Program Studi Teknik Elektro ITB
Nama Kuliah (Kode) : Praktikum Pemecahan Masalah dengan C (EL2208)
Tahun / Semester : 2023-2024 / Genap
Modul : 6 – Linked List
Nama Asisten / NIM : _____
Nama Praktikan / NIM : Pradigta Hisyam Ramadhan / 18322008

Tugas Pendahuluan

1. Pengertian Linked List
Linked list merupakan salah satu jenis struktur data yang penting dalam dunia pemrograman. Linked list terdiri dari node-node dengan masing-masing node-nya menyimpan data dan referensi (link) dari urutan node selanjutnya [1].
Perbedaan antara linked list dan array [2]:

a. Array disimpan di lokasi yang bersebelahan, sedangkan linked list tidak
b. Ukuran dari array tetap (statis), sedangkan linked list dapat berubah-ubah (dinamis)
c. Pada array, memori dialokasikan ketika kode di-*compile*. Sedangkan pada linked list, memori dialokasikan ketika program sedang berjalan
d. Array membutuhkan memori yang lebih sedikit daripada linked list karena pada linked list dibutuhkan alokasi memori untuk menyimpan data dan alamat dari node selanjutnya.
e. Elemen dari array dapat diakses dengan lebih mudah ketimbang linked list, karena pada linked list elemen harus diakses satu-persatu dari kesuluran node.
f. Operasi untuk menyisipkan dan menghapus elemen lebih mudah diaplikasikan oleh linked list ketimbang menggunakan array

2. Salah satu keuntungan dari implementasi linked-list ketimbang array adalah ketika kita akan melakukan insersi ataupun delesi dari suatu data. Selain itu, linked-list membolehkan kita untuk dapat membuat struktur data dari beberapa gabungan tipe-data bawaan dari suatu bahasa. Berikut ini contoh implementasi **delesi** suatu data:

Implementasi menggunakan linked list:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Mahasiswa {
    char nama[50];
    int nim;
    float ipk;
    struct Mahasiswa *next; // Pointer untuk mengakses alamat selanjutnya dari
linked list
} Mahasiswa;

void hapusMahasiswa(Mahasiswa **head, int nim) {
    Mahasiswa *current = *head;
    Mahasiswa *prev = NULL;

    // Mencari node dengan NIM yang sesuai dengan input pengguna
    while (current != NULL && current->nim != nim) {
        prev = current;
        current = current->next;
```

```
    }

    // Apabila node dengan NIM tersebut ketemu
    if (current != NULL) {
        // Apabila node adalah head/elemen pertama linked list
        if (prev == NULL) {
            *head = current->next;
        }
        // Apabila node yang dihapus bukan head
        else {
            prev->next = current->next;
        }
        // Bebaskan memori dari node yang dihapus
        free(current);
        printf("Mahasiswa dengan NIM %d berhasil dihapus!\n", nim);
    }
    // Apabila node yang mengadung NIM yang dimasukan pengguna tida ada dalam
liked list
    else {
        printf("Mahasiswa dengan NIM %d tidak ada!\n", nim);
    }
}

int main(void) {
    Mahasiswa *head = NULL;

    Mahasiswa *mahasiswa1 = malloc(sizeof(Mahasiswa));
    strcpy(mahasiswa1->nama, "John");
    mahasiswa1->nim = 12345;
    mahasiswa1->ipk = 3.75;
    mahasiswa1->next = NULL;
    head = mahasiswa1;

    Mahasiswa *mahasiswa2 = malloc(sizeof(Mahasiswa));
    strcpy(mahasiswa2->nama, "Alice");
    mahasiswa2->nim = 54321;
    mahasiswa2->ipk = 3.95;
    mahasiswa2->next = NULL;
    mahasiswa1->next = mahasiswa2;

    // Meminta masukan dari pengguna untuk menghapus mahasiswa yang akan dihapus
berdasarkan NIM
    int hapusNIM;
    printf("Masukkan NIM mahasiswa yang ingin dihapus: ");
    scanf("%d", &hapusNIM);

    // Memanggil fungsi
    hapusMahasiswa(&head, hapusNIM);

    // Menampilkan data mahasiswa setelah menghapus data
    printf("Data Mahasiswa setelah penghapusan:\n");
```

```
    Mahasiswa *current = head;
    while (current != NULL) {
        printf("Nama: %s, NIM: %d, IPK: %.2f\n", current->nama, current->nim,
current->ipk);
        current = current->next;
    }

    // Bebaskan memori
    current = head;
    while (current != NULL) {
        Mahasiswa *temp = current;
        current = current->next;
        free(temp);
    }

    return 0;
}
```

Implementasi menggunakan array:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_MAHASISWA 100 // Jumlah maksimum Mahasiswa

typedef struct Mahasiswa {
    char nama[50];
    int nim;
    float ipk;
} Mahasiswa;

void hapusMahasiswa(Mahasiswa array[], int *size, int nim) {
    int found = 0;
    for (int i = 0; i < *size; i++) {
        if (array[i].nim == nim) {
            found = 1;
            for (int j = i; j < *size - 1; j++) {
                array[j] = array[j + 1];
            }
            (*size)--;
            printf("Mahasiswa dengan NIM %d berhasil dihapus!\n", nim);
            break;
        }
    }
    if (!found) {
        printf("Mahasiswa dengan NIM %d tidak ada!\n", nim);
    }
}

int main(void) {
```

```
    Mahasiswa array[MAX_MAHASISWA];
    int size = 0; // Inisiasi awal jumlah mahasiswa

    // Sampel data mahasiswa dalam bentuk array
    strcpy(array[size].nama, "John");
    array[size].nim = 12345;
    array[size].ipk = 3.75;
    size++;

    strcpy(array[size].nama, "Alice");
    array[size].nim = 54321;
    array[size].ipk = 3.95;
    size++;

    // Meminta masukan dari pengguna untuk menghapus mahasiswa yang akan dihapus
berdasarkan NIM
    int nimToDelete;
    printf("Masukkan NIM mahasiswa yang ingin dihapus: ");
    scanf("%d", &nimToDelete);

    // Memanggil fungsi untuk menghapus mahasiswa
    hapusMahasiswa(array, &size, nimToDelete);

    // Menampilkan array yang tersisa
    printf("Data Mahasiswa setelah penghapusan:\n");
    for (int i = 0; i < size; i++) {
        printf("Nama: %s, NIM: %d, IPK: %.2f\n", array[i].nama, array[i].nim,
array[i].ipk);
    }

    return 0;
}
```

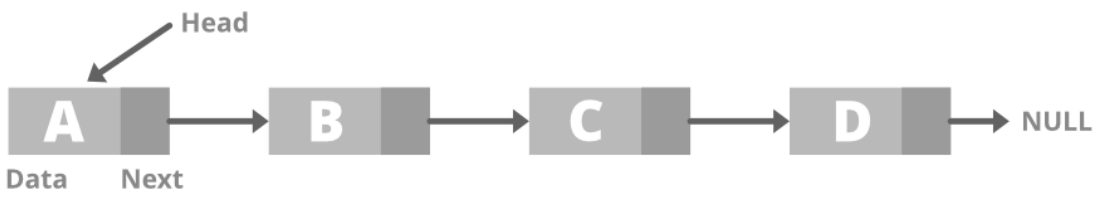
Output yang dihasilkan (kalimat yang digarisbawahi adalah masukan dari pengguna):

Masukkan NIM mahasiswa yang ingin dihapus: 12345
Mahasiswa dengan NIM 12345 berhasil dihapus!
Data Mahasiswa setelah penghapusan:
Nama: Alice, NIM: 54321, IPK: 3.95

Masukkan NIM mahasiswa yang ingin dihapus: 18322008
Mahasiswa dengan NIM 18322008 tidak ada!
Data Mahasiswa setelah penghapusan:
Nama: John, NIM: 12345, IPK: 3.75
Nama: Alice, NIM: 54321, IPK: 3.95

3. Jenis-jenis linked list [3]:

a. Singly Linked list: Merupakan jenis linked list yang paling sederhana. Setiap nodenya berisi beberapa data beserta sebuah pointer yang merujuk ke alamat node berikutnya dengan tipe data yang sama.



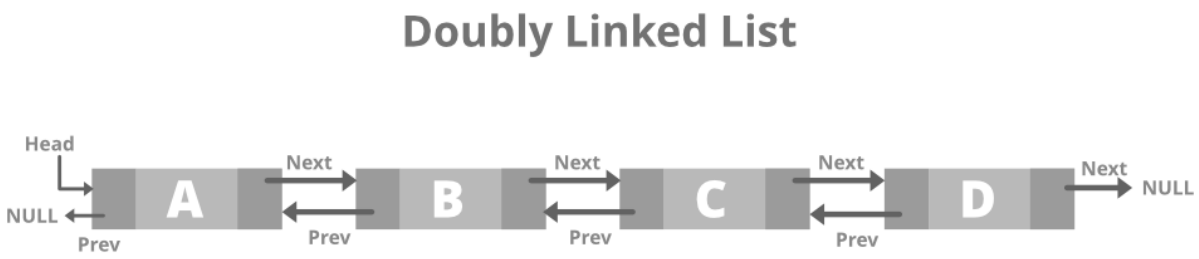
Gambar 3.1 Ilustrasi Singly Linked List [3]

Contoh implementasi Singly Linked List [4]:

```
#include <stdio.h>
#include <stdlib.h>

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

int main(void){

    /*Inisialisasi node*/
    struct node*head;
    struct node*one = NULL;
    struct node*two = NULL;
    struct node*three = NULL;

    /*Alokasi memori*/
    one = malloc(sizeof(struct node));
    two = malloc(sizeof(struct node));
    three = malloc(sizeof(struct node));

    /*Assign nilai tiap node*/
    one->data = 1111;
    two->data = 2222;
    three->data = 3333;

    /*Hubungkan node*/
    one->next = two;
    two->next = three;
    three->next = NULL;

    /*Simpan alamat dari node pertama ke dalam head*/
    head = one;

    /*Iterasi linked list untuk tampilkan semua data*/
    struct node *current = head;
    while (current != NULL) {
        printf("%d ", current->data);
```

```
        current = current->next;
    }
    printf("\n");

    free(one);
    free(two);
    free(three);

    return 0;
}
```

b. Doubly Linked List: Merupakan linked list dua arah karena memiliki pointer yang merujuk ke alamat node sebelumnya dan selanjutnya.



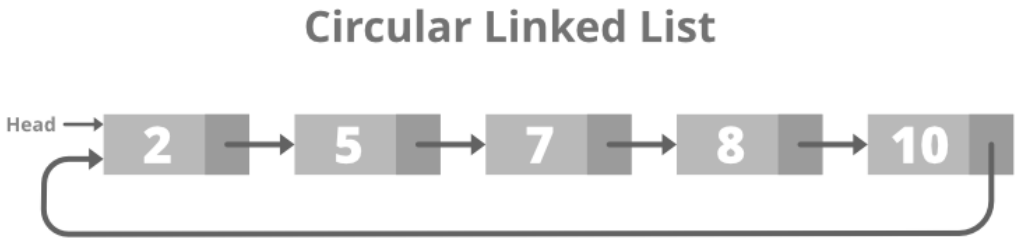
Gambar 3.2 Ilustrasi Doubly Linked List [3]

Contoh implementasi Doubly Linked List [4]:

```
#include <stdio.h>
#include <stdlib.h>

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

int main(void){

    /*Inisialisasi node*/
    struct node*head;
    struct node*one = NULL;
    struct node*two = NULL;
    struct node*three = NULL;

    /*Alokasi memorii*/
    one = malloc(sizeof(struct node));
    two = malloc(sizeof(struct node));
    three = malloc(sizeof(struct node));

    /*Assign nilai tiap data*/
    one->data = 1111;
    two->data = 2222;
    three->data = 3333;
```

```
    /*Hubungkan node*/
    one->next = two;
    one->prev = NULL;

    two->next = three;
    two->prev = one;

    three->next = NULL;
    three->prev = two;

    /*Simpan alamat node pertama ke dalam head*/
    head = one;

    /*Iterasi untuk menampilkan semua data pada linked list*/
    struct node *current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");

    free(one);
    free(two);
    free(three);

    return 0;
}
```

c. Circular Linked List: Merupakan jenis linked list yang node terakhirnya memiliki pointer yang merujuk ke alamat node pertama (head) dari linked list.



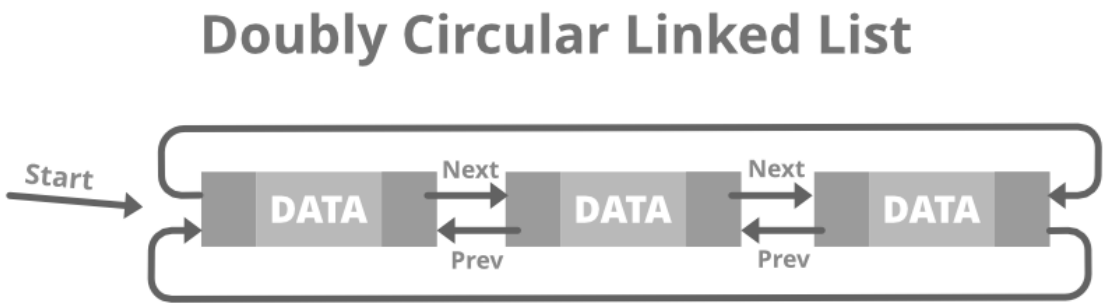
Gambar 3.2 Ilustrasi Circular Linked List [3]

Contoh implementasi Circular Linked List [4]:

```
#include <stdio.h>
#include <stdlib.h>

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

```c
int main(void){

    /*Inisialisasi node*/
    struct node*head;
    struct node*one = NULL;
    struct node*two = NULL;
    struct node*three = NULL;

    /*Alokasi memori*/
    one = malloc(sizeof(struct node));
    two = malloc(sizeof(struct node));
    three = malloc(sizeof(struct node));

    /*Assign niai dari data*/
    one->data = 1111;
    two->data = 2222;
    three->data = 3333;

    /*Hubungkan node*/
    one->next = two;
    two->next = three;
    three->next = one;

    /*Simpan alamat dari node pertama ke dalam head*/
    head = one;

    /*Iterasi untuk menampilkan semua data pada linked list*/
    struct node *current = one;
    do {
        printf("%d ", current->data);
        current = current->next;
    } while (current != one);
    printf("\n");


    free(one);
    free(two);
    free(three);

    return 0;
}
```

d. Doubly Circular Linked List: Merupakan linked list sirkular dua arah. Sama seperti doubly linked list, doubly circular linked list memiliki pointer yang merujuk ke alamat node sebelumnya dan selanjutnya. Perbedaannya dari doubly linked list biasa adalah alamat *previous* dari node pertama bukanlah NULL, tetapi alamat dari data terakhir.Selain itu, alamat *next* dari node terakhir juga bukanlah NULL, tetapi alamat dari node pertama (head).



Gambar 3.4 Ilustrasi Doubly Circular Linked List

Contoh implementasi Doubly Circular Linked List [4]:

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
    struct node *prev; // Add a pointer to the previous node
};

int main(void){

    /* Inisialisasi node */
    struct node *head = NULL;
    struct node *one = NULL;
    struct node *two = NULL;
    struct node *three = NULL;

    /* Alokasi memori */
    one = malloc(sizeof(struct node));
    two = malloc(sizeof(struct node));
    three = malloc(sizeof(struct node));

    /* Assign nilai dari data */
    one->data = 1111;
    two->data = 2222;
    three->data = 3333;

    /* Hubungkan node */
    one->next = two;
    two->next = three;
    three->next = one;

    one->prev = three; // Hubungkan node terakhir dengan node pertama
    two->prev = one;
    three->prev = two;

    /* Simpan alamat dari node pertama ke dalam head */
```

```
    head = one;

    /* Iterasi untuk menampilkan semua data pada linked list */
    struct node *current = head;
    do {
        printf("%d ", current->data);
        current = current->next;
    } while (current != head);
    printf("\n");

    free(one);
    free(two);
    free(three);

    return 0;
}
```

Soal Pemrograman

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Mahasiswa{
    char nama[50];
    int nim;
    float ipk;
    struct Mahasiswa *next; // Pointer ke node berikutnya dalam linked list
}Mahasiswa;

void tambahMahasiswa(Mahasiswa **head, char addNama[50], int addNim, float
addIpk){
    // Alokasi memori untuk list baru
    Mahasiswa *mahasiswa_baru = (Mahasiswa*)malloc(sizeof(Mahasiswa));
    if (mahasiswa_baru == NULL){
        printf("Alokasi memori gagal!\n");
        exit(1);
    }

    strcpy(mahasiswa_baru->nama, addNama);
    mahasiswa_baru->nim = addNim;
    mahasiswa_baru->ipk = addIpk;
    mahasiswa_baru->next = NULL;

    // Inisiasi data pada linked list apabila tidak ada head
    if(*head == NULL){
        *head = mahasiswa_baru;
    }
    // Apabila head sudah terisi
    else{
        Mahasiswa *temp = *head;
        while(temp->next != NULL){
```

```
            temp = temp->next;
        }
        temp->next = mahasiswa_baru;
    }

}

void hapusMahasiswa(Mahasiswa **head, int nim){
    Mahasiswa *current = *head;
    Mahasiswa *prev = NULL;

    // mencari mahasiswa pada linked list berdasarkan nim yang diberikan
    while (current != NULL && current->nim != nim){
        prev = current;
        current = current->next;
    }

    // Apabila node ditemukan
    if (current != NULL){
        // Node yang dihapus adalah head
        if (prev == NULL){
            *head = current->next;
        }
        // Apabila node yang dihapus bukan head
        else{
            prev->next = current->next;
        }
        // bebaskan memori untuk node current
        free(current);
        printf("Mahasiswa dengan NIM %d berhasil dihapus!\n", nim);
    }
    else{
        printf("Mahasiswa dengan NIM %d tidak ada!\n", nim);
    }
}

void tampilkanMahasiswa(Mahasiswa *head){
    Mahasiswa *current = head;
    printf("Data mahasiswa Universitas B\n");
    while(current != NULL){
        printf("NIM: %d, Nama: %s, IPK: %.2f\n", current->nim, current->nama,
current->ipk);
        current = current->next;
    }
}

int main(void){
    Mahasiswa *head = NULL;
    int opsi, nim;
    float ipk;
    char nama[50];
```

```c
do{
    printf("-------------------------------------------");
    printf("\nProgram Pendataan Mahasiswa Universitas B\n");
    printf("1. Tambah Mahasiswa\n");
    printf("2. Hapus data mahasiswa berdasarkan NIM\n");
    printf("3. Tampilkan data mahasiswa\n");
    printf("4. Keluar program\n");
    printf("Masukkan menu yang ingin dipilih: ");
    scanf("%d", &opsi);

    switch(opsi){
    case 1:
        printf("-------------------------------------------\n");
        // Mendapatkan nama, nim, dan IPK mahasiswa
        printf("Masukkan nama mahasiswa: ");
        getchar();  // mengambil karakter newline dari input sebelumnya
        fgets(nama, 50, stdin);
        nama[strcspn(nama, "\n")] = '\0';

        printf("Masukkan NIM mahasiswa: ");
        scanf("%d", &nim);

        printf("Masukkan IPK mahasiswa (koma menggunakan titik): ");
        scanf("%f", &ipk);

        tambahMahasiswa(&head, nama, nim, ipk);
        break;

    case 2:
        printf("-------------------------------------------\n");
        // Mendapatkan NIM mahasiswa
        printf("Masukkan NIM mahasiswa yang ingin dihapus datanya: ");
        scanf("%d", &nim);

        hapusMahasiswa(&head, nim);
        break;
    case 3:
        printf("-------------------------------------------\n");
        tampilkanMahasiswa(head);
        break;
    case 4:
        printf("-------------------------------------------\n");
        printf("Keluar dari program.");
        break;
    default:
        printf("-------------------------------------------\n");
        printf("Silakan masukkan opsi menu diantara angka 1-4!\n");
        break;
    }
```

```
    } while (opsi != 4);


    return 0;
}
```

Referensi

[1] GfG, “Linked List Data Structure,” *GeeksforGeeks*, Apr. 10, 2024. https://www.geeksforgeeks.org/data-structures/linked-list/ (accessed Apr. 15, 2024).

[2] GfG, “Linked List vs Array,” *GeeksforGeeks*, Jul. 10, 2023. https://www.geeksforgeeks.org/linked-list-vs-array/ (accessed Apr. 15, 2024).

[3] GfG, “Types of Linked List,” *GeeksforGeeks*, Jan. 29, 2024. https://www.geeksforgeeks.org/types-of-linked-list/(accessed Apr. 15, 2024).

[4] “Linked List Data Structure.” https://www.programiz.com/dsa/linked-list (accessed Apr. 15, 2024).