : SD.A4.STB.TUGAS4

Deadline : Selasa, 21 Mei 2019

Contoh subject : SD.A1.13020160084.TUGAS4

Tugas Kelas B5

Buat algoritma operasi untuk double linked list:

- Penghapusan node di akhir
- Penghapusan node di awal
- Penghapusan node di tengah

Tugas dikirim ke : lutfibudi.ilmawan@umi.ac.id

Dengan subject : SD.B5.STB.TUGAS6

Deadline : Selasa, 22 Mei 2019

Contoh subject : SD.B5.13020160084.TUGAS6