# LAPORAN PRAKTIKUM

# MODUL 4

# SINGLE LINKED LIST (BAGIAN PERTAMA)


**Nama :**

Candra Dinata (2311104061)

**Kelas :**

S1SE 07 02

**Dosen :**

Wahyu Andi Saputra



PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

# I. TUJUAN

Membantu mahasiswa memahami dasar-dasar pemrograman, sepertisintaks, tipe data, operator, dan strukturkontrol, serta menguasaipenggunaan fungsi untuk modularitasprogram. Selain itu, mahasiswa jugadiperkenalkan dengan konsepObject-Oriented Programming (OOP) sepertikelas, objek, pewarisan, dan enkapsulasi. Melalui latihan ini, mahasiswadiharapkan mampu menerapkan konseppemrograman untuk menyelesaikanmasalah dan memahami carakerja kompilator C++ dalam mengubah kodesumber menjadiprogramyangdapatdijalankan.

# II. LANDASAN TEORI

C++ adalah bahasa pemrograman yang dikembangkansebagaipengembangan dari bahasa C dengan menambahkan fitur-fiturpemrograman berorientasi objek (Object-OrientedProgramming, OOP). Dalam C++, konsep OOP seperti enkapsulasi, pewarisan, dan polimorfismemenjadi dasar untuk membuatprogram yang lebih modular, terstruktur, dandapat digunakankembali. Bahasa ini mendukung penggunaan fungsi, manipulasimemori secara langsung melalui pointer, sertamemilikikemampuan untuk menangani berbagai tipe data dasardan struktural. C++juga mendukung konsep pemrogramanprosedural dan modular, yangmemudahkan pengembanganprogram yang kompleks. Kompilasi programC++ dilakukanmelalui kompilator yang menerjemahkan kode sumbermenjadi fileeksekusi yang bisa dijalankanolehkomputer.

## III. GUIDED

1.

Program ini mengimplementasikan struktur data linked list untuk menyimpan data mahasiswa yang terdiri dari nama dan NIM menggunakan struct mahasiswa dan struct Node untuk menyusun node dalam list. Program memiliki beberapa fungsi, seperti insertDepan() dan insertBelakang() untuk menambahkan mahasiswa ke awal atau akhir list, hitungList() untuk menghitung jumlah node, serta hapusDepan() dan hapusBelakang() untuk menghapus node dari awal atau akhir list. Fungsi tampil() digunakan untuk menampilkan data mahasiswa dalam list, sedangkan clearList() untuk menghapus semua node di list. Program ini juga memiliki fungsi inisialisasi init() untuk mengosongkan list dan pengecekan apakah list kosong dengan isEmpty().

2.

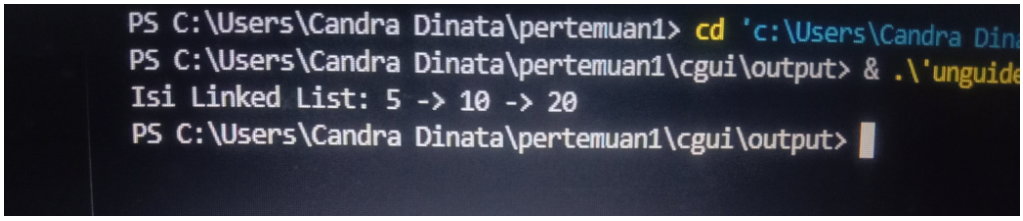
```cpp
#include <iostream>
using namespace std;

// 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;
}
```

Program ini mengimplementasikan struktur data linked list untuk menyimpan data integer menggunakan struct Node, yang terdiri dari elemen data dan pointer ke elemen berikutnya. Fungsi insertFirst() menambahkan elemen di awal list, sedangkan insertLast() menambahkan elemen di akhir list. Fungsi printList() digunakan untuk menampilkan seluruh elemen yang ada dalam list, sementara countElements() menghitung jumlah elemen dalam list. Program juga memiliki fungsi clearList() untuk menghapus seluruh elemen dalam list dan mendekalokasi memori yang digunakan. Fungsi utama menambahkan beberapa elemen, menampilkan list, menghitung jumlah elemen, lalu menghapus semua elemen dari list, dan menampilkan kembali list setelah penghapusan.

## IV. UNGUIDED

1.

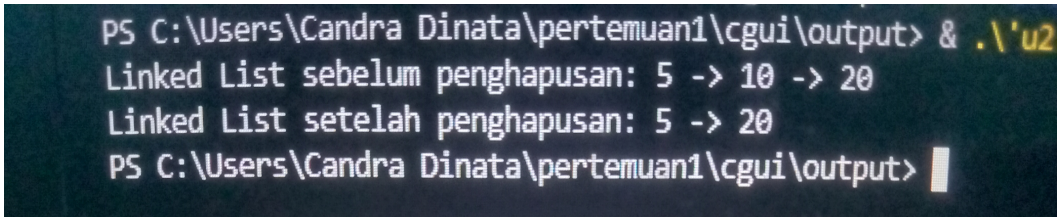
```cpp
#include <iostream>

using namespace std;

// Definisi Node untuk Single Linked List
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambah node di depan
void insertDepan(Node*& head, int nilai) {
    Node* newNode = new Node();
    newNode->data = nilai;
    newNode->next = head;
    head = newNode;
}

// Fungsi untuk menambah node di belakang
void insertBelakang(Node*& head, 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;
    }
}

// Fungsi untuk mencetak isi linked list
void cetakList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data;
        if (temp->next != nullptr)
            cout << " -> ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr; // Inisialisasi linked list

    // Input contoh
    insertDepan(head, 10);  // Tambah node di depan dengan nilai 10
    insertBelakang(head, 20);  // Tambah node di belakang dengan nilai 20
    insertDepan(head, 5);  // Tambah node di depan dengan nilai 5

    // Cetak linked list
    cout << "Isi Linked List: ";
    cetakList(head);

    return 0;
}
```




Program ini mengimplementasikan operasi dasar pada linked list menggunakan struct Node yang menyimpan data integer dan pointer ke node berikutnya. Fungsi insertDepan() menambahkan node baru di awal list, sementara insertBelakang() menambahkan node baru di akhir list. Dalam fungsi cetakList(), program mencetak elemen-elemen linked list dengan format terhubung menggunakan tanda " -> ". Pada fungsi utama (main()), linked list diinisialisasi dengan nilai kosong, kemudian tiga node ditambahkan: satu di depan (nilai 10), satu di belakang (nilai 20), dan satu lagi di depan (nilai 5), sebelum akhirnya mencetak isi linked list yang terdiri dari urutan node.

2.

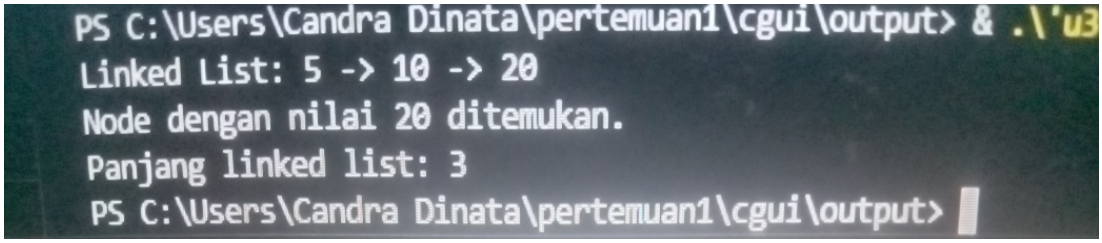
```cpp
#include <iostream>

using namespace std;

// Definisi Node untuk Single Linked List
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambah node di depan
void insertDepan(Node*& head, int nilai) {
    Node* newNode = new Node();
    newNode->data = nilai;
    newNode->next = head;
    head = newNode;
}

// Fungsi untuk menambah node di belakang
void insertBelakang(Node*& head, 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;
    }
}

// Fungsi untuk menghapus node dengan nilai tertentu
void hapusNode(Node*& head, int nilai) {
    if (head == nullptr) {
        cout << "Linked list kosong." << endl;
        return;
    }

    // Jika node yang akan dihapus adalah head
    if (head->data == nilai) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }

    // Mencari node yang memiliki 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;
    }

    // Menghapus node
    prev->next = temp->next;
    delete temp;
}

// Fungsi untuk mencetak isi linked list
void cetakList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data;
        if (temp->next != nullptr)
            cout << " -> ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr; // Inisialisasi linked list

    // Input contoh
    insertDepan(head, 10);  // Tambah node di depan dengan nilai 10
    insertBelakang(head, 20);  // Tambah node di belakang dengan nilai 20
    insertDepan(head, 5);  // Tambah node di depan dengan nilai 5

    cout << "Linked List sebelum penghapusan: ";
    cetakList(head);

    // Hapus node dengan nilai tertentu
    hapusNode(head, 10);  // Menghapus node dengan nilai 10

    // Cetak linked list setelah penghapusan
    cout << "Linked List setelah penghapusan: ";
    cetakList(head);

    return 0;
}
```




Program ini mengimplementasikan operasi dasar linked list, termasuk menambah dan menghapus node. Fungsi insertDepan() menambahkan node baru di awal list, dan insertBelakang() menambahkan node di akhir list. Fungsi hapusNode() digunakan untuk menghapus node dengan nilai tertentu. Jika node yang ingin dihapus adalah head (elemen pertama), maka head diupdate ke node berikutnya. Jika node yang dihapus bukan head, program mencari node dengan nilai yang sesuai, menghubungkan node sebelumnya dengan node setelahnya, lalu menghapus node yang diinginkan. Fungsi cetakList() menampilkan elemen-elemen linked list. Dalam fungsi utama, beberapa node ditambahkan ke linked list, lalu node dengan nilai 10 dihapus, dan list ditampilkan sebelum dan sesudah penghapusan.

3.

```cpp
#include <iostream>

using namespace std;

// Definisi Node untuk Single Linked List
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambah node di depan
void insertDepan(Node*& head, int nilai) {
    Node* newNode = new Node();
    newNode->data = nilai;
    newNode->next = head;
    head = newNode;
}

// Fungsi untuk menambah node di belakang
void insertBelakang(Node*& head, 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;
    }
}

// Fungsi untuk mencari node dengan nilai tertentu
bool cariNode(Node* head, int nilai) {
    Node* temp = head;
    while (temp != nullptr) {
        if (temp->data == nilai) {
            return true;  // Nilai ditemukan
        }
        temp = temp->next;
    }
    return false;  // Nilai tidak ditemukan
}

// Fungsi untuk menghitung panjang linked list
int hitungPanjang(Node* head) {
    int panjang = 0;
    Node* temp = head;
    while (temp != nullptr) {
        panjang++;
        temp = temp->next;
    }
    return panjang;
}

// Fungsi untuk mencetak isi linked list
void cetakList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data;
        if (temp->next != nullptr)
            cout << " -> ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr; // Inisialisasi linked list

    // Input contoh
    insertDepan(head, 10);  // Tambah node di depan dengan nilai 10
    insertBelakang(head, 20);  // Tambah node di belakang dengan nilai 20
    insertDepan(head, 5);  // Tambah node di depan dengan nilai 5

    // Cetak linked list
    cout << "Linked List: ";
    cetakList(head);

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

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

    return 0;
}
```




Program ini mengimplementasikan operasi dasar pada linked list, termasuk menambah node, mencari nilai, menghitung panjang list, dan mencetak isinya. Fungsi insertDepan() menambahkan node baru di awal list, sedangkan insertBelakang() menambahkan node di akhir list. Fungsi cariNode() mencari node dengan nilai tertentu dalam linked list dan mengembalikan true jika ditemukan, serta false jika tidak ditemukan. Fungsi hitungPanjang() menghitung berapa banyak node yang ada dalam linked list. Program juga mencetak isi linked list dengan fungsi cetakList(). Pada fungsi utama, program menambahkan beberapa node ke linked list, mencari node dengan nilai 20, mencetak hasil pencarian, dan menghitung serta mencetak panjang linked list.

## V. KESIMPULAN

Pada bagian pertama praktikum tentang Single Linked List, kami telah mempelajari dan mengimplementasikan operasi dasar pada struktur data ini. Single Linked List merupakan struktur data linier yang terdiri dari node-node yang terhubung secara berurutan, di mana setiap node hanya menunjuk ke node berikutnya. Operasi dasar yang dilakukan meliputi penambahan node di awal dan akhir list, pencarian node berdasarkan nilai, serta penghitungan jumlah node dalam list. Implementasi ini memperkenalkan konsep penggunaan pointer dan dynamic memory allocation untuk mengelola node secara efisien. Single Linked List sangat berguna dalam situasi yang membutuhkan struktur data dinamis, karena memungkinkan penambahan dan penghapusan elemen tanpa harus menggeser elemen lainnya seperti pada array.