LAPORAN PRAKTIKUM

PERTEMUAN 4

Pengenalan C++ : Single_Linked_List_Bagian01

Nama :

Ahmad Ufii Lestari Ma'ruf (2311104015)

Dosen :

YUDHA ISLAMI SULISTYA, S.Kom., M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1. **Praktek Bersama**
   a. **Praktek bersama satu**

```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;
    }
}
```

```cpp
// 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;
    }
}

// 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;
    }
}

// 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

```
.exe" "--interpreter=mi"
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 E:\ITTP\Modul ITTP\semester3
```

b. Praktek bersama dua

```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
        }
    }
}
```

```cpp
// 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 :

```
Isi List: 10 20 30
Jumlah elemen: 3
Isi List setelah pengh
```

2. Tugas Pendahuluan

a. List.h file

```cpp
#include "list.h"

void createList(List &L) {
    first(L) = NULL;
}

address allocate(infotype x) {
    address P = new elmlist;
    info(P) = x;
    next(P) = NULL;
    return P;
}

void insertFirst(List &L, address P) {
    next(P) = first(L);
    first(L) = P;
}

void printInfo(List L) {
    address P = first(L);
    while (P != NULL) {
        cout << info(P) << " ";
        P = next(P);
    }
    cout << endl;
```

b. File list.cpp

```cpp
#ifndef LIST_H_INCLUDED
#define LIST_H_INCLUDED

#include <iostream>

#define first(L) L.first
#define next(P) P->next
#define info(P) P->info

using namespace std;

typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};

struct List {
    address first;
};

void createList(List &L);
address allocate(infotype x);
void insertFirst(List &L, address P);
void printInfo(List L);

#endif // LIST_H_INCLUDED
```

c. **File main.cpp**

```cpp
#include <iostream>
#include "list.h"

int main() {
    List L;
    address P;
    infotype x;

    createList(L);

    // Input 3 digits of NIM (assuming the last 3 digits are 742)
    x = 2;
    P = allocate(x);
    insertFirst(L, P);

    x = 4;
    P = allocate(x);
    insertFirst(L, P);

    x = 7;
    P = allocate(x);
    insertFirst(L, P);

    cout << "Isi list: ";
    printInfo(L);

    return 0;
}
```

Output :

```
Masukkan angka ke-1: 4
List setelah menambahkan angka ke-1: 4,
Masukkan angka ke-2: 6
List setelah menambahkan angka ke-2: 6, 4,
Masukkan angka ke-3: 2
List setelah menambahkan angka ke-3: 2, 6, 4,
```

Tambahan tugas pendahuluan HAVE FUN

a. list.h

```h
#ifndef LIST_H_INCLUDED
#define LIST_H_INCLUDED

#include <iostream>

#define first(L) L.first
#define next(P) P->next
#define info(P) P->info

using namespace std;

typedef int infotype;
typedef struct elmlist *address;

struct elmlist {
    infotype info;
    address next;
};

struct List {
    address first;
};

void createList(List &L);
address allocate(infotype x);
void insertFirst(List &L, address P);
void printInfo(List L);

#endif // LIST_H_INCLUDED
```

b. list.cpp

```cpp
#include "list.h"

void createList(List &L) {
    first(L) = NULL;
}

address allocate(infotype x) {
    address P = new elmlist;
    info(P) = x;
    next(P) = NULL;
    return P;
}

void insertFirst(List &L, address P) {
    next(P) = first(L);
    first(L) = P;
}

void printInfo(List L) {
    address P = first(L);
    while (P != NULL) {
        cout << info(P) << " ";
        P = next(P);
    }
    cout << endl;
```

c. **main.cpp**

```cpp
#include <iostream>
#include "list.h"

int main() {
    List L;
    address P;
    infotype x;

    createList(L);

    // Input 3 digits of NIM (assuming the last 3 digits are 742)
    x = 2;
    P = allocate(x);
    insertFirst(L, P);

    x = 4;
    P = allocate(x);
    insertFirst(L, P);

    x = 7;
    P = allocate(x);
    insertFirst(L, P);

    cout << "Isi list: ";
    printInfo(L);

    return 0;
}
```

Output :

```
Masukkan NIM perdigit:
Digit 1 : 2
Digit 2 : 3
Digit 3 : 1
Digit 4 : 1
Digit 5 : 1
Digit 6 : 0
Digit 7 : 4
Digit 8 : 0
Digit 9 : 1
Digit 10 : 5
Isi list: 2311104015
```

3. **Unguaided**

pada kode program ini kita dapat melakukan insert depan untuk menambahkan node dari depan dan insert belakang yang berfungsi untuk menambahkan node dari belakang. Pada code ini kita juga bisa melakukan delete pada node, lau juga ada logic untuk melakukan peritungan panjang node dan mencari node yang dicari.

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

// Struktur untuk node
struct Node {
    int data;
    Node* next;
    Node(int val) : data(val), next(nullptr) {}
};

// Kelas LinkedList
class LinkedList {
private:
    Node* head;

public:
    LinkedList() : head(nullptr) {}

    // Insert node di depan
    void insertFront(int value) {
        Node* newNode = new Node(value);
        newNode->next = head;
        head = newNode;
    }

    // Insert node di belakang
    void insertBack(int value) {
        Node* newNode = new Node(value);
        if (!head) {
            head = newNode;
            return;
        }
        Node* temp = head;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = newNode;
    }

    // Cetak linked list
    void printList() {
        Node* temp = head;
        while (temp) {
            cout << temp->data;
            if (temp->next) cout << " -> ";
            temp = temp->next;
        }
        cout << endl;
    }

    // Hapus node dengan nilai tertentu
    void deleteNode(int value) {
        if (!head) return;
        if (head->data == value) {
            Node* temp = head;
            head = head->next;
            delete temp;
            return;
        }
        Node* temp = head;
        while (temp->next && temp->next->data != value) {
            temp = temp->next;
        }
        if (temp->next) {
            Node* toDelete = temp->next;
            temp->next = temp->next->next;
            delete toDelete;
        }
    }

    // Cari node dengan nilai tertentu
    bool searchNode(int value) {
        Node* temp = head;
        while (temp) {
            if (temp->data == value) return true;
            temp = temp->next;
        }
        return false;
    }

    // Hitung panjang linked list
    int length() {
        int count = 0;
        Node* temp = head;
        while (temp) {
            count++;
            temp = temp->next;
        }
        return count;
    }
};
```

```cpp
int main() {
    LinkedList list;
    int choice, value;

    while (true) {
        cout << "\nOperasi Linked List:\n";
        cout << "1. Tambah node di depan\n";
        cout << "2. Tambah node di belakang\n";
        cout << "3. Cetak linked list\n";
        cout << "4. Hapus node dengan nilai tertentu\n";
        cout << "5. Cari node dengan nilai tertentu\n";
        cout << "6. Cetak panjang linked list\n";
        cout << "7. Keluar\n";
        cout << "Pilih operasi: ";
        cin >> choice;

        switch(choice) {
            case 1:
                cout << "Masukkan nilai: ";
                cin >> value;
                list.insertFront(value);
                break;
            case 2:
                cout << "Masukkan nilai: ";
                cin >> value;
                list.insertBack(value);
                break;
            case 3:
                cout << "Isi linked list: ";
                list.printList();
                break;
            case 4:
                cout << "Masukkan nilai yang akan dihapus: ";
                cin >> value;
                list.deleteNode(value);
                break;
            case 5:
                cout << "Masukkan nilai yang akan dicari: ";
                cin >> value;
                if (list.searchNode(value))
                    cout << "Node dengan nilai " << value << " ditemukan.\n";
                else
                    cout << "Node dengan nilai " << value << " tidak ditemukan.\n";
                break;
            case 6:
                cout << "Panjang linked list: " << list.length() << endl;
                break;
            case 7:
                cout << "Terima kasih, program selesai.\n";
                return 0;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    }

    return 0;
}
```

makan output yang keluar adalah

output untuk soal nomor satu :

```
Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 1
Masukkan nilai: 10

Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 2
Masukkan nilai: 20

Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 1
Masukkan nilai: 5

Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 3
Isi linked list: 5 -> 10 -> 20
```

output untuk soal nomor dua

```
Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 4
Masukkan nilai yang akan dihapus: 10

Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 3
Isi linked list: 5 -> 20
```

output untuk soal nomor tiga

```
Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 5
Masukkan nilai yang akan dicari: 20
Node dengan nilai 20 ditemukan.

Operasi Linked List:
1. Tambah node di depan
2. Tambah node di belakang
3. Cetak linked list
4. Hapus node dengan nilai tertentu
5. Cari node dengan nilai tertentu
6. Cetak panjang linked list
7. Keluar
Pilih operasi: 6
Panjang linked list: 2
```