

LAPORAN PRAKTIKUM
Modul 5
SINGLE LINKED LIST (BAGIAN KEDUA)



Disusun Oleh:

Nama : Ganes Gemi Putra

NIM : 2311104075

Kelas : SE-07-02

Dosen : WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

Memahami penggunaan linked list dengan pointer operator- operator dalam program.

Memahami operasi-operasi dasar dalam linked list.

Membuat program dengan menggunakan linked list dengan prototype yang ada

2. Landasan Teori

Operasi Utama:

Pencarian: Operasi ini melibatkan traversal (penelusuran) node satu per satu untuk menemukan nilai tertentu. Pencarian dilakukan hingga node yang diinginkan

ditemukan atau akhir dari list tercapai.

Contoh pencarian: Mengunjungi setiap node dan membandingkan nilainya dengan nilai target.

Penyisipan: Menambahkan elemen ke linked list. Ada berbagai cara untuk menyisipkan node, termasuk di awal, setelah node tertentu, atau di akhir list.

Penghapusan: Menghapus elemen dari list. Seperti penyisipan, node bisa dihapus dari awal, setelah node tertentu, atau dari akhir.

Manajemen Memori: Operasi seperti alokasi dan dealokasi memori untuk node, memastikan sumber daya digunakan secara efisien.

Tipe Data Abstrak (ADT):

ADT mendefinisikan perilaku dari linked list, termasuk:

Memeriksa apakah list kosong.

Menyisipkan dan menghapus node pada berbagai posisi.

Mencari elemen di dalam list.

Mencetak atau menelusuri semua elemen dalam list.

Penggunaan Praktis:

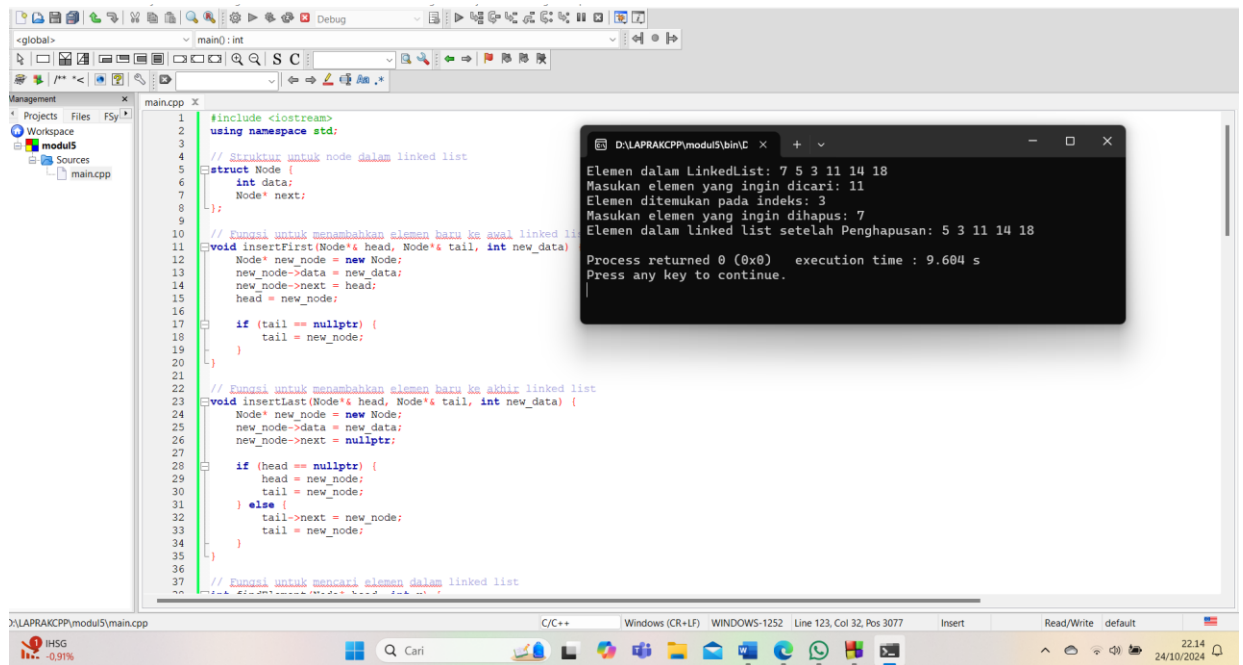
Modul ini memandu Anda dalam membuat fungsi seperti:

InsertFirst: Menyisipkan node di awal list.

FindElm: Mencari node dengan data tertentu.

Operasi penghapusan: Menghapus elemen dari lokasi tertentu dalam list.

3. Guided



The screenshot shows a C++ IDE with a file named `main.cpp` containing the following code:

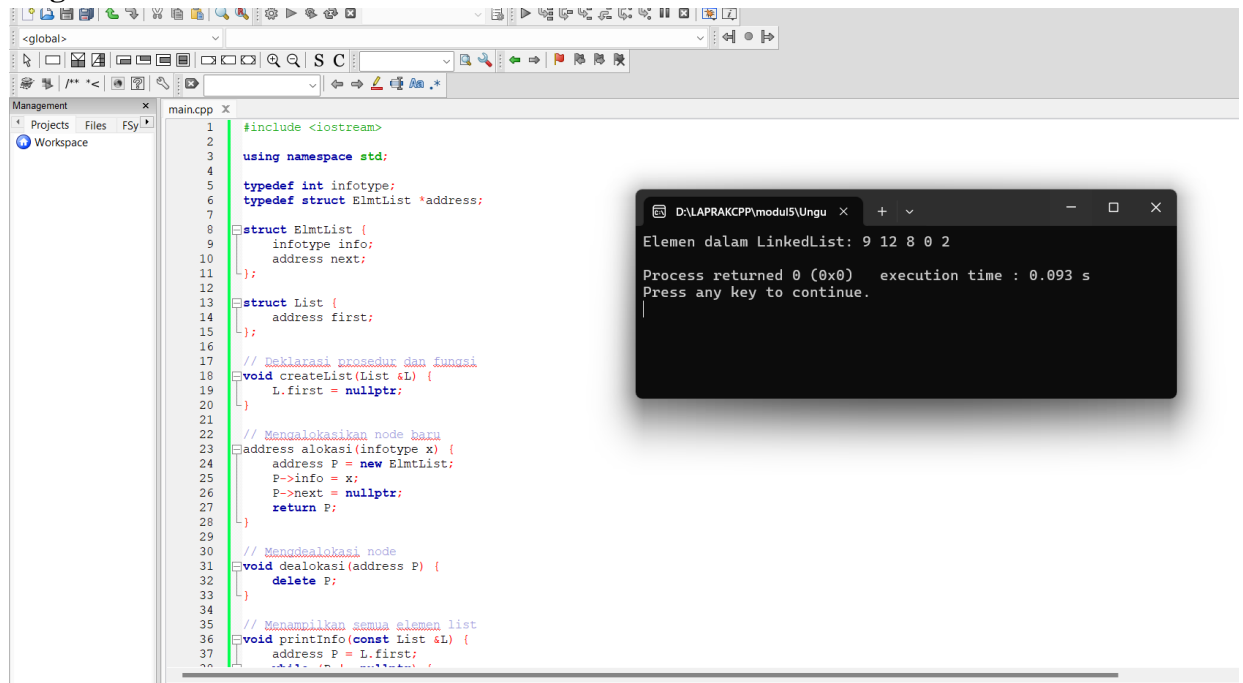
```
1 #include <iostream>
2 using namespace std;
3
4 // Struktur untuk node dalam linked list
5 struct Node {
6     int data;
7     Node* next;
8 };
9
10 // Fungsi untuk menambahkan elemen baru ke awal linked list
11 void insertFirst(Node*& head, Node*& tail, int new_data) {
12     Node* new_node = new Node;
13     new_node->data = new_data;
14     new_node->next = head;
15     head = new_node;
16
17     if (tail == nullptr) {
18         tail = new_node;
19     }
20 }
21
22 // Fungsi untuk menambahkan elemen baru ke akhir linked list
23 void insertLast(Node*& head, Node*& tail, int new_data) {
24     Node* new_node = new Node;
25     new_node->data = new_data;
26     new_node->next = nullptr;
27
28     if (head == nullptr) {
29         head = new_node;
30         tail = new_node;
31     } else {
32         tail->next = new_node;
33         tail = new_node;
34     }
35 }
36
37 // Fungsi untuk mencari elemen dalam linked list
```

The terminal window shows the following output:

```
Elemen dalam LinkedList: 7 5 3 11 14 18
Masukan elemen yang ingin dicari: 11
Elemen ditemukan pada indeks: 3
Masukan elemen yang ingin dihapus: 7
Elemen dalam linked list setelah Penghapusan: 5 3 11 14 18

Process returned 0 (0x0)   execution time : 9.684 s
Press any key to continue.
```

4. Unguided



The screenshot shows a C++ IDE with a file named `main.cpp` containing the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 typedef int infotype;
5 typedef struct ElmtList *address;
6
7 struct ElmtList {
8     infotype info;
9     address next;
10 };
11
12 struct List {
13     address first;
14 };
15
16 // Deklarasi prosedur dan fungsi
17 void createList(List &L) {
18     L.first = nullptr;
19 }
20
21 // Mengalokasikan node baru
22 address alokasi(infotype x) {
23     address P = new ElmtList;
24     P->info = x;
25     P->next = nullptr;
26     return P;
27 }
28
29 // Mengalokasikan node
30 void dealokasi(address P) {
31     delete P;
32 }
33
34 // Menampilkan semua elemen list
35 void printInfo(const List &L) {
36     address P = L.first;
37     while (P != nullptr) {
38         cout << P->info << " ";
39         P = P->next;
40     }
41     cout << endl;
42 }
```

The terminal window shows the following output:

```
Elemen dalam LinkedList: 9 12 8 0 2

Process returned 0 (0x0)   execution time : 0.093 s
Press any key to continue.
```

5. Kesimpulan

- **Pemahaman Dasar Linked List:** Modul ini memberikan pemahaman mendalam tentang **single linked list**, di mana setiap elemen dalam list disebut sebagai **node**, yang terdiri dari data dan pointer ke elemen berikutnya. Linked list sangat berguna untuk

mengelola data dinamis karena memungkinkan penambahan dan penghapusan elemen secara efisien tanpa perlu menggeser data lain seperti pada array.

- **Operasi Dasar Linked List:** Operasi dasar yang dijelaskan meliputi **penyisipan (insertion)**, **penghapusan (deletion)**, **pencarian (searching)**, dan **penelusuran elemen (traversing)**. Modul ini menunjukkan bagaimana operasi tersebut diterapkan menggunakan bahasa pemrograman C/C++, serta bagaimana memanfaatkan pointer untuk memanipulasi elemen-elemen dalam linked list.
- **Manajemen Memori:** Dalam linked list, alokasi dan dealokasi memori merupakan bagian penting dari manajemen data. Setiap node baru dialokasikan memori saat dibutuhkan, dan ketika node dihapus, memori yang digunakan harus dikembalikan (deallocated) untuk menghindari kebocoran memori.
- **Penerapan ADT (Abstract Data Type):** Modul ini juga mengajarkan penerapan ADT untuk mengelola operasi-operasi pada linked list dengan cara yang terstruktur. ADT membuat kode lebih modular dan mudah dipahami, memungkinkan implementasi dan manipulasi linked list menjadi lebih efisien.
- **Latihan Praktis:** Latihan-latihan dalam modul memberikan pemahaman praktis tentang bagaimana membangun dan mengimplementasikan **single linked list** dari awal, termasuk penyusunan struktur data, pembuatan fungsi untuk setiap operasi, dan penerapannya dalam program nyata. Latihan tersebut membantu memperkuat konsep yang dipelajari.