

## Linked List: Gambaran Sederhana

Kontribusi: Taufik Abidin  
Wednesday, 13 December 2006

Linked List adalah struktur data yang berbeda dengan struktur data array. Linked list dapat dibayangkan seperti rantai yang bersambung satu sama lainnya. Tiap-tiap rantai (node) terhubung dengan pointer.

Gambar berikut memperlihatkan sebuah Linked List.

Sebelum kita membahas lebih jauh tentang Linked List, alangkah baiknya bila kita mengulang sedikit tentang struct dan typedef. Struct atau structure dalam bahasa pemrograman C/C++ digunakan untuk mendefinisikan tipe data yang memiliki anggota (member) bertipe tertentu. Berikut contoh dan cara mendeklarasi sebuah struct:

```
struct tgl {
    int hari;
    int bulan;
    int tahun;
} struct tgl Tanggal;
```

Bila divisualisasikan kira-kira sebagai berikut:

Contoh di atas memperlihatkan bagaimana mendeklarasi sebuah struct dengan nama struct tgl yang memiliki tiga member yaitu hari, bulan dan tahun yang bertipe int (integer). Kemudian, sebuah variabel Tanggal dideklarasikan bertipe struct tgl. Untuk mempersingkat dan menyederhanakan pendeklarasian sebuah struct, kata cadang typedef biasa digunakan. Sesuai namanya, typedef digunakan untuk mendefinisikan sebuah tipe data menjadi suatu alias tertentu. Perhatikan contoh berikut:

Dengan cara yang sama, pendeklarasian struct tgl di atas dapat disederhanakan menggunakan kata cadang typedef menjadi:

```
typedef struct tgl {
    int hari;
    int bulan;
    int tahun;
} Date;
Date Tanggal;
```

Single Linked List

Linked list dapat dianalogikan sebagai rantai yang terdiri dari beberapa node yang saling terkait. Dengan memegang node terdepan, maka node-node yang saling terkait lainnya dapat kita telesuri.

Dengan hanya mencatat atau memegang alamat dari alokasi data bertipe struct root pada sebuah variabel LL maka keberadaan node-node dalam linked list tersebut dapat diketahui. Bila data-data dalam node berupa bilangan bulat, maka struct yang diperlukan untuk membentuk linked list adalah sebagai berikut:

```
typedef struct node * NodePtr;
typedef struct node {
    int data;
    NodePtr next;
}Node;
typedef struct root {
    NodePtr head;
    unsigned si
}
Root LL;
```

### Penambahan Node Linked List

Bila setiap penambahan simpul pada linked list dilakukan pada bagian depan (paling dekat dengan head) maka kompleksitas yang diperlukan untuk menambah node baru dalam linked list konstan atau  $O(1)$ . Penambahan sebuah node dengan nilai 3 pada linked list di atas dapat divisualisasikan sebagai berikut:

### Penelusuran Node Linked List

Kompleksitas penelusuran (pencarian) node dalam linked list adalah linier atau  $O(n)$ . Hal ini disebabkan karena node-node dalam linked list disusun secara acak (tidak berurut). Seandainya pun simpul-simpul dalam linked list disusun secara berurut, metode pencarian biner (binary search) tetap tidak dapat dipergunakan. Ada 2 alasan mengapa pencarian biner tidak dapat digunakan:

1. Linked list tidak memiliki indeks seperti array, sehingga akses langsung ke node tertentu tidak dapat dilakukan. Untuk menuju ke node tertentu, proses pemeriksaan tetap dimulai dari node head (node terdepan). Oleh karena itu proses pencarian selalu berjalan secara linier.
2. Tidak dapat membagi linked list menjadi 2 bagian yang sama besar seperti saat membagi array menjadi 2 bagian bila metode pencarian biner diaplikasikan pada array terurut.

### Penghapusan Node Linked List

Sebelum proses penghapusan dilakukan, pencarian node yang ingin dihapus harus terlebih dahulu dilakukan. Bila node yang ingin dihapus ditemukan, suatu hal yang selalu harus diperhatikan adalah bagaimana mengeluarkan node tersebut dari linked list dan kemudian menyambung kembali linked list kembali. Kompleksitas menghapus sebuah node dalam linked list adalah  $O(n)$ . Perhatikan gambar berikut ini:

### Implementasi Linked List

```
/* linkedlist.h: header file */ typedef struct node { int data; NodePtr next; }Node;
typedef struct list { NodePtr head; unsigned size; }List; void initList(List *); int addList(List *, int); void
displayList(List *); void freeList(List *); /* linkedlist.c: implementasi method dilakukan dalam file ini */ #include
"linkedlist.h" #include <stdlib.h> #include <stdio.h> void initList(List * lptr) { lptr->head = NULL; lptr->size = 0; } int
addList(List * lptr, int data) { NodePtr new; new = malloc(sizeof(Node)); if(new == NULL) { printf("Error dalam
membuat alokasi memori\n"); return 0; } new->data = data; new->next = lptr->head; lptr->head = new; lptr-
>size++; return 1; } void displayList(List * lptr) { NodePtr current = lptr->head; printf("\n"); while(current != NULL) {
printf("%d -> ", current->data); current = current->next; } printf("null\n"); } void freeList(List * lptr) { NodePtr
next=lptr->head; NodePtr current=next; while(current != NULL) { next = current->next; free(current); current = next;
} } /* main.c */ #include <stdlib.h> #include <stdio.h> #include "linkedlist.h" int main(void) { int i, n, data; List
LL; initList(&LL); /* Buat initial linked list */ printf("Jumlah simpul = "); scanf("%d", &n); /* Input data simpul */
for(i=0; i<n; i++) { printf("Data = "); scanf("%d", &data); addList(&LL, data); } displayList(&LL); freeList(&LL);
return EXIT_SUCCESS;
```

Compile program di atas dengan pernyataan berikut (diasumsikan OS linux digunakan):

```
%> gcc -Wall -pedantic -g -o mylist main.c linkedlist.c
```

Hasil kompilasi di atas akan membuat sebuah objek mylist (yang dibangkitkan dari dua file main.c dan linkedlist.c). Kemudian objek mylist tersebut dapat dijalankan melalui command line. Selamat mencoba.