LAPORAN PRAKTIKUM
Modul IV
“Single Linked List”

Disusun Oleh:
Fahmi Hasan Asagaf -2311104074
SE 07 02

Dosen :
Wahyu Andi Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

Tujuan dari praktikum ini adalah untuk memahami dan mengimplementasikan struktur data Single Linked List menggunakan bahasa pemrograman C++. Praktikum ini bertujuan untuk:

- Mempelajari cara membuat, mengelola, dan memanipulasi linked list.
- Mengimplementasikan operasi dasar seperti penambahan, penghapusan, pencarian, dan penghitungan elemen dalam linked list.
- Memahami pengelolaan memori secara manual dalam konteks struktur data dinamis.

2. Landasan Teori

Single Linked List adalah salah satu jenis struktur data yang berbasis pada konsep rantai (linked) di mana setiap elemen (disebut node) saling terhubung secara berurutan melalui pointer atau referensi. Pada Single Linked List, setiap node hanya memiliki satu referensi/pointer yang mengarah ke node berikutnya, dan tidak ada referensi yang mengarah ke node sebelumnya.

Karakteristik Single Linked List:

1. Node: Setiap node pada single linked list terdiri dari dua bagian:
    - Data: Berisi informasi atau nilai yang disimpan di node tersebut.
    - Pointer (Next): Berisi alamat atau referensi yang menunjuk ke node berikutnya dalam daftar.
2. Head: Merupakan pointer khusus yang digunakan untuk menunjukkan node pertama pada linked list. Jika linked list kosong, head akan bernilai null atau None.
3. Tail: Node terakhir dari linked list memiliki pointer next yang mengarah ke null, menandakan akhir dari daftar.

Operasi-Operasi Dasar pada Single Linked List:

1. Traversal: Mengakses atau membaca elemen-elemen dalam linked list dimulai dari head hingga tail.
2. Insertion: Menambahkan elemen baru di awal (head), di akhir (tail), atau di tengah daftar.
3. Deletion: Menghapus node dari linked list, baik itu di awal, di akhir, atau di posisi tertentu.
4. Search: Mencari node yang mengandung nilai tertentu dengan menelusuri linked list.

Kelebihan:

- Penggunaan Memori Dinamis: Memori dialokasikan secara dinamis sesuai kebutuhan sehingga menghemat penggunaan memori.
- Insert/Delete Cepat di Awal: Proses penambahan atau penghapusan node di awal linked list sangat cepat karena hanya memodifikasi pointer head.

Kekurangan:

- Akses Lambat: Tidak mendukung akses langsung (random access) ke elemen tertentu seperti array. Untuk mengakses elemen tertentu, harus melakukan traversal dari head.
- Penggunaan Memori Tambahan: Setiap node menyimpan pointer tambahan yang memakan lebih banyak memori dibandingkan array sederhana.

3. Guided

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

// Deklarasi Struct untuk mahasiswa
struct mahasiswa
{
    char nama[30];
    char nim[10];
};

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

Node *head;
Node *tail;

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

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

// Tambah Depan
void insertDepan(const mahasiswa &data)
{
    Node *baru = new Node;
    baru->data = data;
    baru->next = nullptr;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

// Tambah Belakang
void insertBelakang(const mahasiswa &data)
{
    Node *baru = new Node;
    baru->data = data;
    baru->next = nullptr;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

// Hapus Depan
void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        head = head->next;
        delete hapus;
        if (head == nullptr)
        {
            tail = nullptr; // Jika list menjadi kosong
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

```

Telkom University Purwokerto

```cpp
// Hapus Belakang
void hapusBelakang()
{
    if (!isEmpty())
    {
        if (head == tail)
        {
            delete head;
            head = tail = nullptr; // List menjadi kosong
        }
        else
        {
            Node *bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            delete tail;
            tail = bantu;
            tail->next = nullptr;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

// Tampilkan List
void tampil()
{
    Node *current = head;
    if (!isEmpty())
    {
        while (current != nullptr)
        {
            cout << "Nama: " << current->data.nama << ", NIM: " << current->data.nim << endl;
            current = current->next;
        }
    }
    else
    {
        cout << "list masih kosong!" << endl;
    }
    cout << endl;
}

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

// Main function
int main()
{
    init();

    // Contoh data mahasiswa
    mahasiswa m1 = {"Alice", "123456"};
    mahasiswa m2 = {"Bob", "654321"};
    mahasiswa m3 = {"Charlie", "112233"};

    // Menambahkan mahasiswa ke dalam list
    insertDepan(m1);
    tampil();
    insertBelakang(m2);
    tampil();
    insertDepan(m3);
    tampil();

    // Menghapus elemen dari list
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();

    // Menghapus seluruh list
    clearList();

    return 0;
}
```

Output

```
Nama: Alice, NIM: 123456

Nama: Alice, NIM: 123456
Nama: Bob, NIM: 654321

Nama: Charlie, NIM: 112233
Nama: Alice, NIM: 123456
Nama: Bob, NIM: 654321

Nama: Alice, NIM: 123456
Nama: Bob, NIM: 654321

Nama: Alice, NIM: 123456

List berhasil terhapus!
PS D:\guided modul 4>
```

Telkom University Purwokerto

Guided 2

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

// Definisi struktur untuk elemen list
struct Node
{
    int data;   // Menyimpan nilai elemen
    Node *next; // Pointer ke elemen berikutnya
};

// Fungsi untuk mengalokasikan memori untuk node baru
Node *alokasi(int value)
{
    Node *newNode = new Node; // Alokasi memori untuk elemen baru
    if (newNode != nullptr)
    {                         // Jika alokasi berhasil
        newNode->data = value;   // Mengisi data node
        newNode->next = nullptr; // Set next ke nullptr
    }
    return newNode; // Mengembalikan pointer node baru
}

// Fungsi untuk dealokasi memori node
void dealokasi(Node *node)
{
    delete node; // Mengembalikan memori yang digunakan oleh node
}

// Pengecekan apakah list kosong
bool isListEmpty(Node *head)
{
    return head == nullptr; // List kosong jika head adalah nullptr
}

// Menambahkan elemen di awal list
void insertFirst(Node *&head, int value)
{
    Node *newNode = alokasi(value); // Alokasi memori untuk elemen baru
    if (newNode != nullptr)
    {
        newNode->next = head; // Menghubungkan elemen baru ke elemen pertama
        head = newNode;       // Menetapkan elemen baru sebagai elemen pertama
    }
}

// Menambahkan elemen di akhir list
void insertLast(Node *&head, int value)
{
    Node *newNode = alokasi(value); // Alokasi memori untuk elemen baru
    if (newNode != nullptr)
    {
        if (isListEmpty(head))
        {                     // Jika list kosong
            head = newNode; // Elemen baru menjadi elemen pertama
        }
        else
        {
            Node *temp = head;
            while (temp->next != nullptr)
            { // Mencari elemen terakhir
                temp = temp->next;
            }
            temp->next = newNode; // Menambahkan elemen baru di akhir list
        }
    }
}

// Menampilkan semua elemen dalam list
void printList(Node *head)
{
    if (isListEmpty(head))
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        Node *temp = head;
        while (temp != nullptr)
        {                                   // Selama belum mencapai akhir list
            cout << temp->data << " "; // Menampilkan data elemen
            temp = temp->next;          // Melanjutkan ke elemen berikutnya
        }
        cout << endl;
    }
}

// Menghitung jumlah elemen dalam list
int countElements(Node *head)
{
    int count = 0;
    Node *temp = head;
    while (temp != nullptr)
    {
        count++;             // Menambah jumlah elemen
        temp = temp->next; // Melanjutkan ke elemen berikutnya
    }
    return count; // Mengembalikan jumlah elemen
}

// Menghapus semua elemen dalam list dan dealokasi memori
void clearList(Node *&head)
{
    while (head != nullptr)
    {
        Node *temp = head; // Simpan pointer ke node saat ini
        head = head->next; // Pindahkan ke node berikutnya
        dealokasi(temp);   // Dealokasi node
    }
}

int main()
{
    Node *head = nullptr; // Membuat list kosong

    // Menambahkan elemen ke dalam list
    insertFirst(head, 10); // Menambahkan elemen 10 di awal list
    insertLast(head, 20);  // Menambahkan elemen 20 di akhir list
    insertLast(head, 30);  // Menambahkan elemen 30 di akhir list

    // Menampilkan isi list
    cout << "Isi List: ";
    printList(head);

    // Menampilkan jumlah elemen
    cout << "Jumlah elemen: " << countElements(head) << endl;

    // Menghapus semua elemen dalam list
    clearList(head);

    // Menampilkan isi list setelah penghapusan
    cout << "Isi List setelah penghapusan: ";
    printList(head);

    return 0;
}
```

Output

```
PS D:\guided modul 4>  & 'c:\Users\fahmi\.vsco
her.exe' '--stdin=Microsoft-MIEngine-In-yuxgwe
ror-nm1afngm.bjo' '--pid=Microsoft-MIEngine-Pi
Isi List: 10 20 30
Jumlah elemen: 3
Isi List setelah penghapusan: List kosong!
PS D:\guided modul 4>
```

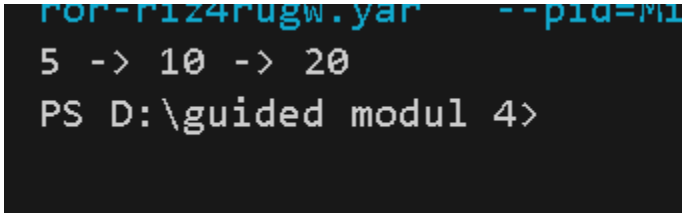
Telkom University Purwokerto

4. Unguided

1.

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

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

// Head pointer to the linked list
Node *head = nullptr;

// Function to insert a node at the front
void insertDepan(int nilai)
{
    Node *newNode = new Node();
    newNode->data = nilai;
    newNode->next = head;
    head = newNode;
}

// Function to insert a node at the back
void insertBelakang(int nilai)
{
    Node *newNode = new Node();
    newNode->data = nilai;
    newNode->next = nullptr;

    if (head == nullptr)
    {
        head = newNode;
    }
    else
    {
        Node *temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to print the entire linked list
void cetakLinkedList()
{
    Node *temp = head;
    if (temp == nullptr)
    {
        cout << "Linked list kosong" << endl;
    }
    else
    {
        while (temp != nullptr)
        {
            cout << temp->data;
            if (temp->next != nullptr)
            {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

int main()
{
    // Contoh operasi
    insertDepan(10);    // Tambah node di depan (nilai: 10)
    insertBelakang(20); // Tambah node di belakang (nilai: 20)
    insertDepan(5);     // Tambah node di depan (nilai: 5)

    // Cetak isi linked list
    cetakLinkedList(); // Output: 5 -> 10 -> 20

    return 0;
}
```

Output



Telkom University Purwokerto

2.

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

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

// Head pointer to the linked list
Node *head = nullptr;

// Function to insert a node at the front
void insertDepan(int nilai)
{
    Node *newNode = new Node();
    newNode->data = nilai;
    newNode->next = head;
    head = newNode;
}

// Function to insert a node at the back
void insertBelakang(int nilai)
{
    Node *newNode = new Node();
    newNode->data = nilai;
    newNode->next = nullptr;

    if (head == nullptr)
    {
        head = newNode;
    }
    else
    {
        Node *temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to delete a node with a specific value
void hapusNode(int nilai)
{
    if (head == nullptr)
    {
        cout << "Linked list kosong, tidak ada node untuk dihapus" << endl;
        return;
    }

    // Jika node yang harus dihapus adalah head
    if (head->data == nilai)
    {
        Node *temp = head;
        head = head->next;
        delete temp;
        cout << "Node dengan nilai " << nilai << " berhasil dihapus" << endl;
        return;
    }

    // Mencari node dengan nilai tertentu
    Node *temp = head;
    Node *prev = nullptr;
    while (temp != nullptr && temp->data != nilai)
    {
        prev = temp;
        temp = temp->next;
    }

    // Jika node tidak ditemukan
    if (temp == nullptr)
    {
        cout << "Node dengan nilai " << nilai << " tidak ditemukan" << endl;
        return;
    }

    // Hapus node
    prev->next = temp->next;
    delete temp;
    cout << "Node dengan nilai " << nilai << " berhasil dihapus" << endl;
}

// Function to print the entire linked list
void cetakLinkedList()
{
    Node *temp = head;
    if (temp == nullptr)
    {
        cout << "Linked list kosong" << endl;
    }
    else
    {
        while (temp != nullptr)
        {
            cout << temp->data;
            if (temp->next != nullptr)
            {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

int main()
{
    // Contoh operasi
    insertDepan(10);    // Tambah node di depan (nilai: 10)
    insertBelakang(20); // Tambah node di belakang (nilai: 20)
    insertDepan(5);     // Tambah node di depan (nilai: 5)

    // Cetak isi linked list sebelum penghapusan
    cetakLinkedList(); // Output: 5 -> 10 -> 20

    // Hapus node dengan nilai tertentu
    hapusNode(10); // Hapus node dengan nilai 10

    // Cetak isi linked list setelah penghapusan
    cetakLinkedList(); // Output: 5 -> 20

    return 0;
}
```

Telkom University Purwokerto

Output

```
ror-fneifgmh.sgs' '--pid=Microsoft-MIEngine-Pi
5 -> 10 -> 20
Node dengan nilai 10 berhasil dihapus
5 -> 20
PS D:\guided modul 4>
```

Telkom University Purwokerto

3.

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

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

// Head pointer to the linked list
Node *head = nullptr;

// Function to insert a node at the front
void insertDepan(int nilai)
{
    Node *newNode = new Node();
    newNode->data = nilai;
    newNode->next = head;
    head = newNode;
}

// Function to insert a node at the back
void insertBelakang(int nilai)
{
    Node *newNode = new Node();
    newNode->data = nilai;
    newNode->next = nullptr;

    if (head == nullptr)
    {
        head = newNode;
    }
    else
    {
        Node *temp = head;
        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to search for a node with a specific value
bool cariNode(int nilai)
{
    Node *temp = head;
    while (temp != nullptr)
    {
        if (temp->data == nilai)
        {
            return true;
        }
        temp = temp->next;
    }
    return false;
}

// Function to count the length of the linked list
int hitungPanjangLinkedList()
{
    Node *temp = head;
    int count = 0;
    while (temp != nullptr)
    {
        count++;
        temp = temp->next;
    }
    return count;
}

// Function to print the entire linked list
void cetakLinkedList()
{
    Node *temp = head;
    if (temp == nullptr)
    {
        cout << "Linked list kosong" << endl;
    }
    else
    {
        while (temp != nullptr)
        {
            cout << temp->data;
            if (temp->next != nullptr)
            {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
}

int main()
{
    // Contoh operasi
    insertDepan(10);    // Tambah node di depan (nilai: 10)
    insertBelakang(20); // Tambah node di belakang (nilai: 20)
    insertDepan(5);     // Tambah node di depan (nilai: 5)

    // Cetak isi linked list
    cetakLinkedList(); // Output: 5 -> 10 -> 20

    // Mencari node dengan nilai tertentu
    int nilaiDicari = 20;
    if (cariNode(nilaiDicari))
    {
        cout << "Node dengan nilai " << nilaiDicari << " ditemukan." << endl;
    }
    else
    {
        cout << "Node dengan nilai " << nilaiDicari << " tidak ditemukan." << endl;
    }

    // Hitung panjang linked list
    int panjang = hitungPanjangLinkedList();
    cout << "Panjang linked list: " << panjang << endl;

    return 0;
}
```

Output

```
5 -> 10 -> 20
Node dengan nilai 20 ditemukan.
Panjang linked list: 3
PS D:\guided modul 4>
```

5. Kesimpulan

Kesimpulan dari tujuan praktikum ini adalah untuk:

1. Memahami dan mengimplementasikan struktur data Single Linked List dengan bahasa pemrograman C++.
2. Mempelajari cara membuat, mengelola, dan memanipulasi linked list.
3. Mengimplementasikan operasi dasar seperti penambahan, penghapusan, pencarian, dan penghitungan elemen dalam linked list.
4. Memahami pengelolaan memori secara manual dalam konteks struktur data dinamis.