

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN  
ALGORITMA**

**MODUL III  
SINGLE AND DOUBLE LINKED  
LIST**



**Disusun Oleh :**

Muhammad Rusdiyanto Asatman  
2311102053

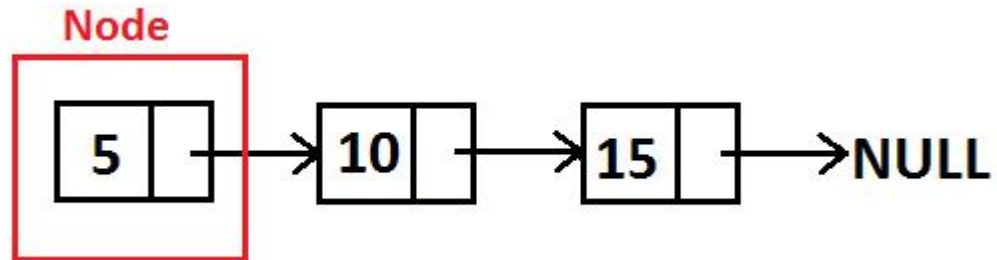
**Dosen :**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## A. Dasar Teori

### 1. Single Linked List

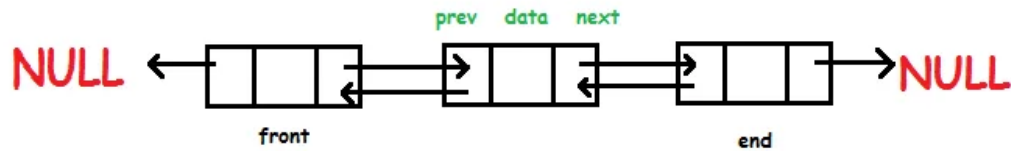


Single linked list adalah struktur data linier di mana setiap elemen, atau “node,” terhubung secara sekuensial melalui pointer. Setiap node menyimpan dua informasi: satu bagian data dan satu pointer ke node berikutnya dalam list. Di single linked list, navigasi dilakukan dalam satu arah, dari node pertama (head) hingga node terakhir yang pointer-nya menunjuk ke NULL yang menandakan akhir dari list. Dengan struktur seperti itu, single linked list menyediakan cara yang efisien untuk menyimpan dan mengakses data berurutan. Namun, satu kelemahan utama dari single linked list adalah bahwa untuk mencapai elemen tertentu, program harus melintasi list dari awal hingga elemen yang diinginkan. Ini membatasi kemampuan untuk pencarian acak dan mempengaruhi kinerja dalam beberapa kasus penggunaan. Berikut adalah karakteristik utama dari single linked list:

- Dinamis: Ukuran list dapat bertambah atau berkurang selama runtime.
- Efisien dalam Penyisipan/Penghapusan: Menyisipkan atau menghapus node hanya memerlukan perubahan pada pointer, tidak seperti array yang memerlukan pergeseran elemen.
- Penggunaan Memori: Tidak memerlukan blok memori yang berdekatan; node dapat tersebar di memori.

Single linked list sering digunakan dalam situasi di mana penyisipan dan penghapusan data yang efisien diperlukan, seperti dalam implementasi stack, queue, dan daftar lainnya.

## 2. Double Linked List



Double Linked List (dikenal juga sebagai Doubly Linked List) adalah sebuah struktur data yang terdiri dari simpul-simpul yang saling terhubung secara linear. Setiap simpul dalam double linked list memiliki dua buah pointer, yaitu prev (menunjuk ke simpul sebelumnya) dan next (menunjuk ke simpul sesudahnya). Dengan ini, setiap simpul dalam double linked list dapat diakses dari kedua arah: dari simpul sebelumnya dan dari simpul berikutnya.

Keuntungan utama dari double linked list adalah kemampuannya untuk mendukung pencarian maju dan mundur secara efisien. Hal ini karena setiap simpul memiliki referensi ke simpul sebelumnya dan simpul berikutnya, sehingga tidak perlu melintasi seluruh list dari awal untuk mencapai simpul tertentu. Dalam beberapa aplikasi, ini dapat meningkatkan kinerja dan efisiensi operasi-operasi seperti pencarian, penghapusan, dan penyisipan data. Namun, keuntungan ini juga datang dengan biaya tambahan dalam hal penggunaan memori, karena setiap simpul dalam double linked list memerlukan penyimpanan tambahan untuk dua pointer. Selain itu, implementasi double linked list juga sedikit lebih kompleks daripada single linked list.

Double linked list sering digunakan dalam berbagai aplikasi seperti penyimpanan data terurut, dalam implementasi beberapa struktur data seperti deque (double-ended queue), dan dalam implementasi beberapa algoritma yang memerlukan kemampuan untuk mengakses data maju dan mundur secara efisien.

## B. Guided

### Guided 1

```
#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;
}

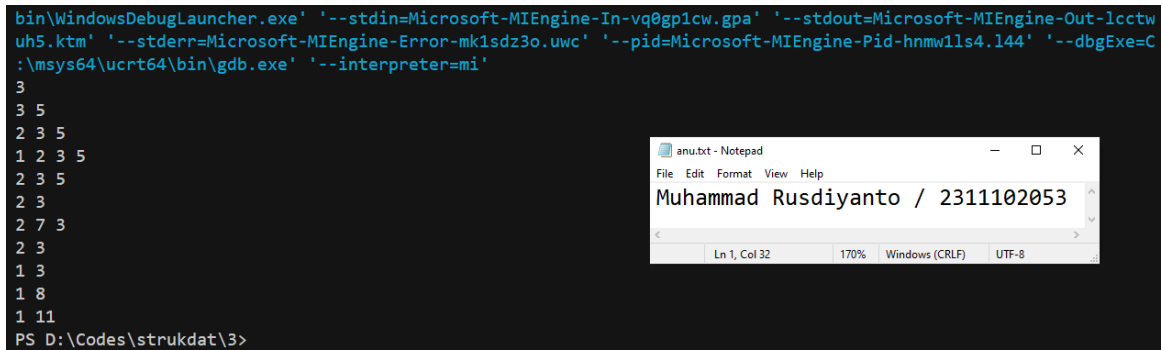
// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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;
}

```



## Screenshots Output



The screenshot shows a debugger window with assembly code on the left and a Notepad window on the right. The assembly code is for a program named 'bin\WindowsDebugLauncher.exe' and shows instructions for loading and executing a program. The Notepad window, titled 'anu.txt - Notepad', displays the output of the program: 'Muhammad Rusdiyanto / 2311102053'.

## Deskripsi:

Program di atas adalah program yang digunakan untuk mendemonstrasikan penggunaan serta pembuatan program yang berbasis Single Linked List. Dalam program tersebut terdapat 14 fungsi (tidak termasuk main) yang terdiri dari :

1. *init* : Membantu inisialisasi program dengan memberikan nilai NULL ke head dan tail.
2. *isEmpty* : mengecek apakah list masih kosong dengan mengecek nilai head. Jika NULL, maka mengembalikan nilai true. Jika tidak NULL, maka false.
3. *insertDepan* : Fungsi yang satu ini berguna untuk menambahkan node atau data ke depan dari list, sehingga data baru menjadi head.
4. *insertBelakang* : Sama seperti *insertDepan*, hanya saja dilakukan di belakang list, sehingga data baru menjadi tail.
5. *hitungList* : menghitung jumlah data atau Node dalam list dengan cara looping ke semua Node disertai counter untuk menghitung Node.
6. *insertTengah* : menambahkan data di posisi tertentu dalam list dengan looping sampai di Node ke sekian (sesuai nilai parameter), lalu data baru ditambahkan.
7. *hapusDepan* : menghapus data paling awal (head).
8. *hapusBelakang* : menghapus data paling akhir (tail).
9. *hapusTengah* : menghapus data di posisi tertentu.
10. *ubahDepan* : mengubah data dalam Node head.
11. *ubahTengah* : mengubah data Node di posisi tertentu Looping akan berhenti jika counter
12. *ubahBelakang* : mengubah data dalam Node tail.
13. *clearList* : menghapus semua data / Node dengan cara looping ke semua Node dan menghapusnya satu per satu. Setelah semua Node terhapus, head dan tail akan menjadi NULL lagi (karena tidak ada Node).

14. *tampil* : menampilkan semua data dengan cara looping ke setiap Node, lalu menampilkan data dalam Node tersebut. Setelah ditampilkan, program akan looping ke Node selanjutnya untuk ditampilkan lagi hingga Node selanjutnya adalah NULL atau kosong.

Program tersebut berjalan tanpa input dari pengguna (karena untuk demonstrasi saja).

## Guided 2

```
#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;
}

```

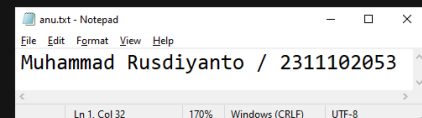
## Screenshots Output

### - Menambahkan data

```

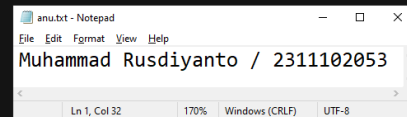
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 1
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5.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

```



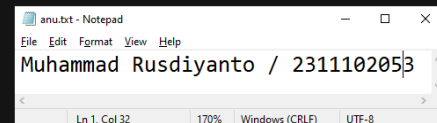
## - Menghapus data

```
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
2 1
```



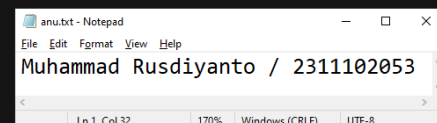
## - Ubah data

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



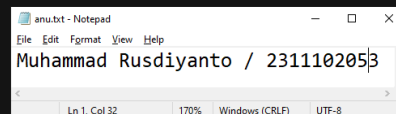
## - Hapus semua data

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



## - Exit program

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS D:\Codes\strukdat\3> █
```



Deskripsi:

Program tersebut merupakan sebuah program yang mendemonstrasikan penggunaan Double Linked List. Berbeda dengan program Guided 1, program ini sudah bisa berinteraksi dengan input dari pengguna. Untuk daftar fungsi dalam program ini adalah sebagai berikut :

1. *DoublyLinkedList* : Sebuah fungsi spesial dalam class, mirip seperti fungsi init dalam Guided 1. Jadi, fungsi ini akan dijalankan pertama kali setelah class dideklarasikan. Dalam hal ini, fungsi berguna untuk memberikan nilai NULL atau nullptr ke variabel head dan tail.
2. *Push* : menambahkan data baru di depan list (head).
3. *pop* : menghapus data di depan list (head).
4. *update* : mengubah data sesuai dengan kriteria data lama yang diberikan. Hal ini dilakukan dengan cara looping ke setiap Node untuk mencari data yang sesuai. Setelah Node ditemukan, data akan diubah dan looping berhenti. Lalu, fungsi akan mengembalikan nilai true sebagai indikator bahwa data berhasil diubah. Jika tidak berhasil, maka mengembalikan false.
5. *deleteAll* : menghapus semua data dengan cara looping dan hapus satu per satu Node.
6. *display* : menampilkan semua nilai Node dengan looping.

Selain fungsi di atas, ada juga fungsi main yang menjadi titik awal dalam program. Fungsi main dalam program ini berguna untuk menampilkan menu serta mengolah input dari pengguna.

Sebagai catatan tambahan, program tersebut menggunakan class, bukan struct. Hal ini menyebabkan semua atribut dan method dalam class bersifat private, sehingga diperlukanlah keyword public supaya bisa diakses

## C. Unguided

### Unguided 1

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

// Deklarasi Struct Node
struct Node {
    string nama;
    int umur;
    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(string nama, int umur) {
    Node* baru = new Node;
    baru->nama = nama;
    baru->umur = umur;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Tambah Node di belakang
void insertBelakang(string nama, int umur) {
```



```

    Node* baru = new Node;
    baru->nama = nama;
    baru->umur = umur;
    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(string nama, int umur, 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->umur = umur;
        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(string nama, int umur) {
    if (!isEmpty()) {
        head->nama = nama;
        head->umur = umur;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di posisi tengah
void ubahTengah(string nama, int umur, 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->nama = nama;
            bantu->umur = umur;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di belakang

```

```

void ubahBelakang(string nama, int umur) {
    if (!isEmpty()) {
        tail->nama = nama;
        tail->umur = umur;
    } 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;
        cout << "Daftar data :" << endl;
        while (bantu != NULL) {
            cout << bantu->nama << ends << bantu->umur;
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

string inNama() {
    string nama;
    cout << endl << "Masukkan nama -> ";
    cin >> nama;
    return nama;
}

```

```

int inUmur() {
    int umur;
    cout << endl << "Masukkan umur -> ";
    cin >> umur;
    return umur;
}

int inPosisi() {
    int posisi;
    cout << endl << "Masukkan posisi data -> ";
    cin >> posisi;
    return posisi;
}

int main() {
    bool runApp = true;
    string line = "===== ";
    int choice;
    init();

    cout << "Selamat datang!" << endl;
    while(runApp) {
        cout << line << "[ List perintah ]" << line << endl;
        cout << "1. Masukkan data (depan)" << endl;
        cout << "2. Masukkan data (belakang)" << endl;
        cout << "3. Masukkan data (tengah)" << endl;
        cout << "4. Hapus data (depan)" << endl;
        cout << "5. Hapus data (belakang)" << endl;
        cout << "6. Hapus data (tengah)" << endl;
        cout << "7. Ubah data (depan)" << endl;
        cout << "8. Ubah data (belakang)" << endl;
        cout << "9. Ubah data (tengah)" << endl;
        cout << "10. Tampil data" << endl;
        cout << line << line << line;
        cout << endl << "Masukkan pilihan [1 - 9] -> ";
        cin >> choice;

        switch(choice) {
            case 1: insertDepan(inNama(), inUmur()); break;
            case 2: insertBelakang(inNama(), inUmur()); break;
            case 3: insertTengah(inNama(), inUmur(), inPosisi());
break;
            case 4: hapusDepan(); break;

```

```

        case 5: hapusBelakang(); break;
        case 6: hapusTengah(inPosisi()); break;
        case 7: ubahDepan(inNama(), inUmur()); break;
        case 8: ubahBelakang(inNama(), inUmur()); break;
        case 9: ubahTengah(inNama(), inUmur(), inPosisi());
break;

        case 10: tampil(); break;
        default: cout << "Input tidak valid!" << endl << endl;

    }
}
return 0;
}

```

## Screenshots Output

### - Menambahkan data

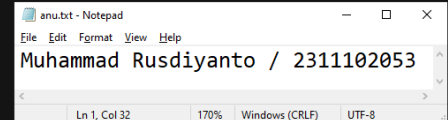
```

Masukkan nama -> Hoshino
===== [ List perintah ] =====
1. Masukkan data (depan)
2. Masukkan data (belakang)
3. Masukkan data (tengah)
4. Hapus data (depan)
5. Hapus data (belakang)
6. Hapus data (tengah)
7. Ubah data (depan)
8. Ubah data (belakang)
9. Ubah data (tengah)
10. Tampil data
=====
Masukkan pilihan [1 - 9] -> 2

Masukkan umur -> 18

Masukkan nama -> Karin

```



anu.txt - Notepad

File Edit Format View Help

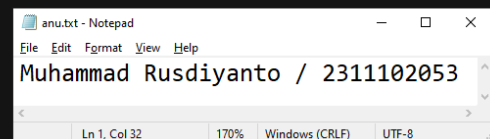
Muhammad Rusdiyanto / 2311102053

Ln 1, Col 32 170% Windows (CRLF) UTF-8

```

Masukkan pilihan [1 - 9] -> 10
Daftar data :
John : 19
Jane : 20
Michael : 18
Yusuke : 19
Akechi : 20
Hoshino : 18
Karin : 18

```



anu.txt - Notepad

File Edit Format View Help

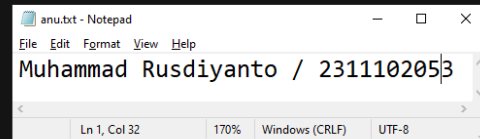
Muhammad Rusdiyanto / 2311102053

Ln 1, Col 32 170% Windows (CRLF) UTF-8

## - Hapus data akechi

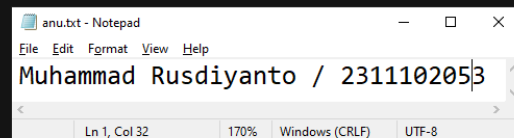
```
===== [ List perintah ] =====
1. Masukkan data (depan)
2. Masukkan data (belakang)
3. Masukkan data (tengah)
4. Hapus data (depan)
5. Hapus data (belakang)
6. Hapus data (tengah)
7. Ubah data (depan)
8. Ubah data (belakang)
9. Ubah data (tengah)
10. Tampil data
=====
Masukkan pilihan [1 - 9] -> 6

Masukkan posisi data -> 5
```



anu.txt - Notepad  
File Edit Format View Help  
Muhammad Rusdiyanto / 2311102053  
Ln 1, Col 32 170% Windows (CRLF) UTF-8

```
Masukkan pilihan [1 - 9] -> 10
Daftar data :
John : 19
Jane : 20
Michael : 18
Yusuke : 19
Hoshino : 18
Karin : 18
```



anu.txt - Notepad  
File Edit Format View Help  
Muhammad Rusdiyanto / 2311102053  
Ln 1, Col 32 170% Windows (CRLF) UTF-8

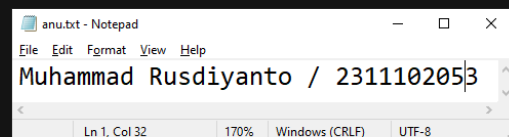
## - Tambah data futaba

```
===== [ List perintah ] =====
1. Masukkan data (depan)
2. Masukkan data (belakang)
3. Masukkan data (tengah)
4. Hapus data (depan)
5. Hapus data (belakang)
6. Hapus data (tengah)
7. Ubah data (depan)
8. Ubah data (belakang)
9. Ubah data (tengah)
10. Tampil data
=====
Masukkan pilihan [1 - 9] -> 3

Masukkan posisi data -> 2

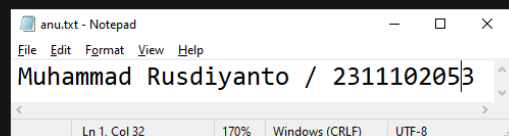
Masukkan umur -> 18

Masukkan nama -> Futaba
```



anu.txt - Notepad  
File Edit Format View Help  
Muhammad Rusdiyanto / 2311102053  
Ln 1, Col 32 170% Windows (CRLF) UTF-8

```
Masukkan pilihan [1 - 9] -> 10
Daftar data :
John : 19
Futaba : 18
Jane : 20
Michael : 18
Yusuke : 19
Hoshino : 18
Karin : 18
```



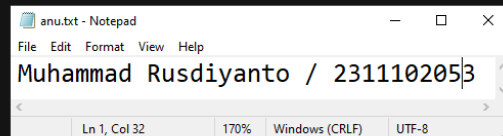
anu.txt - Notepad  
File Edit Format View Help  
Muhammad Rusdiyanto / 2311102053  
Ln 1, Col 32 170% Windows (CRLF) UTF-8

## - Tambah data Igor

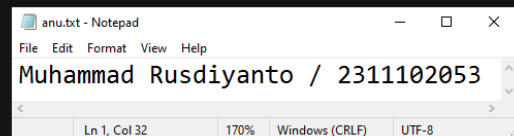
```
===== [ List perintah ] =====
1. Masukkan data (depan)
2. Masukkan data (belakang)
3. Masukkan data (tengah)
4. Hapus data (depan)
5. Hapus data (belakang)
6. Hapus data (tengah)
7. Ubah data (depan)
8. Ubah data (belakang)
9. Ubah data (tengah)
10. Tampil data
=====
Masukkan pilihan [1 - 9] -> 1

Masukkan umur -> 20

Masukkan nama -> Igor
```



```
Masukkan pilihan [1 - 9] -> 10
Daftar data :
Igor : 20
John : 19
Futaba : 18
Jane : 20
Michael : 18
Yusuke : 19
Hoshino : 18
Karin : 18
```



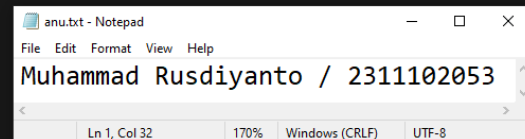
## - Ubah michael

```
Masukkan pilihan [1 - 9] -> 9

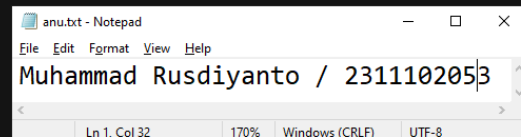
Masukkan posisi data -> 5

Masukkan umur -> 18

Masukkan nama -> Reyn
```



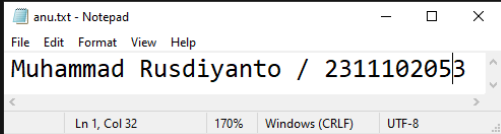
```
Masukkan pilihan [1 - 9] -> 10
Daftar data :
Igor : 20
John : 19
Futaba : 18
Jane : 20
Reyn : 18
Yusuke : 19
Hoshino : 18
Karin : 18
```





## - Tampil data

```
===== [ List perintah ] =====
1. Masukkan data (depan)
2. Masukkan data (belakang)
3. Masukkan data (tengah)
4. Hapus data (depan)
5. Hapus data (belakang)
6. Hapus data (tengah)
7. Ubah data (depan)
8. Ubah data (belakang)
9. Ubah data (tengah)
10. Tampil data
=====
Masukkan pilihan [1 - 9] -> 10
Daftar data :
Igor : 20
John : 19
Futaba : 18
Jane : 20
Reyn : 18
Yusuke : 19
Hoshino : 18
Karin : 18
```



## Deskripsi:

Program di atas merupakan program pendataan mahasiswa yang menggunakan Single Linked List. Program ini kurang lebih sama dengan program Guided 1, hanya saja terdapat beberapa modifikasi. Beberapa modifikasi program di atas adalah :

1. Mengubah struktur data Node yang awalnya hanya menyimpan `int data` menjadi `string nama` dan `int umur`. Dalam kata lain, sekarang Node bisa menyimpan nama dan umur mahasiswa.
2. Menambahkan menu program.
3. Memodifikasi fungsi supaya bisa memproses nama dan umur mahasiswa, baik itu untuk membuat, mengubah, atau menghapus Node.
4. Program sekarang bisa menerima input pengguna untuk navigasi menu, memasukan data, menghapus data, dan mengubah data. Input diproses melalui fungsi `inNama`, `inUmur`, dan `inIndex`, supaya kode program menjadi lebih ringkas.

## Unguided 2

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

class Node {
public:
    string name;
```

```

    int price;
    Node* prev;
    Node* next;
};

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

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

    void push(string name, int price) {
        Node* newNode = new Node;
        newNode->name = name;
        newNode->price = price;
        newNode->prev = nullptr;
        newNode->next = head;

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

        head = newNode;
    }

    void pushAt(string name, int price, int index) {
        Node* newNode = new Node;
        newNode->name = name;
        newNode->price = price;

        if (head != nullptr) {
            Node* current = head;
            int i = 0;
            while (current != nullptr && i < index - 1) {
                current = current->next;
                i++;
            }

```

```

        if (current == nullptr) {
            newNode->next = nullptr;
            newNode->prev = tail;
            tail->next = newNode;
            tail = newNode;
        } else {
            newNode->next = current;
            newNode->prev = current->prev;
            current->prev->next = newNode;
            current->prev = newNode;
        }
    } else {
        push(name, price);
    }
}

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 index) {
    if (head == nullptr) {
        return;
    }
    Node* temp = head;

    int i = 0;
    while (temp != nullptr && i < index - 1) {
        temp = temp->next;
        i++;
    }
}

```

```

        if (temp == nullptr) {
            return;
        }

        if (temp->prev != nullptr && temp->next != nullptr) {
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
        }

        delete temp;
    }

    bool update(string oldName, string newName, int oldprice, int
newprice) {
        Node* current = head;

        while (current != nullptr) {
            if (current->price == oldprice && current->name ==
oldName) {
                current->name = newName;
                current->price = newprice;
                return true;
            }
            current = current->next;
        }
        return false;
    }

    bool updateAt(string newName, int newprice, int index) {
        Node* current = head;

        int i = 0;
        while (current != nullptr && i < index - 1) {
            current = current->next;
            i++;
        }
        if (current != nullptr) {
            current->name = newName;
            current->price = newprice;
            return true;
        }
    }

```

```

        return false;
    }

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

    void display() {
        string line = string(51, '-');
        Node* current = head;
        cout << line << endl;
        cout << setw(32) << left << "| Nama Produk " << setw(18)
<< left << "| Harga " << "|" << endl;
        cout << line << endl;
        while (current != nullptr) {
            cout << "|" << setw(30) << left << current->name <<
            "|" << setw(16) << left << current->price << "|" << endl;
            current = current->next;
        }
        cout << line << endl;
    }
};

int main() {
    DoublyLinkedList list;
    string indent = string(3, ' ');
    string name, nameNew;
    int price, priceNew, index;
    while (true) {
        cout << "Toko Skincare Purwokerto" << endl;
        cout << indent << "1. Tambah data" << endl;
        cout << indent << "2. Hapus data" << endl;
        cout << indent << "3. Update data" << endl;
        cout << indent << "4. Tambah data urutan tertentu" <<
endl;
        cout << indent << "5. Hapus data urutan tertentu" << endl;
    }
}

```

```

        cout << indent << "6. Hapus semua data" << endl;
        cout << indent << "7. Tampilkan data" << endl;
        cout << indent << "8. Exit" << endl;

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

        switch (choice) {
            case 1: {
                cout << "Masukkan nama produk : ";
                cin >> name;
                cout << "Masukkan harga produk : ";
                cin >> price;
                list.push(name, price);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                cout << "Masukkan nama produk lama : ";
                cin >> name;
                cout << "Masukkan harga produk lama : ";
                cin >> price;
                cout << "Masukkan nama produk baru : ";
                cin >> nameNew;
                cout << "Masukkan harga produk baru : ";
                cin >> priceNew;
                bool updated = list.update(name, nameNew, price,
priceNew);

                if (!updated) {
                    cout << "Data not found" << endl;
                }
                break;
            }
            case 4: {
                cout << "Masukkan nama produk : ";
                cin >> name;
                cout << "Masukkan harga produk : ";
                cin >> price;
                cout << "Masukkan urutan produk : ";

```

```

        cin >> index;
        list.pushAt(name, price, index);
        break;
    }
    case 5: {
        cout << "Masukkan urutan produk : ";
        cin >> index;
        list.popAt(index);
        break;
    }
    case 6: {
        list.deleteAll();
        break;
    }
    case 7: {
        list.display();
        break;
    }
    case 8: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}

return 0;
}

```

## Screenshots Output

- Data awal

Enter your choice: 7

Nama Produk	Harga
Originote	60000
Somethinc	150000
Skintific	100000
Wardah	50000
Hanasui	30000

anu.txt - Notepad

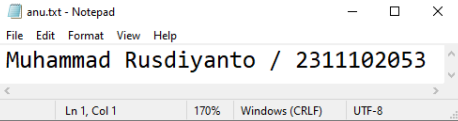
File Edit Format View Help

Muhammad Rusdiyanto / 2311102053

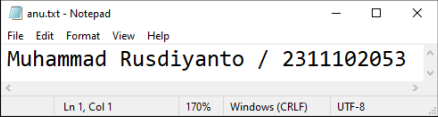
Ln 1, Col 1 170% Windows (CRLF) UTF-8

## - Tambah azarine

```
Toko Skincare Purwokerto
1. Tambah data
2. Hapus data
3. Update data
4. Tambah data urutan tertentu
5. Hapus data urutan tertentu
6. Hapus semua data
7. Tampilkan data
8. Exit
Enter your choice: 4
Masukkan nama produk : Azarine
Masukkan harga produk : 65000
Masukkan urutan produk : 3
```

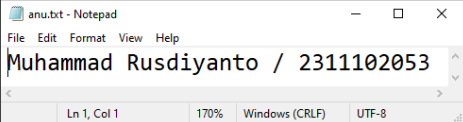


Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Wardah	50000
Hanasui	30000



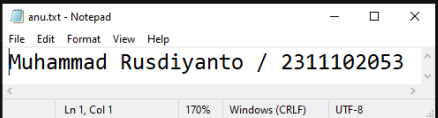
## - Hapus wardah

```
Toko Skincare Purwokerto
1. Tambah data
2. Hapus data
3. Update data
4. Tambah data urutan tertentu
5. Hapus data urutan tertentu
6. Hapus semua data
7. Tampilkan data
8. Exit
Enter your choice: 5
Masukkan urutan produk : 3
```



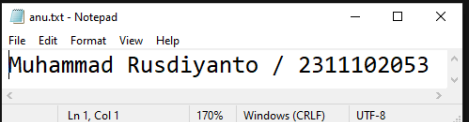
Enter your choice: 7

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Hanasui	30000



## - Update hanasui

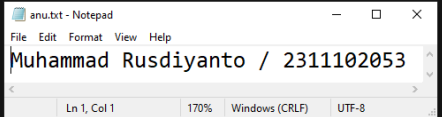
```
Toko Skincare Purwokerto
1. Tambah data
2. Hapus data
3. Update data
4. Tambah data urutan tertentu
5. Hapus data urutan tertentu
6. Hapus semua data
7. Tampilkan data
8. Exit
Enter your choice: 3
Masukkan nama produk lama : Hanasui
Masukkan harga produk lama : 30000
Masukkan nama produk baru : Cleora
Masukkan harga produk baru : 55000
```





```
Enter your choice: 7

-----
| Nama Produk      | Harga      |
-----
| Originote        | 60000      |
| Somethinc         | 150000     |
| Azarine          | 65000      |
| Skintific         | 100000     |
| Cleora           | 55000      |
-----
```



- Tampil menu

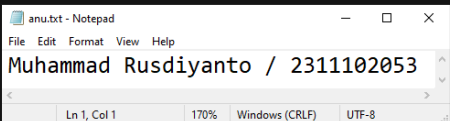
```
Toko Skincare Purwokerto
1. Tambah data
2. Hapus data
3. Update data
4. Tambah data urutan tertentu
5. Hapus data urutan tertentu
6. Hapus semua data
7. Tampilkan data
8. Exit
```



- Tampil data

```
Enter your choice: 7

-----
| Nama Produk      | Harga      |
-----
| Originote        | 60000      |
| Somethinc         | 150000     |
| Azarine          | 65000      |
| Skintific         | 100000     |
| Cleora           | 55000      |
-----
```



Deskripsi:

Program di atas merupakan program katalog toko skincare yang menggunakan Double Linked List. Program tersebut merupakan modifikasi dari program Guided 2 dengan penambahan fungsi, modifikasi fungsi lama, dan modifikasi struktur data. Selengkapnya, perubahan yang telah dilakukan adalah :

1. Mengubah struktur data Node layaknya Unguided 1, hanya saja untuk konteks ini menggunakan `name` untuk nama produk dan `price` untuk harga produk.
2. Memodifikasi fungsi lama untuk bisa memproses struktur data yang baru.
3. Memodifikasi output supaya menyerupai sebuah tabel.
4. Menambahkan fungsi `pushAt` (menambahkan berdasarkan index yang dimasukan), `popAt` (menghapus berdasarkan index), dan `updateAt` (mengubah berdasarkan index).

## **D. Kesimpulan**

Single linked list adalah struktur data dengan simpul yang saling terhubung secara berurutan, dengan setiap simpul hanya memiliki satu pointer ke simpul berikutnya. Hal ini efisien untuk operasi di ujung depan list, tetapi kurang efisien untuk pencarian acak. Sementara itu, double linked list memiliki dua pointer di setiap simpul, memungkinkan akses maju dan mundur yang efisien, cocok untuk operasi tengah list, namun memerlukan lebih banyak memori. Pilihan antara keduanya tergantung pada situasi dan kondisi pengembangan aplikasi. Penggunaan single linked list cocok untuk operasi di ujung list dan double linked list untuk akses maju dan mundur yang efisien.

## **E. Referensi**

Asisten Praktikum, “Modul 3 Single and Double Linked List”

Reema Thareja. (2014). Data Structures using C. OXFORD. New Delhi.