LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA

MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR

DISUSUN OLEH:
MOHAMMAD HARITS. T
2311102016
S1 IF-11-A

DOSEN PENGAMPU:

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

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

2024

BAB I

TUJUAN PRAKTIKUM

1. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
2. Praktikan dapat membuat linked list circular dan non circular.
3. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

DASAR TEORI

1. Linked List Non Circular

   Linked list non circular merupakan linked list dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada Linked List ini selalu bernilai ‘NULL’ sebagai pertanda data terakhir dalam list-nya.

2. Linked List Circular

   Linked list circular merupakan linked list yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai ‘NULL’, tetapi terhubung dengan node pertama (head). Saat menggunakan linked list circular kita membutuhkan dummy node atau node pengecoh yang biasanya dinamakan dengan node current supaya program dapat berhenti menghitung data ketika node current mencapai node pertama (head). Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi.

BAB III

GUIDED

Guided 1

Source

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

// Program for a non-circular single linked list

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

Node* head;
Node* tail;

// Initialize the head and tail pointers to NULL
void init() {
    head = NULL;
    tail = NULL;
}

// Check if the linked list is empty
bool isEmpty() {
    if (head == NULL) {
        return true;
    } else {
        return false;
    }
}

// Add a new node at the beginning of the list
void insertDepan(int value) {
    Node* baru = new Node();
    baru->data = value;
    baru->next = NULL;

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

```
        head = baru;
    }
}

// Add a new node at the end of the list
void insertBelakang(int value) {
    Node* baru = new Node();
    baru->data = value;
    baru->next = NULL;

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

// Count the number of nodes in the list
int hitungList() {
    Node* hitung = head;
    int count = 0;

    while (hitung != NULL) {
        count++;
        hitung = hitung->next;
    }

    return count;
}

// Insert a new node in the middle of the list
void insertTengah(int data, int position) {
    if (position < 1 || position > hitungList()) {
        cout << "Position out of range" << endl;
    } else if (position == 1) {
        cout << "Position is not in the middle" << endl;
    } else {
        Node* baru = new Node();
        baru->data = data;

        Node* traversal = head;
        int nomor = 1;

        while (nomor < position - 1) {
            traversal = traversal->next;
            nomor++;
```

```
        }

        baru->next = traversal->next;
        traversal->next = baru;
    }
}

// Delete the first node in the list
void hapusDepan() {
    Node* toDelete;

    if (isEmpty() == false) {
            if (head->next != NULL) {
                toDelete = head;
                head = head->next;
                delete toDelete;
            } else {
                head = tail = NULL;
            }
        } else {
            cout << "Linked list is still empty" << endl;
        }
    }

    // Delete the last node in the list
    void hapusBelakang() {
        Node* toDelete;
        Node* traversal;

        if (isEmpty() == false) {
            if (head != tail) {
                toDelete = tail;
                traversal = head;

                while (traversal->next != tail) {
                    traversal = traversal->next;
                }

                tail = traversal;
                tail->next = NULL;
                delete toDelete;
            } else {
                head = tail = NULL;
            }
        } else {
            cout << "Linked list is still empty" << endl;
        }
    }
```

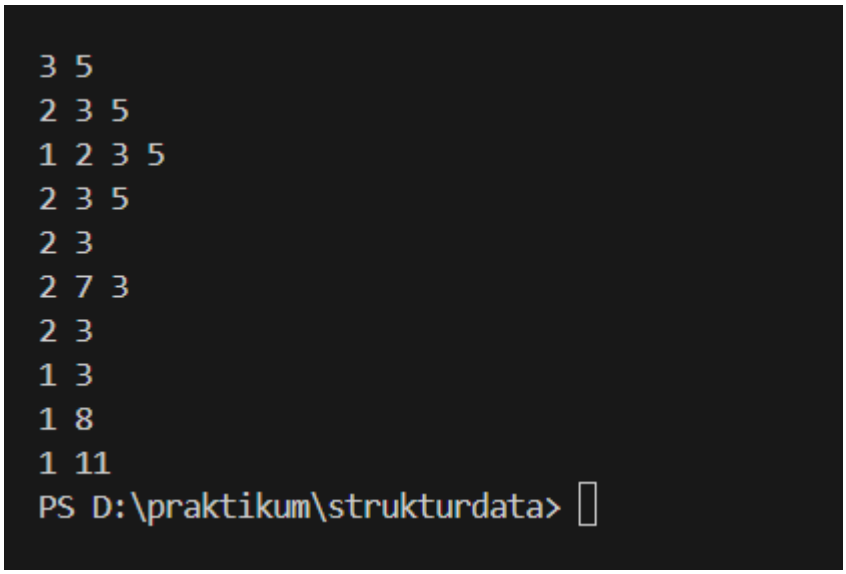
```
136.    // Hapus tengah
137.    void hapusTengah(int posisi)
138.    {
139.        Node *hapus, *bantu, *sebelum;
140.        if (posisi < 1 || posisi > hitungList())
141.        {
142.            cout << "Posisi di luar jangkauan" << endl;
143.        }
144.        else if (posisi == 1)
145.        {
146.            cout << "Posisi bukan posisi tengah" << endl;
147.        }
148.        else
149.        {
150.            int nomor = 1;
151.            bantu = head;
152.            while (nomor <= posisi)
153.            {
154.                if (nomor == posisi - 1)
155.                {
156.                    sebelum = bantu;
157.                }
158.                if (nomor == posisi)
159.                {
160.                    hapus = bantu;
161.                }
162.                bantu = bantu->next;
163.                nomor++;
164.            }
165.            sebelum->next = bantu;
166.            delete hapus;
167.        }
168.    }
169.
170.    // ubah depan
171.    void ubahDepan(int data)
172.    {
173.        if (isEmpty() == 0)
174.        {
175.            head->data = data;
176.        }
177.        else
178.        {
179.            cout << "Linked list masih kosong" << endl;
180.        }
181.    }
182.
183.    // ubah tengah
```

```
184.    void ubahTengah
185.    (int data, int posisi)
186.    {
187.        Node *bantu;
188.        if (isEmpty() == 0)
189.        {
190.            if (posisi < 1 || posisi > hitungList())
191.            {
192.                cout << "Posisi di luar jangkauan" << endl;
193.            }
194.            else if (posisi == 1)
195.            {
196.                cout << "Posisi bukan posisi tengah" << endl;
197.            }
198.            else
199.            {
200.                int nomor = 1;
201.                bantu = head;
202.                while (nomor < posisi)
203.                {
204.                    bantu = bantu->next;
205.                    nomor++;
206.                }
207.                bantu->data = data;
208.            }
209.        }
210.        else
211.        {
212.            cout << "Linked list masih kosong" << endl;
213.        }
214.    }
215.
216.    // ubah belakang
217.    void ubahBelakang(int data)
218.    {
219.        if (isEmpty() == 0)
220.        {
221.            tail->data = data;
222.        }
223.        else
224.        {
225.            cout << "Linked list masih kosong" << endl;
226.        }
227.    }
228.
229.    // Hapus list
230.    void clearList()
231.    {
```

```
232.        Node *bantu, *hapus;
233.        bantu = head;
234.        while (bantu != NULL)
235.        {
236.            hapus = bantu;
237.            bantu = bantu->next;
238.            delete hapus;
239.        }
240.        head = tail = NULL;
241.        cout << "List berhasil terhapus!" << endl;
242.    }
243.
244.    // Tampilkan list
245.    void tampilList()
246.    {
247.        Node *bantu;
248.        bantu = head;
249.        if (isEmpty() == false)
250.        {
251.            while (bantu != NULL)
252.            {
253.                cout << bantu->data << " ";
254.                bantu = bantu->next;
255.            }
256.            cout << endl;
257.        }
258.        else
259.        {
260.            cout << "Linked list masih kosong" << endl;
261.        }
262.    }
263.
264.    int main()
265.    {
266.        init();
267.        insertDepan(3);
268.        tampilList();
269.        insertBelakang(5);
270.        tampilList();
271.        insertDepan(2);
272.        tampilList();
273.        insertDepan(1);
274.        tampilList();
275.        hapusDepan();
276.        tampilList();
277.        hapusBelakang();
278.        tampilList();
279.        insertTengah(7, 2);
```

```
280.        tampilList();
281.        hapusTengah(2);
282.        tampilList();
283.        ubahDepan(1);
284.        tampilList();
285.        ubahBelakang(8);
286.        tampilList();
287.        ubahTengah(11, 2);
288.        tampilList();
289.
290.        return 0;
291.    }
```

Screenshoot program



Deskripsi program

Program ini menggunakan linked list non-circular dan memiliki fitur untuk menambah, menghapus, mengubah. Program tersebut juga menggunakan beberapa fungsi if, else if, dan while do. Selain itu, program tersebut juga menggunakan Struct.

Guided 2

Source code

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

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

Node* head = nullptr;
Node* tail = nullptr;
Node* baru = nullptr;
Node* bantu = nullptr;
Node* hapus = nullptr;

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

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

// Membuat node baru
void buatNode(string data) {
    baru = new Node;
    baru->data = data;
    baru->next = nullptr;
}

// Menghitung jumlah node dalam list
int hitungList() {
    int jumlah = 0;
    bantu = head;
    while (bantu != nullptr) {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

// Menambah node di depan list
```

```
void insertDepan(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = baru;
        baru->next = head;
    } else {
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

// Menambah node di belakang list
void insertBelakang(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = baru;
        baru->next = head;
    } else {
        tail->next = baru;
        baru->next = head;
        tail = baru;
    }
}

// Menambah node di tengah list pada posisi tertentu
void insertTengah(string data, int posisi) {
    if (isEmpty() || posisi == 1) {
        insertDepan(data);
    } else if (posisi > hitungList()) {
        insertBelakang(data);
    } else {
        baru = new Node;
        baru->data = data;
        bantu = head;
        for (int i = 1; i < posisi - 1; i++) {
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Menghapus node di depan list
void hapusDepan() {
    if (!isEmpty()) {
```

```
        hapus = head;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        } else {
            head = head->next;
            tail->next = head;
        }
        delete hapus;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Menghapus node di belakang list
void hapusBelakang() {
    if (!isEmpty()) {
        hapus = head;
        if (head == tail) {
            head = nullptr;
            tail = nullptr;
        } else {
            bantu = nullptr;
            while (hapus->next != head) {
                bantu = hapus;
                hapus = hapus->next;
            }
            bantu->next = head;
            tail = bantu;
        }
        delete hapus;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Menghapus node di tengah list pada posisi tertentu
void hapusTengah(int posisi) {
    if (!isEmpty()) {
        if (posisi == 1) {
            hapusDepan();
        } else if (posisi == hitungList()) {
            hapusBelakang();
        } else {
            bantu = head;
            for (int i = 1; i < posisi - 1; i++) {
                bantu = bantu->next;
            }
```

```
            hapus = bantu->next;
            bantu->next = hapus->next;
            delete hapus;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Menghapus seluruh node dalam list
void clearList() {
    while (!isEmpty()) {
        hapusDepan();
    }
    cout << "List berhasil terhapus!" << endl;
}

// Menampilkan isi list
void tampil() {
    if (!isEmpty()) {
        bantu = head;
        do {
            cout << bantu->data << " ";
            bantu = bantu->next;
        } while (bantu != head);
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    init();
    insertDepan("Ayam");
    tampil();

    insertDepan("Bebek");
    tampil();

    insertBelakang("Cicak");
    tampil();

    insertBelakang("Domba");
    tampil();

    hapusBelakang();
    tampil();
```

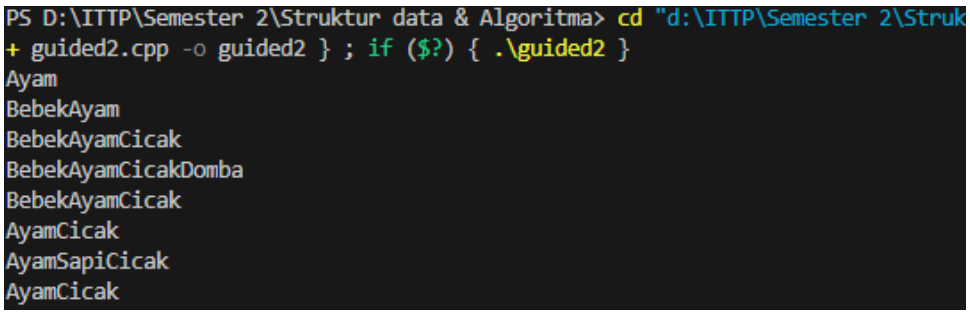
```
    hapusDepan();
    tampil();

    insertTengah("Sapi", 2);
    tampil();

    hapusTengah(2);
    tampil();

    return 0;
}
```

Screenshoot program



Deskripsi Program

Program ini dibuat dengan menggunakan Linked List Circular, program tersebut dapat menambah dan menghapus pada depan dan belakang. Struktur data tersebut terdiri dari Struct Node yang berisi data dan pointer ke Node selanjutnya

LATIHAN KELAS - UNGUIDED

Unguided

Source code

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

class Mahasiswa {
public:
    string nama;
    string nim;
    Mahasiswa* next;

    Mahasiswa(string n, string i) {
        nama = n;
        nim = i;
        next = NULL;
    }
};

class LinkedList {
private:
    Mahasiswa* head;
    Mahasiswa* tail;

public:
    LinkedList() {
        head = NULL;
        tail = NULL;
    }

    void tambahDepan(string n, string i) {
        Mahasiswa* baru = new Mahasiswa(n, i);
        if (head == NULL) {
            head = baru;
            tail = baru;
        } else {
            baru->next = head;
            head = baru;
        }
    }

    void tambahBelakang(string n, string i) {
        Mahasiswa* baru = new Mahasiswa(n, i);
```

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

    void tambahTengah(string n, string i, int pos) {
        Mahasiswa* baru = new Mahasiswa(n, i);
        if (pos == 0) {
            baru->next = head;
            head = baru;
        } else {
            Mahasiswa* bantu = head;
            for (int a = 1; a < pos; a++) {
                if (bantu->next == NULL) {
                    cout << "Posisi tidak ditemukan!" << endl;
                    return;
                }
                bantu = bantu->next;
            }
            baru->next = bantu->next;
            bantu->next = baru;
        }
    }

    void ubahDepan(string n, string i) {
        if (head == NULL) {
            cout << "List kosong!" << endl;
            return;
        }
        head->nama = n;
        head->nim = i;
    }

    void ubahBelakang(string n, string i) {
        if (head == NULL) {
            cout << "List kosong!" << endl;
            return;
        }
        if (head == tail) {
            head->nama = n;
            head->nim = i;
        } else {
            Mahasiswa* bantu = head;
            while (bantu->next != tail) {
```

```
            bantu = bantu->next;
        }
        tail->nama = n;
        tail->nim = i;
    }
}

void ubahTengah(string n, string i, int pos) {
    if (head == NULL) {
        cout << "List kosong!" << endl;
        return;
        }
        if (pos == 0) {
            head->nama = n;
            head->nim = i;
            return;
        }
        Mahasiswa* bantu = head;
        for (int a = 1; a < pos; a++) {
            if (bantu->next == NULL) {
                cout << "Posisi tidak ditemukan!" << endl;
                return;
            }
            bantu = bantu->next;
       }
        bantu->nama = n;
        bantu->nim = i;
    }

    void hapusDepan() {
        if (head == NULL) {
            cout << "List kosong!" << endl;
            return;
        }
        Mahasiswa* hapus = head;
        head = head->next;
        delete hapus;
    }

    void hapusBelakang() {
        if (head == NULL) {
            cout << "List kosong!" << endl;
            return;
        }
        if (head == tail) {
            delete head;
            head = tail = NULL;
        } else {
```

```
137.                 Mahasiswa* bantu = head;
138.                 while (bantu->next != tail) {
139.                     bantu = bantu->next;
140.                 }
141.                 delete tail;
142.                 tail = bantu;
143.                 tail->next = NULL;
144.             }
145.         }
146.
147.         void hapusTengah(int pos) {
148.             if (head == NULL) {
149.                 cout << "List kosong!" << endl;
150.                 return;
151.             }
152.             if (pos == 0) {
153.                 Mahasiswa* hapus = head;
154.                 head = head->next;
155.                 delete hapus;
156.                 return;
157.             }
158.             Mahasiswa* bantu = head;
159.             for (int a = 1; a < pos; a++) {
160.                 if (bantu->next == NULL) {
161.                     cout << "Posisi tidak ditemukan!" << endl;
162.                     return;
163.                 }
164.                 bantu = bantu->next;
165.             }
166.             Mahasiswa* hapus = bantu->next;
167.             bantu->next = bantu->next->next;
168.             delete hapus;
169.         }
170.
171.         void hapusList() {
172.             while (head != NULL) {
173.                 Mahasiswa* hapus = head;
174.                 head = head->next;
175.                 delete hapus;
176.             }
177.             tail = NULL;
178.         }
179.
180.         void tampilkan() {
181.             Mahasiswa* bantu = head;
182.             int i = 1;
183.             while (bantu != NULL) {
184.                 cout << "DATA MAHASISWA" << endl;
```

```
                cout << "NO. " << i << endl;
                cout << "NAMA\t: " << bantu->nama << endl;
                cout << "NIM\t: " << bantu->nim << endl;
                bantu = bantu->next;
                i++;
            }
        }
    };

    int main() {
        LinkedList list;
        int pilihan, pos;
        string nama, nim;

        // Masukkan data sesuai urutan
        list.tambahBelakang("Jawad", "23300001");
        list.tambahBelakang("harits", "2311102016");
        list.tambahBelakang("Farrel", "23300003");
        list.tambahBelakang("Denis", "23300005");
        list.tambahBelakang("Anis", "23300008");
        list.tambahBelakang("Bowo", "23300015");
        list.tambahBelakang("Gahar", "23300040");
        list.tambahBelakang("Udin", "23300048");
        list.tambahBelakang("Ucok", "23300050");
        list.tambahBelakang("Budi", "23300099");

        do {
            system("cls");
            cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" <<
    endl;
            cout << "1. Tambah Depan" << endl;
            cout << "2. Tambah Belakang" << endl;
            cout << "3. Tambah Tengah" << endl;
            cout << "4. Ubah Depan"<< endl;
            cout << "5. Ubah Belakang" << endl;
            cout << "6. Ubah Tengah" << endl;
            cout << "7. Hapus Depan" << endl;
            cout << "8. Hapus Belakang" << endl;
            cout << "9. Hapus Tengah" << endl;
            cout << "10. Hapus List" << endl;
            cout << "11. TAMPILKAN" << endl;
            cout << "0. KELUAR" << endl;
            cout << "Pilih Operasi : ";
            cin >> pilihan;

            switch (pilihan) {
                case 1:
                    cout << "Tambah Depan" << endl;
```

```
232.                 cout << "Masukkan Nama : ";
233.                 cin >> nama;
234.                 cout << "Masukkan NIM : ";
235.                 cin >> nim;
236.                 list.tambahDepan(nama, nim);
237.                 system("pause");
238.                 break;
239.             case 2:
240.                 cout << "Tambah Belakang" << endl;
241.                 cout << "Masukkan Nama : ";
242.                 cin >> nama;
243.                 cout << "Masukkan NIM : ";
244.                 cin >> nim;
245.                 list.tambahBelakang(nama, nim);
246.                 system("pause");
247.                 break;
248.             case 3:
249.                 cout << "Tambah Tengah" << endl;
250.                 cout << "Masukkan Nama : ";
251.                 cin >> nama;
252.                 cout << "Masukkan NIM : ";
253.                 cin >> nim;
254.                 cout << "Masukkan Posisi : ";
255.                 cin >> pos;
256.                 list.tambahTengah(nama, nim, pos);
257.                 system("pause");
258.                 break;
259.             case 4:
260.                 cout << "Ubah Depan" << endl;
261.                 cout << "Masukkan Nama : ";
262.                 cin >> nama;
263.                 cout << "Masukkan NIM : ";
264.                 cin >> nim;
265.                 list.ubahDepan(nama, nim);
266.                 system("pause");
267.                 break;
268.             case 5:
269.                 cout << "Ubah Belakang" << endl;
270.                 cout << "Masukkan nama : ";
271.                 cin >> nama;
272.                 cout << "Masukkan NIM : ";
273.                 cin >> nim;
274.                 list.ubahBelakang(nama, nim);
275.                 system("pause");
276.                 break;
277.             case 6:
278.                 cout << "Ubah Tengah" << endl;
279.                 cout << "Masukkan nama : ";
```

```
                    cin >> nama;
                    cout << "Masukkan NIM : ";
                    cin >> nim;
                    cout << "Masukkan posisi : ";
                    cin >> pos;
                    list.ubahTengah(nama, nim, pos);
                    system("pause");
                    break;
                case 7:
                    cout << "Hapus Depan" << endl;
                    list.hapusDepan();
                    system("pause");
                    break;
                case 8:
                    cout << "Hapus Belakang" << endl;
                    list.hapusBelakang();
                    system("pause");
                    break;
                case 9:
                    cout << "Hapus Tengah" << endl;
                    cout << "Masukkan posisi : ";
                    cin >> pos;
                    list.hapusTengah(pos);
                    system("pause");
                    break;
                case 10:
                    cout << "Hapus List" << endl;
                    list.hapusList();
                    system("pause");
                    break;
                case 11:
                    cout << "TAMPILKAN" << endl;
                    list.tampilkan();
                    system("pause");
                   break;
                case 0:
                    cout << "Terima kasih!" << endl;
                    break;
                default:
                    cout << "Pilihan tidak valid!" << endl;
                    system("pause");
            }
        } while (pilihan != 0);

        return 0;
    }
```

Screenshoot program

a. Tambahkan data berikut di antara farrel dan denis

Wati 2330004

```
Pilih Operasi : 3
Tambah Tengah
Masukkan Nama : wati
Masukkan NIM : 2330004
Masukkan Posisi : 4
```

```
NO. 3
NAMA    : Farrel
NIM     : 23300003
DATA MAHASISWA
NO. 4
NAMA    : wati
NIM     : 2330004
DATA MAHASISWA
NO. 5
NAMA    : Denis
NIM     : 23300005
DATA MAHASISWA
```

b. Hapus data denis

```
Pilih Operasi : 9
Hapus Tengah
Masukkan posisi : 4
Press any key to continue . . .
```

```
Pilih Operasi : 11
TAMPILKAN
DATA MAHASISWA
NO. 1
NAMA    : Jawad
NIM     : 23300001
DATA MAHASISWA
NO. 2
NAMA    : harits
NIM     : 2311102016
DATA MAHASISWA
NO. 3
NAMA    : Farrel
NIM     : 23300003
DATA MAHASISWA
NO. 4
NAMA    : wati
NIM     : 2330004
DATA MAHASISWA
NO. 5
NAMA    : Anis
NIM     : 23300008
DATA MAHASISWA
NO. 6
NAMA    : Bowo
NIM     : 23300015
```

c. **Tambahkan data berikut di awal**

owi 2330000

```
Pilih Operasi : 1
Tambah Depan
Masukkan Nama : owi
Masukkan NIM : 2330000
Press any key to continue . . .
```

```
0. KELUAR
Pilih Operasi : 11
TAMPILKAN
DATA MAHASISWA
NO. 1
NAMA    : owi
NIM     : 2330000
DATA MAHASISWA
NO. 2
NAMA    : Jawad
NIM     : 23300001
DATA MAHASISWA
NO. 3
NAMA    : harits
NIM     : 2311102016
DATA MAHASISWA
```

d. Tambahkan data berikut di akhir

David 23300100

```
Pilih Operasi : 2
Tambah Belakang
Masukkan Nama : david
Masukkan NIM : 23300100
Press any key to continue . . .
```

```
NAMA    : Udin
NIM     : 23300048
DATA MAHASISWA
NO. 10
NAMA    : Ucok
NIM     : 23300050
DATA MAHASISWA
NO. 11
NAMA    : Budi
NIM     : 23300099
DATA MAHASISWA
NO. 12
NAMA    : david
NIM     : 23300100
Press any key to continue . . .
```

e. **Ubah data udin menjadi data berikut**

Idin 23300045

```
Pilih Operasi : 6
Ubah Tengah
Masukkan nama : idin
Masukkan NIM : 23300045
Masukkan posisi : 9
Press any key to continue . . .
```

```
NIM     : 2330004
DATA MAHASISWA
NO. 6
NAMA    : Anis
NIM     : 23300008
DATA MAHASISWA
NO. 7
NAMA    : Bowo
NIM     : 23300015
DATA MAHASISWA
NO. 8
NAMA    : Gahar
NIM     : 23300040
DATA MAHASISWA
NO. 9
NAMA    : idin
NIM     : 23300045
DATA MAHASISWA
NO. 10
NAMA    : Ucok
NIM     : 23300050
DATA MAHASISWA
NO. 11
NAMA    : Budi
NIM     : 23300099
DATA MAHASISWA
NO. 12
NAMA    : david
```

f. Ubah data terakhir menjadi berikut

Lucy 23300101

```
Pilih Operasi : 5
Ubah Belakang
Masukkan nama : lucy
Masukkan NIM : 23300101
Press any key to continue . . .
```

```
NO. 9
NAMA    : idin
NIM     : 23300045
DATA MAHASISWA
NO. 10
NAMA    : Ucok
NIM     : 23300050
DATA MAHASISWA
NO. 11
NAMA    : Budi
NIM     : 23300099
DATA MAHASISWA
NO. 12
NAMA    : lucy
NIM     : 23300101
Press any key to continue . . .
```

g. Hapus data awal

```
Pilih Operasi : 10
Hapus List
Press any key to continue . . .
```

h. **Ubah data awal menjadi berikut**

Bagas 2330002

```
Pilih Operasi : 4
Ubah Depan
Masukkan Nama : bagas
Masukkan NIM : 2330002
Press any key to continue . . .
```

```
Pilih Operasi : 11
TAMPILKAN
DATA MAHASISWA
NO. 1
NAMA    : bagas
NIM     : 2330002
DATA MAHASISWA
NO. 2
NAMA    : harits
NIM     : 2311102016
DATA MAHASISWA
NO. 3
NAMA    : Farrel
NIM     : 23300003
DATA MAHASISWA
NO. 4
NAMA    : wati
NIM     : 2330004
DATA MAHASISWA
NO. 5
```

i. **Hapus data akhir**

```
Pilih Operasi : 8
Hapus Belakang
Press any key to continue . . .
```

j. Tampilkan seluruh data

```
Pilih Operasi : 11
TAMPILKAN
DATA MAHASISWA
NO. 1
NAMA    : bagas
NIM     : 2330002
DATA MAHASISWA
NO. 2
NAMA    : harits
NIM     : 2311102016
DATA MAHASISWA
NO. 3
NAMA    : Farrel
NIM     : 23300003
DATA MAHASISWA
NO. 4
NAMA    : wati
NIM     : 2330004
DATA MAHASISWA
NO. 5
NAMA    : Anis
NIM     : 23300008
DATA MAHASISWA
NO. 6
NAMA    : Bowo
NIM     : 23300015
DATA MAHASISWA
NO. 7
NAMA    : Gahar
NIM     : 23300040
DATA MAHASISWA
NO. 8
NAMA    : idin
NIM     : 23300045
DATA MAHASISWA
NO. 9
NAMA    : Ucok
NIM     : 23300050
DATA MAHASISWA
NO. 10
NAMA    : Budi
NIM     : 23300099
Press any key to continue . . .
```

Deskripsi program

Program ini menggunakan struktur data dari Single Linked List non-circular. Program ini dibuat untuk menyimpan data dan alamat dari setiap node. Inputan program ini berupa informasi Mahasiswa yang terdiri dari Nama dan NIM (Nomor Induk Mahasiswa). Dalam Program ini terdapat beberapa operasi yaitu, Menambahkan Node, Menghapus Node, Mengubah Node, dan Menampilkan data-data.

BAB IV

KESIMPULAN

Linked list adalah strukur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.

Single linked list circular adalah Single Linked List yang pointer nextnya menunjuk pada dirinya sendiri., double linked list memiliki dua pointer yang menghubungkan node secara berurutan dan dua arah. Meskipun fitur tambahan ini memungkinkan lebih fleksibilitas saat menjalankan operasi seperti menambah atau menghapus node dengan mudah, ini juga memerlukan memori tambahan untuk menyimpan kedua pointer tersebut.

Perbedaan antara circular dengan non circular adalah jika non circular pada node terakhir semulanya menunjuk ke NULL. Jika circular terletak pada node terakhir yang semulanya menunjuk ke kepala atau head.

DAFTAR PUSTAKA

Asisten Praktikum. (2024). MODUL 4 LINKED LIST CIRCULAR DAN NON CIRCULAR, Learning Managament System