

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**MODUL 3  
“SINGLE AND DOUBLE LINKED LIST”**



**Disusun Oleh :**

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas : 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

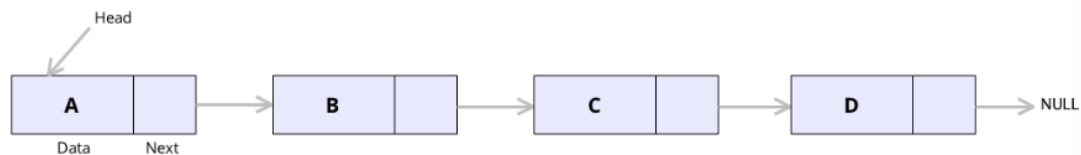
Linked List adalah struktur data linier yang elemen-elemennya dihubungkan menggunakan pointer, bukan disimpan berdekatan satu sama lain. Linked list membentuk serangkaian node yang terhubung, di mana setiap node menyimpan data dan alamat node berikutnya. Linked list terdiri dari:

- Struktur node : Sebuah node dalam linked list biasanya terdiri dari dua komponen:
- Data : berisi nilai aktual atau data yang terkait dengan node tersebut.
- Next Pointer : Menyimpan alamat memori (referensi) node berikutnya dalam urutan.
- Head and Tail : Linked list diakses melalui node head yang menunjuk ke node pertama dalam daftar. Node terakhir dari daftar menunjuk ke NULL atau nullptr, yang menunjukkan akhir dari daftar. Node ini disebut node akhir.

Dalam linked list juga terdapat beberapa jenis diantaranya yaitu single linked list dan double linked list.

### ➤ Single linked list

Single link list merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya dan pointer pada tail menunjuk ke NULL. Single linked list terdiri dari sejumlah elemen (node) dimana setiap node memiliki penunjuk berikutnya ke elemen (node) berikutnya. Penunjuk node terakhir adalah NULL, yang menunjukkan akhir dari single linked list.

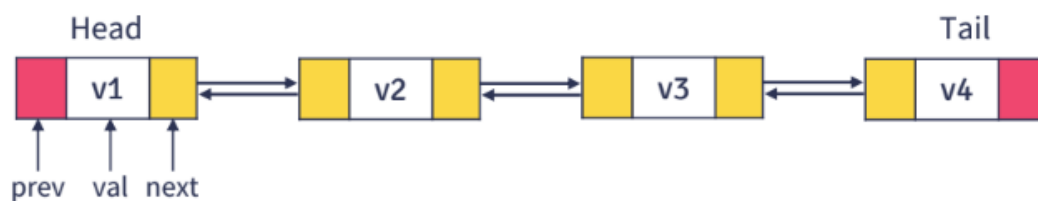


Karakteristik Single linked list :

- a. Setiap node menyimpan nilai dan referensi ke node berikutnya dalam daftar.
- b. Daftar mempunyai head yang menunjuk ke node pertama dalam daftar dan tail yang menunjuk ke node terakhir dalam daftar.
- c. Node tidak disimpan dalam blok memori yang berdekatan melainkan, setiap node menyimpan alamat node berikutnya dalam daftar.
- d. Mengakses elmen dalam single linked list memerlukan penelusuran daftar dari head ke node yang diinginkan, karena tidak ada akses langsung ke node tertentu di memori.

### ➤ Double linked liist

Double Linked List terdiri dari sejumlah elemen (node) yang terhubung satu sama lain melalui 2 pointer. Dalam double linked list suatu linked list memiliki 2 variabel pointer yang menunjuk ke node sebelum dan selanjutnya (prev & next). penunjuk pada node head menunjuk ke NULL, menandakan bahwa node head (node awal). dan juga penunjuk next pada tail menunjuk ke NULL, menandakan bahwa node tail (node akhir) atau bisa di artikan tidak ada yang di tunjuk setelahnya



Karakteristik Doubly linked list :

- Traversal maju dan mundur: Memungkinkan traversal maju dan mundur melalui node.
- Alokasi Dinamis: Mengaktifkan alokasi memori dinamis untuk menambah dan menghapus elemen.
- Memori Tambahan: Memori tambahan diperlukan untuk penunjuk ke node sebelumnya.
- Operasi efisien: Performa bagus saat menambah dan menghapus item di tengah daftar.
- Operasi fleksibel: Cocok untuk operasi manipulasi data seperti menyisipkan, menghapus, menyusun ulang, dan membalik elemen.

## B. GUIDED

### 1. Guided I

#### 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;
}

```

Ouput Program :

The screenshot shows a Windows command prompt window with the following output:

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\
tempCodeRunnerFile }
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:\strukdat>

```

Overlaid on the terminal is a Notepad++ window titled 'Nama Fitr'. The text inside the window is:

```

Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B

```

The status bar at the bottom of the Notepad++ window indicates 'Ln 3, Col 15 | 60 characters | 100% | Window | UTF-8'.

Deskripsi Program :

- `init()`: Fungsi ini menginisialisasi awal dan akhir linked list sebagai NULL.
- `isEmpty()`: Fungsi ini memeriksa apakah linked list kosong dengan memeriksa apakah head memiliki nilai NULL.
- `insertFront(int value)`: Fungsi ini menambahkan node baru dengan nilai yang ditentukan ke depan linked list.
- `insertBack(int value)`: Fungsi ini menambahkan node baru dengan nilai yang ditentukan ke akhir linked list.



- HitungList(): Fungsi ini menghitung jumlah node dalam linked list.
- insertMiddle(int data, int Position): Fungsi ini menambahkan node baru dengan nilai yang ditentukan pada posisi yang ditentukan dalam linked list.
- deleteFront(): Fungsi ini menghapus node pertama dari linked list.
- deleteBack(): Fungsi ini menghapus node terakhir dari linked list.
- deleteMiddle(int Position): Fungsi ini menghapus node pada posisi yang ditentukan dari linked list.
- changeFront(int data): Fungsi ini memperbarui nilai node pertama dari linked list.
- changeCenter(int data, int Position): Fungsi ini memperbarui nilai node pada posisi yang ditentukan dalam linked list.
- changeBack(int data): Fungsi ini memperbarui nilai node terakhir dalam linked list.
- clearList(): Fungsi ini menghapus semua node dari linked list.
- display(): Fungsi ini menampilkan semua nilai node dalam linked list.
- Fungsi main() menggunakan beberapa fungsi ini dengan melakukan hal berikut:
  - Tambahkan elemen 3, 5, 2, dan 1 ke awal linked list.
  - linked list pada setiap penambahan.
  - Item dihapus dari linked list.
  - Menambahkan elemen pada posisi tertentu.
  - Mengembalikan linked list pada setiap penambahan, penghapusan, dan pembaruan.
  - Output program menampilkan linked list setelah setiap operasi dilakukan.

## 2. Guided II

### 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;

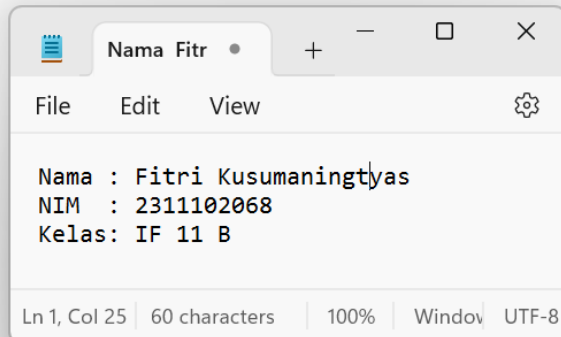
        switch (choice) {
            case 1: {
                int data;
                cout << "Enter data to add: ";
                cin >> data;
                list.push(data);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                int oldData, newData;

```

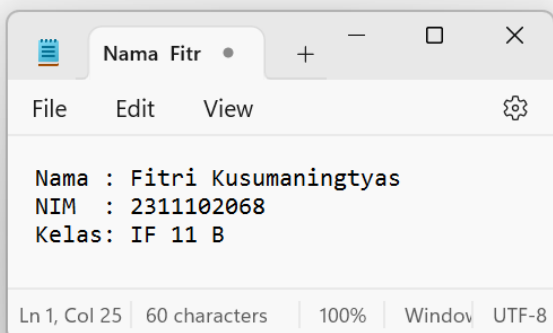
```
        cout << "Enter old data: ";
        cin >> oldData;
        cout << "Enter new data: ";
        cin >> newData;
        bool updated = list.update(oldData, newData);
        if (!updated) {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4: {
       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

## Output Program :

```
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak3g2dll.cpp -o prak3g2dll } ; if ($?) { .\prak3g2dll }
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: 17
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 17
Enter new data: 30
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
30
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: 1
Enter data to add: 87
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
87
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS D:\strukdat> 
```



Deskripsi Program :

Program ini merupakan implementasi dari linked list ganda (dua arah) di C++. Di dalamnya ada kelas Node yang mewakili setiap elemen dari linked list. Setiap node memiliki tiga properti: data, yang menyimpan nilai elemen, prev, yang menyimpan pointer ke node sebelumnya, dan next, yang menyimpan pointer ke node berikutnya. Selain itu, ada kelas DoublyLinkedList yang mewakili linked list asli. Kelas ini memiliki lima fungsi public yaitu:

- Push(int data): Fungsi ini digunakan untuk menambahkan node baru ke linked list. Node baru selalu ditambahkan di awal.
- Pop(): Fungsi ini menghapus node pertama dari linked list.
- update(int oldData, int newData): Fungsi ini digunakan untuk mengganti nilai node yang berisi data tertentu dengan data baru.
- deleteAll(): Fungsi ini menghapus semua node dari linked list.
- display(): Fungsi ini mencetak semua data dalam linked list.

Di dalam main() terdapat loop yang menampilkan daftar menu hingga pengguna keluar dari program.

- Pengguna dapat memilih beberapa opsi yaitu Tambahkan data ke linked list.
- Menghapus data pertama dari linked list.
- Mengganti nilai data tertentu dengan data baru.
- Menghapus semua data dari linked list.
- Menampilkan semua data dalam linked list.
- Menghentikan program.

Pengguna diminta memasukkan opsi yang diinginkan dan diarahkan ke fungsi yang sesuai. Saat program berjalan, pengguna dapat melakukan operasi pada linked list dan melihat hasilnya.

## C. UNGUIDED

### 1. Unguided 1

#### Source Code

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

struct node
{
    string name;
    int age;
    node* next;
};

class linked_list
{
private:
    node* head, * tail;
public:
    linked_list()
    {
        head = NULL;
        tail = NULL;
    }
// tambah awal
    void add_first(string name, int age)
    {
        node* temp = new node;
        temp->name = name;
        temp->age = age;
        temp->next = head;
        head = temp;
        if (tail == NULL)
        {
            tail = temp;
        }
    }
// tambah akhir
    void add_last(string name, int age)
    {
        node* temp = new node;
        temp->name = name;
        temp->age = age;
        temp->next = NULL;
        if (tail == NULL)
        {
            head = temp;
        }
    }
}
```

```

        tail = temp;
    }
    else
    {
        tail->next = temp;
        tail = temp;
    }
}
// tambah tengah
void add_middle(int pos, string name, int age)
{
    node* prev = new node;
    node* cur = new node;
    node* temp = new node;
    cur = head;
    for (int i = 1; i < pos; i++)
    {
        prev = cur;
        cur = cur->next;
    }
    temp->name = name;
    temp->age = age;
    prev->next = temp;
    temp->next = cur;
}
// ubah tengah
void edit_middle(int pos, string name, int age)
{
    node* temp = new node;
    temp = head;
    for (int i = 1; i < pos; i++)
    {
        temp = temp->next;
    }
    temp->name = name;
    temp->age = age;
}
// tampilkan jumlah data
void count_data()
{
    int count = 0;
    node* temp = new node;
    temp = head;
    while (temp != NULL)
    {
        count++;
        temp = temp->next;
    }
}

```



```

        cout << "Jumlah Data: " << count << endl;
    }
// hapus data
void delete_data(int pos)
{
    node* prev = new node;
    node* cur = new node;
    cur = head;
    for (int i = 1; i < pos; i++)
    {
        prev = cur;
        cur = cur->next;
    }
    prev->next = cur->next;
}
// tampilkan data
void display()
{
    node* temp = new node;
    temp = head;
    while (temp != NULL)
    {
        cout << "Nama: " << temp->name << ", Usia: " <<
temp->age
<< endl;
        temp = temp->next;
    }
}
};
int main()
{
    linked_list l;
// input data
    int choice;
    string name;
    int age;
    int pos;
    do {
        cout << endl;
        cout << "Menu Utama: " << endl;
        cout << "1. Tambahkan Awal" << endl;
        cout << "2. Tambahkan Akhir" << endl;
        cout << "3. Tambahkan Tengah" << endl;
        cout << "4. Ubah Tengah" << endl;
        cout << "5. Hapus Data" << endl;
        cout << "6. Tampilkan Jumlah Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Keluar" << endl;
        cout << "Pilih Menu: ";
    }
}

```

```

        cin >> choice;
    switch (choice)
    {
        case 1:
            cout << "Masukkan Nama: ";
            cin >> name;
            cout << "Masukkan Usia: ";
            cin >> age;
            l.add_first(name, age);
            break;
        case 2:
            cout << "Masukkan Nama: ";
            cin >> name;
            cout << "Masukkan Usia: ";
            cin >> age;
            l.add_last(name, age);
            break;
        case 3:
            cout << "Masukkan Posisi: ";
            cin >> pos;
            cout << "Masukkan Nama: ";
            cin >> name;
            cout << "Masukkan Usia: ";
            cin >> age;
            l.add_middle(pos, name, age);
            break;
        case 4:
            cout << "Masukkan Posisi: ";
            cin >> pos;
            cout << "Masukkan Nama: ";
            cin >> name;
            cout << "Masukkan Usia: ";
            cin >> age;
            l.edit_middle(pos, name, age);
            break;
        case 5:
            cout << "Masukkan Posisi: ";
            cin >> pos;
            l.delete_data(pos);
            break;
        case 6:
            l.count_data();
            break;
        case 7:
            l.display();
            break;
        case 8:
            cout << "Program Selesai" << endl;
            break;
    }

```

```

        default:
            cout << "Pilihan tidak tersedia" << endl;
            break;
        }
    } while (choice != 8);

    return 0;
}

```

### Output Program :

```

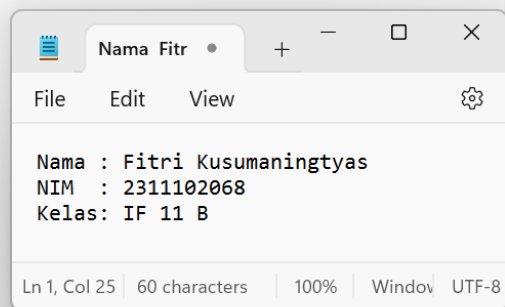
PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak3ug1.cpp -o prak3ug1
} ; if ($?) { .\prak3ug1 }

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: KARIN
Masukkan Usia: 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: HOSHINO
Masukkan Usia: 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data

```



```
8. Keluar
Pilih Menu: 1
Masukkan Nama: AKECHI
Masukkan Usia: 20

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: YUSUKE
Masukkan Usia: 19

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: MICHAEL
Masukkan Usia: 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
```

Nama Fitr

File Edit View

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas: IF 11 B

Ln 1, Col 25 | 60 characters | 100% | Window | UTF-8

0 0 0 0 Ln 200, Col 1 | Spaces: 4 | UTF-8 | CRLF | {} C++ Win32

```
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: JANE
Masukkan Usia: 20

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: JOHN
Masukkan Usia: 19

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: TYAS
Masukkan Usia: 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
```

Nama Fitr

File Edit View

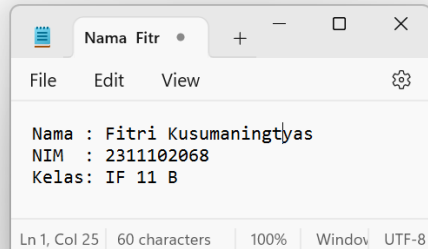
Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas: IF 11 B

Ln 1, Col 25 | 60 characters | 100% | Window | UTF-8

```
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 7
Nama: TYAS, Usia: 18
Nama: JOHN, Usia: 19
Nama: JANE, Usia: 20
Nama: MICHAEL, Usia: 18
Nama: YUSUKE, Usia: 19
Nama: AKECHI, Usia: 20
Nama: HOSHINO, Usia: 18
Nama: KARIN, Usia: 18
```

```
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 5
Masukkan Posisi: 6
```

```
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 7
```

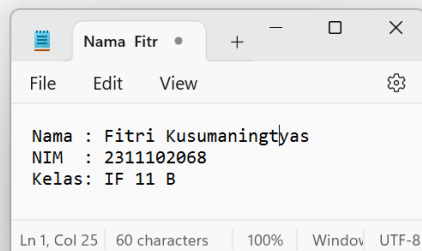


```
Nama Fitr
File Edit View
Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
Ln 1, Col 25 | 60 characters | 100% | Window UTF-8
```

```
Nama: TYAS, Usia: 18
Nama: JOHN, Usia: 19
Nama: JANE, Usia: 20
Nama: MICHAEL, Usia: 18
Nama: YUSUKE, Usia: 19
Nama: HOSHINO, Usia: 18
Nama: KARIN, Usia: 18
```

```
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 3
Masukkan Posisi: 3
Masukkan Nama: FUTABA
Masukkan Usia: 18
```

```
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 7
Nama: TYAS, Usia: 18
Nama: JOHN, Usia: 19
Nama: FUTABA, Usia: 18
Nama: JANE, Usia: 20
Nama: MICHAEL, Usia: 18
```



```
Nama Fitr
File Edit View
Nama : Fitri Kusumaningtyas
NIM : 2311102068
Kelas: IF 11 B
Ln 1, Col 25 | 60 characters | 100% | Window UTF-8
```

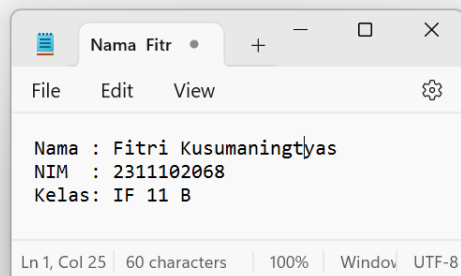
Nama: YUSUKE, Usia: 19  
Nama: HOSHINO, Usia: 18  
Nama: KARIN, Usia: 18

Menu Utama:

1. Tambahkan Awal
  2. Tambahkan Akhir
  3. Tambahkan Tengah
  4. Ubah Tengah
  5. Hapus Data
  6. Tampilkan Jumlah Data
  7. Tampilkan Data
  8. Keluar
- Pilih Menu: 1  
Masukkan Nama: IGOR  
Masukkan Usia: 20

Menu Utama:

1. Tambahkan Awal
  2. Tambahkan Akhir
  3. Tambahkan Tengah
  4. Ubah Tengah
  5. Hapus Data
  6. Tampilkan Jumlah Data
  7. Tampilkan Data
  8. Keluar
- Pilih Menu: 7  
Nama: IGOR, Usia: 20  
Nama: TYAS, Usia: 18  
Nama: JOHN, Usia: 19  
Nama: FUTABA, Usia: 18  
Nama: JANE, Usia: 20  
Nama: MICHAEL, Usia: 18  
Nama: YUSUKE, Usia: 19  
Nama: HOSHINO, Usia: 18  
Nama: KARIN, Usia: 18



Menu Utama:

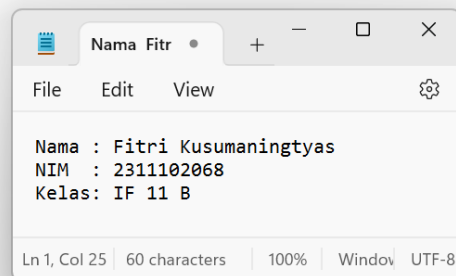
1. Tambahkan Awal
  2. Tambahkan Akhir
  3. Tambahkan Tengah
  4. Ubah Tengah
  5. Hapus Data
  6. Tampilkan Jumlah Data
  7. Tampilkan Data
  8. Keluar
- Pilih Menu: 4  
Masukkan Posisi: 6  
Masukkan Nama: REYN  
Masukkan Usia: 18

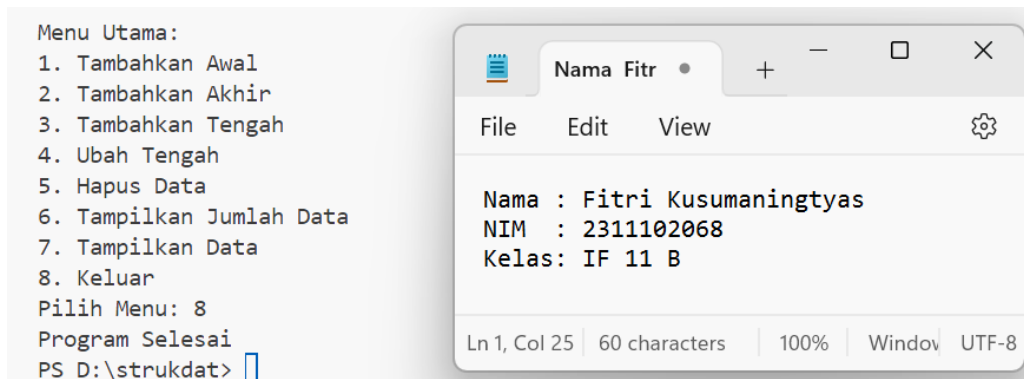
Menu Utama:

1. Tambahkan Awal
  2. Tambahkan Akhir
  3. Tambahkan Tengah
  4. Ubah Tengah
  5. Hapus Data
  6. Tampilkan Jumlah Data
  7. Tampilkan Data
  8. Keluar
- Pilih Menu: 7  
Nama: IGOR, Usia: 20  
Nama: TYAS, Usia: 18  
Nama: JOHN, Usia: 19  
Nama: FUTABA, Usia: 18  
Nama: JANE, Usia: 20  
Nama: REYN, Usia: 18  
Nama: YUSUKE, Usia: 19  
Nama: HOSHINO, Usia: 18  
Nama: KARIN, Usia: 18

Menu Utama:

1. Tambahkan Awal
2. Tambahkan Akhir





### Deskripsi Program :

Program ini memiliki kelas `Linked_list` dengan beberapa metode seperti:

- `add_first(nama string, usia int)`: digunakan untuk menambahkan data baru ke awal daftar tertaut.
- `add_last(nama string, usia int)`: digunakan untuk menambahkan data baru ke akhir daftar tertaut.
- `add_middle(int pos, nama string, int umur)`: digunakan untuk menambahkan data baru pada posisi tertentu dalam daftar tertaut.
- `edit_middle(int pos, nama string, int usia)`: digunakan untuk mengubah data pada posisi tertentu dalam daftar tertaut.
- `count_data()`: digunakan untuk menampilkan jumlah data dalam daftar tertaut.
- `delete_data(int pos)`: digunakan untuk menghapus data pada posisi tertentu dalam daftar tertaut.
- `display()`: digunakan untuk menampilkan semua data dalam daftar tertaut.

Selain itu, program ini memiliki menu utama dimana pengguna dapat memilih operasi untuk dilakukan pada daftar tertaut.

## 2. Unguided 2

### Source Code

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

class Node {
public:
    string product_name;
    float price;
    Node* prev;
    Node* next;
};
```

```

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(string product_name, float price) {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        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(string oldProduct, string newProduct, float
newPrice)
    {
        Node* current = head;
        while (current != nullptr) {
            if (current->product_name == oldProduct) {
                current->product_name = newProduct;
                current->price = newPrice;
                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->product_name << " (Rp" << current->price
        << ")" << endl;
        current = current->next;
    }
}

void insert(string product_name, float price, int position) {
    if (position <= 0) {
        push(product_name, price);
        return;
    }
    Node* current = head;
    for (int i = 1; i < position && current != nullptr; i++) {
        current = current->next;
    }
    if (current == nullptr) {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        newNode->prev = tail;
        newNode->next = nullptr;
        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    } else {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        newNode->prev = current->prev;
        newNode->next = current;
    }
}

```

```

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

void remove(int position) {
    if (position <= 0) {
        pop();
        return;
    }
    Node* current = head;
    for (int i = 1; i < position && current != nullptr; i++) {
        current = current->next;
    }
    if (current == nullptr) {
        return;
    }
    if (current == head) {
        head = current->next;
    } else {
        current->prev->next = current->next;
    }

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

int main()
{
    DoublyLinkedList list;
    int choice;
    string productName, newProductName;
    float price, newPrice;
    int position;

    do {
        cout << "1. tambah data\n";
        cout << "2. hapus data\n";
        cout << "3. Update data \n";
        cout << "4. tambah data urutan tertentu insert\n";
        cout << "5. hapus data urutan tertentu remove\n";
        cout << "6. hapus seluruh data\n";
    }
}

```

```

cout << "7. tampilkan data\n";
cout << "8. Exit\n";
cout << "Enter your choice: ";
cin >> choice;

switch (choice) {
case 1:
    cout << "masukan nama produk: ";
    cin >> productName;
    cout << "masukan harga: ";
    cin >> price;
    list.push(productName, price);
    break;
case 2:
    list.pop();
    break;
case 3:
    cout << "masukan nama produk yang lama: ";
    cin >> productName;
    cout << "masukan nama produk yang baru: ";
    cin >> newProductName;
    cout << "masukan harga baru: ";
    cin >> newPrice;
    if (list.update(productName, newProductName,
newPrice)){
        cout << "Produk berhasil ditambahkan\n";
    } else {
        cout << "produk tidak ada\n";
    }
    break;
case 4:
    cout << "masukan nama produk: ";
    cin >> productName;
    cout << "masukan harga: ";
    cin >> price;
    cout << "masukan posisi: ";
    cin >> position;
    list.insert(productName, price, position);
    break;
case 5:
    cout << "masukan urutan ke-: ";
    cin >> position;
    list.remove(position);
    break;
case 6:
    list.deleteAll();
    cout << "semua produk berhasil di hapus\n";
    break;
}

```

```

        case 7:
            list.display();
            break;
        case 8:
            cout << "keluar dari progam...\n";
            break;
        default:
            cout << "pilihan salah\n";
            break;
    }
} while (choice != 8);
return 0;
}

```

### Output Program :

```

PS D:\strukdat> cd "d:\strukdat\" ; if ($?) { g++ prak3ug2.cpp -o prak3ug2
} ; if ($?) { .\prak3ug2 }
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: HANASUI
masukan harga: 30000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: WARDAH
masukan harga: 50000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: SKINTIFIC
masukan harga: 100000

```

Nama Fitr • + - □  
 File Edit View  
 Nama : Fitri Kusumaningtyas  
 NIM : 2311102068  
 Kelas: IF 11 B  
 Ln 1, Col 25 | 60 characters | 100% | Windo

```
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: SOMETHINC
masukan harga: 150000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 1
masukan nama produk: ORIGINOTE
masukan harga: 60000
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 7
ORIGINOTE (Rp60000)
SOMETHINC (Rp150000)
SKINTIFIC (Rp100000)
WARDAH (Rp50000)
HANASUI (Rp30000)
```

Nama Fitr

File Edit View

Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas: IF 11 B

Ln 1, Col 25 | 60 characters | 100% | Window

```
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 4
masukan nama produk: AZARINE
masukan harga: 65000
masukan posisi: 3
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 7
ORIGINOTE (Rp60000)
SOMETHINC (Rp150000)
AZARINE (Rp65000)
SKINTIFIC (Rp100000)
WARDAH (Rp50000)
HANASUI (Rp30000)
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
```

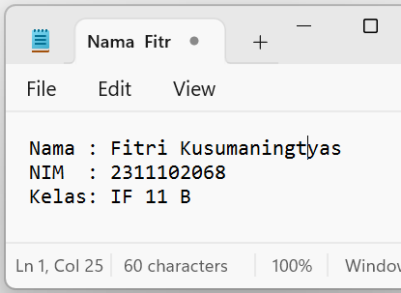
Nama Fitr

File Edit View

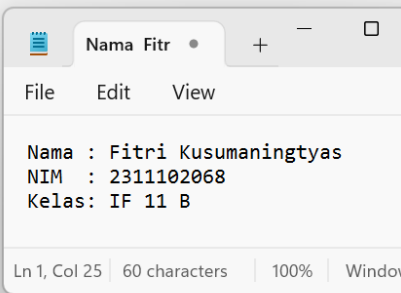
Nama : Fitri Kusumaningtyas  
NIM : 2311102068  
Kelas: IF 11 B

Ln 1, Col 25 | 60 characters | 100% | Window

```
Enter your choice: 5
masukan urutan ke-: 5
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 7
ORIGINOTE (Rp60000)
SOMETHINC (Rp150000)
AZARINE (Rp65000)
SKINTIFIC (Rp100000)
HANASUI (Rp30000)
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 3
masukan nama produk yang lama: HANASUI
masukan nama produk yang baru: CLEORA
masukan harga baru: 55000
Produk berhasil ditambahkan
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
```



```
Enter your choice: 7
ORIGINOTE (Rp60000)
SOMETHINC (Rp150000)
AZARINE (Rp65000)
SKINTIFIC (Rp100000)
CLEORA (Rp55000)
1. tambah data
2. hapus data
3. Update data
4. tambah data urutan tertentu insert
5. hapus data urutan tertentu remove
6. hapus seluruh data
7. tampilkan data
8. Exit
Enter your choice: 8
keluar dari program...
PS D:\strukdat>
```



### Deskripsi Program :

Program ini memiliki kelas Node dan kelas DoublyLinkedList. Kelas Node memiliki tiga atribut: nama produk, harga, dan penunjuk ke node sebelumnya dan berikutnya. Kelas DoublyLinkedList memiliki beberapa metode yaitu :

- Push(string product\_name, float Price): digunakan untuk menambahkan data baru ke awal daftar tertaut.
- Pop(): digunakan untuk menghapus data pertama dalam daftar tertaut.
- Pembaruan (string oldProduct, string newProduct, float newPrice): digunakan untuk mengubah nama dan harga produk.

- `deleteAll()`: digunakan untuk menghapus semua data dalam daftar tertaut.
- `display()`: digunakan untuk menampilkan semua data dalam daftar tertaut.
- `insert(string product_name, float Price, int Position)`: digunakan untuk memasukkan data baru pada posisi tertentu dalam daftar tertaut.
- `delete (int position)`: digunakan untuk menghapus data pada posisi tertentu dalam daftar tertaut.

Selain itu, program ini memiliki menu utama dimana pengguna dapat memilih operasi untuk dilakukan pada daftar tertaut. Menu ini terdiri dari delapan pilihan: Tambah data, Hapus data, Ubah data, Tambah data di lokasi tertentu, Hapus data di lokasi tertentu, Hapus semua data, Lihat data, dan Keluar dari program.

#### D. KESIMPULAN

Single Linked List (SLL): SLL adalah struktur data sederhana yang terdiri dari node yang terhubung satu sama lain melalui pointer.

- Setiap node terdiri dari dua bagian: data dan penunjuk ke node berikutnya.
- Hanya mendukung gerakan maju dari awal hingga akhir.
- Operasi penambahan dan penghapusan di awal daftar sangat efisien, namun operasi di tengah atau di belakang memerlukan jalur dari awal daftar.
- Overhead memori rendah karena hanya ada satu penunjuk pada setiap node.

Double Linked List (DLL): DLL merupakan evolusi dari SLL, dimana setiap node mempunyai pointer tambahan ke node sebelumnya.

- Memungkinkan penjelajahan maju dan mundur, meningkatkan fleksibilitas dalam manipulasi data.
- Menambah dan menghapus item di tengah daftar lebih efisien karena Anda tidak harus memulai dari awal daftar.
- Pointer tambahan memerlukan memori tambahan, sehingga overhead memori lebih tinggi.
- Lebih kompleks untuk diimplementasikan dibandingkan SLL.

Kedua jenis daftar tertaut memiliki kegunaan dan manfaatnya masing-masing, bergantung pada persyaratan aplikasi dan kompleksitas operasi yang dilakukan. SLL cocok untuk aplikasi yang memerlukan overhead memori rendah dan hanya traversal maju, sedangkan DLL cocok untuk aplikasi yang memerlukan traversal maju dan mundur serta penambahan dan penghapusan daftar tengah yang efisien.



## E. REFERESNSI JURNAL

Almadhoun, E., & Parham-Mocello, J. (2021, October). Exploratory Study on Accuracy of Students' Mental Models of a Singly Linked List. In *2021 IEEE Frontiers in Education Conference (FIE)* (pp. 1-9). IEEE.

Almadhoun, E., & Parham-Mocello, J. (2023). Students' difficulties with inserting and deleting nodes in a singly linked list in the C programming language. *Journal of Computer Languages*, 74, 101184.

Sinha, P. (2005). A memory-efficient doubly linked list. *Linux Journal*, 2005(129), 10.

Sundell, H., & Tsigas, P. (2008). Lock-free dequeues and doubly linked lists. *Journal of Parallel and Distributed Computing*, 68(7), 1008-1020.

Karimov, E., & Karimov, E. (2020). Linked Lists. *Data Structures and Algorithms in Swift: Implement Stacks, Queues, Dictionaries, and Lists in Your Apps*, 41-54.