LAPORAN PRAKTIKUM STRUKTUR DATA
MODUL VII

Laporan ini disusun untuk memenuhi Tugas Mata Kuliah

Praktikum struktur data

Disusun Oleh :

M HAMKA ZAINUL ARDHI

NIM: 2311103156

PROGRAM STUDI S1 SISTEM INFORMASI

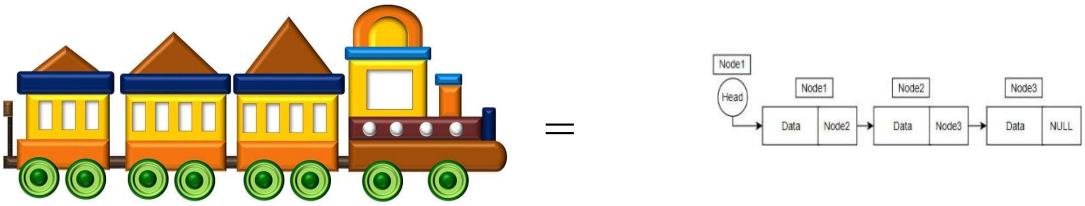
FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Anda tau kereta ? atau senang naik kereta ketika berpergian? Seperti yang kita ketahui sebuah kereta yang terdiri dari gerbong-gerbong yang saling terhubung satu sama lain. Setiap gerbong memiliki informasi yang disimpan di dalamnya dan tautan ke gerbong sebelumnya dan sesudahnya. Inilah cara kerja linked list:

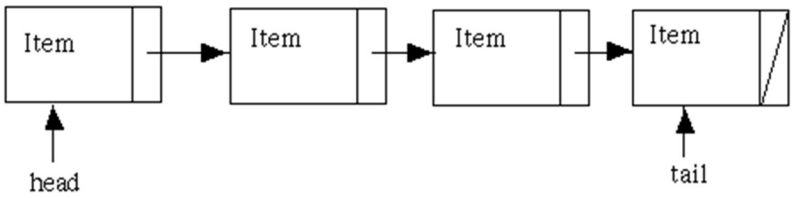


Linked list adalah struktur data linier yang terdiri dari sejumlah simpul (node) yang saling terhubung melalui referensi atau pointer. Setiap simpul dalam linked list menyimpan data dan memiliki sebuah pointer yang menunjuk ke simpul berikutnya dalam urutan linear.

Konsep dasar dari linked list adalah bahwa setiap simpul mengandung dua bagian utama: data dan pointer. Data mewakili informasi yang ingin disimpan, misalnya bilangan, teks, atau objek lainnya. Pointer, juga disebut sebagai “next pointer,” menunjuk ke simpul berikutnya dalam urutan.

Perbedaan utama antara linked list dengan struktur data lainnya, seperti array, adalah kemampuannya untuk mengalokasikan ruang secara dinamis saat program berjalan. Hal ini memungkinkan penggunaan memori yang efisien, karena linked list dapat tumbuh atau menyusut sesuai kebutuhan.

Dalam linked list, simpul pertama disebut sebagai “head” atau “kepala,” sedangkan simpul terakhir dalam urutan disebut sebagai “tail” atau “ekor.” Ketika tail memiliki nilai pointer yang menunjuk ke null, itu menandakan akhir dari linked list.



Ada beberapa macam Linked List, yaitu :

1. Single Linked List
2. Double Linked List
3. Circular Linked List
4. Multiple Linked List

1. Single Linked List

Single Linked List merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya. Biasanya field pada tail menunjuk ke NULL.

Contoh Code nya

```
struct Mahasiswa{
    char nama[25];
    int usia;
    struct Mahasiswa *next;
}*head,*tail;
```

2. Double Linked List

Double Linked List merupakan suatu linked list yang memiliki dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya. Setiap head dan tailnya juga menunjuk ke NULL.

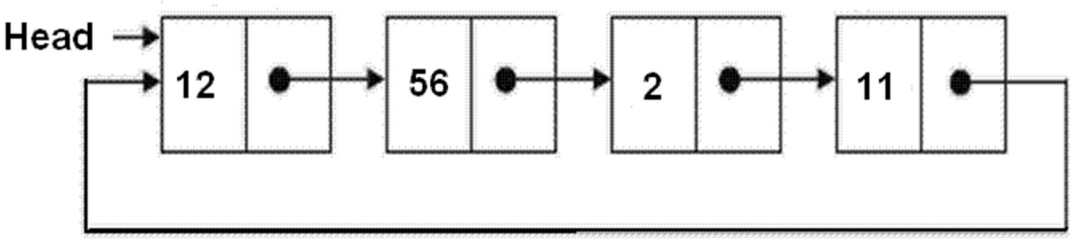
Contoh Code nya

```
struct Mahasiwa{
    char nama[25];
    int usia;
    struct Mahasiswa *next,*prev;
}*head,*tail;
```

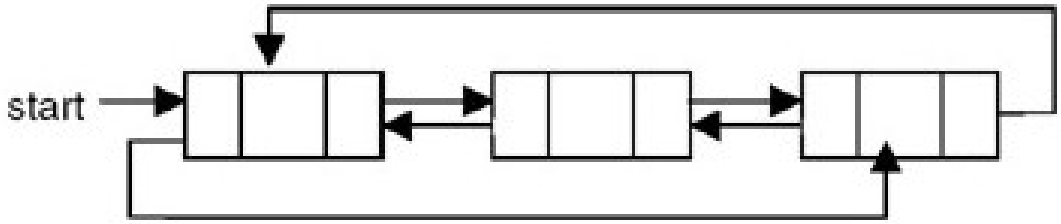
3. Circular Linked List

Circular Linked List merupakan suatu linked list dimana tail (node terakhir) menunjuk ke head (node pertama). Jadi tidak ada pointer yang menunjuk NULL. Ada 2 jenis Circular Linked List, yaitu

- Singel Circular Linked List



- Double Circular Linked List



4. Multi Linked list

Multi-linked lists adalah struktur data di mana setiap simpul memiliki lebih dari satu tautan ke simpul lainnya. Konsep ini memungkinkan untuk memiliki lebih dari satu arah navigasi antara simpul-simpul dalam daftar. Ini berbeda dari linked list biasa di mana setiap simpul hanya memiliki satu tautan ke simpul berikutnya.

- Operasi Dasar

Linked list mempunyai struktur coding yang Mana implementasi linked list melibatkan beberapa struktur coding dasar untuk membuat dan mengelola linked list berikut adlah oprasi dasar linked list seperti penambahan data head, penmambahan data belakang dan menghapus data

1. Penambahan data depan (head)

```
void tambah_depan(string data_user) {
    buat_simpul(data_user);
    if (list_kosong() == true) {
        head = simpul_baru;
        tail = simpul_baru;
        simpul_baru->next = head;
    } else {
        simpul_baru->next = head;
        head = simpul_baru;
        tail->next = simpul_baru;
    }
}
```

2. Penambahan data Belakang (tail)

```
void tambah_belakang(string data_user) {
    buat_simpul(data_user);
    if (list_kosong() == true) {
        head = simpul_baru;
        tail = simpul_baru;
        simpul_baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        tail->next = simpul_baru;
        simpul_baru->next = head;
    }
```

3. Penghapusan Data Belakang (Tail)

```
void hapus_belakang() {
    if (list_kosong() == true) {
        cout << "Tidak ada data" << endl;
    } else {
        melist *helper;
        helper = head;
        tail = head;
        if (helper->next == head) {
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            while (helper->next != head) {
                helper = helper->next;
            }
            while (tail->next != helper) {
                tail = tail->next;
            }
            tail->next = head;
            helper->next = NULL;
            delete helper;
        }
    }
}
```

5. Penghapus Data Depan (Head)

```
void hapus_depan() {
    if (list_kosong() == true) {
        cout << "Tidak ada data" << endl;
    } else {
        melist *helper;
        helper = head;
        tail = head;
        if (helper->next == head) {
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            while (tail->next != helper) {
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
            helper->next = NULL;
            delete helper;
        }
    }
}
```

II. Pembahasan Tugas guided

- **GUIDED I**

membuat program yang menerapkan struktur data single linkedlist non circular, yang terdapat fungsi tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data mengunakan bahasa pemograman c++.

SOURCE CODE

```
#include <iostream>
using namespace std;

struct linkedlist {
    string data;
    linkedlist* next;
};

linkedlist* head = nullptr;
linkedlist* tail = nullptr;

void displayData() {
    if (head == nullptr) {
        cout << "Tidak ada data" << endl;
    } else {
        linkedlist* helper = head;
        while (helper != nullptr) {
            cout << helper->data << " ";
            helper = helper->next;
        }
        cout << endl;
    }
}

void initialize() {
    head = nullptr;
    tail = nullptr;
}

bool isListEmpty() {
    return (head == nullptr && tail == nullptr);
}

void addBack(string data) {
    linkedlist* node = new linkedlist{data, nullptr};
    if (isListEmpty()) {
        head = node;
        tail = node;
    } else {
        tail->next = node;
        tail = node;
    }
}
```

```
void addFront(string data) {
    linkedlist* node = new linkedlist{data, nullptr};
    if (isListEmpty()) {
        head = node;
        tail = node;
    } else {
        node->next = head;
        head = node;
    }
}

void removeBack() {
    if (isListEmpty()) {
        cout << "Tidak ada simpul" << endl;
    } else {
        if (head == tail) {
            delete head;
            head = nullptr;
            tail = nullptr;
        } else {
            linkedlist* helper = head;
            while (helper->next != tail) {
                helper = helper->next;
            }
            delete tail;
            tail = helper;
            tail->next = nullptr;
        }
    }
}

void removeFront() {
    if (isListEmpty()) {
        cout << "Tidak ada simpul" << endl;
    } else {
        linkedlist* helper = head;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        } else {
            head = head->next;
        }
        delete helper;
    }
}

int main() {
    initialize();
    addFront("Rabu");
    addBack("Kamis");
    addFront("Selasa");
    removeBack();
```

```
    displayData();
    return 0;
}
```

Hasil Program

```
PS D:\Praktikum Strukdat> cd "d:\Praktikum Strukdat\Week7\" ;
Selasa Rabu
PS D:\Praktikum Strukdat\Week7>
```

Program di atas adlah single linkedlist non-circular, yang terdapat fungsi tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data. Kita bisa melihat di int main () ada add front, add back, remove back, display data yang masing masing telah diisi oleh naman nama hari, untuk pertama tama diisi oleh ‘rabu’ untuk bagian head nya lalu bagian tail nya di isi dengan ‘kamis’dan sekali lagi penambahan data lagi di depan yaitu selasa namun perintah selanjuta nya adalah menghapus data yang ada di belakang yaitu ‘kamis’ dan Ketika ditampilkan lewat pernytaan ‘display data’ yang tampil hanya lah ‘rabu selasa’ karna data belkang sudah di hapus
Ini lah yang dinamakan linked list bisa menambakan dan menghapus data dimanapun kita mau didepan ditengah atau bahkan dibelakang ini bertujuan untuk menunjukkan konsep-konsep dasar dari pengelolaan linked list dalam code di atas

GUIDED II

Seperti hal nya guided 1 program ini bertujuan sama sama membuat linked list singel namun dengan versi circular yang berarti pointer nextnya menunjuk pada dirinya sendiri. Jika Single Linked List tersebut terdiri dari beberapa node, maka pointer next pada node terakhir akan menunjuk ke node terdepannya.

SOURCE CODE

```
#include <iostream>
using namespace std;

struct NodeList {
    string data;
    NodeList* next;
};

NodeList *head = nullptr;
NodeList *tail = nullptr;

void inisialisasi() {
    head = nullptr;
    tail = nullptr;
}

bool listKosong() {
    return (head == nullptr);
}

void buatSimpul(string dataMasukkan) {
    NodeList* simpulBaru = new NodeList;
    simpulBaru->data = dataMasukkan;
    simpulBaru->next = nullptr;

    if (listKosong()) {
        head = tail = simpulBaru;
        simpulBaru->next = head;
    } else {
        tail->next = simpulBaru;
        tail = simpulBaru;
        tail->next = head;
    }
}

void tambahBelakang(string dataUser) {
    buatSimpul(dataUser);
}

void tambahDepan(string dataUser) {
    NodeList* simpulBaru = new NodeList;
    simpulBaru->data = dataUser;
```

```
    simpulBaru->next = nullptr;

    if (listKosong()) {
        head = tail = simpulBaru;
        simpulBaru->next = head;
    } else {
        simpulBaru->next = head;
        head = simpulBaru;
        tail->next = head;
    }
}

void hapusBelakang() {
    if (listKosong()) {
        cout << "Tidak ada data" << endl;
    } else if (head == tail) {
        delete head;
        head = tail = nullptr;
    } else {
        NodeList *current = head;
        while (current->next != tail) {
            current = current->next;
        }
        delete tail;
        tail = current;
        tail->next = head;
    }
}

void hapusDepan() {
    if (listKosong()) {
        cout << "Tidak ada data" << endl;
    } else if (head == tail) {
        delete head;
        head = tail = nullptr;
    } else {
        NodeList* temp = head;
        head = head->next;
        tail->next = head;
        delete temp;
    }
}

void tampilList() {
    if (listKosong()) {
        cout << "Tidak ada data" << endl;
    } else {
        NodeList* current = head;
        do {
            cout << current->data << " ";
            current = current->next;
        } while (current != head);
```

```
        cout << endl;
    }
}

int main() {
    inisialisasi();
    tambahBelakang("Rabu");
    tambahBelakang("Kamis");
    tambahBelakang("Jum'at");
    tambahDepan("Selasa");
    tampilList();
    hapusBelakang();
    tampilList();
    hapusDepan();
    tampilList();
    return 0;
```

Hasil Program

```
PS D:\Praktikum Strukdat> cd "d:\Praktikum Strukdat\Week7\" ;
Selasa Rabu Kamis Jum'at
Selasa Rabu Kamis
Rabu Kamis
PS D:\Praktikum Strukdat\Week7>
```

Sekilas untuk tujuan dan pengoprasian program di guided 1 dan guided 2 adalah sama karana sama sama menerapkan struktur data single linkedlist yang terdapat fungsi tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data mengunakan bahasa pemograman c++.

Namun perbedaan dari kedua nya adalah jika guided 1 adalah linkedlist non circular dan guided 2 adalah linkedlist circular yang artinya perbedaan nya terletak pada struktur kode nya masing masing dan mengatur dan memanipulasi node. Struktur kode untuk linked list cirkular memerlukan penanganan khusus untuk memastikan bahwa list selalu tertutup (circular), khususnya dalam mengatur node terakhir untuk selalu menunjuk ke head. Ini menambah kompleksitas dalam pengelolaan node terutama pada penambahan dan penghapusan node.

UNGUIDED

1. Unguided 1

memodifikasi tugas guided I dan 2 sehingga menjadi program yang menyajikan tampilan pilihan menu, tambah depan, tambah belakang, hapus depan, hapus belakang, tampil data linkedlist, dan keluar, yang dapat dipilih oleh user/pengguna.

SOURCE CODE

```
#include <iostream>
using namespace std;

struct melist {
    string data;
    melist* next;
};

melist *head;
melist *tail;
melist *simpul_baru;

void inisialisasi() {
    head = NULL;
    tail = NULL;
}

bool list_kosong() {
    if (head == NULL) {
        return true;
    } else {
        return false;
    }
}

void buat_simpul(string dataMasukkan) {
    simpul_baru = new melist;
    simpul_baru->data = dataMasukkan;
    simpul_baru->next = NULL;
}

void tambah_belakang(string data_user) {
    buat_simpul(data_user);
    if (list_kosong() == true) {
        head = simpul_baru;
        tail = simpul_baru;
        simpul_baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        tail->next = simpul_baru;
        simpul_baru->next = head;
    }
```

```
}

void tambah_depan(string data_user) {
    buat_simpul(data_user);
    if (list_kosong() == true) {
        head = simpul_baru;
        tail = simpul_baru;
        simpul_baru->next = head;
    } else {
        simpul_baru->next = head;
        head = simpul_baru;
        tail->next = simpul_baru;
    }
}

void hapus_belakang() {
    if (list_kosong() == true) {
        cout << "Tidak ada data" << endl;
    } else {
        melist *helper;
        helper = head;
        tail = head;
        if (helper->next == head) {
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
            while (helper->next != head) {
                helper = helper->next;
            }
            while (tail->next != helper) {
                tail = tail->next;
            }
            tail->next = head;
            helper->next = NULL;
            delete helper;
        }
    }
}

void hapus_depan() {
    if (list_kosong() == true) {
        cout << "Tidak ada data" << endl;
    } else {
        melist *helper;
        helper = head;
        tail = head;
        if (helper->next == head) {
            head = NULL;
            tail = NULL;
            delete helper;
        } else {
```

```
            while (tail->next != helper) {
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
            helper->next = NULL;
            delete helper;
        }
    }
}

void tampil_list() {
    if (list_kosong() == true) {
        cout << "Tidak ada data" << endl;
    } else {
        tail = head;
        do {
            cout << tail->data << " ";
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
}

int main() {
    int pilih;
    string data_user;
    bool lanjut = true;
    do {
        cout << endl;
        cout << "MENU" <<endl;
        cout << "1. tambah_belakang" <<endl;
        cout << "2. tambah_depan" <<endl;
        cout << "3. hapus_belakang" <<endl;
        cout << "4. hapus_depan" <<endl;
        cout << "5. tampilkan list" <<endl;
        cout << "6. keluar" <<endl;
        cout << "pilih : ";
        cin >> pilih ;
        cout <<endl;

        switch (pilih) {

            case 1:
            cout << "tambah_belakang : " ;
            cin >> data_user;
            tambah_belakang(data_user);
            break;

            case 2:
            cout << "tambah_depan : " ;
            cin >> data_user;
```

```
            tambah_depan(data_user);
            break;

            case 3:
            hapus_belakang();
            cout << "data belakang telah terhapus" << endl;
            break;

            case 4:
            hapus_depan();
            cout << "data depan telah terhapus" << endl;
            break;

            case 5:
            cout << "daftar list" <<endl;
            tampil_list();
            break;

            case 6:
            cout << "terima kasih " ;
            lanjut = false;
            break;


        }
    } while (lanjut);

    }
```

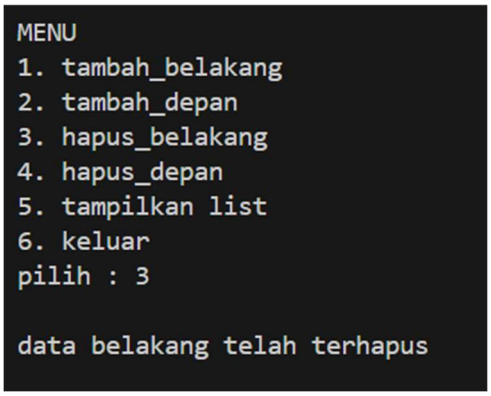
```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 1

tambah_belakang : rabu
```

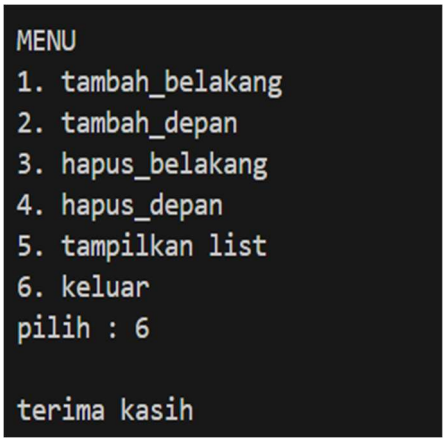
```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 2

tambah_depan : kamis
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 5

daftar list
kamis rabu
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 5

daftar list
Tidak ada data
```

```
MENU
1. tambah_belakang
2. tambah_depan
3. hapus_belakang
4. hapus_depan
5. tampilkan list
6. keluar
pilih : 4

data depan telah terhapus
```





Program ini tidak jauh berbeda dengan guided 1 dan guided 2 disini hanya lah ada penambahan oprasi memilih ketika program di run, sehingaa user bisa memilih opsi yang telah disediakan, untuk memulai nya user diharus kan memilih opsi 1-6 yang telah di deklarasikan dengan ‘int pilih’, di sini untuk variable pemilihan nya menggunkan ‘switch case’ case akan memanggil setiap opsi nya lalu di lanjutkan ke langkah selanjutnya dan ‘variabel fungsi’ di panggil untuk menginisiasi input dari user setiap pilihan akan memicu fungsi yang sesuai untuk melakukan operasi yang diminta, dengan umpan balik yang sesuai diberikan kepada pengguna setelah setiap operasi. Program ini memungkinkan pengguna untuk dengan mudah melakukan manipulasi pada linked list sesuai kebutuhan mereka dan ini lah yang disebut linklist dalam C++.

2. Unguided 2

Membuat pengdeklarasian struct yang menghasilkan node/simpul yang memiliki yang memiliki empat anggota yaitu prev (pointer ke node sebelumnya), data (int), info (char), dan next (pointer ke node berikutnya) lalu definisikan dalam bentuk struktur 'Node'

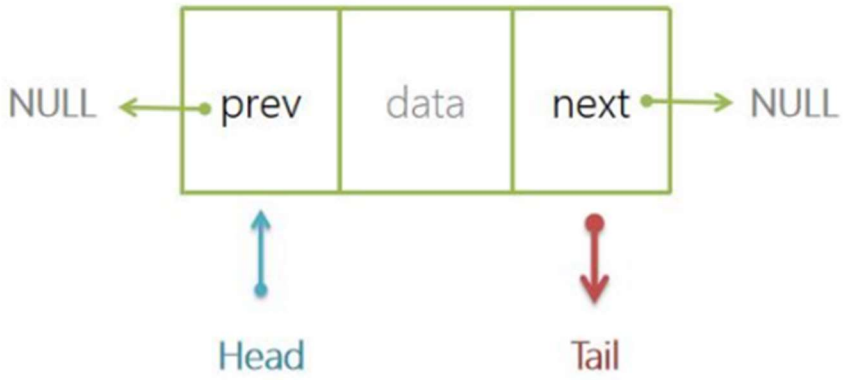
SOURCE CODE

```
#include <iostream>

using namespace std;

struct Node {
    Node* prev;
    int data;
    char info;
    Node* next;
};

int main() {

    Node* newNode = (Node*) malloc(sizeof(Node));


    newNode->prev = NULL;
    newNode->data = 1;
    newNode->info = 'I';
    newNode->next = NULL;


    cout << "Node: " << endl;
    cout << "prev: " << (newNode->prev ? "not NULL" : "NULL") << endl;
    cout << "data: " << newNode->data << endl;
    cout << "info: " << newNode->info << endl;
    cout << "next: " << (newNode->next ? "not NULL" : "NULL") << endl;

    return 0;
}
```

Penjelasan Program

Program ini adlh program yang merepresentasikan node dalam suatu linked list. Struktur Node sendiri ditetapkan untuk menyimpan informasi tentang setiap simpul dalm linked list. Saat membuat simpul baru, program menggalokasikan ruang di memori untuknya dan mengisi nilainya. Untuk simpul pertama, tidak ada simpul sebelumnya, jadi nilai prev diatur menjadi tidak ada (NULL). terus, data diatur menjadi 1. Karena ini juga adalah simpul pertama dan satu-satunya dalam contoh diatas, nilai next juga diatur menjadi tidak ada (NULL). Dengan cara ini, kita memiliki simpul pertama dalam linked list yang siap digunakan.

Jika kita ilutrasi program presentasi node ini ke dalam bentuk gambar maka akan seperti gambar yang ada di modul 7 jadi nya seperti ini :



Kesimpulan

Linked list adalah struktur data linier yang terdiri dari sekelompok simpul-simpul yang saling terhubung melalui tautan atau pointer. Setiap simpul menyimpan data dan tautan ke simpul berikutnya dalam urutan linmked list juga mempunya data depan yang di sebut head dan data belakang yang disebut tail. Keunggulan utama linked list adalah kemampuannya untuk menyisipkan dan menghapus elemen secara efisien di berbagai posisi tanpa perlu menggeser elemen-elemen lainnya, meskipun akses acak ke elemen-elemen tersebut biasanya lebih lambat daripada array.

Refrensi

1. Modul 7 Struktur Data
2. https://medium.com/@furatamarizuki/panduan-lengkap-mengenai-linked-list-definisi-implementasi-dan-penggunaan-dalam-pemrograman-a6159d548941#:~:text=Linked%20list%20adalah%20struktur%20data,simpul%20berikutnya%20dalam%20urutan%20linear.
3. https://suciantinovi.blogspot.com/2014/03/linked-list-i_14.html
4. https://youtu.be/VVemCxif9vg?si=quYpX2emAuxO62ER

Refrensi

1. Modul 5 Struktur Data
2. https://bakrie.ac.id/articles/552-kenalan-dengan-pengertian-queue-dalam-pemrograman.html#:~:text=Queue%20merupakan%20sebuah%20struktur%20data,Last%20In%2C%20First%20Out).
3. https://medium.com/@furatamarizuki/memahami-queue-dalam-pemrograman-konsep-implementasi-dan-penggunaannya-f1cd57c3e375