PRAKTIKUM ALGORITMA PEMROGRAMAN DAN STRUKTUR DATA

MATERI : Single dan Double Linked List

I. Tujuan

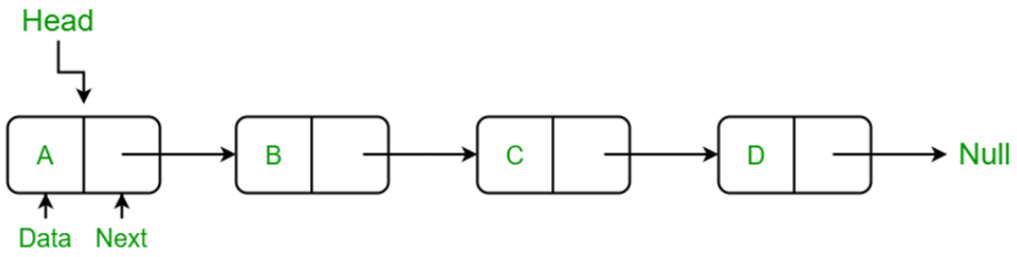
1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

II. Dasar Teori

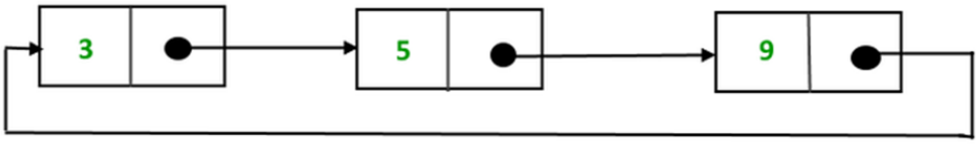
A. Single Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitrary bertipe pointer sebagai penunjuk node selanjutnya.

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Untuk menghubungkan satu node dengan node yang lainnya Linked List menggunakan pointer sebagai penunjuk node selanjutnya. Node sendiri merupakan sebuah struct yang terdiri dari beberapa field, minimal ada 2 buah field yaitu field untuk isi dari struct datanya sendiri, dan 1 field arbitrary bertipe pointer sebagai penunjuk node selanjutnya.



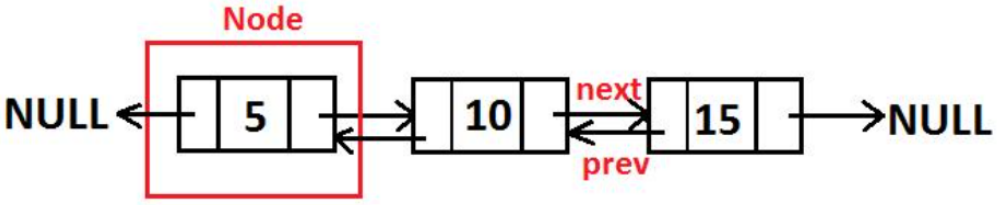
Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



B. Double Linked List

Dalam pembahasan artikel sebelumnya telah diperkenalkan Single Linked List, yakni linked list dengan sebuah pointer penghubung. Dalam artikel ini, dibahas pula varian linked list dengan 2 pointer penunjuk, yakni Doubly linked list yang memiliki pointer penunjuk 2 arah, yakni ke arah node sebelum (previous/prev) dan node sesudah (next).

Representasi sebuah doubly linked list dapat dilihat pada gambar berikut ini:



Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

III. Guided

A. Latihan Single Linked List

```
#include <iostream>
using namespace std;
///PROGRAM SINGLE LINKED LIST NON-CIRCULAR
//Deklarasi Struct Node

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

Node *head;
Node *tail;

//Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
//Pengecekan
bool isEmpty()
{
    if (head == NULL)
  return true;
    else
```

```
        return false;
}

//Tambah Depan
void insertDepan(int nilai)
{
            //Buat Node baru
            Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;

            if (isEmpty() == true){
            head = tail = baru;
        tail->next = NULL;
            }
            else{
        baru->next = head;
            head = baru;
            }
}

//Tambah Belakang
void insertBelakang(int nilai)
{
            //Buat Node baru
            Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
```

```
            if (isEmpty() == true){
            head = tail = baru;
    tail->next = NULL;
            }
            else{
    tail->next = baru;
            tail = baru;
            }
}

//Hitung Jumlah List
int hitungList()
{
            Node *hitung;
            hitung = head;
            int jumlah = 0;
            while( hitung != NULL ){
        jumlah++;
        hitung = hitung->next;
            }
            return jumlah;
}

//Tambah Tengah
void insertTengah(int data, int posisi)
            {
            if( posisi < 1 || posisi > hitungList() ){
            cout << "Posisi diluar jangkauan" << endl;
            }else if( posisi == 1){
```

```
            cout << "Posisi bukan posisi tengah" << endl;
            }else{
            Node *baru, *bantu;
            baru = new Node();
        baru->data = data;

            // tranversing
            bantu = head;
            int nomor = 1;
            while( nomor < posisi - 1 ){
            bantu = bantu->next;
            nomor++;
            }
    baru->next = bantu->next;
    bantu->next = baru;
            }
}

//Hapus Depan
void hapusDepan() {
            Node *hapus;

            if (isEmpty() == false){
            if (head->next != NULL){
           hapus = head;
           head = head->next;
           delete hapus;
            }
            else{
```

```
            head = tail = NULL;
            }
            }
            else{
            cout << "List kosong!" << endl;
            }
}

//Hapus Belakang
void hapusBelakang() {
            Node *hapus;
            Node *bantu;

            if (isEmpty() == false){
            if (head != tail){
            hapus = tail;
            bantu = head;
            while (bantu->next != tail){
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
            }
            else{
            head = tail = NULL;
            }
            }
            else{
```

```
            cout << "List kosong!" << endl;
            }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
            Node *hapus, *bantu, *bantu2;
            if( posisi < 1 || posisi > hitungList() ){
            cout << "Posisi di luar jangkauan" << endl;
            }else if( posisi == 1){
            cout << "Posisi bukan posisi tengah" << endl;
        }else{
            int nomor = 1;
            bantu = head;
            while( nomor <= posisi ){
                if( nomor == posisi-1 ){
                    bantu2 = bantu;
                }
                if( nomor == posisi ){
                    hapus = bantu;
                }
                bantu = bantu->next;
                nomor++;
            }
            bantu2->next = bantu;
            delete hapus;
            }
}
```

```
//Ubah Depan
void ubahDepan(int data)
{
            if (isEmpty() == false){
        head->data = data;
            }
            else{
            cout << "List masih kosong!" << endl;
            }
}

//Ubah Tengah
void ubahTengah(int data, int posisi)
{
            Node *bantu;

            if (isEmpty() == false){
            if( posisi < 1 || posisi > hitungList() ){
            cout << "Posisi di luar jangkauan" << endl;
            }
            else if( posisi == 1){
            cout << "Posisi bukan posisi tengah" << endl;
            }
            else{
            bantu = head;
            int nomor = 1;
            while (nomor < posisi){
               bantu = bantu->next;
```

```
            nomor++;
        }
        bantu->data = data;
        }
        }
        else{
        cout << "List masih kosong!" << endl;
        }
}



//Ubah Belakang
void ubahBelakang(int data)
{
        if (isEmpty() == false){
   tail->data = data;
        }
        else{
        cout << "List masih kosong!" << endl;
        }
}


//Hapus List
void clearList()
{
        Node *bantu, *hapus;
        bantu = head;
        while (bantu != NULL){
        hapus = bantu;
```

```
            bantu = bantu->next;
        delete hapus;
            }
            head = tail = NULL;
            cout << "List berhasil terhapus!" << endl;
}

//Tampilkan List
void tampil()
{
            Node *bantu;
            bantu = head;

            if (isEmpty() == false){
            while (bantu != NULL){
           cout << bantu->data << ends;
           bantu = bantu->next;
            }
            cout << endl;
            }
            else{
            cout << "List masih kosong!" << endl;
            }
}

int main()
{
            init();
    insertDepan(3);
```

```
        tampil();
    insertBelakang(5);
        tampil();
    insertDepan(2);
        tampil();
    insertDepan(1);
        tampil();
    hapusDepan();
        tampil();
    hapusBelakang();
        tampil();
    insertTengah(7,2);
        tampil();
    hapusTengah(2);
        tampil();
    ubahDepan(1);
        tampil();
    ubahBelakang(8);
        tampil();
    ubahTengah(11, 2);
        tampil();


        return 0;
}
```

B. Latihan Double Linked List

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

class Node {
public:
```

```
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }

        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }

        Node* temp = head;
        head = head->next;

        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr;
        }

        delete temp;
    }

    bool update(int oldData, int newData) {
        Node* current = head;
```

```
        while (current != nullptr) {
            if (current->data == oldData) {
                current->data = newData;
                return true;
            }

            current = current->next;
        }

        return false;
    }

    void deleteAll() {
        Node* current = head;

        while (current != nullptr) {
            Node* temp = current;
            current = current->next;
            delete temp;
        }

        head = nullptr;
        tail = nullptr;
    }

    void display() {
        Node* current = head;

        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }

        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;

    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;
```

```
        int choice;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int data;
                cout << "Enter data to add: ";
                cin >> data;
                list.push(data);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                int oldData, newData;
                cout << "Enter old data: ";
                cin >> oldData;
                cout << "Enter new data: ";
                cin >> newData;
                bool updated = list.update(oldData, newData);
                if (!updated) {
                    cout << "Data not found" << endl;
                }
                break;
            }
            case 4: {
                list.deleteAll();
                break;
            }
            case 5: {
                list.display();
                break;
            }
            case 6: {
                return 0;
            }
            default: {
                cout << "Invalid choice" << endl;
                break;
            }
        }
    }

    return 0;
}
```

IV. Unguided

1. Soal mengenai Single Linked List

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut:

a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda.

[Nama_anda]	[Usia_anda]
Budi	19
Carol	20
Ann	18
Yusuke	19
Akechi	20
Hoshino	18
Karin	18

b. Hapus data Akechi

c. Tambahkan data berikut diantara Carol dan Ann : Futaba 18

d. Tambahlan data berikut diawal : Igor 20

e. Ubah data Carol menjadi : Reyn 18

f. Tampilkan seluruh data

2. Soal mengenai Double Linked List

Modifikasi Guided Double Linked List ditambahkan fungsi menambahkan data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

Nama Produk	Harga

Originote	60.000
Somethinc	150.000
Skintific	100.000
Wardah	50.000
Hanasui	30.000

Case:

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini

   Toko Skincare Purwokerto

   1. ***Tambah Data***
   2. ***Hapus Data***
   3. ***Update Data***
   4. ***Tambah Data Urutan Tertentu***
   5. ***Hapus Data Urutan Tertentu***
   6. ***Hapus Seluruh Data***
   7. ***Tampilkan Data***
   8. ***Exit***

   Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Azarine	65.000
Skintific	100.000
Cleora	55.000