Laporan Praktikum Struktur Data
Modul 4

SINGLE LINKED LIST (BAGIAN PERTAMA)

Disusun Oleh:

Dwi Candra Pratama – 2211104035

FAKULTAS iNFORMATIKA PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK TELKOM UNIVERSITY PURWOKERTO

GUIDED

A. Tujuan

1. Memahami penggunaan linked list dengan pointer operator- operator dalam program.
2. Memahami operasi-operasi dasar dalam linked list.
3. Membuat program dengan menggunakan linked list dengan prototype yang ada

- **Linked List**

Linked list (biasa disebut list saja) adalah salah satu bentuk struktur data (representasi penyimpanan) berupa serangkaian elemen data yang saling berkait (berhubungan) dan bersifat fleksibel karena dapat tumbuh dan mengerut sesuai kebutuhan. Data yang disimpan dalam Linked list bisa berupa data tunggal atau data majemuk. Data tunggal merupakan data yang hanya terdiri dari satu data (variabel), misalnya: nama bertipe string. Sedangkan data majemuk merupakan sekumpulan data (record) yang di dalamnya terdiri dari berbagai tipe data, misalnya: Data Mahasiswa, terdiri dari Nama bertipe string, NIM bertipe long integer, dan Alamat bertipe string. Linked list dapat diimplementasikan menggunakan Array dan Pointer (Linked list).

Model-model dari ADT Linked list yang di pelajari adalah :

1. Single Linked list
2. Double Linked list
3. Circular Linked list
4. Multi Linked list
5. Stack (Tumpukan)
6. Queue (Antrian)
7. Tree
8. Graph

Setiap model ADT Linked list di atas memiliki karakteristik tertentu dan dalam penggunaannya
disesuaikan dengan kebutuhan.

Secara umum operasi-operasi ADT pada Linked list, yaitu :

1. Penciptaan dan inisialisasi list (Create List).
2. Penyisipan elemen list (Insert).
3. Penghapusan elemen list (Delete).
4. Penelusuran elemen list dan menampilkannya (View).
5. Pencarian elemen list (Searching).
6. Pengubahan isi elemen list (Update).

- **Linked List Guided 1 (yang tadi dipraktekan)**

```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
Tabnine | Edit | Test | Explain | Document | Ask
void init() {
    head = nullptr;
    tail = nullptr;
}

// Pengecekan apakah list kosong
Tabnine | Edit | Test | Explain | Document | Ask
bool isEmpty() {
    return head == nullptr;
}
```

```
31  // Tambah Depan
    Tabnine | Edit | Test | Explain | Document | Ask
32  void insertDepan(const mahasiswa &data) {
33      Node *baru = new Node;
34      baru->data = data;
35      baru->next = nullptr;
36      if (isEmpty()) {
37          head = tail = baru;
38      } else {
39          baru->next = head;
40          head = baru;
41      }
42  }
43
44  // Tambah Belakang
    Tabnine | Edit | Test | Explain | Document | Ask
45  void insertBelakang(const mahasiswa &data) {
46      Node *baru = new Node;
47      baru->data = data;
48      baru->next = nullptr;
49      if (isEmpty()) {
50          head = tail = baru;
51      } else {
52          tail->next = baru;
53          tail = baru;
54      }
55  }
56
57  // Hitung Jumlah List
    Tabnine | Edit | Test | Explain | Document | Ask
58  int hitungList() {
59      Node *current = head;
60      int jumlah = 0;
61      while (current != nullptr) {
62          jumlah++;
63          current = current->next;
64      }
65      return jumlah;
66  }
```

```
68  // Hapus Depan
    Tabnine | Edit | Test | Explain | Document | Ask
69  void hapusDepan() {
70      if (!isEmpty()) {
71          Node *hapus = head;
72          head = head->next;
73          delete hapus;
74          if (head == nullptr) {
75              tail = nullptr; // Jika list menjadi kosong
76          }
77      } else {
78          cout << "List kosong!" << endl;
79      }
80  }
81
82  // Hapus Belakang
    Tabnine | Edit | Test | Explain | Document | Ask
83  void hapusBelakang() {
84      if (!isEmpty()) {
85          if (head == tail) {
86              delete head;
87              head = tail = nullptr; // List menjadi kosong
88          } else {
89              Node *bantu = head;
90              while (bantu->next != tail) {
91                  bantu = bantu->next;
92              }
93              delete tail;
94              tail = bantu;
95              tail->next = nullptr;
96          }
97      } else {
98          cout << "List kosong!" << endl;
99      }
100 }
```

```
// Tampilkan List
Tabnine | Edit | Test | Explain | Document | Ask
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
Tabnine | Edit | Test | Explain | Document | Ask
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
Tabnine | Edit | Test | Explain | Document | Ask
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:

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\GUIDED\output> & .\'modul4.exe'
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:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\GUIDED\output>
```

- **Linked List Guided 2 (yang tadi dipraktekan)**

```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
Tabnine | Edit | Test | Explain | Document | Ask
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
Tabnine | Edit | Test | Explain | Document | Ask
void dealokasi(Node* node) {
    delete node; // Mengembalikan memori yang digunakan oleh node
}

// Pengecekan apakah list kosong
Tabnine | Edit | Test | Explain | Document | Ask
bool isListEmpty(Node* head) {
    return head == nullptr; // List kosong jika head adalah nullptr
}

// Menambahkan elemen di awal list
Tabnine | Edit | Test | Explain | Document | Ask
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
    }
}
```

```
39  // Menambahkan elemen di akhir list
    Tabnine | Edit | Test | Explain | Document | Ask
40  void insertLast(Node* &head, int value) {
41      Node* newNode = alokasi(value); // Alokasi memori untuk elemen baru
42      if (newNode != nullptr) {
43          if (isListEmpty(head)) {  // Jika list kosong
44              head = newNode;       // Elemen baru menjadi elemen pertama
45          } else {
46              Node* temp = head;
47              while (temp->next != nullptr) { // Mencari elemen terakhir
48                  temp = temp->next;
49              }
50              temp->next = newNode; // Menambahkan elemen baru di akhir list
51          }
52      }
53  }
54
55  // Menampilkan semua elemen dalam list
    Tabnine | Edit | Test | Explain | Document | Ask
56  void printList(Node* head) {
57      if (isListEmpty(head)) {
58          cout << "List kosong!" << endl;
59      } else {
60          Node* temp = head;
61          while (temp != nullptr) {  // Selama belum mencapai akhir list
62              cout << temp->data << " "; // Menampilkan data elemen
63              temp = temp->next; // Melanjutkan ke elemen berikutnya
64          }
65          cout << endl;
66      }
67  }
```

```
68
69  // Menghitung jumlah elemen dalam list
    Tabnine | Edit | Test | Explain | Document | Ask
70  int countElements(Node* head) {
71      int count = 0;
72      Node* temp = head;
73      while (temp != nullptr) {
74          count++;           // Menambah jumlah elemen
75          temp = temp->next; // Melanjutkan ke elemen berikutnya
76      }
77      return count;          // Mengembalikan jumlah elemen
78  }
79
80  // Menghapus semua elemen dalam list dan dealokasi memori
    Tabnine | Edit | Test | Explain | Document | Ask
81  void clearList(Node* &head) {
82      while (head != nullptr) {
83          Node* temp = head;  // Simpan pointer ke node saat ini
84          head = head->next;  // Pindahkan ke node berikutnya
85          dealokasi(temp);    // Dealokasi node
86      }
87  }
88
    Tabnine | Edit | Test | Explain | Document | Ask
89  int main() {
90      Node* head = nullptr;  // Membuat list kosong
91
92      // Menambahkan elemen ke dalam list
93      insertFirst(head, 10); // Menambahkan elemen 10 di awal list
94      insertLast(head, 20);  // Menambahkan elemen 20 di akhir list
95      insertLast(head, 30);  // Menambahkan elemen 30 di akhir list
96
97      // Menampilkan isi list
98      cout << "Isi List: ";
99      printList(head);
100
101     // Menampilkan jumlah elemen
102     cout << "Jumlah elemen: " << countElements(head) << endl;
103
104     // Menghapus semua elemen dalam list
105     clearList(head);
106
107     // Menampilkan isi list setelah penghapusan
108     cout << "Isi List setelah penghapusan: ";
109     printList(head);
110
111     return 0;
112 }
```

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\GUIDED\output> cd 'd:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\GUIDED\output'
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\GUIDED\output> & .\'Guided2M4.exe'
Isi List: 10 20 30
Jumlah elemen: 3
Isi List setelah penghapusan: List kosong!
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\GUIDED\output>
```

Unguided

1. **Membuat Single Linked List**

   Buatlah program C++ untuk membuat sebuah single linked list dengan operasi dasar sebagai berikut:

   - Insert Node di Depan: Fungsi untuk menambah node baru di awal linked list.
   - Insert Node di Belakang: Fungsi untuk menambah node baru di akhir linked list.
   - Cetak Linked List: Fungsi untuk mencetak seluruh isi linked list.

   Contoh input dan output:

   Input:

   1. Tambah node di depan (nilai: 10)
   2. Tambah node di belakang (nilai: 20)
   3. Tambah node di depan (nilai: 5)
   4. Cetak linked list

   Output:

   5 -> 10 -> 20

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED\output> & .\'SoalNo1.exe'
1. Tambah node di depan 10
2. Tambah node di belakang 20
3. Tambah node di depan 5
4. Cetak linked list
5 -> 10 -> 20
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED\output>
```

SourceCode

UNGUIDED > SoalNo1.cpp > main()

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

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

class LinkedList {
private:
    Node* head;

public:
    Tabnine | Edit | Test | Explain | Document | Ask
    LinkedList() {
        head = NULL;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void insertDepan(int nilai) {
        Node* newNode = new Node();
        newNode->data = nilai;
        newNode->next = head;
        head = newNode;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void insertBelakang(int nilai) {
        Node* newNode = new Node();
        newNode->data = nilai;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            return;
        }

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

    Tabnine | Edit | Test | Explain | Document | Ask
    void cetakList() {
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->data;
            if (temp->next != NULL) {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

Tabnine | Edit | Test | Explain | Document | Ask
int main() {
    LinkedList list;
    int nilai;

    cout << "1. Tambah node di depan ";
    cin >> nilai;
    list.insertDepan(nilai);

    cout << "2. Tambah node di belakang ";
    cin >> nilai;
    list.insertBelakang(nilai);

    cout << "3. Tambah node di depan ";
    cin >> nilai;
    list.insertDepan(nilai);

    cout << "4. Cetak linked list\n";
    list.cetakList();

    return 0;
}
```

2. Menghapus Node pada Linked List

Buatlah program C++ yang dapat menghapus node tertentu dalam single linked list berdasarkan nilai yang diberikan oleh pengguna. Tugas ini mencakup operasi:

- Delete Node dengan Nilai Tertentu: Fungsi untuk menghapus node yang memiliki nilai tertentu.
- Cetak Linked List: Setelah penghapusan, cetak kembali isi linked list.

Contoh input/output:

Input:

1. Tambah node di depan (nilai: 10)
2. Tambah node di belakang (nilai: 20)
3. Tambah node di depan (nilai: 5)
4. Hapus node dengan nilai (nilai: 10)
5. Cetak linked list

Output:

5 -> 20

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4> cd "d:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED\" ; if ($?) { g++ SoalNo2.cpp -o SoalNo2
} ; if ($?) { .\SoalNo2 }
1. Tambah node di depan ,nilai: 10
2. Tambah node di belakang ,nilai: 20
3. Tambah node di depan ,nilai: 5
4. Hapus node dengan nilai ,nilai: 10
5. Cetak linked list
5 -> 20
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED>
```

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

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

class LinkedList {
private:
    Node* head;

public:
    Tabnine | Edit | Test | Explain | Document | Ask
    LinkedList() {
        head = NULL;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void insertDepan(int nilai) {
        Node* newNode = new Node();
        newNode->data = nilai;
        newNode->next = head;
        head = newNode;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void insertBelakang(int nilai) {
        Node* newNode = new Node();
        newNode->data = nilai;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            return;
        }

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

    Tabnine | Edit | Test | Explain | Document | Ask
    void hapusNode(int nilai) {
        Node* temp = head;
        Node* prev = NULL;

        if (temp != NULL && temp->data == nilai) {
            head = temp->next;
            delete temp;
            return;
        }
```

```
        while (temp != NULL && temp->data != nilai) {
            prev = temp;
            temp = temp->next;
        }

        if (temp == NULL) return;

        prev->next = temp->next;
        delete temp;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void cetakList() {
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->data;
            if (temp->next != NULL) {
                cout << " -> ";
            }
            temp = temp->next;
        }
        cout << endl;
    }
};

Tabnine | Edit | Test | Explain | Document | Ask
int main() {
    LinkedList list;
    int nilai;

    cout << "1. Tambah node di depan ,nilai: ";
    cin >> nilai;
    list.insertDepan(nilai);

    cout << "2. Tambah node di belakang ,nilai: ";
    cin >> nilai;
    list.insertBelakang(nilai);

    cout << "3. Tambah node di depan ,nilai: ";
    cin >> nilai;
    list.insertDepan(nilai);

    cout << "4. Hapus node dengan nilai ,nilai: ";
    cin >> nilai;
    list.hapusNode(nilai);

    cout << "5. Cetak linked list" << endl;
    list.cetakList();

    return 0;
}
```

3. Mencari dan Menghitung Panjang Linked List

Buatlah program C++ yang dapat melakukan operasi berikut:

- Cari Node dengan Nilai Tertentu: Fungsi untuk mencari apakah sebuah nilai ada di dalam linked list.
- Hitung Panjang Linked List: Fungsi untuk menghitung jumlah node yang ada di dalam linked list. Contoh input/output:

Input:

1. Tambah node di depan (nilai: 10)
2. Tambah node di belakang (nilai: 20)
3. Tambah node di depan (nilai: 5)
4. Cari node dengan nilai 20
5. Cetak panjang linked list

Output:

Node dengan nilai 20 ditemukan.

Panjang linked list: 3

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED\output> cd 'd:\Semester5\StrukturData\PraktikumStrukturDa
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED\output> & .\'SoalNo3.exe'
1. Tambah node di depan 10
2. Tambah node di belakang 20
3. Tambah node di depan 5
4. Cari node dengan nilai 20
Node dengan nilai 20 ditemukan.
5. Cetak panjang linked list
Panjang linked list: 3
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 4\UNGUIDED\output>
```

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

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

class LinkedList {
private:
    Node* head;

public:
    Tabnine | Edit | Test | Explain | Document | Ask
    LinkedList() {
        head = NULL;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void insertDepan(int nilai) {
        Node* newNode = new Node();
        newNode->data = nilai;
        newNode->next = head;
        head = newNode;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    void insertBelakang(int nilai) {
        Node* newNode = new Node();
        newNode->data = nilai;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            return;
        }

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

    Tabnine | Edit | Test | Explain | Document | Ask
    bool cariNode(int nilai) {
        Node* temp = head;
        while (temp != NULL) {
            if (temp->data == nilai)
                return true;
            temp = temp->next;
        }
        return false;
    }

    Tabnine | Edit | Test | Explain | Document | Ask
    int hitungPanjang() {
        int count = 0;
        Node* temp = head;
        while (temp != NULL) {
            count++;
            temp = temp->next;
        }
        return count;
    }
};

Tabnine | Edit | Test | Explain | Document | Ask
int main() {
    LinkedList list;
    int nilai;

    cout << "1. Tambah node di depan ";
    cin >> nilai;
    list.insertDepan(nilai);

    cout << "2. Tambah node di belakang ";
    cin >> nilai;
    list.insertBelakang(nilai);

    cout << "3. Tambah node di depan ";
    cin >> nilai;
    list.insertDepan(nilai);

    cout << "4. Cari node dengan nilai ";
    cin >> nilai;
    if (list.cariNode(nilai)) {
        cout << "Node dengan nilai " << nilai << " ditemukan." << endl;
    }

    cout << "5. Cetak panjang linked list" << endl;
    cout << "Panjang linked list: " << list.hitungPanjang() << endl;

    return 0;
}
```