

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 3
SINGLE AND DOUBLE LINKED LIST**



DISUSUN OLEH:

NAMA : SYARIEF RENDI ADITYA ANTONIUS

NIM : 2311102072

S1 IF-11-B

DOSEN:

Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

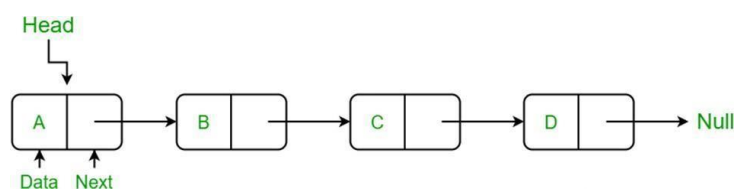
2024

A. DASAR TEORI

Linked list adalah sejumlah objek yang di link atau dihubungkan satu dengan yang lainnya sehingga membentuk suatu list. Sedangkan objek itu sendiri adalah merupakan gabungan beberapa elemen data (variabel) yang dijadikan satu kelompok atau structure atau record yang dibentuk dengan perintah struct. Tiap- tiap elemen dapat memiliki tipe data tersendiri yang berbeda dengan tipe data elemen lain. Untuk menghubungkan objek satu dengan objek lainnya, diperlukan paling tidak sebuah variabel yang 'bertipe' pointer. Variabel pointer tersebut merupakan salah satu variabel dalam struktur objek.

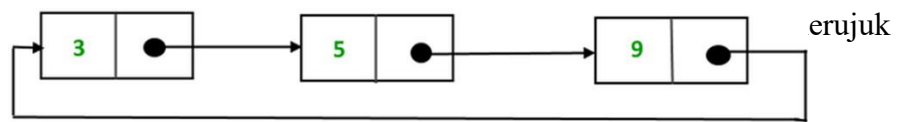
1) Single Linked List

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. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.



Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List.

Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk



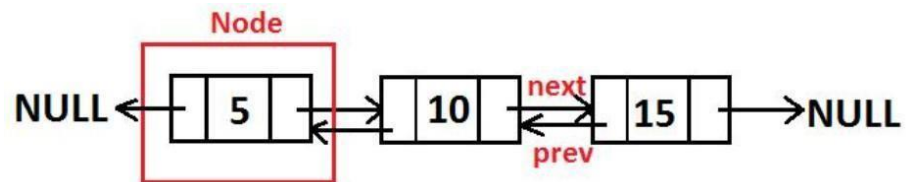
2) Double Linked List

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya.

Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul

membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List

Representasi sebuah double 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

B. Guided

GUIDED 1 :

Latihan Single Linked List

SOURCE CODE

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
Node* tail;

// Inisialisasi Node
void init() {
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
void insertDepan(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Tambah Node di belakang
void insertBelakang(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
```

```

        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

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

// Tambah Node di posisi 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 = new Node();
        baru->data = data;
        Node* bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Node di depan
void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;
        if (head->next != NULL) {
            head = head->next;
            delete hapus;
        } else {
            head = tail = NULL;
        }
    }
}

```

```

        delete hapus;
    }
} else {
    cout << "List kosong!" << endl;
}
}

// Hapus Node di belakang
void hapusBelakang() {
    if (!isEmpty()) {
        if (head != tail) {
            Node* hapus = tail;
            Node* 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 Node di posisi tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++) {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

// Ubah data Node di depan
void ubahDepan(int data) {

```

```

        if (!isEmpty()) {
            head->data = data;
        } else {
            cout << "List masih kosong!" << endl;
        }
    }

// Ubah data Node di posisi tengah
void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node* bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++) {
                bantu = bantu->next;
            }
            bantu->data = data;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

// Hapus semua Node di list
void clearList() {
    Node* bantu = head;
    while (bantu != NULL) {
        Node* hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

```



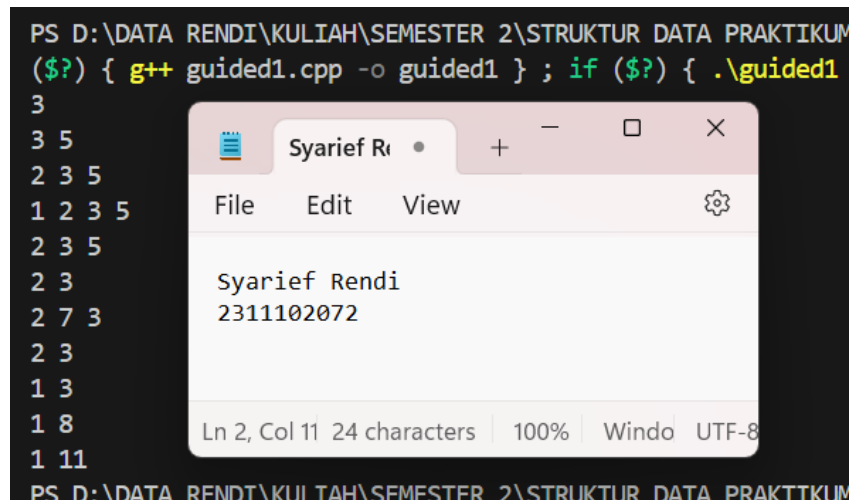
```

// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            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;
} //2311102072syarief

```

SCREENSHOT OUTPUT



```
PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR DATA PRAKTIKUM
($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRUKTUR DATA PRAKTIKUM
```

DESKRIPSI PROGRAM

Dalam program, program memakai struct untuk inisialisasi coding node dan juga linked list terdiri dari node yang berasal dari beberapa fungsi void untuk masing masing algoritma. Pada program int main yaitu pertama terdapat inisialisasi dengan void init. Program juga memanggil dari fungsi void yang telah dibuat seperti insertDepan, insertBelakang, hapusDepan, hapusBelakang, insertTengah, hapusTengah, ubahDepan, ubahBelakang, ubah Tengah. Pada fungsi tampil hanya digunakan untuk algoritma tampilan output. Nilai di hardcode kan dalam source code.

GUIDED 2 :

Latihan Double Linked List

SOURCE CODE

```
#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;
} //2311102072syarief

```

SCREENSHOT OUTPUT

PS D:\DATA RENDI\KULIAH\SEMESTER 2\STRU
ded2.cpp -o guided2 } ; if (\$?) { .\gui

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 1
Enter data to add: 99

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 1
Enter data to add: 98

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5
98 99

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data

Enter your choice: 3
Enter old data: 98
Enter new data: 95

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5
95 99

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5
99

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data

Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit

Enter your choice: 6
PS D:\DATA RENDI\KULIAH\SEMESTER 2\ST

DESKRIPSI PROGRAM

Program di atas merupakan implementasi sederhana dari sebuah Double Linked List. Struktur data Double Linked List terdiri dari node-node yang memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node selanjutnya (next). Pada fungsi int main berisi loop utama yang meminta input dari pengguna untuk melakukan operasi-operasi pada Double Linked List. Pilihan yang tersedia antara lain menambahkan data ke depan linked list, menghapus data dari depan linked list, mengubah data yang sudah ada dengan data baru, menghapus semua data dari linked list, menampilkan semua data yang ada pada linked list, dan keluar dari program.

C. UNGUIDED

UNGUIDED 1 : Soal mengenai Single Linked List. Buatlah program menu single linked list no-circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi

SOURCE CODE

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

struct Node {
    string nama;
    int usia;
    Node* next;
};

Node* head;
Node* tail;

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

bool isEmpty() {
    return head == NULL;
}

void insertDepan(string nama, int usia) {
    Node* baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, int usia) {
    Node* baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty()) {
```

```

        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

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

void insertTengah(string nama, int usia, 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 = new Node();
        baru->nama = nama;
        baru->usia = usia;
        Node* bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusTengah(string nama) {
    if (!isEmpty()) {
        Node* hapus;
        Node* bantu = head;
        Node* prev = nullptr;
        while (bantu != nullptr && bantu->nama != nama) {
            prev = bantu;
            bantu = bantu->next;
        }
        if (bantu == nullptr) {

```



```

        cout << "Nama tidak ditemukan dalam list" << endl;
        return;
    }
    if (prev != nullptr) {
        prev->next = bantu->next;
    } else {
        head = bantu->next;
    }
    if (bantu == tail) {
        tail = prev;
    }
    delete bantu;
    cout << "Node dengan nama '" << nama << "' berhasil
dihapus" << endl;
} else {
    cout << "List masih kosong!" << endl;
}
}

void ubahTengah(string nama, int usia, string nama_lama) {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != nullptr && bantu->nama != nama_lama) {
            bantu = bantu->next;
        }
        if (bantu != nullptr) {
            bantu->nama = nama;
            bantu->usia = usia;
            cout << "Node dengan nama '" << nama_lama << "'
berhasil diubah menjadi '" << nama << "' dengan usia baru " <<
usia << endl;
        } else {
            cout << "Node dengan nama '" << nama_lama << "'
tidak ditemukan" << endl;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->nama << " " << bantu->usia << endl;
            bantu = bantu->next;
        }
    }
}

```

```

        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    init();
    string nama;
    int usia, choice;
    cout << "Masukkan nama anda: ";
    cin >> nama;
    cout << "Masukkan usia anda: ";
    cin >> usia;
    insertBelakang(nama, usia);

    while (true) {
        cout << "1. Tambah Node di depan" << endl;
        cout << "2. Tambah Node di belakang" << endl;
        cout << "3. Tambah Node di posisi tengah" << endl;
        cout << "4. Hapus Node berdasarkan nama" << endl;
        cout << "5. Ubah data Node berdasarkan nama" << endl;
        cout << "6. Tampilkan semua data Node di list" << endl;
        cout << "7. Keluar" << endl;

        cout << "Masukkan pilihan: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                cout << "Masukkan nama: ";
                cin >> nama;
                cout << "Masukkan usia: ";
                cin >> usia;
                insertDepan(nama, usia);
                break;
            }
            case 2: {
                cout << "Masukkan nama: ";
                cin >> nama;
                cout << "Masukkan usia: ";
                cin >> usia;
                insertBelakang(nama, usia);
                break;
            }
            case 3: {
                int posisi;

```

```

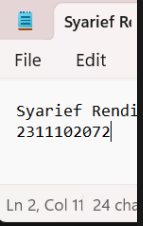
        cout << "Masukkan nama: ";
        cin >> nama;
        cout << "Masukkan usia: ";
        cin >> usia;
        cout << "Masukkan posisi: ";
        cin >> posisi;
        insertTengah(nama, usia, posisi);
        break;
    }
    case 4: {
        cout << "Masukkan nama yang ingin dihapus: ";
        cin >> nama;
        hapusTengah(nama);
        break;
    }
    case 5: {
        string nama_lama;
        cout << "Masukkan nama lama: ";
        cin >> nama_lama;
        cout << "Masukkan nama baru: ";
        cin >> nama;
        cout << "Masukkan usia baru: ";
        cin >> usia;
        ubahTengah(nama, usia, nama_lama);
        break;
    }
    case 6: {
        tampil();
        break;
    }
    case 7: {
        return 0;
    }
    default: {
        cout << "Pilihan tidak valid" << endl;
        break;
    }
}

return 0;
} //2311102072syarief

```

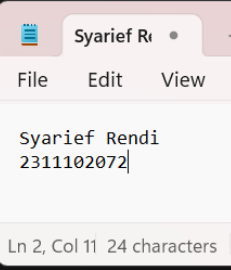
SCREENSHOT OUTPUT

```
unguided1.cpp -o unguided1 } ; if ($?) { .\unguided1 }
Masukkan nama anda: John
Masukkan usia anda: 19
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 2
Masukkan nama: Jane
Masukkan usia: 20
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 2
Masukkan nama: Michael
Masukkan usia: 18
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 2
Masukkan nama: Yusuke
Masukkan usia: 19
1. Tambah Node di depan
```



```
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 2
Masukkan nama: Akechi
Masukkan usia: 20
Masukkan pilihan: 2
Masukkan nama: Hoshino
Masukkan usia: 18
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 2
Masukkan nama: Karin
Masukkan usia: 18
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
```

```
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 6
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
```



```
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 4
Masukkan nama yang ingin dihapus: Akechi
Node dengan nama 'Akechi' berhasil dihapus
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
```

```
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 3
Masukkan nama: Futaba
Masukkan usia: 18
Masukkan posisi: 2
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
```

The image displays two screenshots of a C++ program running in a terminal and a code editor.

Terminal Screenshot (Left):

```
Masukkan pilihan: 6
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18

1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar

Masukkan pilihan: 1
Masukkan nama: Igor
Masukkan usia: 20
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar

Masukkan pilihan: 6
Igor 20
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Code Editor Screenshot (Right):

The code editor shows the source code with a tooltip for the variable `Syarief Rendi` at line 2, column 11, containing the value `2311102072`.

```
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 5
Masukkan nama lama: Michael
Masukkan nama baru: Reyn
Masukkan usia baru: 18
Node dengan nama 'Michael' berhasil diubah menjadi 'Reyn' dengan usia baru 18
1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 6
Igor 20
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18

1. Tambah Node di depan
2. Tambah Node di belakang
3. Tambah Node di posisi tengah
4. Hapus Node berdasarkan nama
5. Ubah data Node berdasarkan nama
6. Tampilkan semua data Node di list
7. Keluar
Masukkan pilihan: 7
```

DESKRIPSI PROGRAM

Program di atas merupakan implementasi sederhana dari sebuah Single Linked List. Struktur data Single Linked List terdiri dari node-node yang memiliki satu pointer, yaitu pointer ke node selanjutnya (next). Pada fungsi `int main` berisi loop utama yang meminta input dari pengguna untuk melakukan operasi-operasi pada Single Linked List. Pilihan yang tersedia antara lain menambahkan data ke depan linked list, menambah data ke belakang linked list, menambah data di posisi tengah linked list, menghapus data berdasarkan nama, mengubah data berdasarkan nama yang sudah ada dengan data baru, dan menampilkan semua data yang ada pada linked list.

UNGUIDED 2 : Soal mengenai Double Linked List. Modifikasi Guided Double Linked List dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

SOURCE CODE

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

class Node {
public:
    string namaProduk;
    double harga;
    Node* prev;
    Node* next;
};

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

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

    void push(string namaProduk, double harga) {
        Node* newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = nullptr;

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

    void pushAt(int pos, string namaProduk, double harga) {
        if (pos < 1) {
```

```

        cout << "Posisi tidak valid" << endl;
        return;
    }

    Node* newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;

    if (pos == 1) {
        newNode->next = head;
        newNode->prev = nullptr;
        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
        return;
    }

    Node* current = head;
    int count = 1;
    while (current != nullptr && count < pos - 1) {
        current = current->next;
        count++;
    }

    if (current == nullptr) {
        cout << "Posisi tidak valid" << endl;
        return;
    }

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

void pop() {
    if (head == nullptr) {
        return;
    }
    Node* temp = head;
    head = head->next;

```

```

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

        delete temp;
    }

    void popAt(int pos) {
        if (pos < 1) {
            cout << "Posisi tidak valid" << endl;
            return;
        }

        if (pos == 1) {
            if (head == nullptr) {
                return;
            }
            Node* temp = head;
            head = head->next;
            if (head != nullptr) {
                head->prev = nullptr;
            } else {
                tail = nullptr;
            }
            delete temp;
            return;
        }

        Node* current = head;
        int count = 1;
        while (current != nullptr && count < pos) {
            current = current->next;
            count++;
        }

        if (current == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }

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

```



```

        current->prev->next = current->next;
        delete current;
    }

    bool update(string oldNamaProduk, string newNamaProduk, double
newHarga) {
        Node* current = head;

        while (current != nullptr) {
            if (current->namaProduk == oldNamaProduk) {
                current->namaProduk = newNamaProduk;
                current->harga = newHarga;
                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() {
        cout << "Nama Produk      Harga" << endl;
        Node* current = head;
        while (current != nullptr) {
            cout << current->namaProduk << "      " << current->harga
<< endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "Toko Tambah Ganteng Purwokerto" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
    }
}

```

```

    cout << "3. Update Data" << endl;
    cout << "4. Tambah Data Urutan Tertentu" << endl;
    cout << "5. Hapus Data Urutan Tertentu" << endl;
    cout << "6. Hapus Seluruh Data" << endl;
    cout << "7. Tampilkan Data" << endl;
    cout << "8. Exit" << endl;

    int choice;
    cout << "Masukkan pilihan Anda: ";
    cin >> choice;

    switch (choice) {
        case 1: {
            string namaProduk;
            double harga;
            cout << "Masukkan Nama Produk: ";
            cin.ignore();
            getline(cin, namaProduk);
            cout << "Masukkan Harga Produk: ";
            cin >> harga;
            list.push(namaProduk, harga);
            break;
        }
        case 2: {
            list.pop();
            break;
        }
        case 3: {
            string oldNamaProduk, newNamaProduk;
            double newHarga;
            cout << "Masukkan Nama Produk Lama: ";
            cin.ignore();
            getline(cin, oldNamaProduk);
            cout << "Masukkan Nama Produk Baru: ";
            getline(cin, newNamaProduk);
            cout << "Masukkan Harga Produk Baru: ";
            cin >> newHarga;
            bool updated = list.update(oldNamaProduk,
newNamaProduk, newHarga);
            if (!updated) {
                cout << "Produk tidak ditemukan" << endl;
            }
            break;
        }
        case 4: {
            int pos;
            string namaProduk;
            double harga;

```

```

        cout << "Masukkan posisi: ";
        cin >> pos;
        cout << "Masukkan Nama Produk: ";
        cin.ignore();
        getline(cin, namaProduk);
        cout << "Masukkan Harga Produk: ";
        cin >> harga;
        list.pushAt(pos, namaProduk, harga);
        break;
    }
    case 5: {
        int pos;
        cout << "Masukkan posisi untuk menghapus: ";
        cin >> pos;
        list.popAt(pos);
        break;
    }
    case 6: {
        list.deleteAll();
        break;
    }
    case 7: {
        list.display();
        break;
    }
    case 8: {
        return 0;
    }
    default: {
        cout << "Pilihan tidak valid" << endl;
        break;
    }
}

return 0;
}
} //2311102072syarief

```

SCREENSHOT OUTPUT

```
g++ unguided2.cpp -o unguided2 } ; if ($?) { .\unguide
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 1
Masukkan Nama Produk: Originote
Masukkan Harga Produk: 60000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 1
Masukkan Nama Produk: Somethinc
Masukkan Harga Produk: 150000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 1
Masukkan Nama Produk: Skintific
Masukkan Harga Produk: 100000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 1
Masukkan Nama Produk: Wardah
Masukkan Harga Produk: 50000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 1
Masukkan Nama Produk: Hanasui
Masukkan Harga Produk: 30000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 7
Nama Produk      Harga
Originote        60000
Somethinc        150000
Skintific        100000
Wardah           50000
Hanasui          30000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 4
Masukkan posisi: 3
Masukkan Nama Produk: Azarine
Masukkan Harga Produk: 65000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 7
Nama Produk      Harga
Originote        60000
Somethinc        150000
Azarine          65000
Skintific        100000
Cleora           55000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 3
Masukkan Nama Produk Lama: Hanasui
Masukkan Nama Produk Baru: Cleora
Masukkan Harga Produk Baru: 55000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 7
Nama Produk      Harga
Originote        60000
Somethinc        150000
Azarine          65000
Skintific        100000
Cleora           55000
Toko Tambah Ganteng 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
Masukkan pilihan Anda: 8
PS D:\DATA_RENDI\KULIAH\SEMESTER_2\STRUKTUR DATA
```

DESKRIPSI PROGRAM

Program di atas merupakan implementasi sederhana dari sebuah Double Linked List. Struktur data Double Linked List terdiri dari node-node yang memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node selanjutnya (next). Pada fungsi main, terdapat loop utama yang meminta input dari user untuk melakukan operasi-operasi pada Double Linked List. Pilihan yang tersedia antara lain menambahkan data ke dalam linked list, menghapus data dari linked list, mengupdate data berdasarkan nama produk yang sudah ada dengan data baru, menambahkan data pada posisi tertentu di dalam linked list, menghapus data dari posisi tertentu di dalam linked list, menghapus semua data yang ada dalam linked list, menampilkan semua data yang ada pada linked list. Dengan adanya pilihan-pilihan tersebut, user dapat mengelola data pada Doubly Linked List sesuai dengan kebutuhan, baik itu menambah, menghapus, mengubah, atau menampilkan data.

D. Kesimpulan

Linked list adalah sejumlah objek yang dihubungkan satu dengan yang lainnya sehingga membentuk suatu list. Variabel pointer tersebut merupakan salah satu variabel dalam struktur objek. Terdapat beberapa macam Linked List seperti : Single Linked List dan Double Linked List.

E. Referensi

Asisten Praktikum, "Modul Single and Double Linked List", Learning Management System, 2024.

Yesi Gusla, dkk. 2022. Konsep Algoritma dan Pemrograman. Bandung : Indie Press.

Budi Raharjo. 2015. Pemrograman C++. Bandung: Informatika.

Rinaldi Munir, Leony Lidya. 2016. Algoritma dan Pemrograman. Bandung: Informatika Bandung.